INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Masterarbeit

Datenvisualisierung zur Unterstützung bei der Linux-Serveradministration in Hinblick auf Informationssicherheit

Friedrich Schmidt

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Masterarbeit

Datenvisualisierung zur Unterstützung bei der Linux-Serveradministration in Hinblick auf Informationssicherheit

Friedrich Schmidt

Aufgabensteller: PD Dr. Wolfgang Hommel

Betreuer: Tanja Hanauer
Daniela Pöhn

Abgabetermin: 03. August 2015

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 03. August 2015

. . . . . .

(Unterschrift des Kandidaten)

Administratoren von Linux-Servern sind zunehmenden Herausforderungen ausgesetzt: Sowohl die Zahl der Dienst-Benutzer als auch der Server nimmt immer mehr zu. Log-Dateien wachsen immer schneller und der Überblick über Konfigurationsdateien verschwindet mit der Zeit. Informationsvisualisierung dient dem Zweck, unüberschaubare Datenmengen beherrschbar und vor allem übersichtlicher zu machen. Diese Masterarbeit befasst sich mit der Frage, inwiefern Methoden der Informationsvisualisierung im Kontext der Linux-Serveradministration unter Berücksichtigung der Informationssicherheit gewinnbringend eingesetzt werden können. Um ein allgemeines Verständnis zu vermitteln, werden zunächst wichtige Grundlagen der Informationsvisualisierung und der Informationssicherheit angesprochen. Nach einer Ermittlung des Istzustands am Leibniz-Rechenzentrum und einer Anforderungsanalyse erfolgt die Evaluation einer Auswahl repräsentativer Administrationstools, die bereits Visualisierungsmethoden einsetzen. Für eine repräsentative Auswahl an Anwendungsfällen aus der Linux-Server-Administration werden schließlich Lösungsmöglichkeiten für Visualisierungen erarbeitet, die einen Mehrwert gegenüber vorhandenen Produkten bieten. Diese Forschungsergebnisse fließen schließlich in eine zu entwickelnde prototypische Web-Anwendung ein, die unter Berücksichtigung von Grundlagen der Informationsvisualisierung und der Informationssicherheit erstellt wird. Eine anschließende Evaluierung der Anwendung gibt Aufschluss über den tatsächlich gewonnenen Mehrwert durch Informationsvisualisierung bei der Administration von Linux-Servern.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1. Einleitung

Hochschulrechenzentren stellen ein breites Sortiment an Diensten für Anwender, wie Hochschulangehörige oder Grid-Nutzer, bereit. Um ihre Aufgaben erfüllen zu können, werden zahlreiche physikalische Server gehostet und virtuelle Maschinen bereitgestellt. Wenn es um die eingesetzten Server-Betriebssysteme geht, haben sich auf Linux basierende Distributionen durchgesetzt [DIE 13]. Durch den Einsatz dieser Open-Source-Betriebssysteme erwartet man Kosteneinsparungen gegenüber kommerziellen Produkten und Unabhängigkeit von Softwareherstellern.[1] Anpassungen an Open-Source-Programmen sind bei Bedarf ohne Weiteres möglich. Diese Flexibilität wirkt sich positiv im Betrieb aus.

Um den ebenfalls gestiegenen Anforderungen an die bereitgestellten Dienste gerecht zu werden, wächst die Anzahl der eingesetzten Linux-Server im Lauf der Zeit immer mehr. Dies stellt die verantwortlichen Systemadministratoren vor immer größere Herausforderungen. Selbst wenn bei einem System nur Standardeingriffe, wie zum Beispiel das Einspielen neuer Software über das Paketsystem, getätigt werden müssen, summiert sich der Zeitaufwand, da zahlreiche Server davon betroffen sind. Wenn beispielsweise eine Übersicht der auf den Servern installierten Software- und Patch-Stände erstellt werden soll, müsste jedes einzelne System angetastet und die notwendige Information über die Kommandozeile abgefragt werden. Der Zeitaufwand steigt in Abhängigkeit von den zu verwaltenden Rechnern. Im Idealfall müssen noch aufwendige Tests vor dem tatsächlichen Einspielen der Updates auf den Produktivsystemen durchgeführt werden.

Die Zusammenstellung von Informationen über kritische Events erfolgt in einigen Fällen zwar automatisch in der Form von fortlaufend aktualisierten Log-Dateien. Die dort erhaltenen textuellen Informationen müssen aber vorgefiltert, in ein gut lesbares Format übertragen und schließlich von Menschen interpretiert werden. Diese Aufgabe kann zeitraubende Ausmaße annehmen. Zudem stechen essentielle Informationen nicht sofort ins Auge, wenn nur textuelle Repräsentationsformen betrachtet werden. Wichtige Informationen werden ggf. erst nach erheblicher Dauer gefunden oder im schlimmsten Fall aufgrund der fehlenden Überschaubarkeit gar übersehen. In einem Hochschulrechenzentrum stellt dies ein ernst zu nehmendes Problem dar, das sich letztendlich auf die Güte der bereitgestellten Dienste auswirkt: Wenn Administratoren primär mit zeitraubenden Routineaufgaben beschäftigt sind, stehen weniger Kapazitäten für andere Aufgaben zur Verfügung. Nicht nur *Quality of Service (QoS)*, sondern auch Sicherheitsaspekte können in Mitleidenschaft gezogen werden.

Das bekannte Sprichwort “Ein Bild sagt mehr als tausend Worte” lässt sich auch auf die Administration von Linux-Servern übertragen. Um große Datenmengen nicht nur übersichtlich, sondern auch beherrschbar zu machen, eignen sich Techniken der Visualisierung. In der Disziplin der computergestützten Informationsvisualisierung befasst man sich seit einigen Jahrzehnten damit, komplexe Datensätze mit multivariaten Daten in übersichtlicher Form auf dem Bildschirm darzustellen und dem Betrachter so essentielle Informationen schnell und effizient zu vermitteln. Nicht nur technologische, sondern auch psychologische Aspekte kommen in der Disziplin der Informationsvisualisierung zum Tragen.

1.1. Motivation

Die zuvor genannten Probleme lassen sich auf das Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften (LRZ) in Garching übertragen. Linux-Server-Installationen sind im Lauf der Zeit historisch angewachsen und erfordern viel Aufmerksamkeit von den Administratoren. Sie arbeiten oft auf der Kommandokonsole und führen dieselben Arbeiten an verschiedenen Systemen immer wieder durch. Sie sind mit vielen unterschiedlichen Konfigurationsdateien konfrontiert und müssen Protokolldateien interpretieren, um die Zustände der Server und der darauf befindlichen Dienste zu ermitteln.

[1] Enterprise-Versionen von Linux-Derivaten mit kommerziellem Support stellen eine Ausnahme dar.

Unterstützung dabei erhalten sie von verschiedenen Administrationstools, die oft unabhängig voneinander eingesetzt werden. Neben Kommandozeilentools handelt es sich dabei auch um Programme mit grafischer Bedienoberfläche, die entweder native Client-Anwendungen oder webbasierte Anwendungen sind. Bei einigen dieser Tools handelt es sich um Eigenentwicklungen, die an die konkreten Bedürfnisse des LRZ angepasst wurden. Diese Programme haben gemeinsam, dass sie einige wichtige Zustandsinformationen über die Linux-Server an zentraler Stelle bereitstellen. Damit können sicherheitsrelevante Zustände einer großen Anzahl von Linux-Servern abgefragt werden, ohne jedes System einzeln anrühren zu müssen. Neben Texten und Tabellen nutzen diese Tools zum Teil auch visuelle Darstellungsformen wie etwa Diagramme, um die Serverzustände auszudrücken.

Grafische Darstellungsformen haben gegenüber unübersichtlichen Texten oder Texten den Vorteil, dass sie schneller zu begreifen sind und die Aufmerksamkeit des Betrachters auf sich ziehen. Je nach Informationsgehalt sind sie auch platzsparender auf einem Bildschirm unterzubringen als es bei Tabellen oder Texten der Fall ist. Für die Art und Weise in der die verschiedenen Administrationstools Serverzustände visuell darstellen gibt es keinen allgemeingültigen Standard. Dies führt einerseits dazu, dass die Tools vollkommen unterschiedliche Serverdaten visualisieren, andererseits besitzen die Visualisierungen sehr unterschiedliche Qualität. Dies ist der Grund dafür, dass viele unterschiedliche Tools parallel und unabhängig voneinander betrieben werden.

Die Motivation dieser Masterarbeit besteht darin, Lösungsansätze für die Visualisierung von sicherheitsrelevanten Server-Zustandsinformationen zu finden, um die Administration großer Linux-Server-Umgebungen sowohl zeitsparender als auch sicherer zu gestalten. Zeitersparnis soll dadurch erreicht werden, dass Administratoren essentielle Informationen über ihre Systeme in visuell ansprechender Form präsentiert bekommen. Dadurch können sie sich besser ihren anderen wichtigen Aufgaben widmen und vergeuden keine kostbare Arbeitszeit mit dem Betrachten von Textkolonnen. Eine höhere Sicherheit resultiert aus einer verbesserten Übersichtlichkeit im Vergleich zu herkömmlichen Methoden.

Da qualitativ hochwertige Visualisierungen gewinnbringender in der Administration von Linux-Servern eingesetzt werden können, besteht eine weitere Motivation dieser Arbeit in der Optimierung bestehender Darstellungsformen. Bei diesem Bestreben kann auf Grundlagen der Informationsvisualisierung zurückgegriffen werden.

Da Menschen beim Betrachten von langen Textdateien auch Informationen übersehen können, profitieren sie von einer visuellen Darstellung sicherheitskritischer Daten. Die menschliche Fähigkeit, bestimmte Bildinhalte präattentiv [ALE 13, S. 238] wahrzunehmen, begünstigt die Suche nach visuellen Darstellungsformen für sicherheitskritische Informationen einer Linux-Server-Umgebung.

Zeitersparnis zu erzielen ist nicht nur aus finanziellen Gründen erstrebenswert, sie wirkt sich auch auf die Informationssicherheit aus. Indem sicherheitsrelevante Informationen durch Methoden der Informationsvisualisierung augenblicklich erfassbar sind, können Administratoren schneller Sicherheitslücken finden und darauf reagieren. Es ist auch denkbar, mit Hilfe der Visualisierung von Server-Zustandsdaten proaktiv tätig zu werden und sich anbahnende Sicherheitsprobleme rechtzeitig zu erkennen. Damit wäre ein weiterer Beitrag zur Informationssicherheit am LRZ geleistet.

1.2. Zielsetzung

Ziel dieser Masterarbeit ist es, herauszufinden, ob bestimmte Methoden der Informationsvisualisierung für die Administration von Linux-Server-Systemen gewinnbringend genutzt werden können. Diese Frage stellt sich insbesondere im Kontext des LRZ, wo eine große Anzahl an Systemen vorhanden ist und eine ungleich höhere Anzahl an Benutzern unterbrechungsfrei mit Diensten versorgt werden muss. In Anbetracht der Bedürfnisse, die am LRZ im Bereich der Linux-Server-Administration vorherrschen, soll aus diesem Grund eine Anforderungsanalyse erstellt werden. Es soll zusammengefasst werden, welche funktionalen und nichtfunktionalen Anforderungen an ein Administrationstool gestellt werden, das zur Überwachung einer großen Anzahl von Linux-Servern dient.

Im nächsten Schritt soll die Tauglichkeit bereits existierender Visualisierungsmethoden für systemadministrative Tätigkeiten im Rahmen dieser Arbeit untersucht und bewertet werden. Dies geschieht an einer repräsentativen Auswahl an Administrationstools, die zum Teil auch am LRZ Verwendung finden. Zu diesen

Tools zählen beispielsweise *Ansible* (siehe Abschnitt 5.2.1) und *Splunk* (siehe Abschnitt 5.4.4). Es soll festgestellt werden, wie zielführend dort bereits Methoden der Informationsvisualisierung eingesetzt werden und wo Defizite herrschen. Bei der Bewertung der Tools soll auf die Ergebnisse zurückgegriffen werden, die in der Anforderungsanalyse erarbeitet wurden.

Die Erkenntnisse aus dieser Evaluierung sowie Grundlagen der Informationsvisualisierung sollen später herangezogen werden, um eine geeignete Visualisierungsmethode für verschiedene typische Anwendungsfälle der Linux-Serveradministration zu entwickeln. Idealerweise sollte eine gemeinsame Visualisierungsmethode entwickelt werden, das sich für mehr als einen Anwendungsfall eignet. Eine Vereinheitlichung bietet den Vorteil, dass es einen geringeren Einarbeitungsaufwand erfordert. Außerdem ermöglicht es eine effizientere Nutzung durch die Administratoren, da sie nicht ständig zwischen mehreren unterschiedlichen Anwendungen oder Ansichten umschalten und umdenken müssen.

Das im Rahmen dieser Arbeit zu entwickelnde Visualisierungsschema soll sich als Grundlage für ein mögliches Produktivsystem eignen. Der Zielgedanke hierbei ist es, nicht einfach nur ein weiteres Administrationstool zu schaffen, weil die bereits vorhandenen nicht ausreichen würden. Stattdessen soll ein Informationssystem, das das erarbeitete Visualisierungsschema implementiert, als zentrale Anlaufstelle dienen, sodass die Administratoren seltener zu einem speziellen Tool greifen müssen.

Ergänzend zu der theoretischen Ausführung in dieser Ausarbeitung soll eine prototypische Anwendung implementiert werden, die die Umsetzbarkeit und den gewinnbringenden Einsatz von Informationsvisualisierung im Linux-Server-Kontext zeigen soll. Neben den Grundlagen der Informationsvisualisierung soll der Prototyp auch ein grundlegendes Maß an Informationssicherheit zur Verfügung stellen. Zudem soll er im Rahmen der Möglichkeiten die Anforderungen aus der zuvor erfolgten Anforderungsanalyse erfüllen. Der Demonstrator soll mit der Hilfe etablierter Open-Source-Technologien implementiert werden. Eine Einfache Erweiterbarkeit bzw. Wiederverwendbarkeit ist wünschenswert, um den Demonstrator ggf. zu einem Wirksystem weiterentwickeln zu können oder um Teile davon in bestehende Eigenentwicklungen übertragen zu können.

Auf lange Sicht sollen die Ergebnisse dieser Masterarbeit dazu dienen, die Administration der Linux-Server am LRZ effizienter und sicherer zu gestalten. Dies kann durch Folgearbeiten erreicht werden, die auf den Ergebnissen dieser Masterarbeit aufbauen.

1.3. Vorgehensweise

Im Rahmen dieser Arbeit werden dem Leser zunächst für das allgemeine Verständnis wichtige Grundlagen der Informationsvisualisierung vermittelt. Dies geschieht im Kapitel 2. Das darauf folgende Kapitel 3 befasst sich kurz mit den im Rahmen dieser Arbeit nennenswerten Grundlagen der Informationssicherheit.

Bevor die prototypische Anwendung implementiert und getestet werden kann, sind zunächst richtungsentscheidende Schritte notwendig. Da der Prototyp für das LRZ entwickelt werden soll und unter Umständen als Grundlage für spätere Produktivsysteme dienen wird, sind zunächst die gegebenen Voraussetzungen und die spezifischen Anforderungen am LRZ zusammenzutragen und zu analysieren. Dies erfolgt im Kapitel 4.

Die Ausgangslage am LRZ wird in Abschnitt 4.1 dargestellt. Zur ihr zählen unter anderem die dort angebotenen Dienste und mit Hilfe welcher Server-Systeme sie erbracht werden. Unter Berücksichtigung der Ausgangssituation am LRZ erfolgt eine Anforderungsanalyse. Es werden funktionale, nichtfunktionale und visuelle Anforderungen zusammengetragen, die an ein Administrationswerkzeug zur Überwachung vieler Linux-Server gestellt werden.

Nach der Anforderungsanalyse erfolgt eine Evaluation einer repräsentativen Auswahl an Administrationstools (siehe Kapitel 5). Die Tools werden anhand der Qualität der Umsetzung informationsvisualisierender Methoden und anhand der Ergebnisse der Anforderungsanalyse bewertet. Als Grundlage zur Bewertung der Visualisierungen und der Bedienoberfläche wird auf das Kapitel 2 zurückgegriffen.

Linux-Administration umfasst sehr vielfältige Anwendungsfälle. Dazu zählen zum Beispiel Benutzerverwaltung, Rechteverwaltung, Firewall-Konfiguration, Software-Aktualisierung, Datenbankadministration, Storage, Backup, das Deployment von Webanwendungen und viele mehr. Um den Rahmen dieser Arbeit nicht zu spren-

gen, werden ein paar repräsentative Use Cases ausgesucht, die aufgrund ihrer Häufigkeit und ihrem Bezug zur Informationssicherheit besonders wichtig sind.

Im Kapitel 6 werden schließlich für die ausgesuchten Anwendungsfälle aus der Linux-Server-Administration Vorschläge für zielführende Visualisierungsmaßnahmen erarbeitet. Dies erfolgt unter Berücksichtigung der Grundlagenkapitel aus dieser Arbeit. Es wird eine Visualisierungsform erarbeitet, das sich gleichermaßen für unterschiedliche Use Cases im Kontext der Linux-Server-Administration eignet.

Da die Grundlage von Informationsvisualisierung unüberschaubare Rohdaten sind, wird für jeden der ausgewählten Anwendungsfälle bestimmt, welche verwertbaren Daten auf dem Server als Quelle zur Visualisierung in Frage kommen. Ggf. muss ein Server erst auf eine geeignete Art konfiguriert werden oder mit zusätzlicher Software ausgestattet werden, um die erforderlichen Informationen zu liefern. In Abhängigkeit der anfallenden Informationen werden dann geeignete Verfahren der Informationsvisualisierung ermittelt. Teile der Ergebnisse dieses Kapitels dienen schließlich als Vorlage für eine praktische Umsetzung in der prototypischen Anwendung.

Bevor die Implementierung des Prototyps beginnen kann, müssen einige Gegebenheiten des LRZ berücksichtigt werden. Nicht jeder Administrator ist für sämtliche vorhandenen Systeme verantwortlich, sondern nur für einen bestimmten Teil. Dies gilt nicht nur, weil eine Person allein nicht die Administration aller Server übernehmen kann, sondern hat auch datenschutzrechtliche Gründe. Diese Tatsache ist bei der Erstellung des Datenmodells und des Rechtesystems sowie bei der Planung der Bedienoberfläche und des Workflows zu berücksichtigen.

Einige technische Rahmenbedingungen für den Prototyp sind ebenfalls vorhanden und werden berücksichtigt. So soll das Ergebnis des praktischen Teils dieser Arbeit eine webbasierte Anwendung werden. Die damit verbundenen Vorteile, wie die leichte Zugreifbarkeit von verschiedenen Geräten aus und die Unabhängigkeit von den zugrundeliegenden Client-Architekturen, sprechen für diesen Ansatz. Nachzulesen sind die konkreten Anforderungen in Abschnitt 4.2.

Sobald alle notwendigen Informationen vorhanden sind, kann mit der Entwicklung der prototypischen Anwendung begonnen werden. Diesem Thema widmet sich Kapitel 7. Die Entwicklung beginnt mit der Planung des Datenmodells, das später noch vorgestellt wird. Um das Rad nicht komplett neu zu erfinden, werden nach Möglichkeit Software-Bibliotheken verwendet, die geeignete Visualisierungstechniken bereits unterstützen oder die Entwicklung eigener Methoden zumindest begünstigen.

Es wird Wert darauf gelegt, hauptsächlich auf quelloffene Software zurückzugreifen, um die Unabhängigkeit von kommerziellen Anbietern weitestgehend zu gewährleisten. Bei der Entwicklung der Anwendung wird auf Werkzeuge und Technologien zurückgegriffen, die sich bereits in dem Umfeld bewährt haben, wo Client-Server-basierte Web-Applikationen im Fokus stehen. Die verwendeten Technologien und Plattformen werden im gleichnamigen Unterkapitel 7.2.1 kurz vorgestellt.

Die konkreten Anforderungen an den Prototyp werden im Rahmen dieser Arbeit berücksichtigt. Es werden Konzepte umgesetzt, die einen hohen Grad an Bedienfreundlichkeit bieten. Es wird gezeigt, dass mit den erarbeiteten Visualisierungsmethoden ein Mehrwert im Vergleich zu herkömmlichen Administrationstools geschaffen werden kann.

Nach der erfolgreichen Implementierung ausgewählter Teile der Ergebnisse aus Kapitel 6 im Prototyp erfolgt eine weitere Evaluation, die in Kapitel 8 nachzulesen ist. Diesmal stehen nicht bereits existierende Produkte, sondern der Prototyp im Vordergrund. Es wird bewertet, inwiefern der angefertigte Demonstrator den Anforderungen, die in Kapitel 4.2 aufgestellt wurden, gerecht wird. Dies geschieht unter Berücksichtigung der Grundlagenkapitel am Anfang dieser Ausarbeitung.

Da der im Rahmen dieser Masterarbeit anzufertigende Prototyp die gewinnbringende Verwendbarkeit der erarbeiteten Visualisierungsmethoden demonstriert, liegt es nahe, Teile daraus in den Wirkbetrieb zu überführen oder den Prototyp zu einem Wirksystem weiterzuentwickeln. Dies ist allerdings nicht mehr Bestandteil dieser Arbeit, stattdessen werden mögliche weitere Ausbaustufen im Kapitel 9 angesprochen. Dieses Kapitel ist auch für die Zusammenfassung der erreichten Ergebnisse dieser Arbeit vorgesehen. Da die Themenbereiche der Linux-Server-Administration, der Informationsvisualisierung und der Informationssicherheit sehr weitreichend sind, werden auch mögliche Folgearbeiten angesprochen, die sich zur Bearbeitung durch andere Studenten anbieten.

2. Grundlagen der Informationsvisualisierung

In diesem Kapitel werden einige wichtige Grundlagen aus der Disziplin der Informationsvisualisierung vorgestellt. Dieser Grundlagenteil orientiert sich sowohl strukturell als auch inhaltlich an der Vorlesung *Informationsvisualisierung* von *Prof. Dr. Florian Alt* aus dem Wintersemester 2013/2014 der Ludwig-Maximilians-Universität München (LMU) [ALT 13] und an der gleichnamigen Vorlesung aus dem vorherigen Jahr von *Prof. Dr. Andreas Butz* [BUTZ 12].

Neben der Motivation für Informationsvisualisierung (siehe Abschnitt 2.1), Begriffserklärungen (siehe Abschnitte 2.2, 2.3 und 2.4), den Zielen und Herausforderungen (siehe Abschnitt 2.5) und einem kurzen Abriss ihrer Geschichte (siehe Abschnitt 2.6) werden Grundlagen vorgestellt, die für die spätere Anfertigung des Prototyps von Bedeutung sind. Dazu zählen zum Beispiel die Designgrundregeln von *Edward Tufte* (siehe Abschnitt 2.7), die sicherstellen sollen, dass die Visualisierungen klar strukturiert und frei von unnötigem Ballast ist.

Um von einer großen Menge an Rohdaten zu einer fertigen Visualisierung zu gelangen, sind mehrere nachgelagerte Schritte erforderlich. Die Abschnitte 2.8 und 2.9 befassen sich mit diesem Thema. Ein notwendiger Zwischenschritt bei diesem Prozess stellt das *Visual Encoding* dar, das in Abschnitt 2.10 beschrieben wird.

Psychologische Aspekte wie die präattentive Wahrnehmung oder die unterbewusste Gruppierung von Objekten durch die Gestaltpsychologie (siehe Abschnitt 2.11) sollen dort wo es Sinn ergibt ebenfalls in den Prototyp einfließen.

Um komplexe Datenstrukturen zu visualisieren, gibt es viele unterschiedliche Möglichkeiten. Eine repräsentative Auswahl davon wird im Abschnitt 2.12 vorgestellt. Dem sinnvollen Einsatz von Farben in Visualisierungen und Bedienoberflächen ist Abschnitt 2.13 gewidmet.

Computergenerierte Visualisierungen gewinnen gegenüber Papierexemplaren an Mehrwert, indem sie Interaktionen des Benutzers zulassen, die sich unmittelbar auf die Darstellung auswirken. Verschiedene Konzepte der Interaktion werden aus diesem Grund in Abschnitt 2.14 vorgestellt.

Die Visualisierung von Datensätzen, die durch Zeit parametrisiert sind, erfordert spezielle Methoden. Dasselbe gilt auch für die Visualisierung von Prosatexten oder Programmcodes. Diesen besonderen Anwendungsfällen sind die Abschnitte 2.15 und 2.16 gewidmet.

In der heutigen Zeit gibt es Computer in den unterschiedlichsten Größenordnungen. Damit Visualisierungen auf Bildschirmen mit unterschiedlichen Größen und Auflösungen zur Verfügung stehen und möglichst viele essentielle Informationen beinhalten, können unterschiedliche Techniken der Präsentation eingesetzt werden. Eine Auswahl der wichtigsten befindet sich in Abschnitt 2.17.

2.1. Datenüberflutung

Dem Fortschritt der Informationstechnik ist es zu verdanken, dass Unmengen an Daten auf Datenträgern existieren oder über Kommunikationsnetze übertragen werden können. Menschen sind in der Lage, auf immer größer werdende Datenmengen zuzugreifen und auch selbst zu produzieren. Einen Eindruck vom Wachstum des Datenaufkommens erhält man, wenn man zwei Infografiken von `domo.com` gegenüberstellt. In einem Abstand von circa zwei Jahren wurden dort verschiedene Kennzahlen, wie etwa die Anzahl von *YouTube*-Uploads, *Google*-Suchanfragen oder versendeter E-Mails, visuell aufbereitet. Die Grafiken entstanden im Juni 2012 (siehe Abbildung 2.1) und im April 2014 (siehe Abbildung 2.2).

Sogar Maschinen stoßen regelmäßig an ihre Grenzen, um mit diesen immensen Datenmengen zurecht zu kommen. Große Online-Dienst-Betreiber wie *Google* oder *Facebook* müssen neue und leistungsfähigere Rechenzentren errichten, um mit dieser Entwicklung mithalten zu können [WIN 13]. Auch die Kommunikationsnetze bleiben von dieser Entwicklung nicht verschont. *Content Delivery Networks* mit tausenden Servern sorgen dafür, dass umfangreiche Multimedia-Daten die zahlreichen Konsumenten erreichen.

Nicht nur Privatkonsumenten erzeugen große Datenmengen, auch in der Wissenschaft fallen diese an. Sie dienen dazu, um konkrete Forschungsfragen zu beantworten, wie etwa in der Genforschung. Neben der Notwendigkeit von Computern, die große Datenmengen verarbeiten können, werden also auch Werkzeuge benötigt, die dem Menschen den Umgang damit erleichtern und ihn bei der Lösung konkreter Probleme unterstützen.

2.2. Daten vs. Informationen

Es wurden bereits zuvor die Begriffe *Daten* und *Informationen* genannt. Diese sind allerdings nicht synonym zu verstehen, denn es gibt eine Abgrenzung zwischen ihnen. Unter dem Begriff Daten versteht man im Allgemeinen Fakten oder Konzepte, die in einer für den Computer verarbeitbaren Form, also auf unterster Ebene binär kodiert, vorliegen.

Informationen bestehen aus Wissen, das aus der Interpretation von Daten abgeleitet wurde. Dazu müssen sie einen Abstraktionsprozess durchlaufen. Hier ein Beispiel: Aus einer Folge von Einsen und Nullen kann ein Mensch in der Regel nicht ablesen, dass es sich zum Beispiel um eine Musikdatei handelt. Für ihn ist diese Menge an Daten praktisch bedeutungslos. Erst wenn er weiß, wie sie zu interpretieren ist, kann er die gewünschte Information extrahieren. Im konkreten Beispiel mit der Musikdatei ist zu diesem Zweck Wissen über das vorliegende Container-Format und den gewählten Audio-Codec erforderlich. Nach dem Demultiplexing und der Dekodierung muss das Musikstück noch in einen Digital-to-Analog Converter (DAC) gespeist werden, denn der dekomprimierte Datenstrom ist für den Menschen immer noch nicht unmittelbar verwertbar. Das dort resultierende analoge Audio-Kleinsignal muss schließlich noch verstärkt und über Lautsprecher oder Kopfhörer wiedergegeben werden. Erst dann ist die eigentliche Information für den Menschen hörbar, nämlich Musik.

Informationen machen einen wesentlichen Bestandteil der menschlichen Kommunikation aus. Bei Informationen handelt es sich zum Beispiel um Wissen über Personen, Konzepte, Gegenstände, Naturgesetze etc. Sie liefern dem Menschen insbesondere nutzbare Antworten auf konkrete Fragestellungen. Der emeritierte Informatik-Professor der ETH Zürich *Carl August Zehnder* spricht genau dann von einer Information, wenn auf eine spezifische Frage eine Antwort gegeben werden kann, die das Verständnisniveau des Fragestellers erhöht und er dadurch einem bestimmten Ziel näher kommen kann [ZEH 98].

2.3. Was ist Informationsvisualisierung?

Bei der Informationsvisualisierung handelt es sich um ein noch verhältnismäßig junges Forschungsgebiet, das sich mit rechnergestützten Methoden befasst, die die grafische Repräsentation von Daten zum Ziel haben. Als interdisziplinäres Forschungsgebiet bedient sich die Informationsvisualisierung verschiedener Erkenntnisse und Methoden. Dazu zählen neben der naheliegenden Informatik auch Mathematik, Statistik, Datamining und Kognitionswissenschaft.

Insbesondere bei großen Datenmengen helfen die Darstellungsmethoden Menschen bei der Auswertung und bei der Gewinnung von essentiellen Informationen aus diesen Daten. Dabei wird großer Wert auf die Effektivität der gewählten Darstellungen gelegt. Das bedeutet, dass sich eine Person bei der Betrachtung in möglichst kurzer Zeit einen Eindruck über die enthaltenen Informationen verschaffen kann.

Nicht selten gibt es Fälle, bei denen Menschen nicht ohne Informationsvisualisierung auskommen. Dies ist zum Beispiel bei Problemen der Fall, die nicht ausreichend präzise definiert sind, damit sie ein Computer mit Hilfe von Algorithmen lösen kann. Ein erfahrener Arzt kann beispielsweise aus einem Elektrokardiogramm (EKG) augenblicklich mehr wichtige Informationen gewinnen als ein Computerprogramm, das sich womöglich noch in der Entwicklung befindet. Die Gesundheit von Menschen steht auf dem Spiel und man

will sie nicht ausschließlich einer Maschine überlassen. In solchen Fällen sind Beurteilungen der konkreten Problemstellungen durch Menschen unerlässlich. Andererseits kann auf Informationsvisualisierung auch verzichtet werden, wenn informationstechnische Probleme vollständig von einem Algorithmus gelöst werden können.

2.4. Definitionen

In der Disziplin der Informationsvisualisierung gibt es Wissenschaftler, die ihre eigenen Vorstellungen von ihrer Definition vertreten. *Stuart K. Card* schrieb zum Beispiel in seinem im Jahr 1999 erschienenen Buch *Readings in Information Visualization: Using Vision to Think* [CARD+ 99, S. 6] folgende Definition:

> "The use of computer-supported, interactive, visual representations of data to amplify cognition."

Neun Jahre später lieferte Card weitere Schlüsselaussagen zum Thema Informationsvisualisierung:[1]

> "Information visualization is a set of technologies that use visual computing to amplify human cognition with abstract information."
>
> "Information visualization promises to help us speed our understanding and action in a world of increasing information volumes."
>
> "The purpose of information visualization is to amplify cognitive performance, not just to create interesting pictures. Information visualizations should do for the mind what automobiles do for the feet."

In dem Paper *Dynamic and Interactive Dimensional Anchors for Spring-Based Visualizations* [GEE+ 05, S. 1] von *Alexander G. Gee et al.* ist folgende Aussage zur Informationsvisualisierung zu finden:

> "Information visualizations attempt to efficiently map data variables onto visual dimensions in order to create graphic representations."

Daniel A. Keim et al. schrieben Folgendes in ihrem Paper *Challenges in Visual Data Analysis* [KEIM+ 06]. Sie bringen interaktive Bedienoberflächen für die Informationsvisualisierung ins Spiel:

> "InfoVis is the communication of abstract data through the use of interactive visual interfaces."

Die Interaktion zwischen dem Computer und dem Anwender steht auch für Helen C. Purchase et al. im Vordergrund. In ihrer Veröffentlichung mit dem Namen *Theoretical Foundations of Information Visualization* [PURC+ 08, S. 58] ist Folgendes diesbezüglich zu lesen:

> "Information visualization utilizes computer graphics and interaction to assist humans in solving problems."

Für Card ist ein wesentlicher Bestandteil der Informationsvisualisierung, dass sie die Wahrnehmung und die Kognition des Menschen deutlich verbessert. Dass die Informationsvisualisierung dabei kein reiner Selbstzweck ist, wird in seinem letzten Zitat klar. Nüchterner fällt das Zitat von Gee aus. Das Ziel, dem Menschen beim Verstehen von Sachverhalten zu unterstützen, lässt sich dort nicht herauslesen. Er spricht nur das zugrundeliegende Vorgehen dieser Disziplin an. Neben Card bringen die beiden zuletzt zitierten Forscher die Interaktivität ins Spiel. Damit Menschen mit Hilfe der Informationsvisualisierung Probleme lösen können, müssen sie in den Prozess eingreifen können.

2.5. Ziele und Herausforderungen

Zusammenfassend lässt sich über die Ziele der Informationsvisualisierung Folgendes aussagen: Sie soll vorhandene Daten so sehr komprimieren, dass am Ende eine für den Menschen interpretierbare visuelle Quintessenz übrig bleibt. Unüberschaubare Datenmengen sollen so verständlich gemacht werden. Die Unterschiede

[1] Quelle: `http://www.infovis-wiki.net/index.php/Information_Visualization`

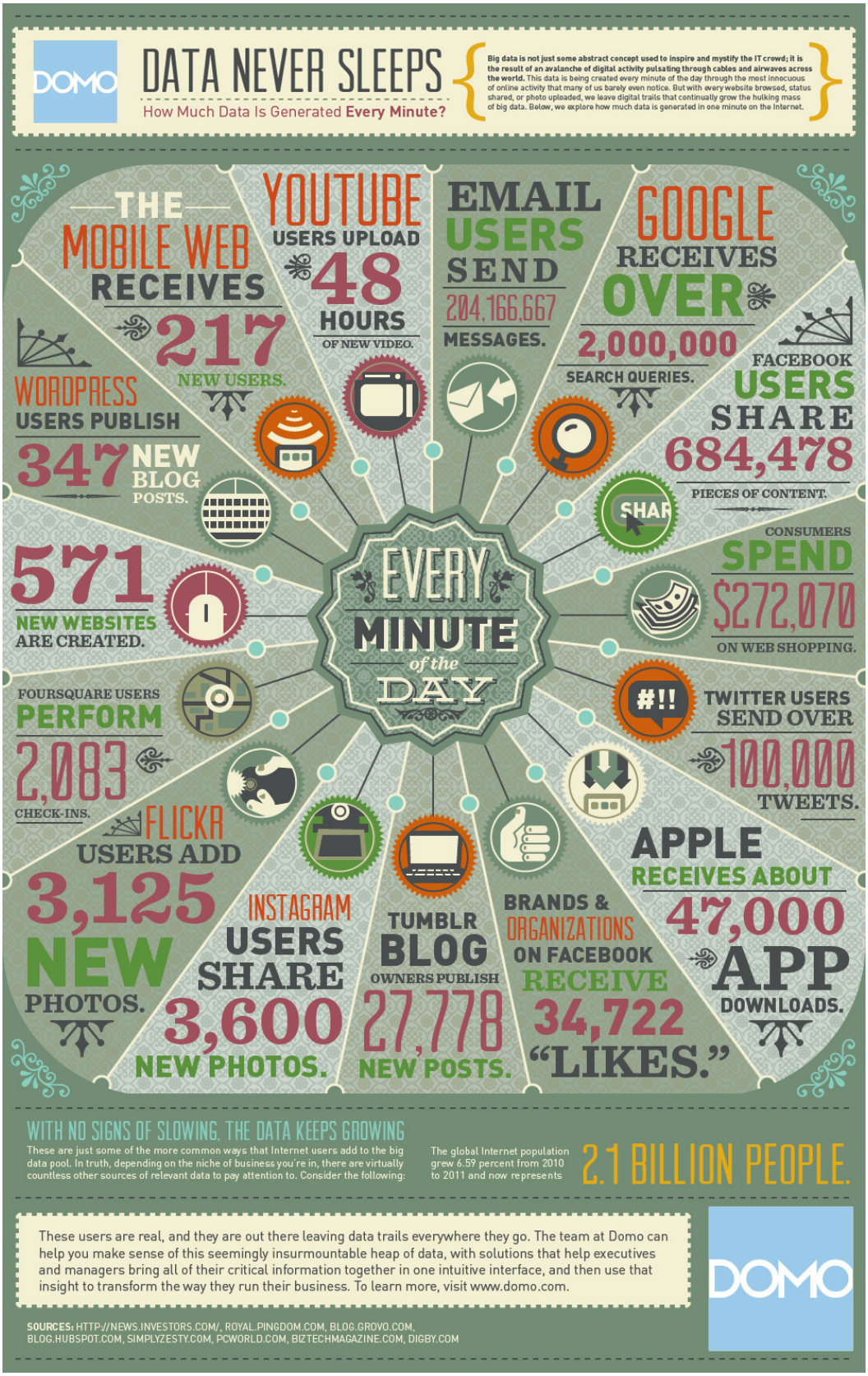


Abbildung 2.1.: Infografik *Data never sleeps* aus dem Jahr 2012. Sie veranschaulicht, wie viele Daten von Nutzern verschiedener Online-Dienste pro Minute generiert wurden (Bildquelle: `http://www.domo.com/blog/2012/06/how-much-data-is-created-every-minute/`).

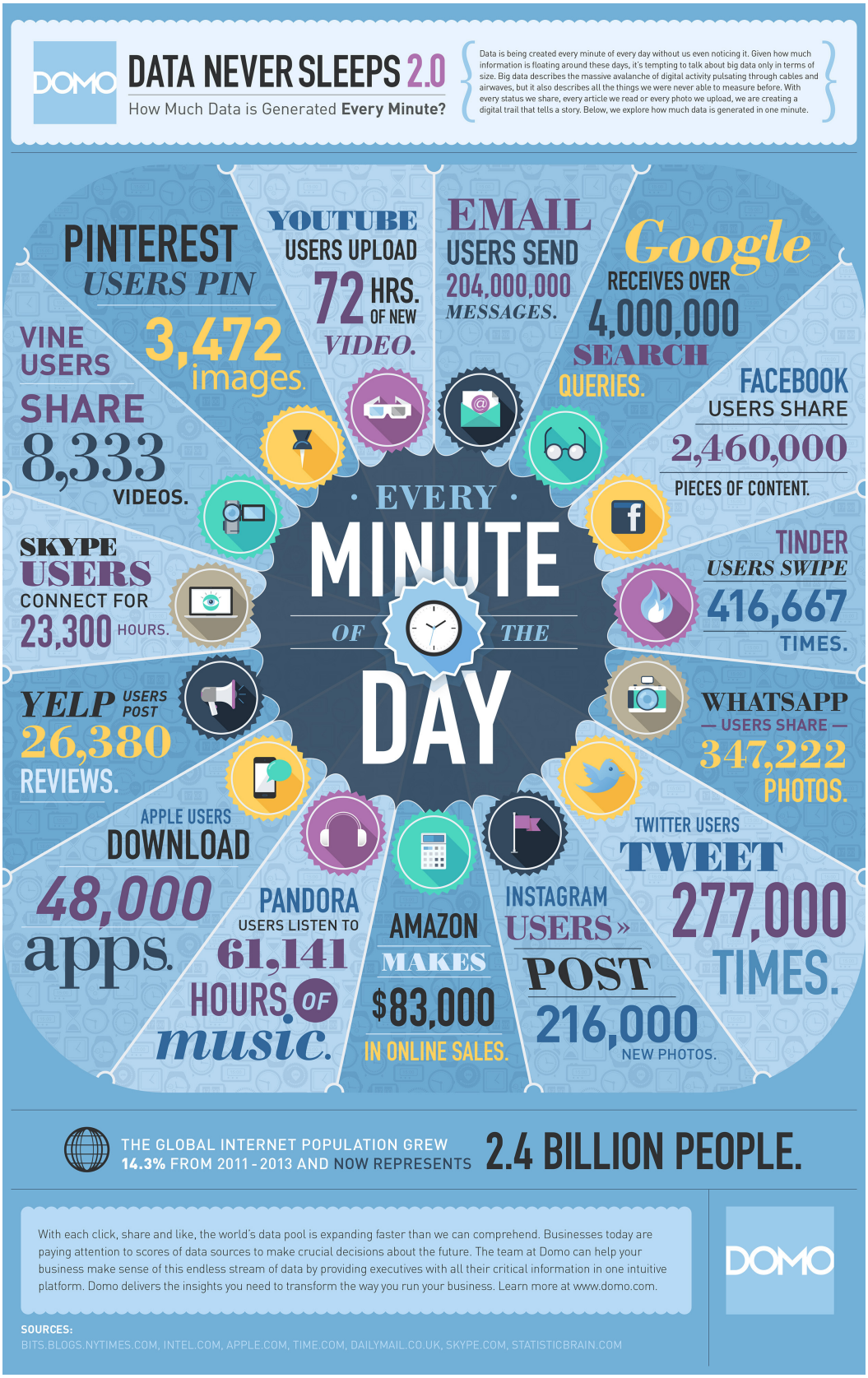


Abbildung 2.2.: Infografik *Data never sleeps 2.0* aus dem Jahr 2014. Im Vergleich zu 2012 hat sich das Datenaufkommen sichtbar erhöht (Bildquelle: `http://www.domo.com/blog/2014/04/data-never-sleeps-2-0/`).

in verschiedenen Datenmengen sollen durch das einfache Vergleichen ihrer gegenübergestellten visuellen Repräsentationen gelingen. Um dem Anwender von Informationsvisualisierungssystemen unter die Arme zu greifen, soll er zudem die Möglichkeit haben, Informationen aus unterschiedlichen Betrachtungswinkeln ansehen zu können. Ebenso soll er in der Lage sein, zwischen verschiedenen Detailgraden zu wählen, um uninteressante Informationen auszublenden. Das hilft ihm insgesamt dabei, die optimale Visualisierung zur Beantwortung einer konkreten Problemstellung zu finden.

Der Erfüllung dieser Ziele stellen sich diverse Probleme in den Weg. Das fängt bereits bei der Dimensionalität der Ausgangsdaten an. Ein- und zweidimensionale Daten lassen sich auf einer zweidimensionalen Bildschirmoberfläche oder auf Papier noch hervorragend darstellen. Hier kommen zum Beispiel sog. *Dotplots* und *Streudiagramme* (Scatterplots) zum Einsatz. Bei dreidimensionalen Datensätzen stößt die Technik bereits an ihre Grenzen und Betrachter sind auf eine zweidimensionale Projektion angewiesen (sofern sie keinen 3D-fähigen Monitor oder Projektor verwenden). Höherdimensionale Datensätze können ebenfalls als *Streudiagramm-Matrizen* dargestellt werden. Hier muss der Betrachter Projektionen aus verschiedenen Blickwinkelkombinationen ansehen, um die ganzheitliche Aussage der Visualisierung zu verstehen. Dies erfordert Übung und viel Vorstellungskraft. In Abbildung 2.3 ist eine beispielhafte Scatterplot-Matrix zu sehen, die einen Demo-Datensatz der für statistische Zwecke ausgelegten Programmiersprache *R* darstellt.

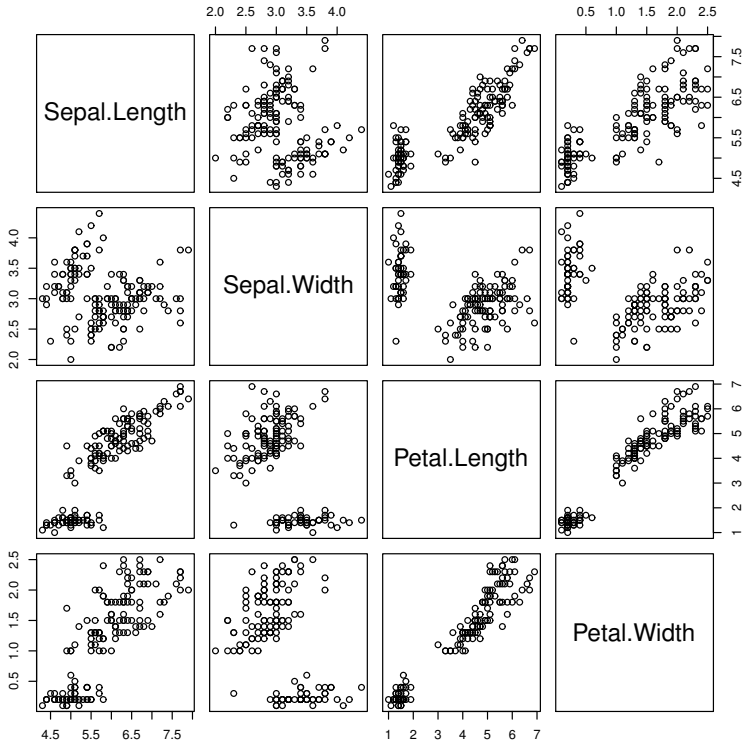


Abbildung 2.3.: Eine Scatterplot-Matrix basierend auf dem R-Demo-Datensatz `iris`. Diese Form der Visualisierung beinhaltet redundante Informationen, da die Grafik entlang der Diagonalen gespiegelt ist.

Bei noch höher dimensionierten Daten müssen geeignete Kodierungen gewählt werden, die die Daten aus den zusätzlichen Dimensionen auf das zweidimensionale Ausgabemedium abbilden können. Mit steigender Dimensionalität erhöht sich potenziell die Unübersichtlichkeit einer Visualisierung. Geeignete Filtermechanismen müssen deswegen gesucht und implementiert werden. Trotz allem müssen Entwickler darauf achten, dass Anwender das Informationsvisualisierungssystem als bedienfreundlich empfinden und ihre konkreten Probleme damit effizient lösen können.

2.6. Geschichte der Informationsvisualisierung

Die Visualisierung von Daten reicht weit in der Menschheitsgeschichte zurück. Bereits vor rund 30.000 Jahren berichteten Höhlenmalereien von erfolgreichen Jagden. Die vermutlich erste grafische Darstellung, die auf statistischen Daten beruht, stammte jedoch erst aus dem 17. Jahrhundert vom flämischen Astronomen *Michael Florent van Langren* (siehe Abbildung 2.4). Die Visualisierung einer größeren Menge abstrakter Datensätze gelang dem französischen Bauingenieur *Charles Minard* 1869 in seiner *Carte Figurative*, die Napoleons Russlandfeldzug Anfang des 19. Jahrhunderts darstellt (Abbildung 2.5). Aus ihr lassen sich vielfältige Variablen ablesen wie zum Beispiel Zeit, Anzahl der Soldaten, Ortskoordinaten, Bewegungsrichtung und Temperatur.

Mitte des 19. Jahrhunderts wurden neue Darstellungsformen entwickelt, um insbesondere Statistiken darzustellen. Thematische Kartographien wurden ebenfalls populär. In der ersten Hälfte des 20. Jahrhunderts wurden weniger Innovationen in der Datenvisualisierung verzeichnet. Eine Renaissance erlebte diese Disziplin durch die Beiträge von *John W. Tukey* und *Jacques Bertin*. Tukey entwickelte zahlreiche neue, einfache und effektive Darstellungsmethoden. Bertin entwickelte Techniken, um visuelle Elemente anhand der zugrundeliegenden Daten anzuordnen.

Computer eröffneten schließlich vollkommen neue Möglichkeiten, um komplexe Datenmengen zu visualisieren. Nicht nur das wachsende Rechen- und Speichervermögen begünstigten ihren Einsatz in dieser Disziplin, sondern auch neu aufgekommene Bedienkonzepte wie Grafiktabletts, hochauflösende grafische Oberflächen und die Computermaus. Viele neue Methoden der Visualisierung entstanden, die auf herkömmlichem Weg mit Papier und Stift nicht realisierbar waren. Auf der Website `visualcomplexity.com` können zahlreiche moderne Visualisierungsformen aus unterschiedlichen Anwendungsfeldern betrachtet werden, um sich eine Vorstellung von der Vielfalt zu verschaffen. Beispiele daraus sind in Abbildung 2.6 zu sehen.

Dies war nur ein kurzer Abriss über die Geschichte der Informationsvisualisierung. Wer sich mit ihren Meilensteinen näher befassen will, kann dies anhand eines empfehlenswerten Papers von *Michael Friendly* von der *York University* in Toronto tun [FRIE 05]. Dort berichtet er vom sog. *Milestones Project*, das sich der Sammlung, Dokumentation, Illustration und Interpretation der historischen Entwicklung dieses Forschungsfelds widmet.

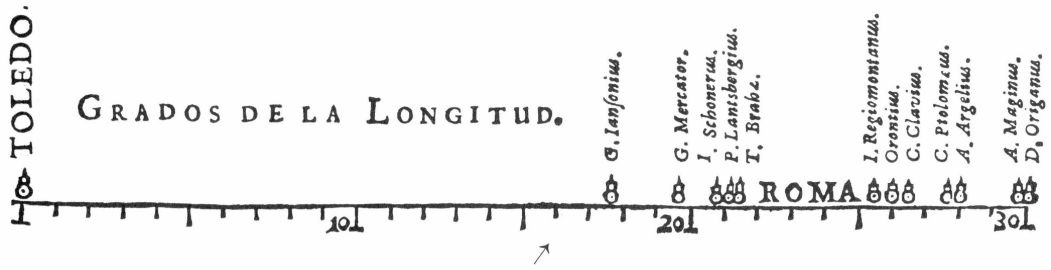


Abbildung 2.4.: Das Schaubild van Langrens zeigt von unterschiedlichen Astronomen geschätzte Entfernungen zwischen Toledo und Rom in Längengraden. Der korrekte Wert beträgt $16°30'$.

2.7. Designgrundregeln

Der US-amerikanische Informationswissenschaftler und Grafikdesigner *Edward Tufte* hat sich bereits sehr früh mit der visuellen Vermittlung von Informationen befasst. Seine Werke gelten als Standardlektüre im Umfeld der Informationsvisualisierung. In seinem Anfang der 1980er Jahre veröffentlichten Buch *The Visual Display of Quantitative Information*, das im Jahr 2001 neu aufgelegt wurde [TUFT 01], stellt er verschiedene Design-Grundregeln für die visuelle Darstellung von Informationen vor. Er gilt als Verfechter der Vermeidung von

[2]Bildquelle: `http://en.wikipedia.org/wiki/Visualization_(computer_graphics)#mediaviewer/File:Minard.png`
[3]Bildquelle: `http://www.visualcomplexity.com`

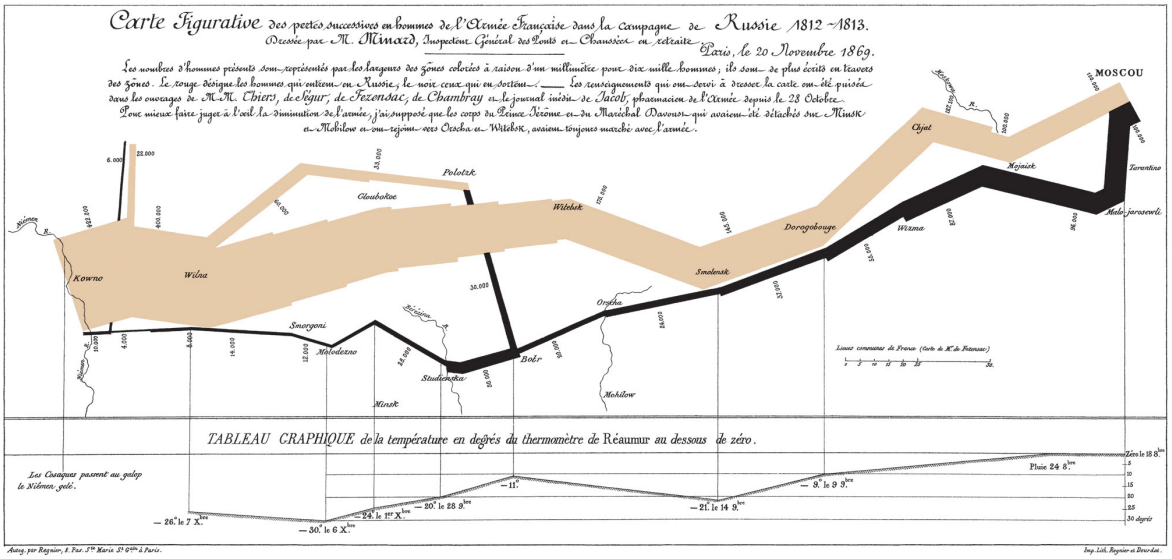


Abbildung 2.5.: Charles Minards Visualisierung des Napoleon-Feldzugs nach Russland von 1812 bis 1813 nutzt zur Positionskennzeichnung eine Landkarte als Hintergrund. Die beige Spur von Westen nach Osten markiert die ursprüngliche Marschroute der Armee. Die Spurbreite kennzeichnet die Truppenstärke. Die Dezimierung der Armee auf dem beschwerlichen Rückweg ist deutlich erkennbar.[2]

dekorativen Elementen in Darstellungen, die keine zusätzliche Information liefern. In diesem Zusammenhang prägte er Begriffe wie *Chartjunk* und *Data-Ink*. Auf einige wichtige dieser Design-Grundregeln wird in diesem Abschnitt eingegangen.

2.7.1. Data-Ink

Der französische Schriftsteller *Antoine de Saint-Exupéry* schrieb in seinem Sammelband *Terre des hommes* [EXU 99] Folgendes:

> "Il semble que la perfection soit atteinte non quand il n'y a plus rien à ajouter, mais quand il n'y a plus à retrancher."

Frei ins Deutsche übersetzt bedeutet dies:

> "Perfektion ist nicht dann erreicht, wenn es nichts mehr hinzuzufügen gibt, sondern wenn man nichts mehr weglassen kann."

Edward Tufte stützt sich scheinbar auf diese Weisheit bei der Definition des Begriffs *Data-Ink*. Die sprichwörtliche Tinte auf dem Papier soll hauptsächlich dazu dienen, Informationen darzustellen und nichts anderes. Dies lässt sich auch auf die Anzahl der Pixel einer Bildschirmdarstellung übertragen. Als *Data-Ink* bezeichnet Tufte den unauslöschbaren Kern einer Grafik, der allein von der darzustellenden Information vorgegeben wird. In diesem Zusammenhang erwähnt er eine Kennzahl mit dem Namen *Data-ink ratio*, die er in seinem Buch *The Visual Display of Quantitative Information* [TUFT 01] wie folgt berechnet:

$$\textit{Data-ink ratio} = \frac{\textit{data-ink}}{\textit{total ink used to print the graphic}} \leq 1.0 \tag{2.1}$$

Mit *Data-ink ratio* beschreibt Tufte das Verhältnis zwischen der Menge an Tinte, die zur redundanzfreien Darstellung der Information genutzt wird (*data-ink*), und der Menge an Tinte, die die gesamte Grafik beansprucht (*total ink used to print the graphic*). Im Idealfall beträgt dieses Verhältnis 1.0. Je kleiner der Wert ist, desto mehr redundante Elemente befinden sich in der jeweiligen Abbildung. Diese Elemente tragen nichts zu

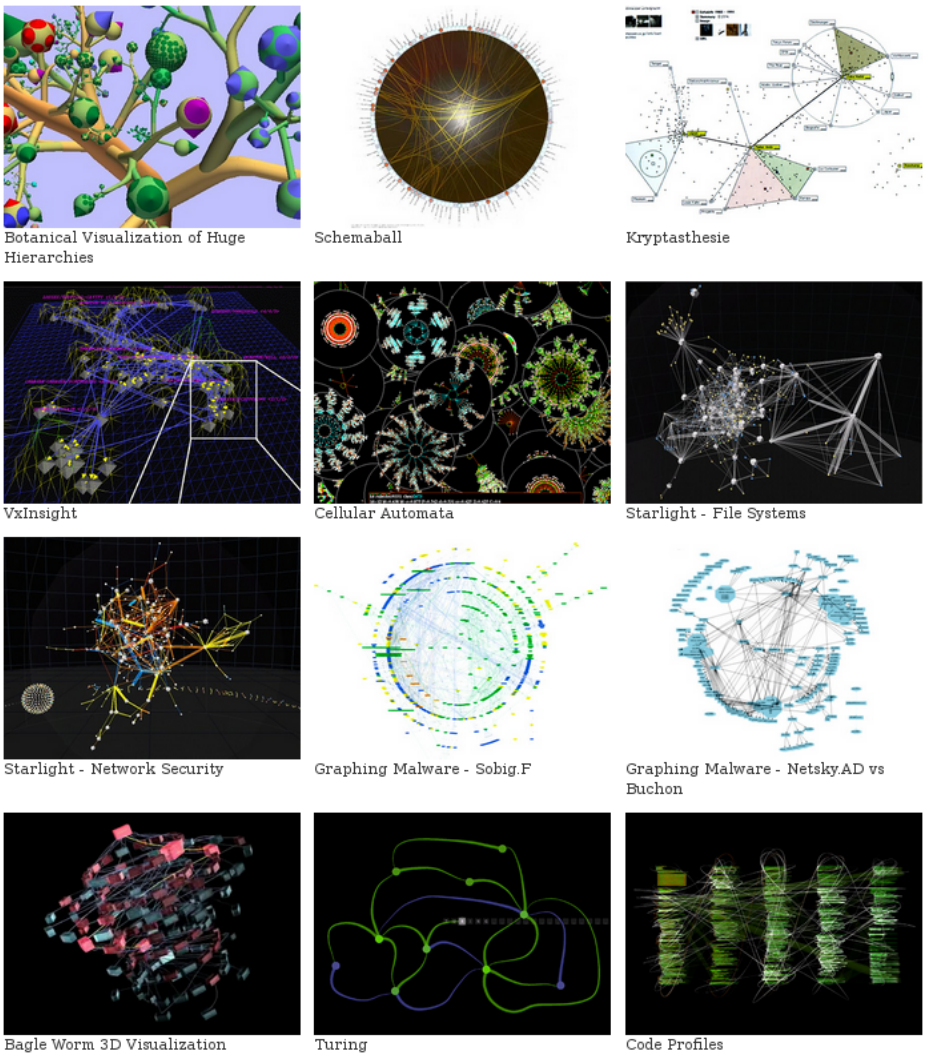


Abbildung 2.6.: Auf der Website `visualcomplexity.com` sind zahlreiche moderne computergenerierte Visualisierungen ausgestellt. Dieser Screenshot zeigt eine Auswahl von Darstellungen aus der Kategorie “Computer Systems”.[3]

den darzustellenden Informationen bei und können deshalb ohne jedweden Informationsverlust aus der Grafik entfernt werden.

Die Motivation für diese Design-Grundregel ist nicht in erster Linie, Geld für teure Druckertinte zu sparen, sondern vielmehr die Übersichtlichkeit einer Grafik zu optimieren. Wie bereits erwähnt wurde, lässt sich diese Design-Grundregel, bei der immer von Tinte die Rede ist, auch auf die Bildschirmdarstellung übertragen. Neben der verbesserten Übersichtlichkeit sind dort auch (messbare) Performance-Vorteile denkbar. Eine dynamisch erzeugte Visualisierung ist im Allgemeinen schneller berechnet, wenn sie weniger Elemente enthält.

2.7.2. Chartjunk

Populärwissenschaftliche Grafiken sowie Schaubilder in den Massenmedien und der Werbung sind oft mit dekorativen Elementen ausgestattet. Dies hat unterschiedliche Gründe: Dekorationen sollen die Grafik lebendiger aussehen lassen und dadurch die Aufmerksamkeit eines flüchtigen Betrachters darauf lenken und möglichst lange fesseln. Manchmal werden sie auch eingesetzt, um beispielsweise den populärwissenschaftlichen Charakter zu unterstreichen oder der Grafik-Designer schöpft einfach aus dem Vollen und will eine unverkennbare

Handschrift hinterlassen. Laut Tufte handelt es sich unabhängig von den genannten Gründen nichtsdestotrotz um *Non-Data Ink*, also überflüssige Tinte, die nicht zur Darstellung der eigentlichen Informationen zum Einsatz kommt. In diesem Zusammenhang spricht Tufte auch etwas abschätzig von sog. *Chartjunk*.

Aus seiner Abneigung gegenüber Chartjunk macht Tufte keinen Hehl, da seiner Meinung nach hübsche Dekorationen scheinbar billiger in der Herstellung sind als die Gewinnung der eigentlichen Informationen. Es gibt aber auch Chartjunk, der keinen künstlerischen Ursprung hat, sondern von vielen Leuten schlichtweg gewohnheitsmäßig in Darstellungen eingefügt wird. Dazu zählen zum Beispiel überflüssige Hilfslinien in Diagrammen oder der übermäßige Gebrauch von Füllfarben oder Hintergrundtexturen. In Abbildung 2.7 ist ein extremes Beispiel zu sehen. Die Übersichtlichkeit leidet hier unter den genannten Design-Fehlern. Streng genommen sind die beiden im Kapitel 2.1 gezeigten Schaubilder über das Datenaufkommen im Internet auch mit reichlich Chartjunk überladen (siehe Abbildung 2.1 und 2.2).

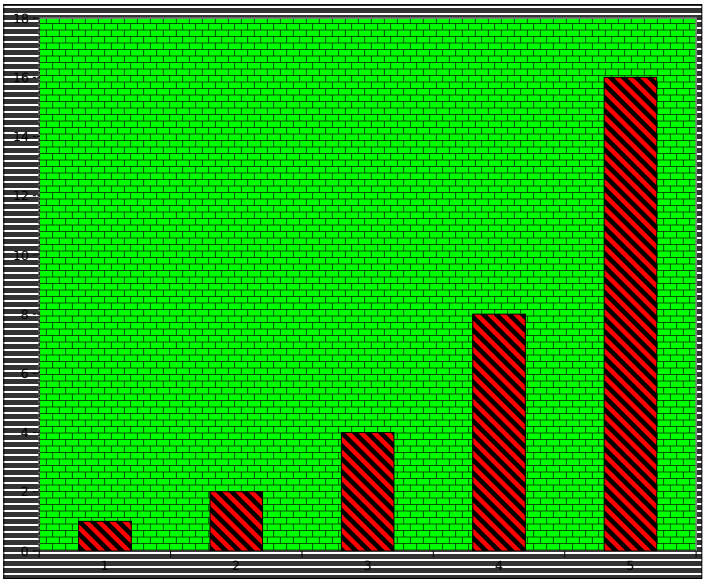


Abbildung 2.7.: Extrembeispiel für ein Diagramm mit Chartjunk. Die grellen Texturen erfüllen keinen Zweck und lenken den Betrachter vom Wesentlichen ab. Die Achsenbeschriftungen sind aufgrund des fehlenden Kontrasts zum Hintergrund kaum zu entziffern.[4]

2.7.3. Lie Factor

Im Kapitel *Graphical Integrity* seines Buchs spricht Tufte die *Verzerrung* einer Grafik an. Unter Verzerrung versteht er aber nicht ein verändertes Seitenverhältnis, das zu Eierköpfen führt, sondern er bezieht sich auf die Repräsentation der Informationen. Eine Grafik ist genau dann nicht verzerrt, wenn die visuelle Repräsentation der Daten mit den zugrundeliegenden Zahlen übereinstimmt. Andernfalls steigt die Wahrscheinlichkeit, dass Betrachter die Grafik falsch interpretieren und aus der Grafik falsche Werte herauslesen. Um die sog. *grafische Integrität* zu optimieren, empfiehlt Tufte folgende Richtlinien:

1. Die direkt in einer Grafik ermittelbaren Maße sollen direkt proportional zu den repräsentierten Werten sein.

2. Saubere, detaillierte und genaue Beschriftungen sollen verwendet werden, um Mehrdeutigkeit auszuschließen. Wichtige Informationen sollen in der Grafik beschriftet werden. Erklärungen sollen sich direkt in der Abbildung befinden.

[4] Bildquelle: `http://upload.wikimedia.org/wikipedia/commons/c/c9/Chartjunk-example.svg`

Im Zusammenhang mit der ersten Richtlinie bringt Tufte eine Kennzahl mit dem Titel *Lie Factor* ins Gespräch. Sie berechnet sich folgendermaßen:

$$\textit{Lie Factor} = \frac{\textit{size of effect shown in graphic}}{\textit{size of effect in data}} \tag{2.2}$$

Der Lie Factor ist ein Indikator für die zuvor genannte Verzerrung einer Grafik. Im Idealfall beträgt er 1.0. Dann gibt die Grafik die zugrundeliegenden Zahlen exakt wieder. Werte kleiner als 1.0 deuten auf eine Abschwächung der darzustellenden Werte hin, größere auf eine Übertreibung. Tufte machte die Beobachtung, dass in der Praxis Grafiken oft zugunsten der Übertreibung verzerrt werden. Lie-Faktoren zwischen 2.0 und 5.0 sind nicht unüblich.

Ein extremes Beispiel aus Tuftes Buch ist in Abbildung 2.8 zu sehen. Es handelt sich um eine Grafik, die im Jahr 1978 in der *New York Times* abgedruckt wurde. Sie stellt Verbrauchsrichtlinien der US-Regierung für Automobilhersteller dar, die beginnend ab 1978 schrittweise verschärft werden sollten. Der Kraftstoffverbrauch wird hier wie in den USA üblich indirekt durch die Reichweite in Meilen angegeben, die das Fahrzeug mit einer Gallone Benzin zurücklegen soll. Die Jahr für Jahr immer breiter werdenden Linien stehen also für eine sukzessive Verringerung des Kraftstoffverbrauchs. Vergleicht man die Reichweite der beiden Jahre 1985 und 1978, stellt man eine Vergrößerung um beinahe 53% fest:

$$\frac{27.5\frac{miles}{gallon} - 18.0\frac{miles}{gallon}}{18.0\frac{miles}{gallon}} \cdot 100 \approx 53\% \tag{2.3}$$

Der relative Längenzuwachs der Datenlinie von 1985 im Vergleich zu 1978 berechnet sich folgendermaßen anhand der Längen, die Tufte damals selbst nachgemessen hat:

$$\frac{5.3in - 0.6in}{0.6in} \cdot 100 \approx 783\% \tag{2.4}$$

Wie zu erkennen ist, fällt der Längenzuwachs der Datenlinie deutlich größer aus als das tatsächliche Wachstum der Reichweite. Die zugrundeliegenden Zahlen werden in der Grafik also stark verzerrt wiedergegeben. Der Lie Factor der Abbildung ist deutlich größer als 1.0:

$$\textit{Lie factor} = \frac{783\%}{53\%} \approx 14.8 \gg 1.0 \tag{2.5}$$

Auf den Betrachter hat ein derartig großer Lie Factor eine manipulierende Wirkung. Er nimmt auf den ersten Blick eine eklatante Steigerung in der Zeitreihe wahr, die in Wirklichkeit viel geringer ausfällt.

2.8. Referenzmodell

Um Aussagen über unterschiedliche Systeme der Informationsvisualisierung zu treffen und sie untereinander objektiv vergleichen zu können, haben Card et al. im Jahr 1999 ein Referenzmodell veröffentlicht [CARD+ 99, S. 17]. Es lässt sich wie folgt zusammenfassen: Das Ausgangsmaterial im Referenzmodell stellen Rohdaten dar, die einer Reihe von Transformationen unterzogen werden. Zuerst werden die Rohdaten durch eine *Datentransformation* in Datentabellen umgewandelt. In der Praxis bedeutet dies, dass zum Beispiel statistische Daten in einer relationalen Datenbank gespeichert werden. Nun folgt das *visuelle Mapping*, das die Datentabellen in visuelle Strukturen überführt. Letztere bestehen aus räumlichen Untergründen, Markierungen und grafischen Eigenschaften. Zuletzt erfolgt die *View-Transformation*. Dieser Prozess nutzt die visuellen Strukturen, um sogenannte *Views* daraus zu generieren. Dies geschieht durch die konkrete Spezifizierung der grafischen Parameter wie Objektposition, Größe, Farbe etc. Abbildung 2.9 ist dem Bild aus Cards Buch nachempfunden und veranschaulicht den Prozess.

Von zentraler Bedeutung im Referenzmodell nach Card et al. ist das Feedback des Anwenders. Er befindet sich nicht nur als passiver Betrachter am Ende dieses Prozesses, sondern sitzt gewissermaßen auch am Kontrollhebel. Er ist in der Lage, auf jeden der zuvor genannten Transformationsschritte Einfluss zu nehmen und

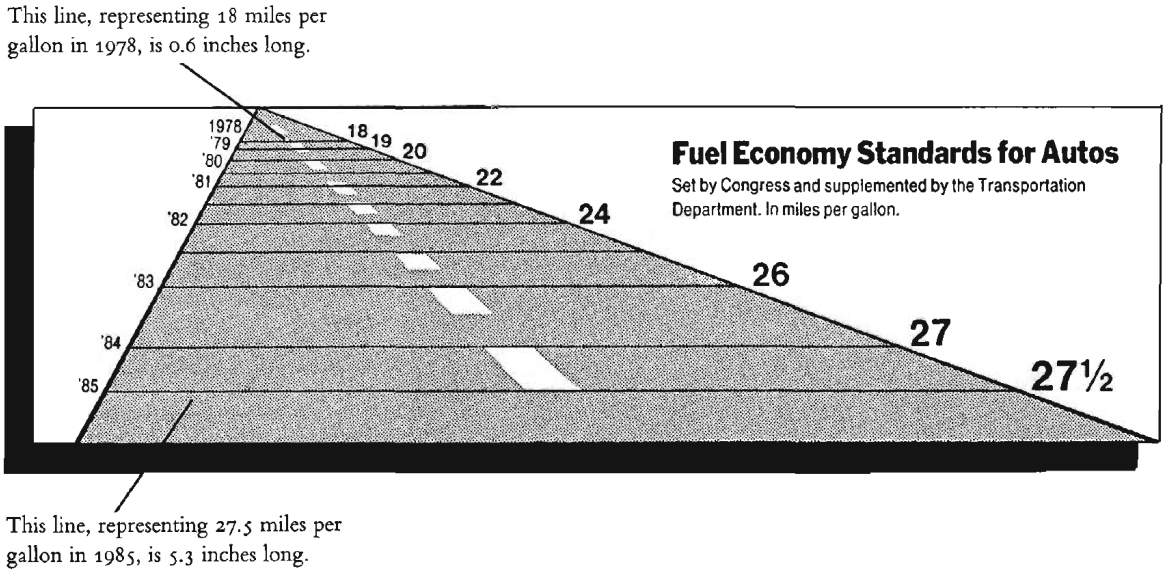


Abbildung 2.8.: Eine Zeitungsgrafik aus dem Jahr 1978 mit extremem Lie Factor. Wenn man die Verbrauchszahlen ignoriert, möchte man meinen, dass sich die Reichweite von Autos mit derselben Menge an Kraftstoff innerhalb weniger Jahre vervielfachen soll.

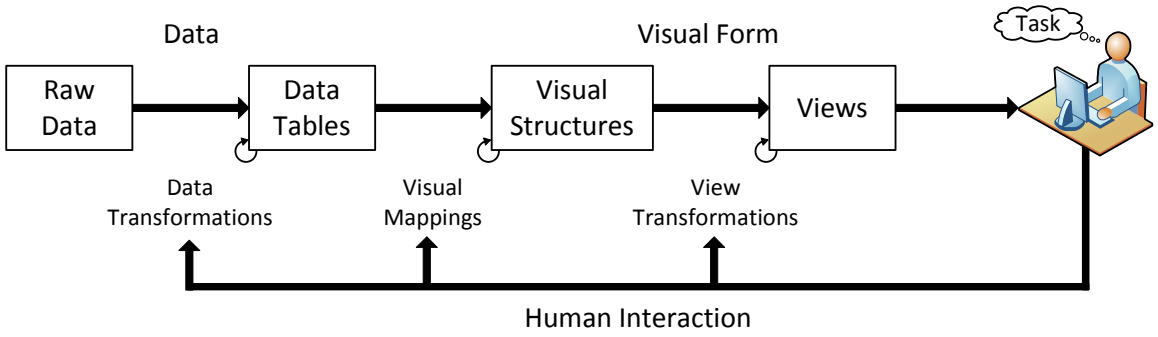


Abbildung 2.9.: Das Referenzmodell der Informationsvisualisierung nach Card et al.

das Ergebnis durch die Änderung der erforderlichen Parameter zu beeinflussen. Das beinhaltet zum Beispiel die Begrenzung der in der View angezeigten Auswahl an Daten oder die Änderung des Betrachtungswinkels.

Rohdaten könnten eigentlich auch direkt visualisiert werden. Sobald man aber mit abstrakten Daten ohne impliziten räumlichen Bezug zu tun hat, ist eine Transformation in Relationen erforderlich. Man stelle sich folgenden Fall vor: Es liegen mehrere Textdokumente vor, die inhaltlich verglichen werden sollen. Die Ähnlichkeit der Dokumente soll schließlich visuell dargestellt werden. Die Rohdaten einfach nebeneinander oder übereinander darzustellen, ist für diesen Anwendungsfall nicht zielführend. Es ist besser, die Dokumente zuerst zu bereinigen, indem unnötige Informationen herausgefiltert werden, und sie in eine abstraktere Form wie zum Beispiel normalisierte Dokumentvektoren zu überführen. Aus diesen Vektoren können schließlich Datentabellen gefüllt werden, die als Grundlage für die Visualisierung dienen. Die Idee, Dokumente derart zu vergleichen, stellten *Gerard A. Salton et al.* im Jahr 1975 in ihrem Paper *A Vector Space Model for Automatic Indexing* [SAL+ 75] vor.

2.9. Visualisierungs-Pipeline

Die *Visualisierungs-Pipeline* nach *Selan dos Santos* und *Ken Brodlie* beschreibt den schrittweisen Vorgang, visuelle Repräsentationen aus Daten zu generieren. Sie stammt aus dem Bericht *Gaining understanding of multivariate and multidimensional data through visualization* [DOS+ 04] und setzt sich aus den folgenden vier Prozessen zusammen:

Datenanalyse bereitet die Rohdaten vor. Fehlende Werte werden möglicherweise interpoliert, eventuell vorhandene Messfehler werden aussortiert, unterschiedliche Glättungsfilter werden angewendet etc. Diese Aufgabe wird hauptsächlich vom Computer erledigt und benötigt nur wenig Eingriff durch den Menschen.

Filterung Der Anwender wählt aktiv eine Teilmenge der zu visualisierenden Daten aus.

Mapping Die ausgewählten Daten werden auf geometrische Objekte wie Linien, Punkte, Rechtecke etc. abgebildet. Dabei werden ihre visuellen Attribute wie z. B. Größe, Position und Farbe in Abhängigkeit von den zu repräsentierenden Daten festgelegt. Die Aussagekraft der Visualisierung wird maßgeblich von diesem Schritt in der Pipeline bestimmt.

Rendering Der Computer berechnet aus den geometrischen Daten eine Projektion auf eine Bildfläche.

Abbildung 2.10 veranschaulicht die einzelnen Phasen der Visualisierungs-Pipeline und die daraus resultierenden Zwischenprodukte.

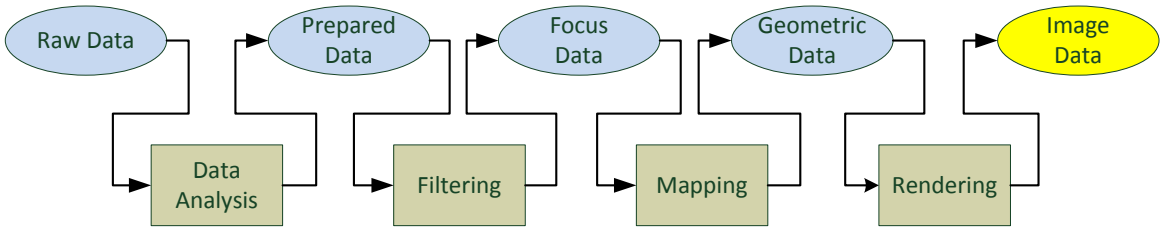


Abbildung 2.10.: Die Visualisierungs-Pipeline nach dos Santos und Brodlie.[5]

2.10. Visual Encoding

Unter *Visual Encoding* versteht man den Vorgang, multivariate Daten in sogenannte visuelle Variablen zu übertragen. Der französische Kartograph *Jacques Bertin* entwickelte die Theorie der grafischen Variablen und veröffentlichte sie 1967 in seinem Buch *Sémiologie graphique: Les diagrammes, les résaux, les cartes* [BER 13]. Das Buch gilt als erstes Standardwerk, das sich mit den Themen Visualisierung und der grafischen Theorie befasst. Bertin zeigt, dass ein sog. *Kartenzeichen* aus den folgenden Variablen zusammengesetzt werden kann: Position, Größe, Form, Helligkeit, Farbe, Ausrichtung und Textur. In Abbildung 2.11 sind beispielhafte Ausprägungen der genannten visuellen Variablen dargestellt.

Der US-amerikanische Informationsvisualisierungsexperte *Jock D. Mackinlay* griff Bertins Theorie der visuellen Variablen auf, um sie später zu erweitern. Er führte nicht nur neue Arten von Variablen ein, sondern sortierte sie zudem noch anhand ihrer Genauigkeit und in Abhängigkeit von ihrer Funktion. Abbildung 2.12 zeigt Mackinlays Liste visueller Variablen, die jeweils nach ihrer quantitativen, ordinalen und nominalen Genauigkeit sortiert sind.

Ein etwas einfacheres Modell visueller Variablen haben *William S. Cleveland* und *Robert McGill* benutzt, um ihre Genauigkeit anhand empirischer Versuche festzustellen. Sie kamen zu folgendem Ergebnis: Bei der Position eines Objekts innerhalb einer Grafik handelt es sich um die visuelle Variable mit der größtmöglichen

[5] Bildvorlage: `http://www.infovis-wiki.net/images/5/59/Dossantos04vis_pipeline.png`
[6] Bildquelle: `http://www.infovis-wiki.net/images/8/89/VisualVariables.png`
[7] Bildquelle: `http://www.infovis-wiki.net/images/0/0b/Mackinlay_PerceptualTask.jpg`

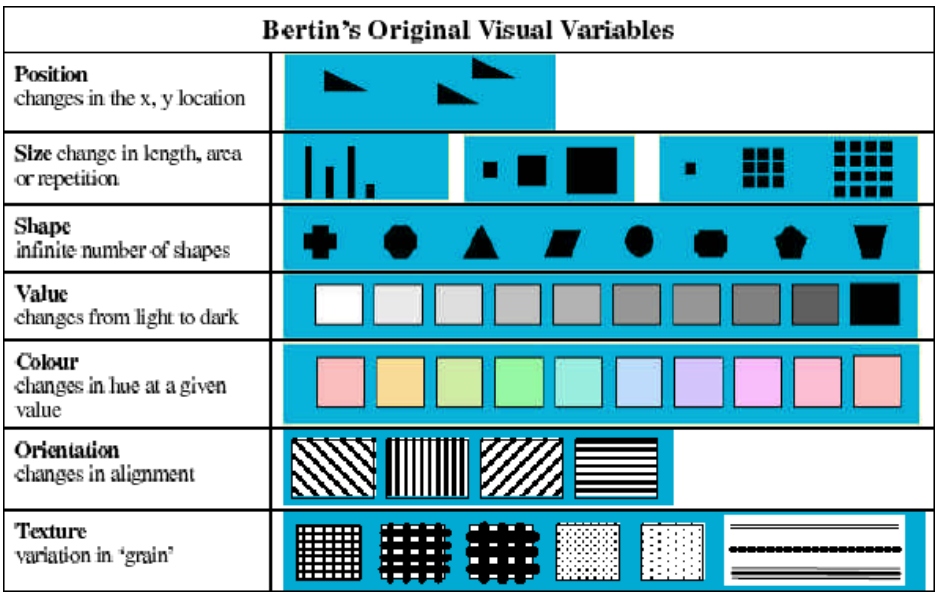


Abbildung 2.11.: Bertins visuelle Variablen sind gegliedert in die Kategorien Position, Größe, Form, Helligkeit, Farbe, Ausrichtung und Textur.[6]

Genauigkeit. Die durch die Position repräsentierte Information lässt sich vom Betrachter gut ermitteln, sogar besser als dies bei unterschiedlichen Seitenlängen von Objekten der Fall ist. Die Genauigkeit nimmt laut Cleveland und McGill bei Winkeldarstellungen und Steigungen ab. Für den Betrachter noch ungenauer zu erfassen sind Flächen, gefolgt von perspektivisch dargestellten Objektvolumina. Besonders schlecht fällt die Genauigkeit von gerasterten Flächen mit unterschiedlicher Dichte aus. Abbildung 2.13 fasst diese Erkenntnisse aus der empirischen Studie zusammen.

Wie anhand der zuvor genannten Genauigkeiten zu sehen ist, dürfen visuelle Variablen nicht beliebig eingesetzt werden, denn das Wahrnehmungsvermögen des Menschen ist ein limitierender Faktor. Dabei spielt es nicht nur eine Rolle, wie viele Ausprägungen einer visuellen Variable vom Betrachter unterschieden werden können. Seine kognitive Kapazität beschränkt auch das, woran er sich später erinnern wird. Auch technologische Hürden gilt es zu meistern: Ein breites Portfolio an visuellen Variablen ist zwar schön und gut, aber man muss auch in der Lage sein, sie auf einem Bildschirm mit beschränkter Größe und Auflösung unterzubringen. Besonders bei dynamisch generierten Grafiken mit vielen visuellen Variablen sind effiziente Algorithmen wünschenswert, die geringe Reaktionszeiten ermöglichen und eine unterbrechungsfreie Bedienung der Anwendung gewährleisten.

2.11. Präattentive Wahrnehmung

Das Wort *präattentiv* setzt sich aus den lateinischen Wörtern *prä* (vor) und *attentio* (Aufmerksamkeit) zusammen. In Zusammenhang mit dem Sehprozess versteht man darunter eine unterschwellige Wahrnehmung von Objekten, die vor ihrer Erkennung durch das menschliche Bewusstsein stattfindet. Umgangssprachlich könnte man die präattentive Wahrnehmung von Objekten so beschreiben, dass sie einem regelrecht *ins Auge stechen*. Im Alltag ist der Mensch ununterbrochen den Sinnesreizen der Augen ausgesetzt. Eine bewusste Wahrnehmung all dieser ungefilterten Reize würde ihn überfordern, deswegen dringen nur wenige relevante Reize so weit vor, dass der Mensch sie bewusst wahrnehmen kann.

Es ist eine Herausforderung, Informationen derart zu präsentieren, dass sie zwar sofort ins Auge stechen, den Betrachter aber nicht von anderen wichtigen Aspekten ablenken. Forschungen haben gezeigt, dass es verschiedene grundlegende visuelle Eigenschaften von Objekten gibt, die vom Menschen präattentiv wahrgenommen

[8]Bildquelle: `http://hcil2.cs.umd.edu/trs/99-20/image157.gif`

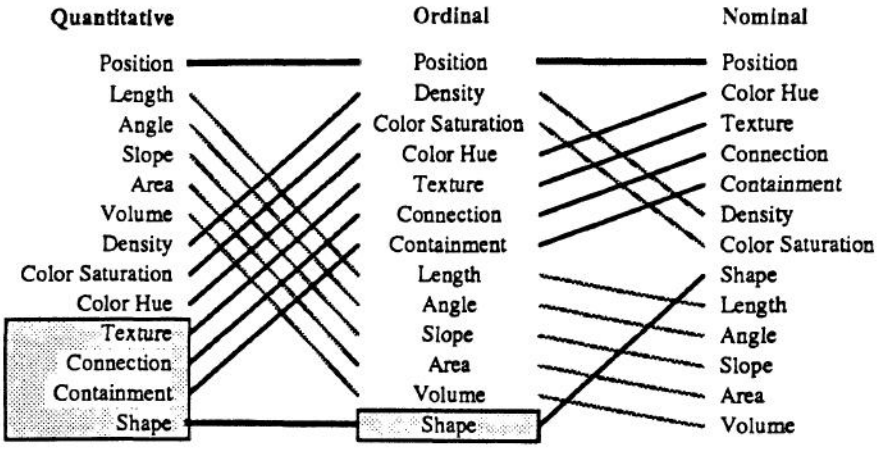


Abbildung 2.12.: Mackinlays visuelle Variablen.[7]

werden. Einige von ihnen werden im Folgenden erwähnt. Ein Ziel der Informationsvisualisierung ist es, diese unterschwellige Fähigkeit des Sehsinns zu nutzen und Visualisierungen zu schaffen, die dem Betrachter die Kernaussagen unmittelbar vermitteln.

2.11.1. Feature-Integration-Theorie

Durch ihre wissenschaftlichen Errungenschaften im Bereich der präattentiven Wahrnehmung ist die britisch-amerikanische Kognitionspsychologin *Anne Treisman* bekannt geworden. Derzeit arbeitet sie als Psychologie-Professorin an der Universität in Princeton. Während ihrer Forschungen hat sie sich zwei grundlegenden Fragestellungen gewidmet:

- Welche visuellen Eigenschaften werden präattentiv wahrgenommen?
- Wie funktioniert die präattentive Wahrnehmung?

Auf dem Weg zu ihrer *Feature-Integration-Theorie* führte Treisman empirische Experimente mit Versuchspersonen durch, die Fragen zu speziell präparierten Testbildern beantworten mussten [TRE+ 80]. Um präattentiv wahrgenommene Attribute zu bestimmen, wurden bei den Experimenten sowohl die Reaktionszeit als auch die Genauigkeit der Wahrnehmung ermittelt.

Die Bestimmung der Reaktionszeit führte sie nach folgendem einheitlichen Schema durch: Die Probanden wurden aufgefordert, inhaltliche Fragen zu einem eingeblendeten Bild so schnell wie möglich zu beantworten (sog. *target detection*). Nacheinander präsentierte Bilder enthielten dabei immer mehr ablenkende Elemente. Wenn die Reaktionszeit bei den Probanden trotz des gesteigerten Chaos in den Bildern relativ konstant und unterhalb eines bestimmten Schwellenwerts blieb, ging Treisman von einer präattentiven Wahrnehmung des untersuchten visuellen Attributs aus. Bei einigen untersuchten Attributen wandten die Probanden bewusste Suchstrategien an, um zu beantworten, ob bestimmte Objekte im Bild vorhanden sind oder nicht. Wenn sich die Suchzeit proportional zur Anzahl der Störeinflüsse erhöhte, konnte die präattentive Wahrnehmung des betrachteten Attributs ausgeschlossen werden.

Die Bestimmung der Genauigkeit fand folgendermaßen statt: Den Probanden wurden auch hier verschiedene präparierte Testbilder eingeblendet. Für den kurzen Zeitraum von maximal $250ms$ wurden einmalig eines oder mehrere Objekte zusätzlich eingeblendet, die von den Testpersonen erkannt werden mussten. Auch bei der Bestimmung der Genauigkeit wurden Schritt für Schritt mehr ablenkende Elemente hinzugefügt. Falls die Probanden korrekt antworteten, wurde das betrachtete Attribut als präattentiv wahrnehmbar angesehen.

Insgesamt trug Treisman eine ganze Reihe an sog. *Features* zusammen, die sie als präattentiv wahrnehmbar eingestuft hat. Dazu zählen unter anderem Folgende:

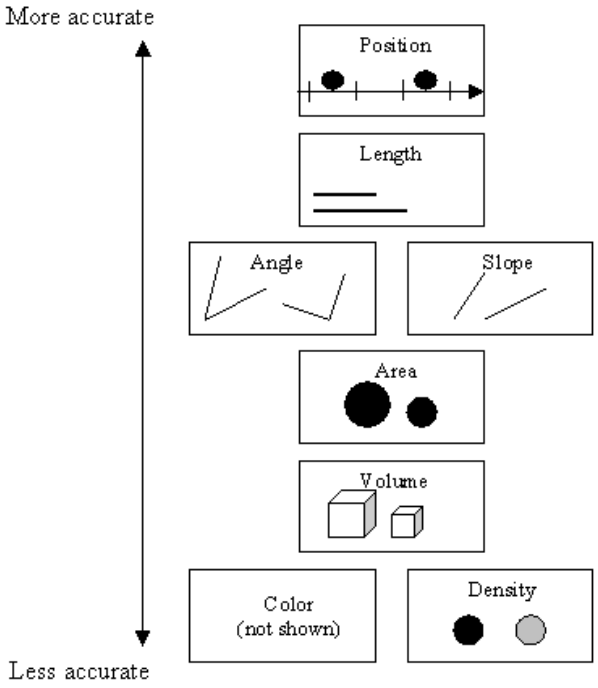


Abbildung 2.13.: Ranking der Genauigkeiten visueller Variablen nach Cleveland und McGill.[8]

- Farbe (Abbildung A.1)
- Form (Abbildung A.2)
- Länge und Breite (Abbildung A.3)
- Abgeschlossenheit und Krümmungen (Abbildung A.4)
- Überschneidungen und Abschlüsse (Abbildung A.5)
- Häufungen und Helligkeit (Abbildung A.6)
- Ausrichtung und räumliche Tiefe (Abbildung A.7)

Es existieren noch weitere präattentive visuelle Variablen wie zum Beispiel die Bewegung von Objekten und ihre Bewegungsrichtung. Blinkende oder flackernde Bildelemente werden ebenfalls augenblicklich wahrgenommen. Diese Eigenschaft machte sich auch die Werbeindustrie bereits vor Jahren bei Reklame auf fremdfinanzierten Webseiten zunutze. Die von vielen PC-Anwendern als zu penetrant wahrgenommene Werbung lenkt sie aber dermaßen vom eigentlichen Inhalt ab, dass Werbeblocker wie zum Beispiel *Adblock Plus* zum Einsatz kommen. Das Plug-in steht mittlerweile für alle gängigen Webbrowser zum Download[9] bereit. Unter den *Mozilla*[10]-Browsern *Firefox*[11] und *SeaMonkey*[12] gilt Adblock Plus interessanterweise als die am häufigsten heruntergeladene Extension mit derzeit knapp 22 Millionen Anwendern (Stand: Oktober 2014) [MOZ 14].

Einzelne Objekte grenzt das Gehirn automatisch voneinander ab, falls sie sich deutlich in ihrer Farbe unterscheiden. Das funktioniert auch dann, wenn sie unterschiedliche Formen haben. Das linke Testbild in Abbildung 2.14 kann als Beispiel herangezogen werden. Die Kreise und Quadrate in der oberen Bildhälfte sind rot gefärbt, die Objekte in der unteren Bildhälfte blau. Es gibt unter allen Objekten keine einheitliche Farbgebung, dennoch nimmt man beim Betrachten unmittelbar eine imaginäre Trennlinie wahr, die zwischen den beiden Bildhälften waagerecht verläuft. Dies geschieht präattentiv.

[9] Adblock-Plus-Website: `https://adblockplus.org/`
[10] Mozilla-Website: `https://www.mozilla.org/de/`
[11] Firefox-Website: `https://www.mozilla.org/de/firefox/new/`
[12] SeaMonkey-Website: `http://www.seamonkey-project.org/`

Anders gestaltet es sich beim rechten Bild in Abbildung 2.14. Es sticht hier keine eindeutige Trennlinie ins Auge. Die Objekte sind allerdings nach einem bestimmten Schema gruppiert. In der linken Bildhälfte sind alle Kreise rot und alle Quadrate blau. In der rechten Bildhälfte ist es genau andersherum. Ein Betrachter, der dieses Bild zum ersten Mal sieht, sucht das Bild bewusst Element für Element ab und versucht dann durch Logik eine Regel zu finden, die eine Trennlinie zwischen den Objekten rechtfertigt. Aufgrund der umgekehrten Farbgebung von Kreisen und Quadraten befindet sich diese logische Trennlinie senkrecht in der Mitte des Bilds. Sie zu finden, ist mit deutlich höherem Zeitaufwand verbunden als beim linken Bild.

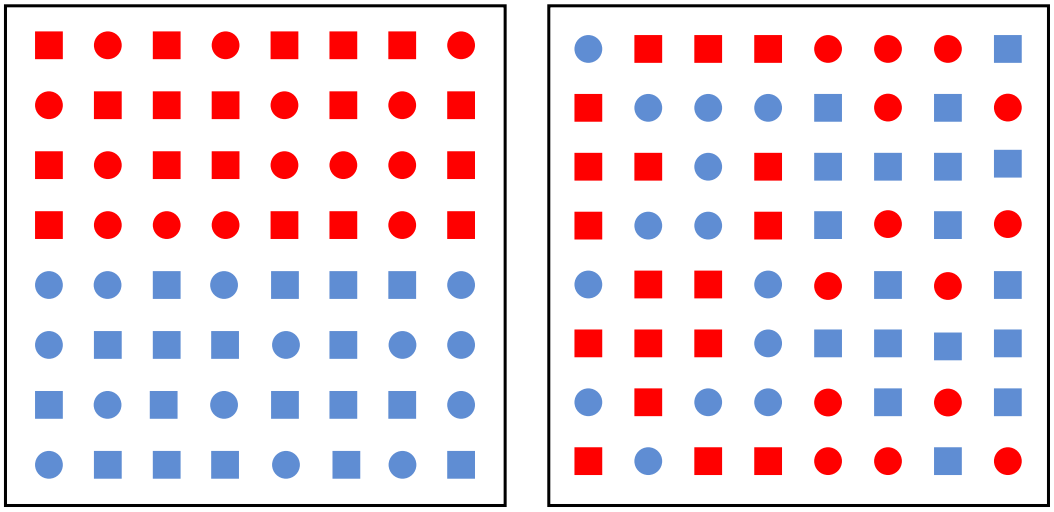


Abbildung 2.14.: Objektgruppierung präattentiv (links) und durch Logik (rechts).

Die zuvor genannten Features wurden bisher einzeln betrachtet. Die Kombination zweier visueller Attribute wurde von Treisman aber auch untersucht. Es stellte sich heraus, dass eine solche Kombination in der Regel nicht präattentiv wahrgenommen wird. Als Beispiel sei an dieser Stelle Abbildung 2.15 genannt. Ist in einem der Bilder ein blauer Kreis abgebildet? Diese Frage bezieht sich auf die beiden visuellen Attribute Farbe und Form. Um die Frage zu beantworten muss der Betrachter so viele Objekte sequenziell betrachten, bis er zu einem eindeutigen Ergebnis kommt. Das setzt ein bewusstes logischen Vorgehen voraus und erfordert etwas Zeit.

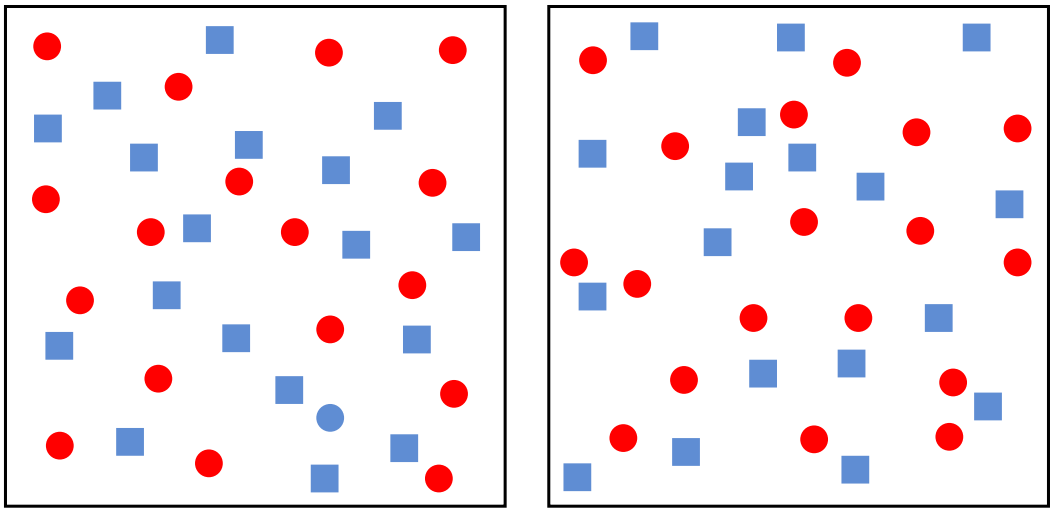


Abbildung 2.15.: Ist ein blauer Kreis im Bild enthalten?

Bei Treismans empirischen Versuchen stellte sich heraus, dass einige Features ein asymmetrisches Verhalten an den Tag legen. Abbildung 2.16 beinhaltet dazu ein Beispiel: Eine schräger Strich fällt in einer Menge von

vertikalen Strichen sofort auf. Ein vergleichbares Bild wurde bereits links in Abbildung A.7 gezeigt. Man könnte es als eine Art Schwarm interpretieren, der sich ein eine Richtung bewegt und mittendrin befindet sich ein Wesen, das aus der Reihe tanzt. Dieser Ausreißer wird präattentiv wahrgenommen. Im Gegensatz dazu dauert es deutlich länger, einen vertikalen Strich unter vielen schrägen Strichen zu finden. Hier muss der Betrachter bewusst danach suchen.

Beim Anfertigen des zuletzt genannten Testbilds stellte sich heraus, dass der vertikale Strich auch beinahe augenblicklich zu erkennen war. Dies ist allerdings der Tatsache geschuldet, dass der Betrachter genau weiß, wo er suchen muss, wenn er das Bild zuvor selbst gezeichnet hat. Um den Effekt der asymmetrischen Wahrnehmung präattentiver Attribute deutlich zu spüren, sollte man sich deswegen Abbildungen im Internet suchen, die nicht von einem selbst stammen. Dies gilt auch für andere Testbilder.

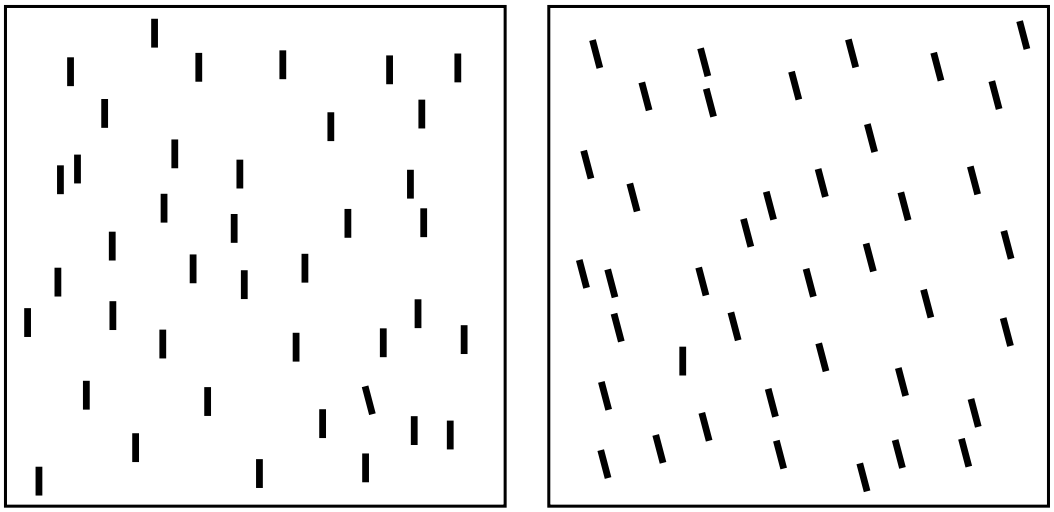


Abbildung 2.16.: Ein Beispiel für asymmetrische Features.

2.11.2. Gestalt-Theorie

Ende des 19. Jahrhunderts kam der Philosoph *Christian Freiherr von Ehrenfels* zu der Erkenntnis, dass die menschliche Wahrnehmung Qualitäten enthält, die sich nicht aufgrund der Anordnung einfacher Sinnesqualitäten ergeben. Um es mit den Worten von *Aristoteles* auszudrücken:

> "Das Ganze ist mehr als die Summe aller Teile [SCHI 91]."

Ehrenfels definierte daraufhin den Begriff *Gestalt* als etwas Ganzes, das die Eigenschaften *Übersummativität* und *Transponierbarkeit* besitzt. Unter Übersummativität bzw. *Emergenz* versteht man in diesem Kontext laut Duden "das Auftreten neuer, nicht voraussagbarer Qualitäten beim Zusammenwirken mehrerer Faktoren"[13]. Transponierbarkeit bedeutet in diesem Zusammenhang, dass eine Reizkonstellation in einen anderen Bereich übertragbar ist.

Ehrenfels bedient sich der Musik, um dies zu verdeutlichen: Eine Melodie besteht zwar aus einzelnen Tönen, sei aber wesentlich mehr als die Summe aller Töne. Aus der gleichen (mengentheoretischen) Menge an Tönen könnte man auch eine vollkommen andere Melodie zusammensetzen. Andererseits bleibe die Melodie auch dann gleich, wenn man sie in eine andere Tonart transponiert, obwohl sie dann allerdings aus anderen Tönen besteht. Es gibt also ein sog. *Wahrnehmungsganzes* neben den einzelnen Sinneseindrücken. Ehrenfels sprach von sog. *Gestaltqualitäten* [WEIN+ 78].

Anfang des 20. Jahrhunderts entstand basierend auf den Erkenntnissen von Ehrenfels die *Berliner Schule der Gestaltpsychologie*, die sich selbst auch *Gestalttheorie* nannte. Umfangreiche empirische Forschungen

[13] Quelle: `http://www.duden.de/rechtschreibung/Emergenz`

wurden auf dem Gebiet der Wahrnehmung durchgeführt. Daraus resultierten einige Forschungsergebnisse, die als *Gestaltgesetze* bekannt wurden[14]

Gesetz der Ähnlichkeit Objekte, die zueinander ähnlich aussehen, werden als zusammengehörig wahrgenommen. Dies hilft dabei, nahe zusammenliegende Objekte voneinander unterscheiden zu können.

Gesetz der Dominanz Einfach strukturierte Objekte dominieren gegenüber komplexeren Formen.

Gesetz der Erfahrung Undefinierte Strukturen werden aufgrund persönlicher Erfahrung als bekannte Objekte wahrgenommen. Dieses Phänomen der Wahrnehmung ist notwendig, um dreidimensionale Objekte auf einer Ebene wahrzunehmen.

Gesetz der Figur-Grund-Beziehung Zur Unterscheidung zwischen einer Figur und ihrem Hintergrund werden die Aspekte Geschlossenheit, Abgrenzung, Textur, Farbe, Kontrast, Helligkeit, Räumlichkeit und Gliederung herangezogen. Eine bewusste Störung dieser Differenzierung bewirken sog. *Vexierbilder*. Figur und Hintergrund werden dort als gleichwertig wahrgenommen.

Gesetz der fortgesetzt durchgehenden Linie Angedeutete Linien werden imaginär derart fortgesetzt, dass sie einem möglichst einfachen Weg folgen. Dies gilt auch dann, wenn ein Hindernis wie etwa eine andere Linie im Weg ist. In diesem Fall nimmt der Betrachter eine Kreuzung wahr.

Gesetz der gemeinsamen Bewegung Mehrere Objekte, die sich mit demselben Geschwindigkeitsvektor vorwärts bewegen, werden als zusammengehörig aufgefasst.

Gesetz der gemeinsamen Region In abgegrenzten Gebieten enthaltene Objekte werden pro Gebiet als zusammengehörig empfunden.

Gesetz der Geschlossenheit Aus lückenhaften Konturen bildet das Gehirn geschlossene Konturen, da es dazu tendiert, komplette Objekte zu erkennen. Das gilt auch für Bilder, bei denen Teile fehlen oder überdeckt sind.

Gesetz der Gleichzeitigkeit Auch Objekte, die sich gleichzeitig verändern, werden unterbewusst gruppiert.

Gesetz der Innenseite Die Innenwinkel von Körpern sind in der Regel kleiner als die Außenwinkel.

Gesetz der Kontinuität Wenn ein Sinnesreiz als die Fortsetzung eines vorherigen Reizes wahrgenommen wird, empfindet man sie als zusammengehörig.

Gesetz der Nähe Die Nähe zwischen verschiedenen Objekten entscheidet über ihre Zusammengehörigkeit. Nahe beieinander liegende Objekte werden als Gruppe empfunden. Sind einige Objekte in einem Raum gleichmäßig mit einer bestimmten Konzentration verteilt und die restlichen weiter entfernt, entsteht ebenfalls der Eindruck einer Gruppierung der ersteren.

Gesetz der Prägnanz Die Wahrnehmung präferiert Figuren aus Strukturen, die möglichst einfach zusammengesetzt sind. Durch die Prägnanztendenz nimmt man in erster Linie Objekte wahr, die sich durch eine bestimmte Eigenschaft von den anderen Objekten unterscheiden.

Gesetz der Symmetrie Objekte, die symmetrisch angeordnet sind, werden als Einheit wahrgenommen.

Gesetz der verbundenen Elemente Objekte, die z. B. durch eine Linie miteinander verbunden sind, werden als zusammengehörig empfunden. Die direkte Verbindung von Objekten ist besonders stark und überschattet oft die anderen Gruppierungsprinzipien.

In Abbildung 2.17 sind einige Beispiele zu sehen, die die Wirksamkeit der Gestaltgesetze demonstrieren. Am bekanntesten dürfte vermutlich das Vexierbild oben rechts sein, in dem abwechselnd ein Kelch bzw. zwei Gesichter zu sehen sind.

[14] Quellen:
`http://www.robaweb.de/gdm/inhalt/VisuelleWahrnehmung/Gestaltwahrnehmung/03-Gestaltgesetze.html`
`http://de.wikipedia.org/wiki/Gestalttheorie`

	Variable_X	Variable_Y	Variable_Z	...
Sample_A	Wert_{AX}	Wert_{AY}	Wert_{AZ}	...
Sample_B	Wert_{BX}	Wert_{BY}	Wert_{BZ}	...
Sample_C	Wert_{CX}	Wert_{CY}	Wert_{CZ}	...
...	...	...	...	...

Tabelle 2.1.: Relationale Datentabelle mit einem Datensatz pro Sample.

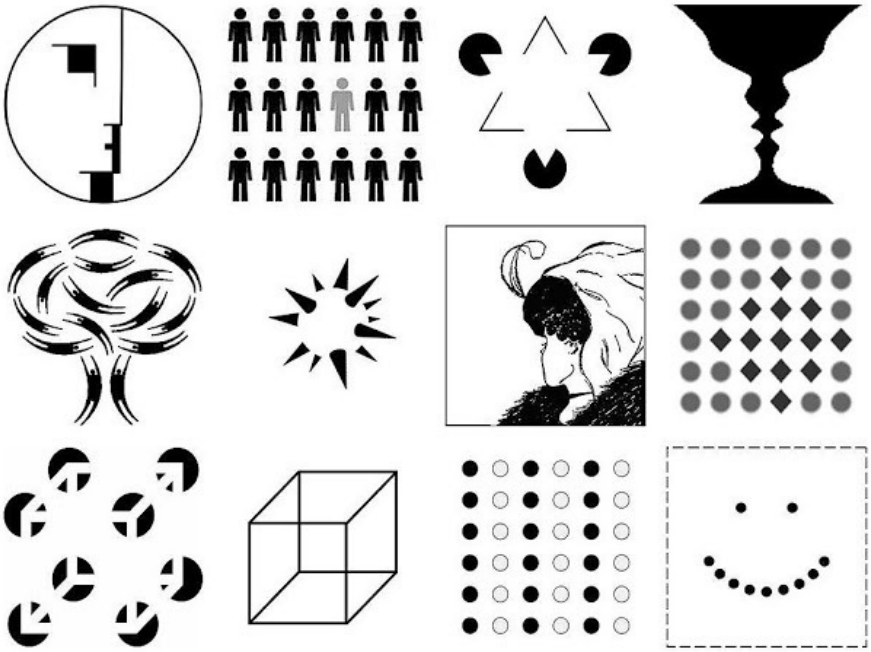


Abbildung 2.17.: Eine Zusammenstellung von Beispielbildern zur Veranschaulichung der Gestaltgesetze.[15]

2.12. Visualisierung multivariater Daten

Dieses Kapitel befasst sich mit den Möglichkeiten der Informationsvisualisierung, multivariate Daten anschaulich darzustellen. Zunächst wird erklärt, was es mit multivariaten Daten und Datentypen überhaupt für eine Bewandtnis hat. Im Anschluss werden konkrete Methoden anhand von Praxisbeispielen vorgestellt. Dabei muss stets im Hinterkopf behalten werden, die Übersichtlichkeit von Visualisierungen zu gewährleisten, indem Überfrachtung und Durcheinander vermieden werden.

2.12.1. Was sind multivariate Daten?

Datensätze, die bei einzelnen Messungen in der realen Welt anfallen, können je nach Anwendungsfall unterschiedlich viele Attribute beinhalten. In einer relationale Datentabelle lassen sich solche Datensätze ohne Probleme unterbringen. Pro Beobachtung bzw. *Sample* erzeugt man einen neuen Datensatz und pro gemessenem Attribut einen Wert in der dafür vorgesehenen Attribut-Spalte. Allgemein sieht dies folgendermaßen aus:

[15] Bildquelle: `http://upload.wikimedia.org/wikipedia/commons/a/a7/Gestalt_Principles_Composition.jpg`

Die Werte eines einzelnen Datensatzes lassen sich auch als Tupel interpretieren. Für den ersten Datensatz in der Beispieltabelle wäre das z. B. $(Wert_{\mathrm{AX}}, Wert_{\mathrm{AY}}, Wert_{\mathrm{AZ}})$.

Die Dimensionalität der Daten gibt an, wie viele Attribute der Datensatz einer Messung enthält. In Abhängigkeit von der Anzahl gibt es dafür auch Namenskonventionen:

- 1 Variable: *univariat*
- 2 Variablen: *bivariat*
- 3 Variablen: *trivariat*
- $>$ 3 Variablen: *multivariat*, *hypervariat*

Visualisierungen, die auf multivariaten Daten basieren, werden auch als multidimensional bezeichnet. Eine der größten Herausforderungen der Informationsvisualisierung ist es, multivariate Daten darzustellen. Dies trifft beispielsweise auch auf sicherheitsrelevante Server-Zustandsinformationen wie Paketlisten und Firewall-Regeln von Linux-Servern zu.

2.12.2. Datentypen

Im vorherigen Kapitel wurde davon berichtet, dass sich ein multivariater Datensatz aus mehreren konkreten Werten zusammensetzt. Diese Werte basieren wiederum auf Datentypen. Bei Datentypen handelt es sich um Mengen zulässiger Werte und den darauf definierten Operationen. Im Kontext der Informationsvisualisierung lassen sich Datentypen in die folgenden Kategorien einteilen [CON 07, S. 6 ff.].

- Kategorisch
- Ordinal
- Quantitativ
- Hierarchisch
- Graphen

Kategorische Datentypen werden auch als *nominale* Datentypen bezeichnet. Es handelt sich dabei um unsortierte Mengen, deren Elemente die möglichen Merkmalsausprägungen darstellen. Das sind beispielsweise Wörter wie Städtenamen oder Autohersteller. Die einzigen Operatoren, die sich auf die Werte dieses Datentyps anwenden lassen, sind Gleichheit und Ungleichheit ($=, \neq$).

Auf *ordinalen* Datentypen ist eine feste Ordnungsrelation definiert. Das heißt, dass die Werte dieses Typs in eine eindeutige Reihenfolge gebracht werden können. Dies trifft beispielsweise auf Schulnoten oder Michelin-Sterne zu. Ein messbares Intervall zwischen ordinalen Werten existiert im Allgemeinen nicht. Bei ordinalen Werten sind zusätzlich zu den zuvor genannten kategorischen Werten Größenvergleiche mit den beiden Operatoren $<$ und $>$ möglich.

Im Gegensatz zu ordinalen Werten besitzen jeweils zwei *quantitative* Werte immer ein messbares Intervall. Sie beschreiben numerische Eigenschaften von Objekten wie zum Beispiel die Anzahl geöffneter Ports auf einem Server. Auf ordinale Werte können arithmetische Operationen wie Addition und Subtraktion ($+, -$) angewendet werden.

Hierarchische Daten repräsentieren eine Rangordnung zwischen einzelnen Elementen. Jedes Element (mit Ausnahme des höchsten Elements in der Hierarchie) ist genau einem anderen Element untergeordnet. Repräsentiert werden hierarchische Daten meistens in Form von Bäumen. Beispiele hierfür sind sie Enthaltenseinsbeziehungen von Dateien und Ordnern in einem Dateisystem.

Graphen setzen sich einfach ausgedrückt aus einer Menge von Knoten und einer Menge von Kanten zusammen. Eine Kante verbindet je zwei Knoten aus der Knotenmenge. Dabei wird unterschieden, ob es sich um einen ungerichteten oder einen gerichteten Graphen handelt. Im Gegensatz zu ungerichteten Graphen wird bei gerichteten Graphen die Reihenfolge von Start- und Endknoten einer Kante berücksichtigt. Eine ausführliche

Einführung in Graphen wurde von *Gerald* und *Susanne Teschl* in einem Lehrbuch für Informatiker veröffentlicht [TES 07, S. 409 ff]. Graphen eignen sich beispielsweise, um Beziehungen zwischen Entitäten in einem Rechnernetz zu modellieren.

Rohdaten müssen nach der Messung in Werte abgebildet werden, die auf den o.g. Datentypen basieren. Dabei kann es vorkommen, dass Informationen verfälscht werden, z. B. durch Rundungen. Bei der Abbildung von quantitativen Daten auf ordinale Werte muss zuvor eine Bildung von sinnvollen Wertebereichen stattfinden, die für den jeweiligen Anwendungsfall ausreichend ist. Bei der Selektierung von Halbleiterkomponenten ist dies zum Beispiel gängige Praxis: Der unter High Fidelity (Hi-Fi)-Enthusiasten begehrte Kleinsignalfeldeffekttransistor *2SK170* wurde von seinem Hersteller *Toshiba* in drei verschiedenen Ausführungen angeboten [TOS 97]. Zur Kategorisierung diente der Drain-Sättigungsstrom I_{DSS}. Exemplare mit einem I_{DSS} von 2,6 – 6,5mA wurden in die Klasse *GR*, von 6,0 – 12mA in die Klasse *BL* und von 10 – 20mA in die Klasse *V* eingeteilt.[16]

2.12.3. Geometrische Transformation

Während sich Daten mit maximal drei Dimensionen noch gut auf einem zweidimensionalen Wiedergabemedium wie einem Bildschirm oder Papier darstellen lassen, stößt man bei den komplexeren multivariaten Daten sowohl an die Grenzen der Technik als auch der Auffassungskraft. Das stellt eine Herausforderung für die Informationsvisualisierung dar. Die *geometrische Transformation* versucht, dieses Problem zu umschiffen, indem unterschiedliche Projektionen der multivariaten Daten angeboten werden. Die folgenden Darstellungsmethoden bieten sich hierfür an:

- Streudiagramm-/Scatterplot-Matrizen
- Prosection-Matrizen
- Parallele Koordinaten

Ein Beispiel für eine Scatterplot-Matrix wurde bereits zuvor in Abbildung 2.3 vorgestellt. Sie besteht aus vielen zweidimensionalen Projektionen von je zwei Attributen, die gegenübergestellt werden. Die Anzahl dieser Teilbilder verhält sich quadratisch zur Anzahl der Variablen. Dass pro Zeile auch jede Variable auf sich selbst gemappt wird, erkennt man an der Diagonalen, die sich quer über das ganze Schaubild legt. Sie bildet gleichzeitig eine Spiegelachse, da jedes Variablen-Paar zweimal abgebildet wird. Der Vorteil dieser Darstellungsmethode ist, dass man in Ruhe eine bestimmte Variable Schritt für Schritt mit jeweils einer anderen Variable vergleichen kann. Das Vergleichen von mehr als zwei Attributen zur selben Zeit ist allerdings nicht möglich.

Prosection-Matrizen bedienen sich dem Konzept von Scatterplot-Matrizen, bringen aber Interaktivität ins Spiel. Vorgestellt wurden Prosection-Matrizen im Jahre 1995 von *Bob Spence et al.* [SPEN+ 95]. Sie bestehen ebenfalls aus einzelnen Streudiagrammen, die alle möglichen Variablen-Paare repräsentieren. Ein einzelner Scatterplot wird beispielsweise aus drei Parametern folgendermaßen gebildet: Die ersten zwei Parameter werden auf Abszisse und Ordinate des Scatterplots abgebildet. Sie stellen eine zweidimensionale Projektionsfläche dar. Werte des dritten Parameters werden nur dann auf diese Fläche projiziert, wenn sie sich innerhalb eines Werteintervalls befinden, das der Anwender interaktiv festgelegt hat (siehe Abbildung 2.18). Da es sich hierbei um die ***Pro****jektion* einer ***Sek****tion* eines Parameterraums handelt, wird das bildgebende Verfahren *Prosection* genannt.

Visualisierungen mit *parallelen Koordinaten* besitzen für jede Variable eine eigene vertikale Achse. Die Achsen sind, wie der Name vermuten lässt, parallel nebeneinander und äquidistant angeordnet. Jede der Achsen umfasst das Intervall zwischen dem kleinsten und dem größten darstellbaren Wert der jeweiligen Variable. Ein Datensatz wird in das Schaubild eingezeichnet, indem auf jeder Achse der entsprechende Attributwert markiert wird. Die Punkte eines Datensatzes zwischen je zwei benachbarten Achsen werden daraufhin durch

[16] Exemplare des Typs *2SK170 BL* waren im Do it yourself (DIY)- und Hi-Fi-Tuning-Bereich besonders begehrt. Spätestens seit der Einstellung der Produktion im Jahr 2008 wurde der Markt von Produktfälschungen überschwemmt.

[17] Bildquelle: `http://www.medien.ifi.lmu.de/lehre/ws1314/iv/slides/iv-ws1314-04-multidimensional.pdf`

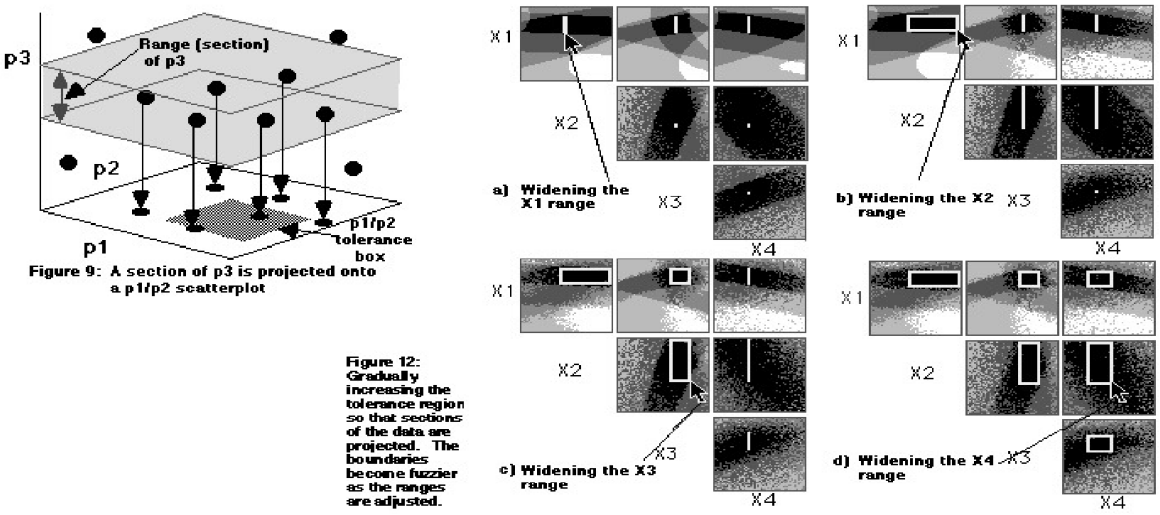


Abbildung 2.18.: Prosection-Matrizen entstehen durch die Projektion der Sektion eines Parameterraums (links). Rechts ist zu sehen, wie die Toleranzregionen einzelner Dimensionen mit dem Mauszeiger geändert werden.[17]

eine Strecke miteinander verbunden. So ergibt sich für jeden Datensatz eine charakteristische Spur in der Visualisierung.

Diese Darstellungsform bietet Vorteile gegenüber den zuvor genannten: Die Komplexität der Grafik verhält sich lediglich direkt proportional zur Anzahl der Variablen und nicht proportional zum Quadrat. Alle Variablen bekommen gleich viel Platz auf ihrer eigenen Achse. Positiv korrelierende Datensätze stechen ins Auge aufgrund der nahezu parallelen Pfade. Andererseits sind Ausreißer auch gut sichtbar. Variablen, die sich zueinander indirekt proportional verhalten, sind ebenfalls gut zu erkennen: Wenn die dazugehörigen Achsen direkt nebeneinander liegen, kreuzen sich die Verbindungslinien.

Doch es gibt auch Nachteile: Je mehr kategorische Variablen ein Datensatz enthält, desto weniger brauchbare Informationen sind aus der Visualisierung ablesbar, da es oft keine sinnvolle Sortierung für kategorische Daten gibt. Mit der steigenden Anzahl an Datensätzen droht bei Visualisierungen mit parallelen Koordinaten auch Unübersichtlichkeit. Dem kann durch sog. *Brushing* entgegengewirkt werden, indem Datensätze, die vom Benutzer interaktiv festgelegten Anforderungen entsprechen, farblich hervorgehoben werden. Durch das Ausblenden nicht benötigter Datensätze steigt die Übersichtlichkeit weiter.

Abbildung 2.19 zeigt den Parallelkoordinaten-Plot des Beispiel-Datensatzes `Seatbelts` des Statistikprogramms R. Es handelt sich dabei um eine Statistik über Autofahrer, die vor und kurz nach Einführung der Anschnallpflicht in England im Straßenverkehr umgekommen sind. Um die Datensätze besser auseinanderhalten zu können, wurden sie unterschiedlich eingefärbt. Die Anzahl an Datensätzen nach Einführung der Anschnallpflicht ist in dem Demo-Datensatz zwar deutlich geringer als zuvor, es lässt sich aber ein Trend herauslesen: Die Samples mit den wenigsten verunglückten Fahrern stammen überwiegend aus der Zeit mit Gurtpflicht. Außerdem geht aus dem Diagramm hervor, dass diese Fahrer auch größere Strecken zurückgelegt haben.

2.12.4. Glyphen

Neben der geometrischen Transformation bieten *Glyphen* die Möglichkeit, multivariate Daten effizient abzubilden. Bei Glyphen handelt es sich um zusammengesetzte, in der Regel kleine grafische Objekte, die über mehrere visuelle oder geometrische Attribute verfügen. Multivariate Daten werden auf diese Attribute gemappt. Die Anzahl der visuellen Attribute entspricht dabei der Anzahl der Dimensionen der Quelldaten.

[18] Bildquelle: `http://upload.wikimedia.org/wikipedia/commons/1/17/Chernoff_faces_for_evaluations_of_US_judges.svg`

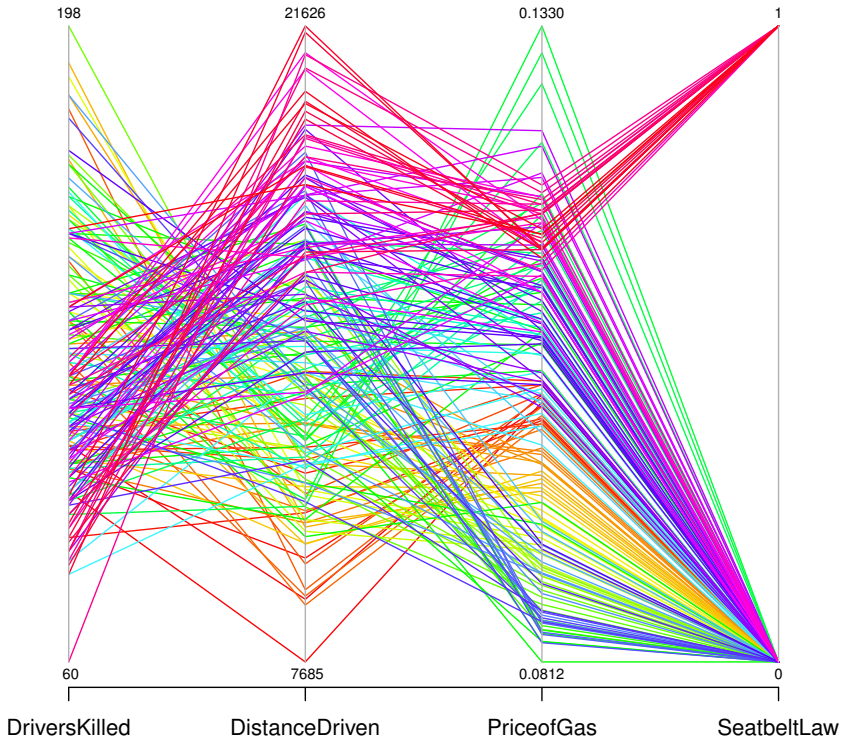


Abbildung 2.19.: Parallel-Koordinaten-Plot basierend auf einem *R*-Beispieldatensatz. Durch farbige Markierungen sind die einzelnen Datensätze besser zu unterscheiden.

Ein einfaches Beispiel für Glyphen sind Pfeile in zweidimensionalen Vektorfeldern. Jeder einzelne Pfeil symbolisiert durch seinen Winkel die Richtung und durch seine Länge die Geschwindigkeit eines Objekts. Neben den Attributen Winkel und Länge lassen sich auch weitere visuelle Attribute unterbringen, die zusätzliche Datenvariablen repräsentieren. Dazu zählen zum Beispiel Farbe, Form etc. Zu den prominenten Vertretern von glyphenbasierten Visualisierungen zählen:

- Chernoff-Gesichter
- Stick-Figure Icons
- Shape Coding
- Color Icons
- Sternglyphen

Chernoff-Gesichter repräsentieren multivariate Daten durch die unterschiedlichen Ausprägungen abstrahierter menschlicher Gesichter. Sie sind nach ihrem Erfinder *Herman Chernoff* benannt [CHER 74]. Die Motivation, Gesichter als Glyphen zu verwenden, liegt darin begründet, dass es Menschen leicht fällt, auch minimale Unterschiede in Gesichtern festzustellen. Maximal 18 Variablen lassen sich auf unterschiedliche Gesichtsmerkmale mappen.

Abbildung 2.20 zeigt Chernoff-Gesichter, die aus einem Beispiel-Datensatz aus *R* resultieren. Die Daten bestehen aus Einschätzungen der Persönlichkeit von Richtern durch Rechtsanwälte, die mit ihnen schon einmal Kontakt hatten.

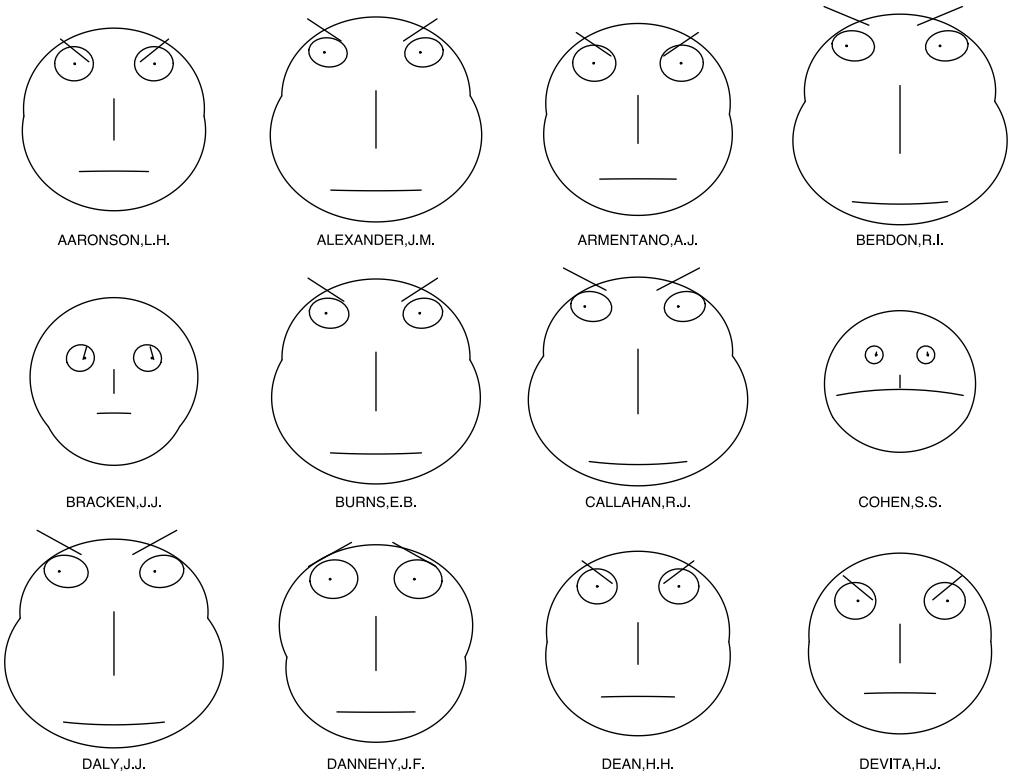


Abbildung 2.20.: Persönlichkeiten von Richtern aus der Sicht von Rechtsanwälten dargestellt als Chernoff-Gesichter (Demo-Datensatz aus *R*).[18]

Wie auch menschliche Gesichter sind Chernoff-Gesichter symmetrisch und beinhalten deswegen redundante Informationen. Es überrascht nicht, dass Edward Tufte dies für Platzverschwendung hält, da es sich hierbei um Non-Data-Ink (siehe Abschnitt 2.7.1) handelt [TUFT 01]. Dass man durch eine Unterscheidung der beiden Gesichtshälften doppelt so viele Variablen unterbringen könnte, stellten auch *Bernhard Flury* und *Hans Riedwyl* fest [FLU+ 81].

Die sog. *Stick-Figure Icons* wurden im Jahr 1988 von *Ronald M. Pickett* und *Georges G. Grinstein* in einem Paper vorgestellt [PIC+ 88]. Jedes einzelne Icon, das aus zusammenhängenden Strecken besteht, repräsentiert einen Datensatz. Jeweils zwei Attribute sind auf die zweidimensionale Position eines Symbols gemappt. Winkel bzw. Längen der vier Anhängsel der Figuren repräsentieren weitere Dimensionen. Wenn sehr viele Datensätze in einer Visualisierung untergebracht werden, entsteht eine hohe Dichte und beim Betrachter der Eindruck einer Textur. Aus diesen Textur-Eigenschaften kann das geschulte Auge bestimmte Merkmale ablesen, die für den gesamten betrachteten Datenraum charakteristisch sind. Abbildung 2.21 stammt aus dem genannten Paper und zeigt eine solche von Stick-Figure Icons dicht besiedelte Grafik. Darin sind die Messwerte der Sonden eines Wettersatelliten kodiert.

Stick-Figure Icons können schnell unübersichtlich werden, sobald sich die einzelnen Figuren überlappen. Man spricht dann auch von sog. *Visual Clutter*. Eine vollständig überlappungsfreie Darstellung ist hingegen durch sog. *Shape Coding* möglich. Pro Datensatz wird ein rechteckiger Glyph gezeichnet, der in einzelne Zellen unterteilt ist. In jeder Zelle wird der Wert eines Attributs durch einen Helligkeitswert kodiert. Bei der Positionierung der einzelnen Glyphen gibt es mehrere Möglichkeiten. Sie können in Zeilen und Spalten angeordnet werden oder in einem Koordinatensystem platziert werden, um weitere Dimensionen durch die Objektposition zu kodieren. Eine geschickte Anordnung der Glyphen unterstützt das Erkennen interessanter Muster in den Datensätzen. Vorgestellt wurde Shape Coding im Jahr 1990 von Jeff Beddow [BED 90].

Abbildung 2.22 zeigt eine Beispielgrafik aus Beddows Paper. In den einzelnen Glyphen sind Messwerte aus der Magnetosphäre und von Sonnenwinden (in leider schlecht zu erkennenden Graustufen) von insgesamt drei Wochen kodiert. Die Glyphen sind derart räumlich angeordnet, dass sich die Muster der einzelnen Tage

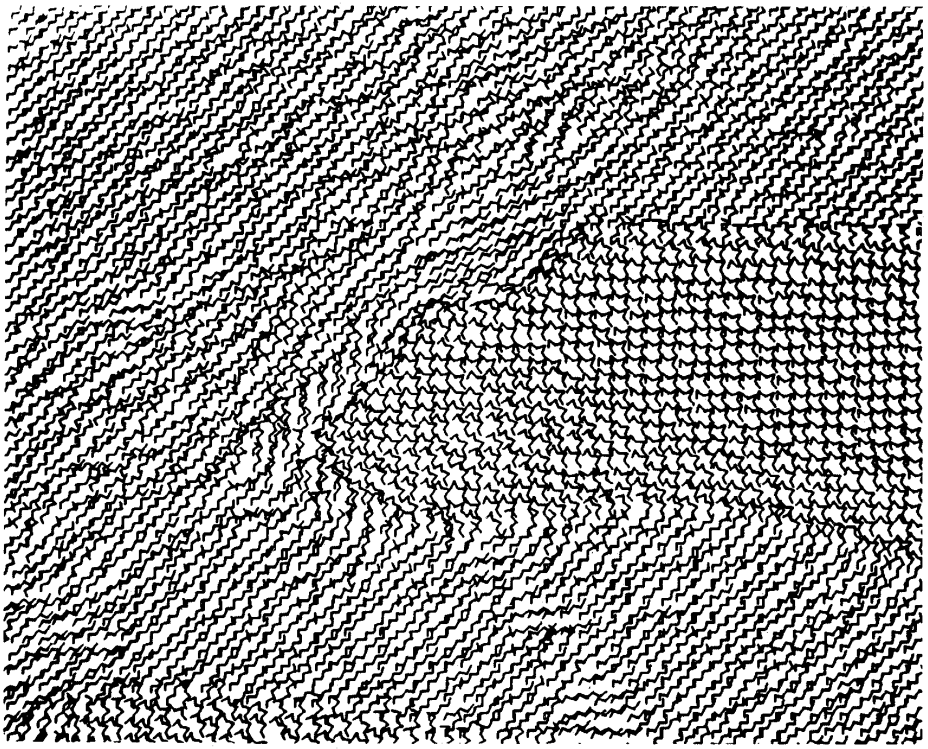


Abbildung 2.21.: Stick-Figure-Visualisierung basierend auf den Messwerten eines Wettersatelliten. Durch die Dichte der einzelnen Figuren ergibt sich eine erkennbare Textur.

(Ordinate) und Stunden (Abszisse) gut miteinander vergleichen lassen.

Das Grundprinzip von *Color Icons* ist mit dem von Shape Coding vergleichbar. Allerdings kommen hier keine Graustufen, sondern Farben zum Einsatz. Farben bieten im Gegensatz zu Graustufen den Vorteil, dass der Betrachter eine größere Anzahl davon sicher unterscheiden kann. Die Glyphen werden so klein gezeichnet, dass die einzelnen Attributwerte gerade noch wahrgenommen werden können und je nach Anwendungsfall angeordnet. Trennlinien helfen dem Betrachter beim Auseinanderhalten der Zellen. Keim und Kriegel stellten eine spiralförmige Anordnung vor, die Abfrageergebnisse einer Datenbank visualisiert [KEIM+ 94]. Die Beispielgrafik aus ihrem Paper in Abbildung 2.23 zeigt ein Abfrageergebnis, das insgesamt acht Dimensionen umfasst. Die erste der jeweils neun Zellen kennzeichnet den sog. *Relevanzfaktor* des entsprechenden Ergebnisses. Die Relevanz der Ergebnisse nimmt von innen nach außen ab.

Sternglyphen, auch *Netzdiagramme* genannt, besitzen ähnlich wie Parallelkoordinatendiagramme für jede Dimension eine Achse. Sie sind allerdings sternförmig mit um einen gemeinsamen Mittelpunkt angeordnet. Die Winkel zwischen den Achsen sind alle gleich. Variablenwerte werden von der Mitte nach außen angetragen [SIEG+ 72]. Netzdiagramme eignen sich gut, um mehrere Datensätze grob miteinander zu vergleichen, indem man sie übereinander stapelt und die Verbindungslinien der Datensätze für eine bessere Übersichtlichkeit unterschiedlich einfärbt. So ist es auch in der Beispiel in Abbildung 2.24 zu sehen.

2.12.5. Pixelbasierte Darstellung

Pixelbasierte Darstellungen sehen auf den ersten Blick den glyphenbasierten Color Icons ähnlich, doch es gibt einen deutlichen Unterschied: Hier existieren gar keine Glyphen. Die Darstellungsfläche ist in Zellen aufgeteilt, die jeweils nur einen Pixel groß sind und den Attributwert einer Dimension aufnehmen. Dies hat zur Folge, dass die Werte eines Datensatzes nicht zusammenhängen, sondern gleichmäßig über alle Dimensionszellen verteilt sind. Trennlinien sind nicht vorgesehen.

[19] Bildquelle: `http://upload.wikimedia.org/wikipedia/commons/5/59/Netzdiagramm-Beispielp.svg`

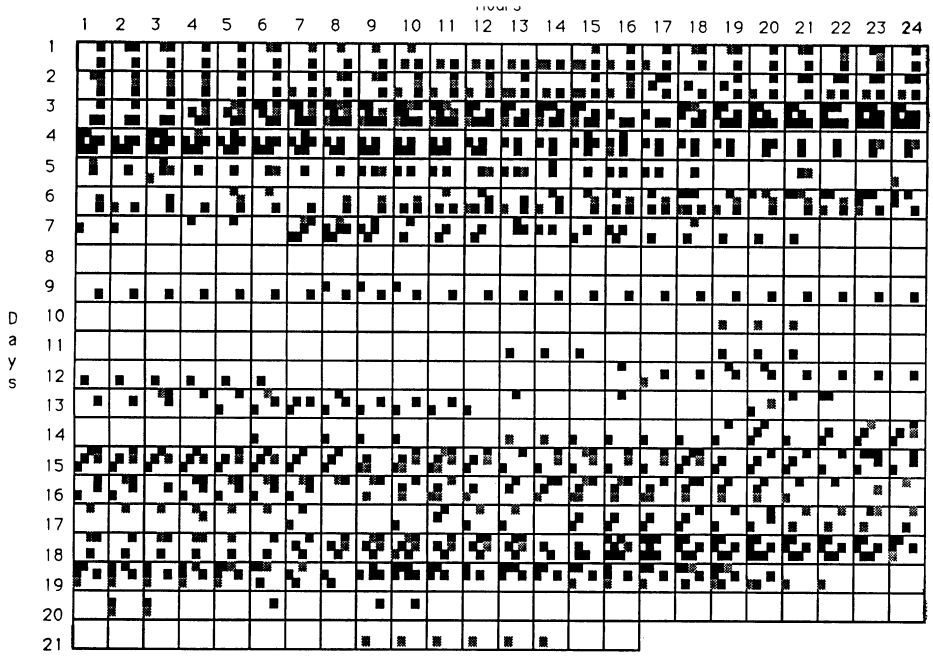


Abbildung 2.22.: Beispiel für eine Shape-Coding-Darstellung. Sie beinhaltet insgesamt 13 Variablen aus Sonnenwinddaten und der Magnetosphäre. Die Attributwerte sind auf die Helligkeitswerte Weiß, Grau und Schwarz abgebildet. Tage sind an der Ordinate und Stunden an der Abszisse angetragen.

Bei der Anordnung der Pixel in den Zellen werden oft sog. *FASS-Kurven* (space-**f**illing, self-**a**voiding, **s**imple and self-**s**imilar bzw. raumfüllend, selbstausweichend, einfach und selbstähnlich) wie etwa *Lebesgue-*, *Peano-* oder *Hilbertkurven* verwendet.

Um die inhärente Struktur bestimmter Daten in der Visualisierung widerzuspiegeln, setzten *Keim* et al. im Jahr 1995 *rekursive Muster* ein [KEIM+ 95]. Bei Daten, die über einen langen Zeitraum gesammelt wurden, ist dies naheliegend. Messwerte, die zu einem Tag gehören, werden in der ersten Ebene untergebracht, Daten einer Woche in der zweiten Ebene, Daten eines Monats in der dritten Ebene u.s.w. Abbildung 2.25 zeigt eine Beispielgrafik aus Keims Paper. Dort wird die Entwicklung verschiedener Börsenkurse über einen Zeitraum von mehreren Jahren bis 1995 dargestellt.

Mit pixelbasierten Techniken lassen sich aufgrund ihres günstigen Data-Ink-Verhältnisses (siehe Abschnitt 2.7.1) problemlos große Datensätze visualisieren. Da keine Überlappung von Objekten stattfindet, können Betrachter darin enthaltene Muster gut erkennen. Mit einer wachsenden Anzahl von Dimensionen geht allerdings auch die Übersichtlichkeit immer mehr verloren. Dass dann auch die Beschriftung immer schwieriger wird, versteht sich von selbst.

2.12.6. Herunterskalierung von Dimensionen

Bereits zuvor wurde erwähnt, dass Visualisierungen auf dem Papier oder dem Bildschirm begrenzte Darstellungsmöglichkeiten für multidimensionale Datensätze bieten. Ein Mapping auf unterschiedliche visuelle Attribute ist bei einer zwei- oder dreidimensionalen Darstellung nötig. Eine weitere Lösungsmöglichkeit stellt die *Herunterskalierung von Dimensionen* dar. Es handelt sich dabei um eine Projektion von n Dimensionen auf eine geringere Dimensionalität $m < n$. Dieser Prozess ist mit Informationsverlust verbunden, deshalb muss berücksichtigt werden, dass die je nach Anwendungsfall relevanten Daten nicht verloren gehen.

Besonders bei Daten mit sehr hoher Dimensionalität stellt sich die Frage, welche Dimensionen bei der Herunterskalierung ausgeblendet werden. Einen auf statistischen Methoden basierenden Lösungsansatz stellt beispielsweise die *Principal-Components-Analyse* dar. Sie findet unter anderem Einsatz bei der Kompression von

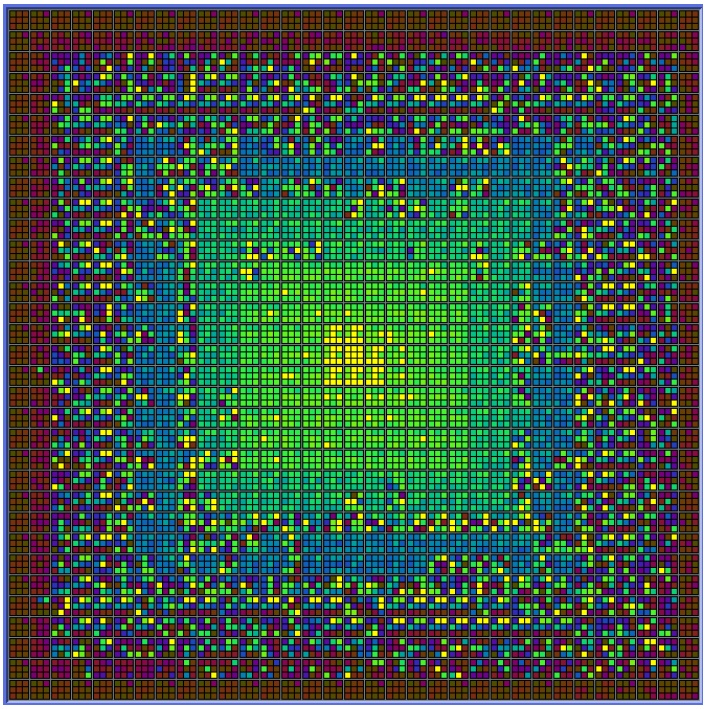


Abbildung 2.23.: Color-Icon-Visualisierung eines Abfrageergebnisses mit ca. 1000 Datensätzen. Jeder Datensatz wird durch ein neunelementiges Color Icon repräsentiert. Acht Elemente verkörpern die Dimensionen des Datensatzes, das erste Element (jeweils ganz oben links) die Relevanz des Ergebnisses. Dieser sog. Relevanzfaktor nimmt spiralförmig im Uhrzeigersinn von innen nach außen ab.

Bilddateien oder der Gesichtserkennung [SMI 02].

Sog. *Kohonen-Karten* basieren auf selbstorganisierenden künstlichen neuronalen Netzen [KOH 95]. Sie sind nach ihrem Erfinder, dem finnischen Ingenieur *Teuvo Kohonen* benannt und werden auch als selbstorganisierende Karten bezeichnet. Sie dienen zur Abbildung eines multidimensionalen Eingangsraums auf einen in der Regel zweidimensionalen Raum. Das zugrundeliegende neuronale Netz wird durch sog. *unüberwachtes Lernen* Schritt für Schritt aufgebaut. Im Gegensatz zu vielen anderen neuronalen Netzen verwenden Kohonen-Karten eine sog. *Nachbarschaftsfunktion*, um die topologischen Eigenschaften des Eingangsraums im Ausgangsraum beizubehalten.

Ein iterativer Algorithmus produziert eine zweidimensionale Karte, in denen ähnliche Datensätze räumlich benachbart werden. Das Ergebnis nach einer ausreichenden Anzahl an Iterationen ist eine topografische Abbildung. Die Interpretation einer solchen Visualisierung ist für den Betrachter zeitaufwendig, da auf die Koordinaten der einzelnen Objekte keine konkrete Dimension abgebildet wird, obwohl die Objektposition nach Cleveland und McGill die kräftigste visuelle Variable ist (siehe Abschnitt 2.10). Die Objekte sind zufällig angeordnet. In Abbildung 2.26 ist eine Visualisierung zu sehen, die auf Worthäufigkeiten in englischsprachigen Wikipedia-Seiten basiert. Die Entfernung der Objekte ist indirekt proportional zur thematischen Ähnlichkeit der Artikel. Die rot eingezeichneten Linien kennzeichnen Hyperlinks zwischen den entsprechenden Wikipedia-Seiten.

2.12.7. Vermeidung von Visual Clutter

Visual Clutter bzw. Wirrwarr in Abbildungen multidimensionaler Daten zu vermeiden, ist eine zentrale Aufgabe der Informationsvisualisierung. Es gibt unterschiedliche Methoden, um dies zu bewerkstelligen:

[20] Bildquelle: `http://upload.wikimedia.org/wikipedia/en/0/07/Self_oraganizing_map_cartography.jpg`

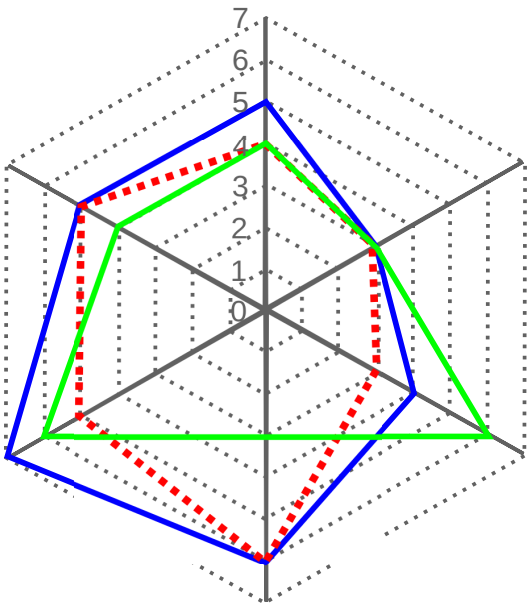


Abbildung 2.24.: Beispiel für ein Netzdiagramm.[19]

- Dimensionen anders anordnen
- Sampling
- Konstante Informationsdichte
- Point Displacement
- Aggregation und Clusterbildung
- Filter

Die *Dimensionen anders anzuordnen* kann abhängig von den betrachteten Datensätzen zu mehr Übersichtlichkeit führen. Bei Parallelkoordinatendiagrammen wird beispielsweise die Reihenfolge der Achsen vertauscht, bis sich ein klareres Bild mit weniger Überschneidungen ergibt. An der Anzahl der angezeigten Datensätze wird nichts geändert.

Anders verhält es sich beim *Sampling*. Um Clutter zu verringern, wird dort nur eine zufällige Teilmenge aller Datensätze als Grundlage für die Visualisierung verwendet. Durch das zufällige Auswählen von Datensätzen bleibt die statistische Verteilung der Daten in der Regel erhalten, sodass Trends wie zum Beispiel Korrelationen weiterhin erkennbar sind.

Sampling kann auch in Ausschnitten einer Visualisierung eingesetzt werden. Die *Sampling-Linse* von *Geoffrey Ellis et al.* wird vom Benutzer auf dem Bildschirm über einer Visualisierung der vollständigen Verteilung bewegt. Bei der Fläche, die sich unter der Linse befindet, wird durch zufälliges Sampling die Dichte der Daten verringert, indem nur eine zufällige Teilmenge der Objekte angezeigt wird [ELL+ 05]. Abbildung 2.27 zeigt eine beispielhafte Verteilung mit drei Häufungspunkten. Auch unter der Sampling-Linse sind die Häufungen weiterhin erkennbar.

In Visualisierungen mit *konstanter Informationsdichte* kommen Glyphen zum Einsatz. Ihre Größe und ihr Detailgrad hängt von der Anzahl der benachbarten Glyphen, also von der örtlichen Informationsdichte, ab. In Regionen mit hoher Dichte werden kleine Glyphen mit wenigen Details eingeblendet und umgekehrt. Während sich der Betrachter am Rechner durch die Darstellung bewegt, bleibt die Anzahl der angezeigten Informationen immer konstant [WOOD+ 98]. Dadurch ist gewährleistet, dass er zu keinem Zeitpunkt aufgrund zu vieler Informationen den Überblick verliert.

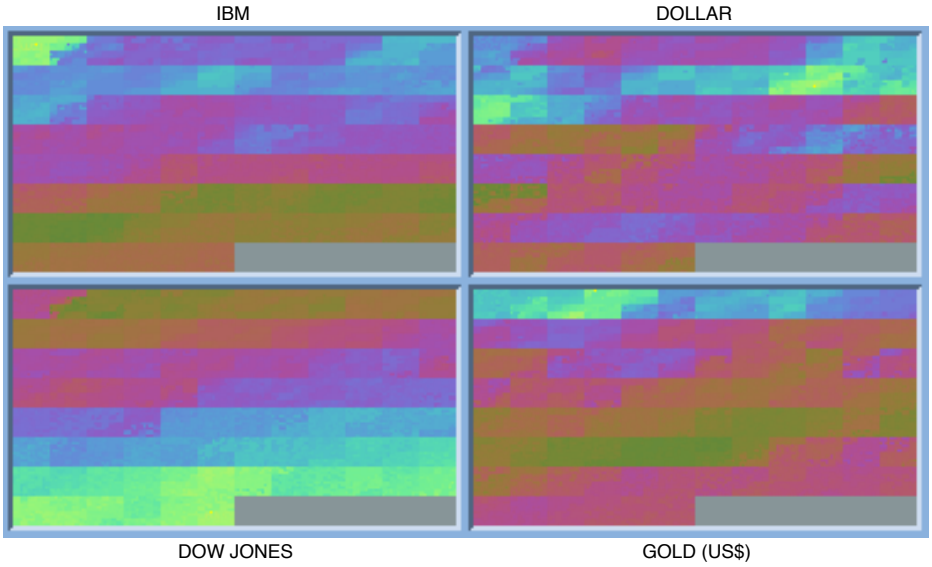


Abbildung 2.25.: Entwicklung von Börsenkursen mit rekursiven Mustern dargestellt: Die acht horizontalen Zeilen entsprechen jeweils acht aufeinanderfolgenden Jahren. Monate eines Jahres sind in zwölf Spalten untergebracht.

Visualisierungen, die auf *Point Displacement* beruhen, bemühen sich im Gegensatz zur Sampling-Linse oder zu Visualisierungen mit konstanter Informationsdichte um eine vollständige Anzeige aller Objekte zur gleichen Zeit. Für jedes Objekt, das der Visualisierung hinzugefügt werden soll, wird vor dem Einzeichnen geprüft, ob an der entsprechenden Position noch Platz ist. Andernfalls wird es an einem möglichst nahegelegenen freien Platz eingesetzt. Falls auch im engen Umkreis kein Platz mehr frei ist, können sich darin befindliche Objekte verschoben werden, um Platz zu schaffen. Die durchgeführten Verschiebungen sollen insgesamt so minimal ausfallen, dass die resultierende Visualisierung einer Darstellung ohne Point Displacement möglichst nahe kommt. Bei einer zu hohen Verzerrung drohen sonst Fehlinterpretationen.

Zur Bestimmung der Position der zu verschiebenden Objekte gibt es unterschiedliche Lösungsansätze. Dazu zählen beispielsweise *Nearest-Neighbor*-Algorithmen. Sie sind einfach zu berechnen, können aber zu starken Verzerrungen führen. Alternativ können Objekte entlang raumfüllender Kurven verschoben werden. Dies führt insgesamt zu geringeren Verzerrungen. Hier nimmt man allerdings einen höheren Rechenaufwand in Kauf. Auf einer hierarchischen Partitionierung des Datenraums basiert der 1998 von Keim et al. vorgestellte *Gridfit*-Algorithmus [KEIM+ 98].

Aggregation und Clusterbildung reduzieren die Anzahl angezeigter Objekte und damit auch Clutter, indem sog. *Stellvertreterobjekte* verwendet werden. Ein Objekt repräsentiert mehrere ursprüngliche Objekte, die anhand bestimmter Kriterien gruppiert wurden. Nicht nur zur Anzeige, sondern auch zur indirekten Manipulation von Objekten können Stellvertreterobjekte verwendet werden (siehe Abschnitt 2.14.2).

Filter dienen dem Betrachter, um sukzessive die Anzahl der anzuzeigenden Datensätze zu verringern, sodass am Ende nur noch die für den Betrachter relevanten Objekte übrig bleiben. Entweder wird dann in der ganzen Grafik nur eine Teilmenge der Objekte angezeigt oder der Betrachter nutzt *Zoom*-Funktionen, um nur einen Ausschnitt der gesamten Darstellung zu sehen.

2.13. Farben

Farben sind aus Visualisierungen und grafischen Bedienoberflächen in der heutigen Zeit nicht mehr wegzudenken. Sie eignen sich hervorragend als visuelles Attribut, da der Mensch Farben besser unterscheiden kann als Graustufen. Zudem fällt es ihm leichter, Farben mit einem bestimmten Namen zu assoziieren oder sie zumindest zu umschreiben als es bei Graustufen der Fall ist. Farben benennen zu können ist ein entscheidender

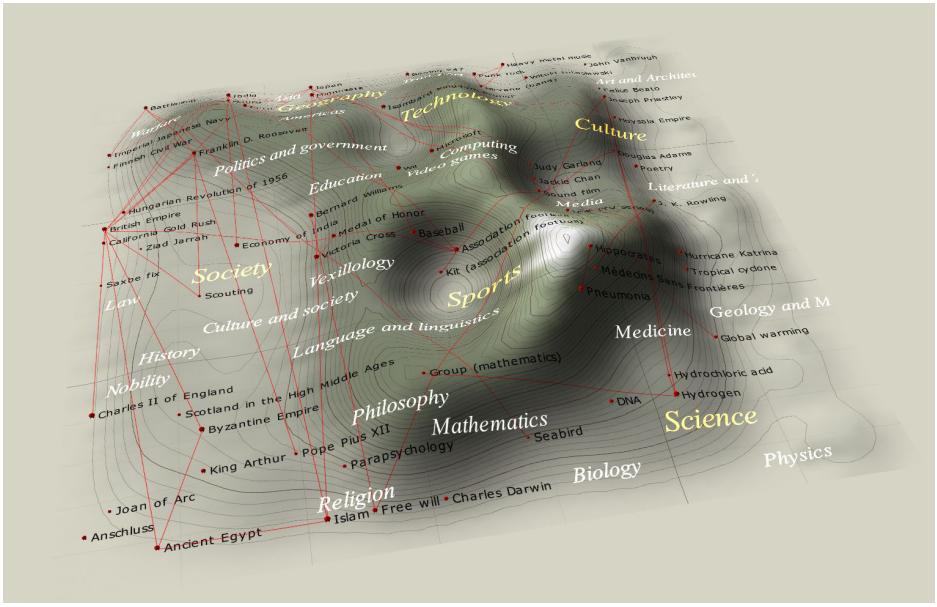


Abbildung 2.26.: Kartografische Repräsentation einer Kohonen-Karte basierend auf der Worthäufigkeit in englischsprachigen Wikipedia-Artikeln. Je näher zwei Artikel in der Abbildung zusammen sind, desto ähnlicher sind sie auch thematisch. Clusterbildung wird durch die Umrisse der Berge angedeutet. Verlinkungen zwischen den Artikeln werden durch rote Linien zum Ausdruck gebracht.[20]

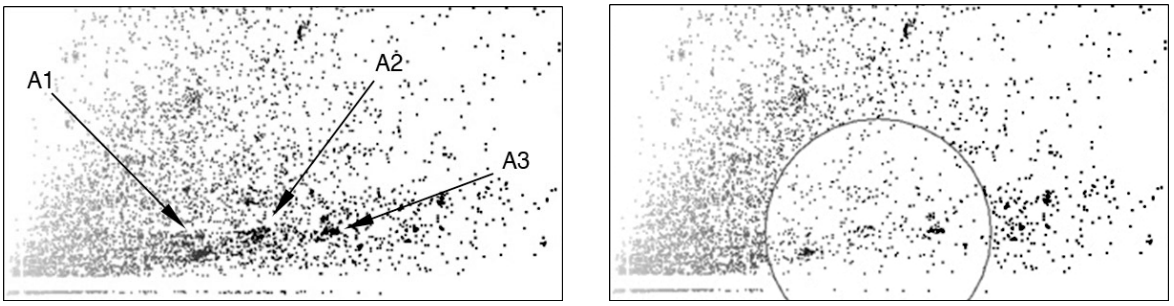


Abbildung 2.27.: Verteilung mit drei Häufungspunkten (links) ohne und mit (rechts) Sampling Lens. Die Häufungspunkte sind rechts immer noch zu erkennen.

Vorteil, wenn sich zwei Personen gemeinsam über eine Visualisierung unterhalten.

Obwohl Farben Vorteile gegenüber Graustufen bieten, ist eine inflationäre Verwendung von Farben kontraproduktiv. Um dennoch übersichtliche Visualisierungen und Bedienoberflächen erstellen zu können, müssen einige Grundregeln eingehalten werden.[21]

2.13.1. Farbskalen

Eine *Farbskala* überführt nach der Definition von *Haim Levkowitz* eine Menge unterschiedlicher kategorischer oder numerischer Werte (siehe Abschnitt 2.12.2) in eine bildliche Repräsentation. Jeder Wert besitzt dort eine eigene Farbe [LEV 92]. Die folgenden Anforderungen sollte eine Farbskala erfüllen:

[21] Beim Erstellen barrierefreier Visualisierungen sollte bedacht werden, dass im Schnitt 5% der Bevölkerung eine Rot-Grün-Sehschwäche besitzt. Der überwiegende Anteil davon sind Männer (Quelle: `http://de.wikipedia.org/wiki/Farbenblindheit`). Eine Vorstellung von Farbfehlsichtigkeit können sich Normalsehende mit Hilfe von einem Simulator verschaffen: `http://www.color-blindness.com/coblis-color-blindness-simulator/`

- Falls die zu repräsentierenden Daten eine Ordnung besitzen, sollten auch die Farbskalen einer bestimmten Ordnung folgen.
- Abstände zwischen zwei Werten sollten als äquivalente Abstände zwischen zwei Farben wahrnehmbar sein.
- Farbskalen dürfen keine Abgrenzungen beinhalten, die nicht auch in den Daten existieren.

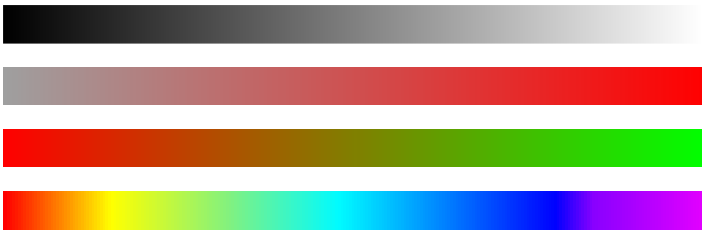


Abbildung 2.28.: Grauskala (oben) und unterschiedliche Farbskalen.

Ganz unten in Abbildung 2.28 ist eine Regenbogenskala zu sehen. Die darin enthaltenen Farben sind analog zu den Farben des sichtbaren Lichtspektrums angeordnet. Physiker haben vermutlich kein Problem damit, die Reihenfolge der Farben dieses Spektrums zu bestimmen. Der Rest der Betrachter benötigt eine Legende, um die Farben zu ordnen.

Zur Abbildung ordinaler Werte auf eine Farbskala bietet es sich an, als Start- und Endfarbe jeweils eine mit geringer und eine mit hoher Sättigung zu wählen. Ein Übergang von dunkel nach hell ist ebenfalls geeignet. In beiden Fällen sollte nur ein Farbton verwendet werden.

Falls mehrdimensionale Daten abgebildet werden sollen, sind mehrere Farbskalen erforderlich, die voneinander gut zu unterscheiden sind. Mit der Anzahl der Dimensionen steigt die Herausforderung. Weiter erschwert wird das Zusammenstellen geeigneter Farbschemas, wenn zusätzliche Rahmenbedingungen hinzukommen. Dazu zählen zum Beispiel die Barrierefreiheit für Rot-Grün-Blinde und die Druckfreundlichkeit. Eine weitere denkbare Randbedingung besteht darin, dass die Farbskalen auch nach dem Anfertigen von Fotokopien noch halbwegs zu erkennen sein sollen. Beim Zusammenstellen von Farbschemas unter komplexen Rahmenbedingungen gibt es im World Wide Web (WWW) verschiedene Generatoren wie zum Beispiel den *Color Scheme Designer*[22] oder den *ColorBrewer*[23].

Grauskalen stellen einen Sonderfall von Farbskalen dar. Die darin enthaltenen Grautöne können vom Menschen zwar nicht so gut unterschieden werden wie Farben, doch sie bieten einen Vorteil: Sie eignen sich besser als Farbskalen, um Ordnungen darzustellen. Dabei muss allerdings berücksichtigt werden, dass die Genauigkeit leidet, wenn der Kontrast zu gering ist. In Abbildung 2.28 ist eine Grauskala (oben) neben verschiedenen Farbskalen zu sehen.

In seinem Buch *Information Visualization: Perception for Design* schreibt *Colin Ware*, dass der "Helligkeitskanal" fundamental für viele Aspekte der visuellen Wahrnehmung ist, darunter für die Erkennung von Formen. Aus diesem Grund hält Ware Grauskalen für eine Verschwendung von Ressourcen der Wahrnehmung [WARE 04].

2.13.2. Farbregeln

Unabhängig vom Darstellungstyp einer Visualisierung sollten folgende Best-Practice-Regeln befolgt werden, um Farben gewinnbringend einzusetzen:

- Gewährleistung eines hohen Kontrasts zwischen Vordergrund- und Hintergrundfarbe.
- Schwarze Rahmen um Farbflächen verringern die Kontrastwirkung.
- Kanonische Farben, die nahe am Ideal liegen, sind gut zu unterscheiden und zu benennen.

[22] Color-Scheme-Designer-Website: `http://colorschemedesigner.com/csd-3.5/`
[23] ColorBrewer-Website: `http://colorbrewer2.org/`

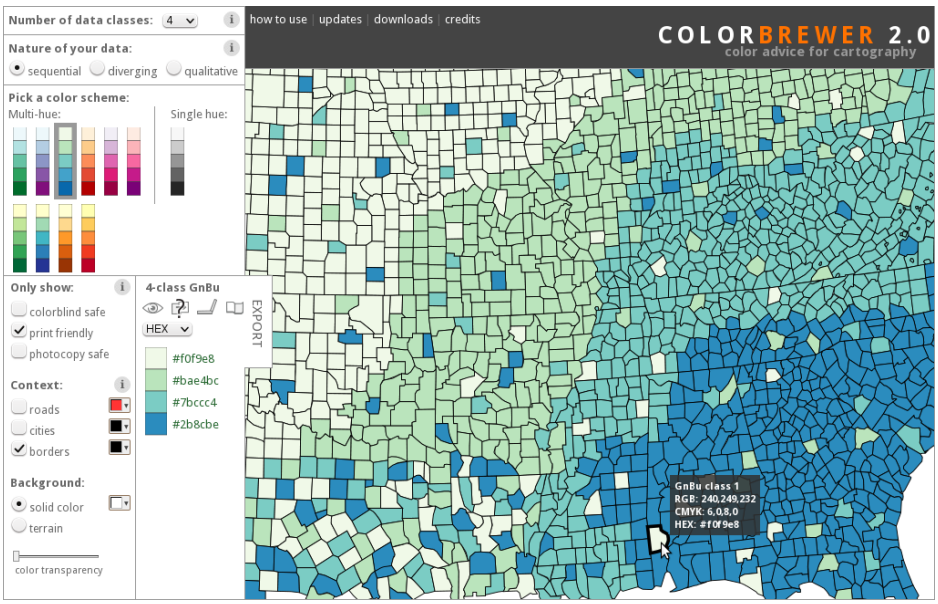


Abbildung 2.29.: Tools wie der ColorBrewer erleichtern die Zusammenstellung von Farbschemas selbst unter komplexen Rahmenbedingungen.

- Zur Repräsentation konkreter Werte nur eine begrenzte Zahl diskreter Farben verwenden.
- Die Farben Rot, Gelb, Grün und Blau sind vor allen anderen zu verwenden, da das die Primärfarben der meisten Betrachter sind.
- Für große Flächen vor einem weißen Hintergrund sind helle Farben mit geringer Sättigung zu verwenden.
- Kleine farbkodierte Objekte werden durch eine hohe Farbsättigung hervorgehoben.
- Die Farben Rot und Grün sollten nicht am Rand des Bildbereichs verwendet werden aufgrund der geringeren Farbempfindlichkeit der Retina im Randbereich.
- Die Farben Schwarz, Weiß und Gelb sollten nur peripher eingesetzt werden, da sie sonst zu viel Aufmerksamkeit auf sich ziehen.
- Farben eines Schemas müssen sich anhand von Sättigung und Helligkeit gut unterscheiden.

2.13.3. Farben in Bedienoberflächen

Mit dem Aufkommen von farbfähigen Grafikkarten und Monitoren hielten Farben in den 1980er Jahren Einzug in PCs. Von den technischen Möglichkeiten inspiriert befassten sich bereits damals Forscher mit dem optimalen Einsatz von Farben in Bedienoberflächen. *Barbara Meier* stellte fest, dass sich die meisten Anwendungsentwickler nicht an etablierte Designprinzipien aus der analogen Welt halten und Farben eher willkürlich auswählen, ohne ihre psychologische Wirkung zu berücksichtigen. Sie entwickelte ein Expertensystem, um zu erproben, ob ein Algorithmus automatisch optimale Farben für eine Bedienoberfläche bestimmen kann [MEI 88].

Wer für das Design von Bedienoberflächen zuständig ist und welche Paradigmen dabei angewendet werden, hat sich im Lauf der Zeit geändert. So wird seit Anfang des Jahrtausends vermehrt auf sog. User Experience (UX) Design gesetzt. Der Begriff wurde durch das Buch *The Invisible Computer* [NOR 99] des Informatikprofessors und Usability-Spezialisten *Donald Norman* geprägt und bezieht sich auf die Herangehensweise, dem Anwender positive Erlebnisse zu vermitteln. Eine Zusammenfassung der Paradigmen zur Entwicklung von Bedienoberflächen haben *Bernhard Preim* und *Raimund Dachselt* in ihrem Buch *Interaktive Systeme: Band 2* veröffentlicht [PRE 15].

Bereits Anfang der 1990er Jahre hat *Ben Shneiderman* ein Buch veröffentlicht, das vom korrekten Design von Bedienoberflächen handelt [SHN 04]. Dort befasst er sich auch mit dem Einsatz von Farben: Sie können auf den Betrachter sowohl beruhigend als auch aufrüttelnd wirken. Sie setzen auf dem Bildschirm Akzente und tragen zur Unterscheidbarkeit von Objekten in komplexen Bildschirminhalten bei. Farben können auch als ein Mittel eingesetzt werden, um die Informationsdichte auf dem Bildschirm zu erhöhen. Farben betonen die logische Organisation von Informationen und dienen auch dazu, um das Auge des Betrachters auf Warnungen zu lenken. Sie können Emotionen wie Freude, Aufregung, Angst oder Wut vermitteln. Im Kontext von Bedienoberflächen macht Shneiderman die folgenden Vorschläge zum Einsatz von Farben:

- Farben sollen sparsam und konsistent eingesetzt werden.
- Die Anzahl unterschiedlicher Farben soll gering sein.
- Farben sollen zur Aufgabenerfüllung beitragen.
- Der Anwender soll die Farben nach eigenen Bedürfnissen verändern können.
- Farben können die Textformatierung verbessern.
- Die Bedürfnisse von farbfehlsichtigen Anwendern müssen berücksichtigt werden.
- Einige Farben sind bei Anwendern mit bestimmten Erwartungen vorbelegt.
- Farbänderungen eignen sich zum Hinweis auf Statusänderungen.

Abbildung 2.30 zeigt Bildschirmfotos von verschiedenen Programmen, die Bestandteil von Microsofts kommendem Büropaket *Office 2016* sind. Deutlich zu erkennen sind die großflächig eingefärbten Titelleisten der einzelnen Programmfenster, wobei jede Anwendung eine eigene Farbe besitzt. Diese Farben erfüllen keinen Zweck für den Anwender, da sie nicht zur Aufgabenerfüllung beitragen. Vermutlich handelt es sich dabei um eine Marketing-Maßnahme. Immerhin bietet Microsoft die Option an, wieder zum klassischen Farbschema umzuschalten.

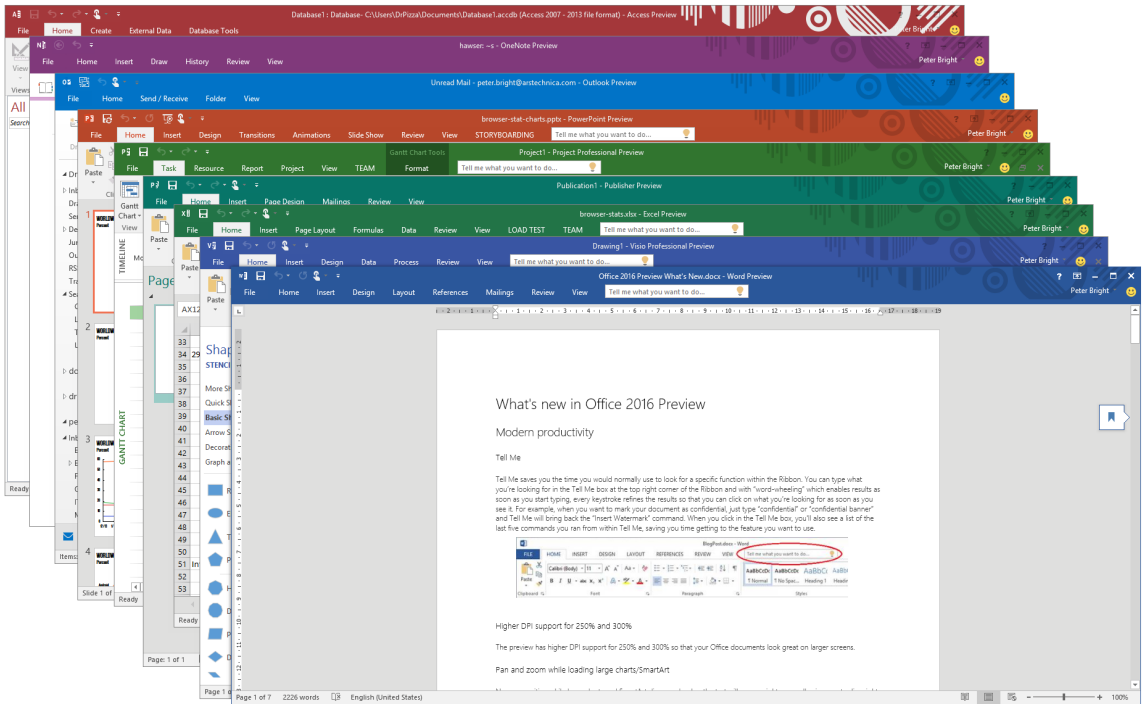


Abbildung 2.30.: Vorschauversionen der einzelnen Programme von Microsoft Office 2016 mit unterschiedlich eingefärbten Titelleisten (Hintergrundbild entfernt).[24]

[24] Bildquelle: `http://cdn.arstechnica.net/wp-content/uploads/2015/03/colorful-titlebars.png`

In Zusammenhang mit den Screenshots von Microsoft Office kann der Begriff *Eye Candy* erklärt werden. Darunter versteht man im Kontext der Werbung attraktive Bildelemente, die die Aufmerksamkeit des Betrachters besonders lange auf sich ziehen sollen, um ihm die zugrundeliegende Werbeinformation eindringlicher zu vermitteln [WIED 10]. Auch in Bedienoberflächen gibt es derartige Elemente, die keinen anderen Zweck erfüllen, als die Optik des Programms zu verschönern. In Abbildung 2.30 zählen die Wasserzeichen in den Titelleisten oben rechts zu dieser Kategorie. Zu viel Eye Candy kann die Bedienung einer Anwendung sogar erschweren. Im Zusammenhang mit überflüssigen Augenschmeichlern in Bedienoberflächen wurde auch der abschätzige Begriff *Klickibunti* geprägt [KLI 13].

Die großzügige Verwendung rötlicher Farbtöne bei den Anwendungen *Access* und *PowerPoint* ist besonders problematisch, da die Farbe Rot eine alarmierende Signalfarbe darstellt. Durch die inflationäre Verwendung der Farbe Rot sticht sie nicht mehr an den Stellen ins Auge, wo sie tatsächlich ihre warnende Wirkung entfalten soll. [MOHR 11].[25]

Ein weiteres aktuelles Beispiel für einen zu sorglosen Umgang mit der Farbe Rot in Bedienoberflächen stellt der Webbrowser *Vivaldi*[26] dar (siehe Abbildung 2.31). Das Browser-Projekt wurde vom langjährigen *Opera*-Chef *Jon von Tetzchner* ins Leben gerufen, nachdem er bei Opera ausgeschieden war. Das Programm besitzt eine komplett rote Navigationsleiste und hebt aktive Tabs ebenfalls rot hervor. Auf der anderen Seite besitzt die Bedienoberfläche zu schwache Kontraste. Auf einem Bildschirm sind beispielsweise die Fensterbuttons oben rechts kaum vom Hintergrund zu unterscheiden. Die Bildlaufleiste ist auch kaum zu erkennen. Da das Browser-Fenster keinen deutlichen Rahmen besitzt, ist es vor einem hellen Hintergrund kaum zu erkennen. Vermutlich soll sich Vivaldi durch diese visuellen Attribute deutlich von der Konkurrenz abheben und das Design zur Marke erhoben werden. Allerdings leidet die Übersichtlichkeit darunter.

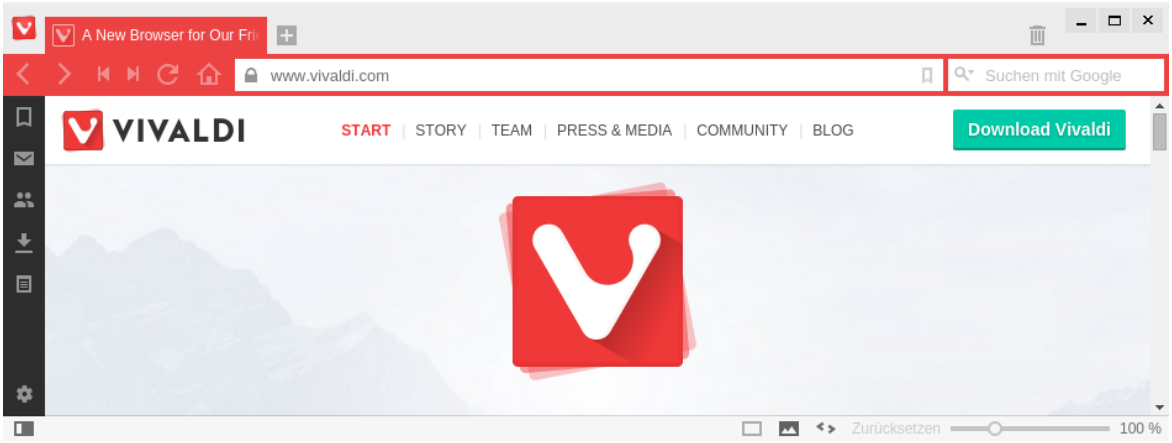


Abbildung 2.31.: Vorschauversion des Webbrowsers Vivaldi.

Zu Shneidermans Vorschlägen zählt unter anderem auch, dass Benutzeroberflächen zuerst für Monochrom-Displays entworfen werden sollen. An den gezielten Einsatz von Farben sollen Entwickler erst danach denken. Dieses Paradigma kommt Entwicklern in der heutigen Zeit antiquiert vor, da nahezu jeder Computer mit einem Farbbildschirm ausgestattet ist. Dennoch macht die Einhaltung dieses Paradigmas auch heute noch Sinn: Es hilft dabei, Farben möglichst sparsam einzusetzen. Eine inflationäre Verwendung von Farben verringert die Übersichtlichkeit und lenkt den Anwender von den wichtigen Informationen ab.

Shneidermans Anforderung an eine Bedienoberfläche, benutzerdefinierte Farben zu ermöglichen, wurde bereits in frühen Versionen von Windows erfüllt. Abbildung 2.32 zeigt das Dialogfenster von Microsoft Windows 3.1, das zu diesem Zweck diente (mit eingeklappter Farbpalette). Benutzerdefinierte Farbschemas wirken sich nur auf Anwendungen aus, die auf gewisse Standardbibliotheken von Windows zurückgreifen. Davon entkoppelte Bedienoberflächen ermöglichen Abweichungen von den Standardfarben nur in wenigen Fällen. Oft können Anwender die Farben dort auch nicht einzeln ändern, sondern müssen unter vorgefertigten *Skins* ein Design auswählen. Eine Ausnahme stellen moderne Webanwendungen dar: Die vor Jahren selbstverständliche

[25] Dass zu viel Rot nicht gut ist, musste auch Terence Hill in dem Film *Der Supercop* erfahren.
[26] Vivaldi-Website: `https://vivaldi.com/`

Möglichkeit, benutzerdefinierte Farben für Bedienelemente einzusetzen, ist in webbasierten Applikationen nahezu verlorengegangen. Der im Rahmen dieser Masterarbeit anzufertigende Prototyp fasst Shneidermans Forderung wieder auf und ermöglicht es dem Anwender, eine Akzentfarbe nach seinem Geschmack auszuwählen.

Abbildung 2.32.: Das vordefinierte Farbschema “Hotdog Stand” von Microsoft Windows 3.1.[27]

2.14. Interaktion

Ursprünglich befasste sich die Informationsvisualisierung mit der Entwicklung neuer visueller Darstellungsformen. Benutzerinteraktionen rücken aber immer mehr in den Vordergrund. Dabei kommt es auch Überschneidungen mit der Disziplin der *Human-Computer Interaction (HCI)*. Dies führt dazu, dass moderne Systeme der Informationsvisualisierung aus zwei Hauptbestandteilen zusammengesetzt sind:

Repräsentation hat ihre Ursprünge in der Computergrafik. Hier werden Daten auf geometrische Objekte gemappt und bestimmt, wie diese Objekte auf dem Ausgabegerät gezeichnet werden.

Interaktion basiert auf Entwicklungen der Mensch-Maschine-Interaktion. Über einen Anwendungsdialog beeinflusst der Benutzer die Repräsentationsform. Er fordert unterschiedliche Perspektiven und Ansichten an, um anhand der Visualisierung ein Problem beantworten zu können.

Die Interaktion ist zu einem wichtigen Bestandteil von Systemen der Informationsvisualisierung geworden. Ohne sie würden diese Systeme lediglich statische Bilder oder zumindest automatisch animierte Bildsequenzen generieren. Gerade bei umfangreichen Datensätzen sind statische Bilder im Nachteil.[28]

2.14.1. Definition

Der auf Mensch-Computer-Interaktion spezialisierte britische Universitätsprofessor *Alan Dix* definiert Interaktion in seinem Buch *Human-Computer Interaction* folgendermaßen [DIX 03, S. 124]:

> “[. . .] the communication between user and the system.”

[27] Bildquelle: `http://media.charlesleifer.com/blog/photos/hotdog-stand.png`

[28] Es ist allerdings nicht ausgeschlossen, dass ein Betrachter mit einem statischen Bild interagiert. Er kann zumindest das Papier, auf dem es sich befindet, aus beliebigen Perspektiven ansehen und Notizen darauf anfertigen.

Becker, *Cleveland* und *Wilks* veröffentlichten im Jahr 1987 ein Paper mit dem Titel *Dynamic Graphics for Data Analysis*. Über Interaktion im Sinn von dynamischen grafischen Methoden erwähnen sie dort die folgenden beiden essentiellen Eigenschaften [BECK+ 87]:

> "[...] direct manipulation and instantaneous change."

Eine kurze und knappe Auffassung von dynamischen Visualisierungen veröffentlichte *Colin Ware* [WARE 04, S. 385]:

> "[...] asymmetry in data rates."

Asymmetrische Datenraten kennt man in der Regel von seinem *Asymmetric Digital Subscriber Line (ADSL)*-Internetanschluss zu Hause. Obwohl Telekommunikationsnetze nicht das Thema von Informationsvisualisierung sind, meint Ware etwas ganz Ähnliches: Bei *asymmetrischem DSL* werden unterschiedliche Datentransferraten für Up- und Downstream verwendet. Wer im Internet surft, sendet beim Aufruf einer Webseite deutlich weniger Daten zum Server, als er von ihm wieder empfängt. Auch dynamische Visualisierungen erfordern vom Anwender deutlich weniger Daten als Eingabe, als sie selbst dem Anwender wieder zurückliefern. Ware bezeichnet das Team aus Mensch und Computer außerdem als "Problemlösungssystem".

Die Tatsache, dass interaktive Systeme der Informationsvisualisierung kein Selbstzweck sind, sondern zur Lösung von Problemen dienen, wird bei Ware besonders klar. Um dieses Ziel zu erreichen, muss der Rechner unmittelbar auf die Eingaben des Anwenders reagieren und die Visualisierung so auf den Bildschirm bringen, wie es sich der Anwender vorstellt. Dabei handelt es sich um eine Form der Kommunikation zwischen Mensch und Maschine: Der Mensch gibt dem Rechner klare Anweisungen und erhält im Gegenzug nicht nur hübsch anzusehende Bilder, sondern nützliche Informationen und Einsicht.

2.14.2. Direct Manipulation

In Zusammenhang mit interaktiven Systemen der Informationsvisualisierung prägte der US-amerikanische Informatiker Ben Shneiderman im Jahr 1982 den Begriff *Direct Manipulation* [SHN 82]. Folgende Merkmale muss ein System, das Direct Manipulation ermöglicht, erfüllen:

- Die für den Benutzer interessanten Objekte müssen angezeigt werden.
- Inkrementelle Aktionen des Anwenders führen zu jeweils unmittelbaren Rückmeldungen des Systems.
- Alle vorherigen Aktionen müssen umkehrbar sein, damit der Benutzer ungehemmt mit dem System experimentieren kann.
- Die syntaktische Korrektheit aller Benutzereingaben muss jederzeit gewährleistet sein. Besonders bei *Dynamic Queries*, die im Anschluss vorgestellt werden, spielt dies eine wichtige Rolle.
- Der Benutzer soll virtuelle Objekte in ähnlicher Weise wie reale Objekte manipulieren können. Reale-Welt-Metaphern sollen die Bedienung intuitiver machen.

Zu den Reale-Welt-Metaphern, die schon vor langer Zeit Einzug in die Bedienung von Computern gehalten haben, zählen virtuelle Schreibtische bzw. Desktops. Auf ihnen legt der Anwender Ordner und Dateien in Form kleiner Piktogramme ab und kann mit ihnen interagieren, indem er sie zum Beispiel durch Klickziehen mit der Maus in einen anderen Ordner verschiebt oder zum geplanten Löschen in einen stilisierten Papierkorb wirft.

Am PC nutzen Anwender auch Jahrzehnte nach ihrer Erfindung immer noch die Maus als primäres Eingabegerät für Bedienoberflächen mit Direct Manipulation. Bereits in einer frühen Studie erwies sie sich als sehr effizientes Eingabegerät zur Auswahl von Text auf dem Bildschirm [ENG+ 67]. Die Maus wurde Ende der 1960er Jahre vom Computer-Pionier *Douglas C. Engelbart* erstmals einem Publikum im Zusammenhang mit dem an der Stanford Universität entwickelten *oN-Line System* vorgestellt.[29]

[29] Das oN-Line System beherrschte bereits damals wegweisende Funktionen, die bei modernen Computern nicht mehr wegzudenken sind. Darunter zählen zum Beispiel effiziente Navigationsmenüs, Textbearbeitung, Hyperlinks, Videokonferenzen und die gemeinsame Bearbeitung von Dokumenten. Douglas C. Engelbarts ca. 100-minütige Demonstration wurde später als die "Mutter aller Demos" bezeichnet (Präsentation auf YouTube: `https://www.youtube.com/watch?v=yJDv-zdhzMY`).

[30] Bildquelle: `https://tctechcrunch2011.files.wordpress.com/2012/01/fujifilm.jpg`

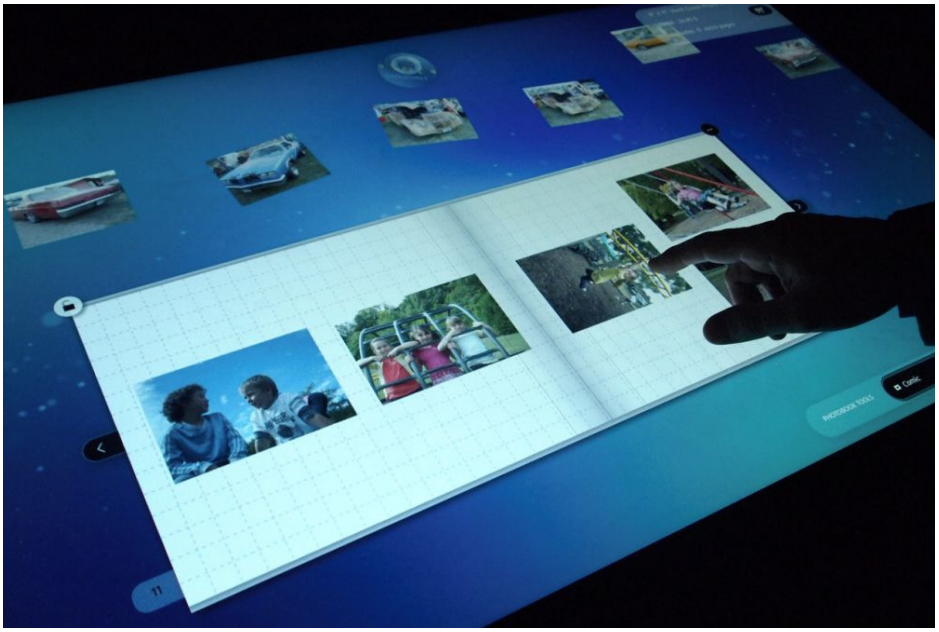


Abbildung 2.33.: Zusammenstellung eines Fotobuchs auf dem Touchscreen eines Samsung SUR40.[30]

Für eine direktere Interaktion mit dem Objekten auf dem Bildschirm sind Touchscreens erforderlich oder andere Geräte, die die Gesten des Anwenders erfassen. Durch den Abbau technischer Hürden zwischen Mensch und Computer wie zum Beispiel durch den Entfall von Zeige- und Eingabegeräten wird der Computer immer mehr zu einem Alltagsgegenstand ohne nennenswerte Einstiegshürden. Ein möglicher Anwendungsfall für Touchscreens mit Direct Manipulation und Reale-Welt-Metaphern sind Selbstbedienungsterminals in Fotogeschäften, die dem Kunden die intuitive Zusammenstellung eigener Fotobücher erlauben. Abbildung 2.33 zeigt einen *Samsung SUR40*, der hier zu diesem Zweck dient.

Metaphern aus der realen Welt werden bei grafischen Bedienoberflächen gerne mit der Absicht eingebaut, eine möglichst intuitive Handhabung auch für Laien zu ermöglichen. Das ist auch gerechtfertigt, solange es der Produktivität des Anwenders dienlich ist. Im Praxisteil dieser Arbeit wird hierauf besonderer Wert gelegt. Die Bedienung soll nicht erst die Lektüre eines dicken Handbuchs erfordern, jedoch bleibt zu berücksichtigen, dass die Zielgruppe erfahrene Systemadministratoren sind, die gewisse Funktionen von der Anwendung erwarten.

Dass das Bestreben, Bedienoberflächen möglichst zu vereinfachen, auch nach hinten losgehen kann, zeigt die für Linux verfügbare Desktop-Umgebung *Gnome*. Der Gnome-Desktop war zwar von Anfang an auf Minimalismus ausgelegt, doch mit dem Release der Version 3 trieben es die Entwickler auf die Spitze: Der Desktop wurde so weit abgespeckt und funktional zurückentwickelt, dass sich selbst *Linus Torvalds* – der Erfinder des Linux-Kernels – zu Wort meldete [THO 11]. Wie es bei Open-Source-Benutzergruppen, die mit der Entwicklung eines Projekts unzufrieden sind, die Regel ist, gründeten sie schließlich einen Fork der Vorgängerversion von Gnome und nannten ihn *Mate*[32]. Populär wurde Mate durch die auf Debian basierende Linux-Distribution *Mint*[33]. Auch von Gnome 3 wurde ein Fork namens *Cinnamon*[34] angelegt, dessen Oberfläche sich ebenfalls mehr am Vorgänger orientiert.

In Kontext von Reale-Welt-Metaphern spricht man auch von sog. *Skeuomorphismus*. Dabei handelt es sich um eine Design-Stilrichtung, die mit Hilfe neuer Technik (oder neuer Materialien) ältere Gegenstände oder Geräte aus der “analogen Welt” nachahmt, ohne dass hierfür eine technische Notwendigkeit besteht. Ein Beispiel hierfür wäre zum Beispiel eine Smartphone-Applikation, die die Wählscheibe eines antiquierten Impulswahltelefons imitiert.

[31] Bildquelle: `http://www.peacockworks.com/wp-content/uploads/2012/05/skeuomorphism.png`
[32] Mate-Website: `http://mate-desktop.org/de/`
[33] Linux-Mint-Website: `http://www.linuxmint.com/`
[34] Cinnamon-Website: `http://cinnamon.linuxmint.com/`

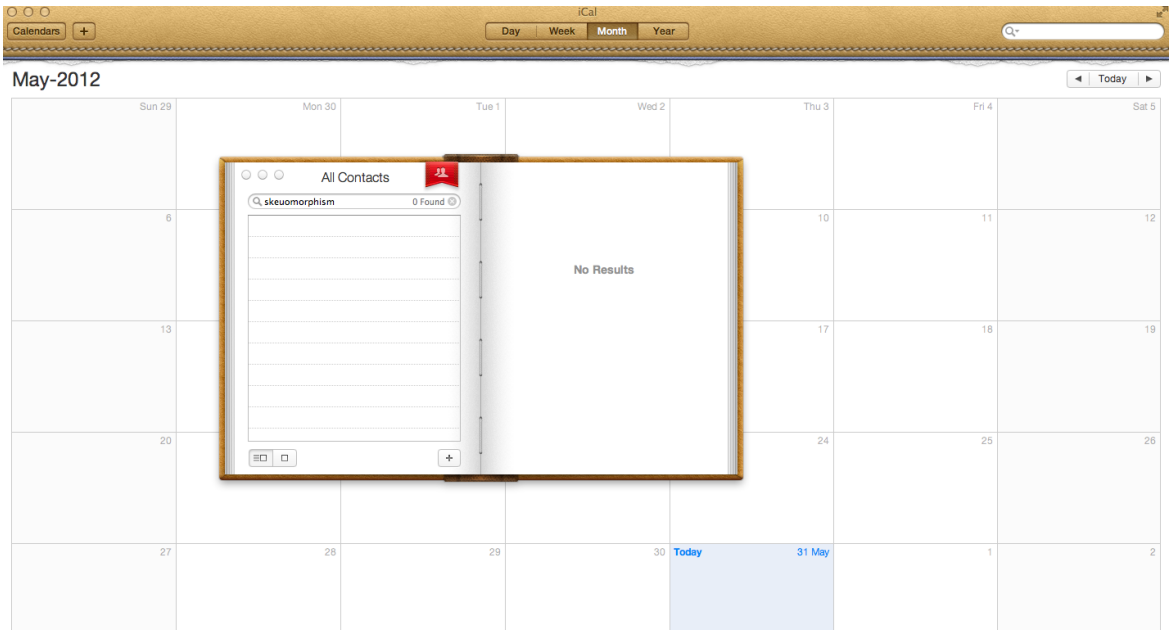


Abbildung 2.34.: Einsatz von Skeuomorphismus bei Apples Kalender und Adressbuch in OS X Lion. Kennzeichnend hierfür sind virtuelles Leder mit einer Ziernaht, die Illusion von abgerissenem Papier und das imitierte Hardcover-Buch mit Lesezeichen.[31]

Auch der *Apple*-Mitgründer *Steve Jobs* schätzte Skeuomorphismen so sehr, dass sie schließlich Einzug in zahlreiche Programme von Apple gehalten haben. Als Beispiel sei hier die allseits bekannte Kalender-App mit dem virtuellen Leder-Einband genannt (siehe Abbildung 2.34). Während die Leder-Textur der Anwendung hier nur zu einem hübschen Gesamteindruck und zur Bildung einer Marke verhilft, wirken Skeuomorphismen wie die virtuelle Wählscheibe im Alltag eher kontraproduktiv, da sie die Bedienung des Geräts unnötig verkomplizieren. Obwohl die Eingabe einer Telefonnummer mit einer Zehnertastatur auf dem Bildschirm deutlich schneller vonstatten geht, gibt es tatsächlich Anwender, die die Illusion einer guten alten Wählscheibe bevorzugen. Solange der Anwender die Wahl hat, ist dagegen nichts einzuwenden. Skeuomorphismen sind Geschmackssache und müssen vor allem bei grafischen Bedienoberflächen sorgsam abgewogen werden, damit kein Nachteil für den Anwender entsteht. Mit der Ankündigung des mobilen Betriebssystems *iOS 7* schlug Apple schließlich einen Weg zu einem vereinfachteren Oberflächen-Design ein [EVA 13].

Durch Direct Manipulation lassen sich Bedienkonzepte realisieren, die für den Benutzer einfach und intuitiv wirken. An ihre Grenzen stößt diese Technik allerdings, wenn der Anwender mit Objekten interagieren will, die besonders klein und/oder detailreich sind oder wenn es sich um eine große Anzahl von Objekten handelt, mit denen er gleichzeitig interagieren will. Die direkte Manipulation von dreidimensionalen Objekten auf einem zweidimensionalen Bildschirm ist auch ein kniffliges Unterfangen. Einen Lösungsansatz hierfür liefern sog. *Stellvertreterobjekte*, die der Benutzer manipuliert, um das oder die repräsentierten Objekte indirekt zu beeinflussen [KWON+ 11].

Stellvertreterobjekte sind oft abstrakte Repräsentanten, die einfach mit der Maus bearbeitet werden können. Sie werden beispielsweise in den Werkzeugfenstern zahlreicher Zeichenprogrammen verwendet, um anhand einer dort abgebildeten einfachen geometrischen Figur wie einem Kreis oder einem Rechteck die Attribute eines beliebig komplexen Zeichenobjekts, das zuvor mit dem Zeigegerät ausgewählt wurde, zu manipulieren.

Auch in Computerspielen finden Stellvertreterobjekte Verwendung: Rollenspiele mit 3D-Grafik ermöglichen dem Anwender oft die Ausstattung des Protagonisten mit Kleidungsstücken, Ringen, Amuletten, Waffen und anderen nützlichen Gegenständen. Das Hinzufügen und Entfernen dieser Gegenstände zur bzw. von der Spielfigur geschieht oft anhand einer zweidimensionalen Abstraktion der Figur, durch einfache Platzhalter oder durch das Selektieren bzw. Deselektieren von Items in einer einfachen Liste. Dieser Vorgang geschieht in der Regel in einer separaten Inventaransicht und nicht direkt im Spiel an der 3D-Figur selbst.

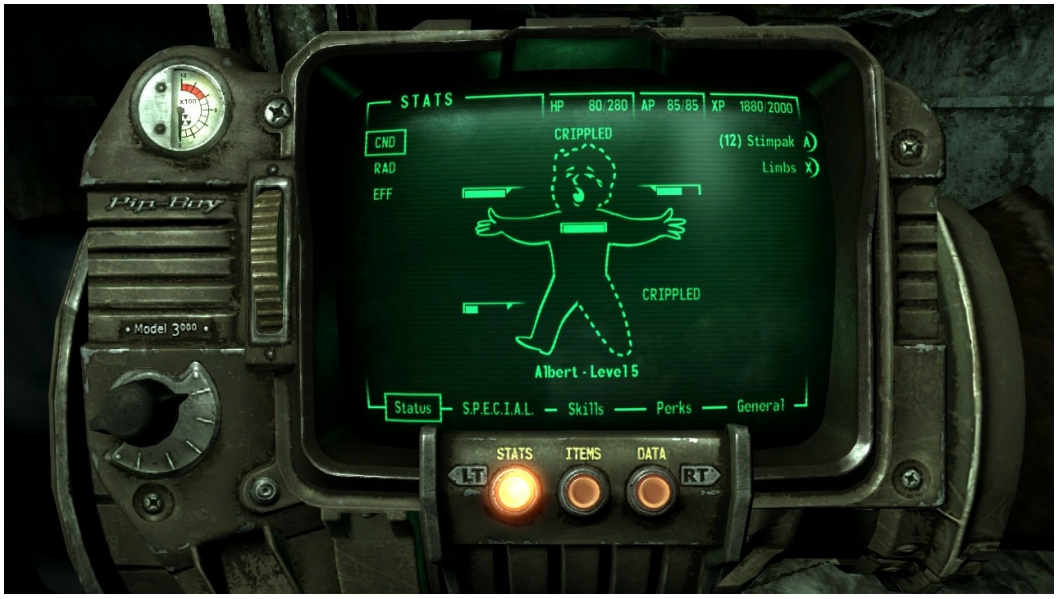


Abbildung 2.35.: Einsatz von Stellvertreterobjekten auf dem “Pip-Boy” im Computerspiel “Fallout 3”.[35]

Ein Beispiel für Stellvertreterobjekte in Computerspielen ist in Abbildung 2.35 zu sehen. Sie zeigt einen Screenshot aus dem 3D-Rollenspiel *Fallout 3*. Durch Gewalteinwirkungen in der postapokalyptischen Welt kann die Spielfigur Verletzungen erleiden. Die Heilung einzelner Körperteile des Protagonisten erfolgt anhand einer abstrahierten Figur (dem “Vault Boy”) auf der Bedienoberfläche des sog. *Pip-Boy*. Anstelle auf separaten Vollbildübersichten werden der Gesundheitszustand, das Inventar, Quests und andere Charaktereigenschaften auf dem Bildschirm eines Geräts eingeblendet, das die Spielfigur ähnlich wie eine Armbanduhr am linken Arm trägt und auf Anforderung durch den Spieler in die Kamera hält.[36] Damit wird der Pip-Boy selbst auch zu einem Stellvertreterobjekt.

2.14.3. Dynamic Queries

Dynamic Queries basieren auf Direct Manipulation Interfaces (DMIs). Sie ermöglichen dem Anwender, auf intuitive Art und Weise Abfragen zu formulieren und unmittelbar eine visuelle Rückmeldung zu erhalten. Das System verlangt zu diesem Zweck keine Eingaben in einer formalen Abfragesprache, die der Benutzer erst erlernen muss, sondern ermöglicht Abfragen über eine überschaubare Menge von grafischen Bedienelementen. Für alle Komponenten der Abfrage müssen demnach entsprechende visuelle Repräsentationen, wie zum Beispiel Slider, Schaltflächen oder Checkboxen, vorhanden sein. Das Zusammenstellen und Ändern von Abfragen soll durch Dynamic Queries besonders schnell von der Hand gehen und unmittelbare Ergebnisse liefern sowie eine hohe Bedienfreundlichkeit gewährleisten.

Eine in der Fachliteratur häufig erwähnte Beispielanwendung, die die Funktionsweise dynamischer Abfragen demonstrieren soll, ist der *Dynamic HomeFinder*. Das Programm wirkte damals revolutionär und zählt zu den ersten mit dynamischen Abfragen. Es wurde von *Christopher Williamson* und *Ben Shneiderman* vorgestellt [WIL+ 92]. Der Dynamic HomeFinder zeigt eine Karte von Washington, D.C. Farbige Punkte kennzeichnen Wohnungen, die zum Verkauf stehen. Auf der Suche nach einer passenden Immobilie kann der Anwender verschiedene Vorgaben über grafische Bedienelemente festlegen. Dazu zählen zum Beispiel die Anzahl der Zimmer, Kosten oder ob eine Garage vorhanden ist. Änderungen durch den Benutzer wirken sich unmittelbar auf die Menge der angezeigten Häuser aus. Testpersonen reagierten äußerst positiv auf dieses neuartige

[35] Bildquelle: `http://static.giantbomb.com/uploads/original/1/17020/545887-pipboy.jpg`

[36] Es gibt auch Computerspiele, bei denen die Heilung von Körperteilen tatsächlich per Direct Manipulation geschieht, indem der Spieler zum Beispiel eine Spritze geschickt im Raum auf den Patienten zusteuert. Ein Beispiel hierfür ist der *Surgeon Simulator* (`http://www.surgeonsim.com/`).

Bedienkonzept[37]. Abbildung 2.36 zeigt die Anwendung in Aktion.

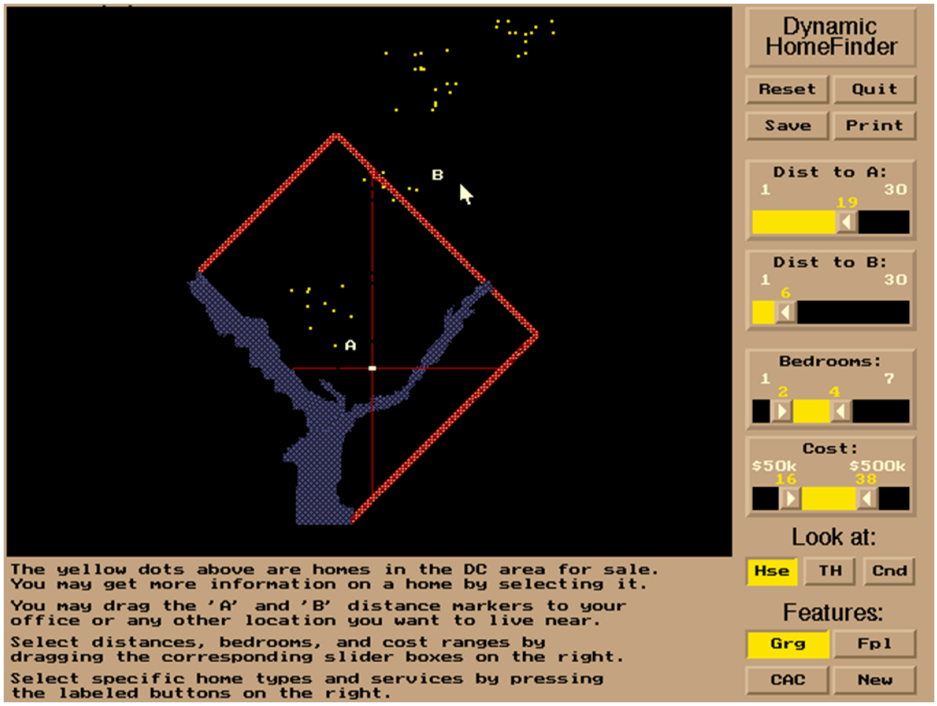


Abbildung 2.36.: Der Dynamic HomeFinder zeigt eine schematische Karte von D. C. und ermöglicht die Suche nach Immobilien anhand von Kriterien, die der Anwender am rechten Bildschirmrand definiert. Die Menge der eingeblendeten Suchergebnisse ändert sich unmittelbar nach der Änderung eines Kriteriums.[38]

2.15. Zeitreihen

Edward Tufte untersuchte von 1974 bis 1980 rund 4000 Schaubilder in den damals populären Zeitungen und Magazinen und kam zu der Erkenntnis, dass rund drei Viertel *zeitlich veränderliche Daten* bzw. *Zeitreihen* darstellen [TUFT 01, S. 28]. Ein Grund hierfür ist vermutlich, dass die Leser an Vergleichen aktueller Kennzahlen mit vorherigen Werten interessiert sind, um ihre jetzige Situation besser beurteilen zu können. Aus Zeitreihen können aber auch Vorhersagen über zukünftige Ereignisse abgeleitet werden.

Eine Zeitreihe besteht aus einer Sequenz diskreter Messungen, die zeitlich nacheinander angefertigt wurden. Allgemein lässt sich eine Zeitreihe wie folgt ausdrücken:

$$D = \{(t_1, y_1), (t_2, y_2), \ldots, (t_n, y_n)\} \tag{2.6}$$

wobei für alle $1 \leq i \leq n$ gilt: $y_i = f(t_i)$. Die Funktion f bildet dabei Zeitschritte auf Daten eines bestimmten Typs ab. Im einfachsten Fall sind dies quantitative Werte. Beispiele hierfür sind Aktienkurse oder ein Elektrokardiogramm. Neben quantitativen Werten kann die Zielmenge der Funktion auch unter anderem folgende Werte enthalten:

- ordinale
- kategorische
- relationale

[37] Quelle: `https://www.youtube.com/watch?v=VuYUG1eDKtg`
[38] Bildquelle: `http://www.aviz.fr/wiki/uploads/Research/homefinder.png`

- textuelle
- multivariate
- Kombinationen davon

Bei der Art der Zielmenge von f sind kaum Grenzen gesetzt. Es stellt eine Herausforderung dar, für solche zeitveränderlichen Daten passende Visualisierungsformen zu finden. Eine gemeinsame Anforderung an alle dieser Formen ist, dass die zeitliche Reihenfolge der zugrundeliegenden Daten auch aus der Darstellung hervorgeht.

Je nach Anwendungsfall muss auch entschieden werden, ob in der Visualisierung *Zeitpunkte* oder *Zeitintervalle* ausgedrückt werden sollen. Dabei handelt es sich um *temporale Primitiven*. Im Gegensatz zu Zeitpunkten besitzen Zeitintervalle eine Ausdehnung, die entweder durch zwei Zeitpunkte oder einen Zeitpunkt und eine Dauer festgelegt wird. Abhängig von dieser Entscheidung können unterschiedliche Analyseverfahren auf die Darstellung angewendet werden. Beim Vergleich zweier Zeitpunkte x und y sind nur zwei Vergleichsfunktionen möglich:

equals(x, y) Die beiden Zeitpunkte sind gleich.

before(x, y) Ein Zeitpunkt ist vor einem anderen.

Beim Vergleich zweier Zeitintervalle können deutlich mehr Aussagen getroffen werden. Auch hier wird zur Veranschaulichung von zwei Zeitpunkten x und y ausgegangen:

equals(x, y) Die Intervalle starten und enden jeweils zum gleichen Zeitpunkt.

meets(x, y) Der Endzeitpunkt eines Intervalls ist der Startzeitpunkt des anderen.

before(x, y) Zwischen zwei Intervallen ist ein weiteres Intervall.

during(x, y) Ein Intervall ist in einem anderen enthalten.

overlaps(x, y) Es gibt ein Intervall, das eine Schnittmenge zweier anderer Intervalle ist.

starts(x, y) Zwei Intervalle starten zum selben Zeitpunkt.

finishes(x, y) Zwei Intervalle enden zum selben Zeitpunkt.

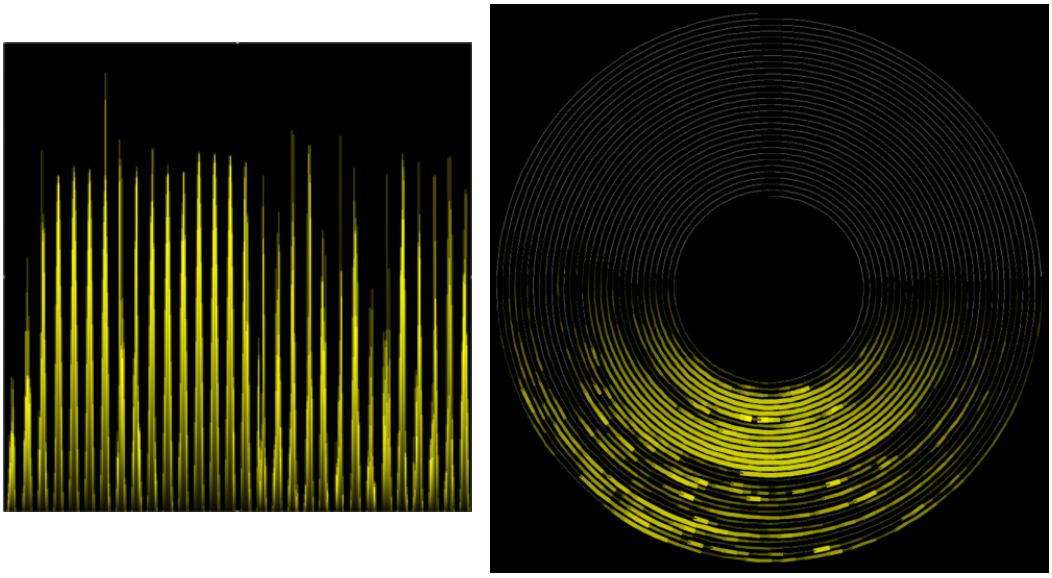


Abbildung 2.37.: Zwei etwa gleich große Visualisierungen der Sonnenintensität mehrerer Tage. Je heller die Spur ist, desto stärker ist die Einstrahlung. Die Spiraldarstellung auf der rechten Seite ähnelt stark einer Schallplatte. In ihr sind Tage besser miteinander zu vergleichen und sowohl Sonnenauf- als auch Untergang gut zu erkennen.

Bei der Betrachtung von Zeitreihen werden auch die möglichen Zeitachsen unterschieden. *Lineare Zeitachsen* kommen der natürlichen Wahrnehmung von Zeit am nächsten. Zeit wird hier als eine geordnete Menge von temporalen Primitiven angesehen. Lineare Zeitachsen eignen sich gut, um anhaltende Trends festzustellen. Dazu gesellen sich *zyklische Zeitachsen*, die sich aus einer endlichen Menge wiederkehrender temporaler Primitiven zusammensetzen. Die Tage einer Woche im *gregorianischen Kalender* sind ein Beispiel hierfür. Saisonale Effekte lassen sich gut aus zyklischen Zeitachsen ablesen. In der Praxis werden zur Verbesserung der Handhabung zyklische Zeitachsen beispielsweise zu linearen Exemplaren "ausgerollt". Eine Möglichkeit zur Unterbringung mehrerer Zyklen in einer einzigen Grafik stellten *Marc Weber et al.* im Jahr 2001 vor [WEB+ 01]. Sie nutzten eine spiralförmige Visualisierungsform, wie sie in Abbildung 2.37 zu sehen ist.

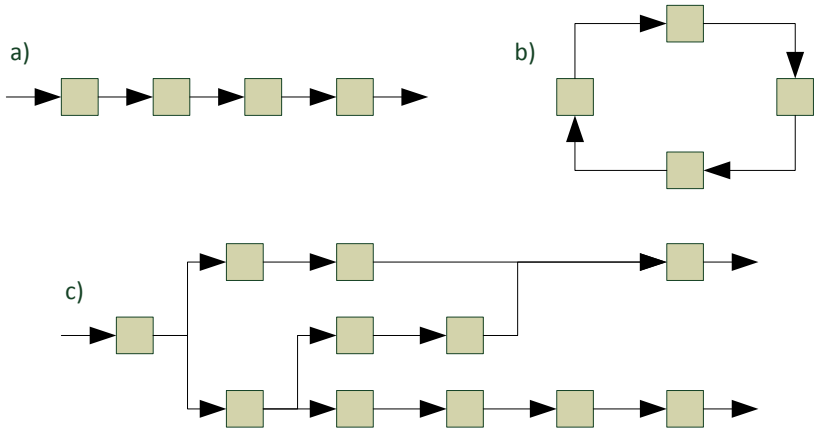


Abbildung 2.38.: Beispiele für lineare (a), zyklische (b) und verzweigte (c) Zeitachsen.

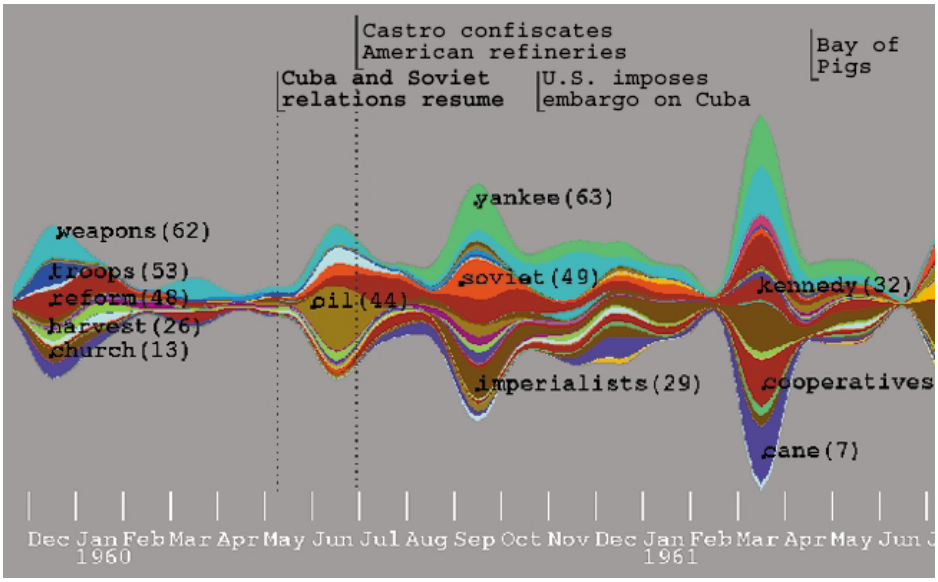


Abbildung 2.39.: Inhaltliche Schwerpunkte von Reden Fidel Castros ab dem Jahr 1960 stellt der ThemeRiver mit Hilfe der Metapher farbiger Flüsse dar, die unterschiedliche Themen repräsentieren. Je breiter ein Fluss ist, desto häufiger kommt das Thema vor.

Verzweigte Zeitachsen werden als Graphen modelliert, deren Kanten die temporalen Primitiven repräsentieren. Gerichtete Kanten kennzeichnen die zeitliche Reihenfolge. Knoten mit einem Ausgangsgrad größer als eins teilen die Zeitachse auf und repräsentieren so mehrere Handlungsalternativen oder mögliche Szenarien. Abbildung 2.38 veranschaulicht die drei zuvor genannten Fälle.

Unabhängig davon, ob man sich bei zeitlich veränderlichen Daten zweidimensionaler oder dreidimensionaler

Darstellungsformen bedient, gibt es einen reichen Fundus an Möglichkeiten, von denen an dieser Stelle nur ein Bruchteil vorgestellt wird. *Susan Havre et al.* stellten im Jahr 2002 den *ThemeRiver* vor [HAV+ 00]. Basierend auf einer großen Dokumentensammlung lassen sich damit die Änderungen thematischer Schwerpunkte durch *Fluss-Metaphern* im Laufe der Zeit visualisieren (siehe Abbildung 2.39).

Eine vergleichbare Visualisierungsform setzt der *NameVoyager*[39] von *CMI Marketing, Inc.* ein. Er gibt Aufschluss über die Popularität von Namen für Neugeborene in den USA seit dem Jahr 1880 (siehe Abbildung 2.40).

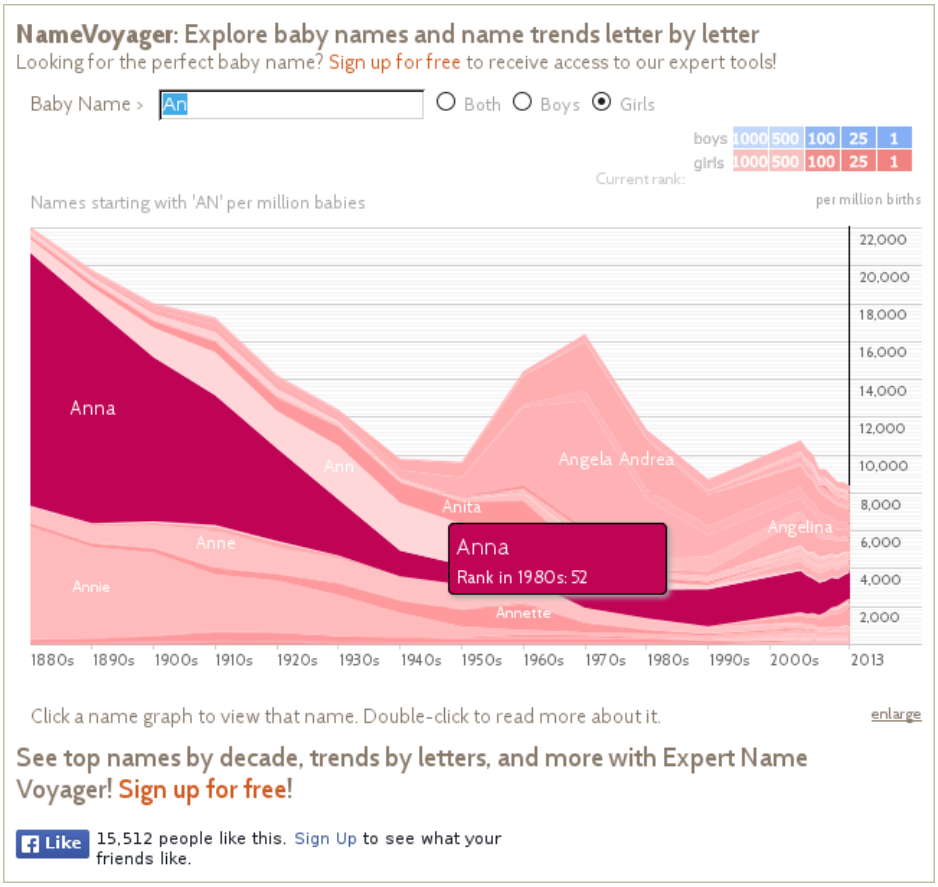


Abbildung 2.40.: Sich ändernde Trends für Babynamen visualisiert der NameVoyager durch Fluss-Metaphern. Die Grafik wird bei jedem Tastendruck im Eingabefeld neu generiert. Beim Überfahren eines Namens mit dem Mauszeiger wird ein Ranking des Namens im jeweiligen Jahrzehnt angezeigt.

Eine für damalige Verhältnisse innovative Art der Dokumentenverwaltung auf dem Computer stellten *Eric Freeman* und *Scott Fertig* im Jahr 1995 mit *Lifestreams* vor [FRE+ 95]. Dokumente werden dort als zeitorientierte, stromförmige Metaphern dargestellt. Bei Änderungen werden alte Versionen von Dokumenten automatisch archiviert. Damit können Anwender sämtliche Änderungen zurückverfolgen. Zu alten Versionen kann der Benutzer sprichwörtlich zurückblättern. Die Notwendigkeit, sich Dateinamen zu merken, soll durch diesen Ansatz ebenfalls entfallen.

Ein Screenshot aus dem Paper ist in Abbildung 2.41 dargestellt. Mehr als zehn Jahre später griff *Apple* dieses Paradigma erneut auf und integrierte es in sein Datensicherungsprogramm *Time Machine*[40], das seit *Mac OS X Leopard* Teil des Betriebssystems ist.

[39] NameVoyager-Website: `http://www.babynamewizard.com/voyager`
[40] Time-Machine-Website: `http://www.apple.com/de/osx/apps/#timemachine`

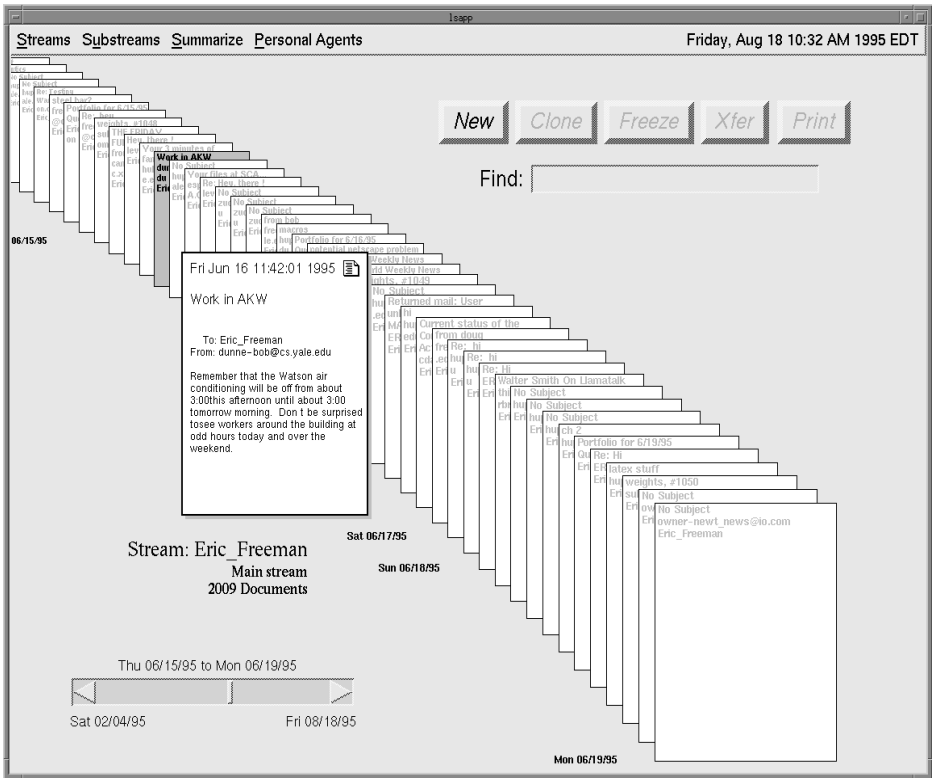


Abbildung 2.41.: Lifestreams war als Alternatives Bedienkonzept zur Schreibtisch-Metapher gedacht. Es unterstützt einheitliche Zugriffsmöglichkeiten über Rechnergrenzen hinweg. Änderungen an Dokumenten führen automatisch zu einer neuen Version. Alte Dateiversionen bleiben erhalten und sind durchsuchbar.

2.16. Text

Die bisher vorgestellten Techniken der Informationsvisualisierung hatten gemeinsam, dass sie Daten aus speziellen Strukturen wie zum Beispiel relationalen Datensätzen oder Graphen veranschaulichen sollen. Dieses Kapitel befasst sich mit den Möglichkeiten der Informationsvisualisierung, Eigenschaften von weniger strukturierten Daten, nämlich Texten, grafisch darzustellen.

2.16.1. Charakteristik von Text

Texte sind ein wichtiger Informationsträger der menschlichen Kommunikation. Sie kommen in allen erdenklichen Medien wie zum Beispiel in Büchern, Zeitschriften und auf Webseiten vor. Um Texte zu visualisieren, muss zuerst untersucht werden, was es dort überhaupt zu visualisieren gibt. Neben dem eigentlichen Text kommen unter anderem folgende Informationen in Frage:

- Metadaten (Autor, Erscheinungsdatum etc.)
- Statistiken (z. B. Worthäufigkeiten)
- Relevanz von Dokumenten bzgl. einer Suchanfrage
- Verhältnis zu anderen Dokumenten (Textverweise, Ähnlichkeit etc.)
- Textstruktur

2.16.2. Programmcode-Statistiken

Stephen G. Eick et al. stellten im Jahr 1993 *SeeSoft* vor, ein Tool zur Visualisierung von Programmcodestatistiken [EICK+ 92]. Das Tool informiert den Programmierer unter anderem über den Urheber bestimmter Codezeilen, wie alt bestimmte Codezeilen innerhalb eines Listings sind und wie oft eine Zeile bei der Ausführung des Programms durchlaufen wurde. Es stellt mehrere Listings stark verkleinert in Spaltenform nebeneinander dar, sodass bis zu 50.000 Zeilen auf eine Bildschirmseite passen. Der Programmcode ist so zwar nicht mehr lesbar, der Anwender hat aber die Möglichkeit, für ihn interessante Codezeilen in einem separaten Fenster in lesbarer Größe abzurufen (siehe Abbildung 2.42). Die Bewertung der Relevanz von Codeabschnitten erfolgt anhand einer Farbskala.

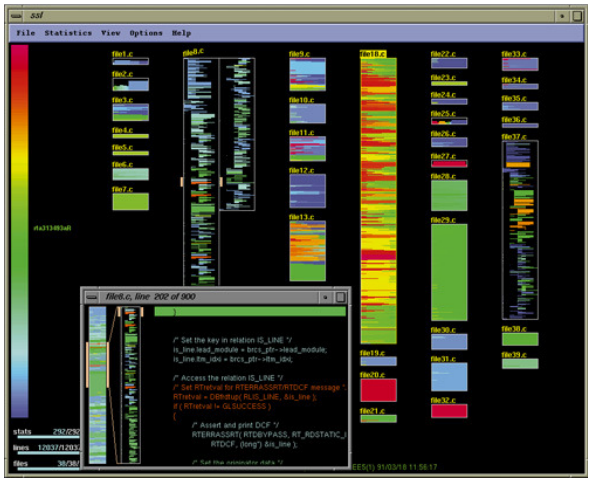


Abbildung 2.42.: Neue Programmzeilen kennzeichnet SeeSoft in der Farbe Rot, alte in Blau. Das eingeblendete Fenster zeigt eine Detailansicht der ausgewählten Stelle im Programmcode.[41]

2.16.3. Textstrukturen

Viele Dokumente, die in Textform vorliegen, enthalten Muster, die sich wiederholen. Es überrascht zwar nicht, dass in einem deutschen oder englischen Dokument die lateinischen Buchstaben immer wieder auftauchen, viel interessanter ist es jedoch, Wiederholungen von komplexen Mustern wie Wörtern oder ganzen Zeilen zu finden. Eine dazu passende Visualisierungsmethode hat *Martin Wattenberg* im Jahr 2002 der Öffentlichkeit vorgestellt [WAT 02]. Seine sog. *Arc-Diagramme* eignen sich, um wiederkehrende Muster in Texten grafisch sichtbar zu machen. Strukturen innerhalb der betrachteten Werke können so besser erkannt werden als es beim Lesen der Fall wäre.

Ein Arc-Diagramm basiert auf einem String, der in einer einzigen Zeile ausgegeben wird. Zwei Teilstrings, die identisch sind und direkt aufeinander folgen, werden von einem halbkreisförmigen Bogen überspannt. Je breiter ein Bogen ist, desto länger ist auch das wiederholt vorkommende Muster, das gefunden wurde. Je größer ein Bogen ist, desto größer ist auch die Entfernung der beiden identischen Muster im kompletten String. Um Visual Clutter zu verringern, wird oft nur eine Teilmenge aller vorhandenen redundanten Muster eingeblendet, zum Beispiel nur diejenigen Muster, die eine bestimmte Länge besitzen. Je länger die Teilstrings sind, desto unwahrscheinlicher sind Übereinstimmungen und desto übersichtlicher wirkt die Visualisierung. Die Wahlmöglichkeit der Musterlänge ist sinnvoll, da es für den Betrachter in der Regel nur in Sonderfällen von Interesse ist, welcher einzelne Buchstabe mehrfach vorkommt. Um die Halbkreisbögen besser voneinander unterscheidbar zu machen, eignet sich eine durchsichtige Hintergrundfarbe.

Wiederkehrende Muster können mit Hilfe von Arc-Diagrammen nicht nur in Prosatexten, sondern auch beispielsweise in Programmcode oder dem Markup Code von Webseiten gefunden werden. Auf Gensequenzen in

[41] Bildquelle: `http://www.cs.umd.edu/class/spring2001/cmsc838b/Project/Parija_Spacco/old_images/seesoft2.jpg`
[42] Bildquelle: `http://www.turbulence.org/Works/song/gallery/gallery.html`

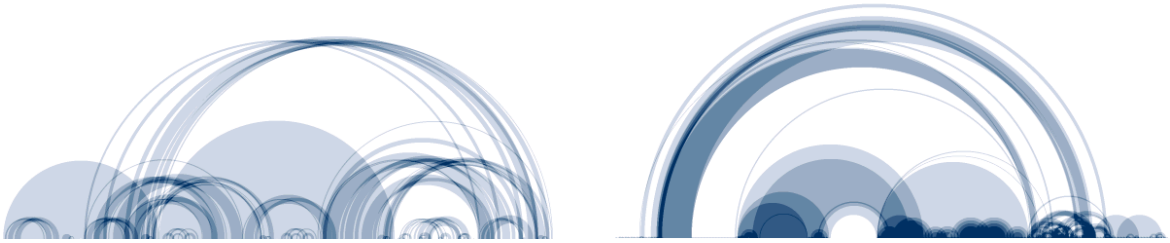


Abbildung 2.43.: Arc-Diagramme von Frédéric Chopins “Mazurka in fis-Moll” (links) und Madonnas “Like A Prayer” (rechts). Man kann bereits an der Farbdichte erkennen, wie sich der prinzipielle Aufbau moderner Popmusik von Klaviermusik des 19. Jahrhunderts unterscheidet.[42]

der Desoxyribonukleinsäure (DNS) lässt sich das Verfahren in begrenztem Maß auch anwenden. Ein weiterer interessanter Einsatzzweck ist Musik. Dort sucht der Arc-Algorithmus nicht nach Substrings, sondern nach identischen Folgen von Musiknoten. Abbildung 2.43 zeigt hierfür ein Beispiel.

2.17. Präsentation

Bei den Methoden der Informationsvisualisierung, die bisher in dieser Arbeit vorgestellt wurden, war die Rede davon, dass sie idealerweise auf einem Computerbildschirm angezeigt werden, damit der Anwender mit ihnen interagieren kann. Da Computer heutzutage in den unterschiedlichsten Größenordnungen existieren, müssen auch Systeme der Informationsvisualisierung auf unterschiedlich große Bildschirme vorbereitet sein. Besonders kleine Bildschirme stellen ein Problem dar, da sie wenig Platz bieten. Dieser Abschnitt befasst sich mit möglichen Lösungsansätzen für das sog. *Präsentationsproblem*.

2.17.1. Das Präsentationsproblem

Nicht selten wird man mit dem Problem konfrontiert, dass es deutlich mehr Informationen darzustellen gibt, als auf den Bildschirm passen. Einerseits liegt dies an den Datenbergen, die ohnehin schon unüberschaubar sind, andererseits daran, dass ein Display für den Anwender nie groß genug sein kann. Smartphones sind heutzutage flächendeckend verbreitet, durch ihre relativ kleinen Bildschirme bleiben sie aber weiterhin Mäusekinos. Die zunehmend hohen Pixeldichten ihrer Displays ändern auch wenig an dieser Tatsache, da man die Bildschirminhalte nicht beliebig verkleinern kann, ohne ein Vergrößerungsglas vorauszusetzen. Extrem hohe *Pixels per Inch (PPI)*-Werte wie etwa bei Apples *Retina Display*[43] oder bei Samsungs GALAXY TabPRO[44] dienen dort in erster Linie der besonders scharfen Darstellung von Text in normaler Größe und natürlich auch dem Marketing.

Um sicherzustellen, dass auch Anwender von Geräten mit besonders kleinen Bildschirmen Systeme der Informationsvisualisierung nutzen können, gibt es folgende Möglichkeiten:

- Platzsparende Kodierung der Daten (siehe vorherige Abschnitte).
- Dem Betrachter eine interaktive Anpassung der Darstellung ermöglichen.

Scrollbare Oberflächen werden schon lange eingesetzt, um dem Anwender eine interaktive Anpassung der Darstellung zu ermöglichen. Sie eignen sich gut, um Informationen auf begrenztem Raum unterzubringen. Entwickler von Bedienoberflächen können davon ausgehen, dass praktisch jeder Benutzer damit schon einmal in Kontakt gekommen ist. Dadurch sind die Einstiegshürden sehr gering. Je größer aber das Verhältnis zwischen den Ausmaßen des kompletten Inhalts und des Scrollfensters ist, desto mehr leidet die Bedienfreundlichkeit darunter, da die Zeit zum Suchen bestimmter Bildinhalte steigt. Wenn der Anwender den Ausschnitt

[43] Quelle: `http://de.wikipedia.org/wiki/Retina-Display`
[44] Quelle: `http://www.samsung.com/de/consumer/mobile-device/tablets/tablets/SM-T520NZWADBT`

einer Visualisierung zum ersten Mal sieht, hat er auch keine Vorstellung davon, wie der komplette Inhalt aussieht, ohne sie zuvor komplett durchgescrollt zu haben. Des Weiteren erlauben traditionelle Scrollbars auch nur Bewegungen in X- oder Y-Richtung und nehmen permanent – wenn auch nur wenig – Platz auf dem Bildschirm weg. Eine Lösung für das zuletzt genannte Problem sind Bildlaufleisten, die sich bei Nichtbenutzung automatisch ausblenden.

2.17.2. Zoombare Benutzeroberflächen

Zoombare Benutzeroberflächen werden auch als *Multiscale Interfaces* bezeichnet. Sie zeichnen sich dadurch aus, dass Datenobjekte im Raum angeordnet sind und unterschiedliche Skalierungen unterstützen. Es liegt in der Hand des Anwenders, welche der Objekte angezeigt werden sollen und wie groß der gewünschte Detailgrad ist. Dabei bedienen sich Anwender der folgenden beiden grundlegenden Operationen:

Panning Bewegung des Sichtfensters im Raum unter Beibehaltung des Vergrößerungsgrads.

Zooming Änderung des Vergrößerungsgrads des Sichtfensters mit Auswirkung auf die Anzahl und Größe der angezeigten Objekte.

George W. Furnas und *Benjamin B. Bederson* schrieben über die Motivation zoombarer Bedienoberflächen folgende Zeilen [FUR+ 95]:

> "[...] by moving through space and changing scale the users can get an integrated notion of a very large structure and its contents, navigating through it in ways effective for their tasks."

Zoombare Oberflächen eignen sich demnach hervorragend, um einerseits die Größe komplexer Datenstrukturen zu vermitteln und andererseits durch die o. g. Methoden mit den dort enthaltenen Objekten zu arbeiten.

Bereits Ende der 1970er Jahre wurde am *Massachusetts Institute of Technology (MIT)* ein zukunftsweisender Testaufbau zur zoombaren Visualisierung von Daten entwickelt [DON 78]. Das *Spatial Data Management System* wurde von einer Person mit Hilfe von berührungsempfindlichen Bildschirmen und zwei Joysticks bedient. Der rechte Joystick diente zum Panning und der linke zum Zooming. Zusätzlich war ein sog. *Tablet* vorhanden. Dabei handelte es sich um ein Eingabegerät, das einem digitalen Zeichenbrett ähnelt. Der große Hauptpräsentationsbereich an der Wand wurde durch Rückprojektion realisiert, damit Personen oder Gegenstände davor keinen Schatten werfen. Insgesamt acht Lautsprecher wurden in den Ecken des Versuchsraums angebracht, um Raumklang bereitzustellen[45]. Abbildung 2.44 zeigt das System in Aktion.

Im Gegensatz zu 1978 waren im Jahr 1993 bereits PCs mit grafischen Bedienoberflächen weit verbreitet. Eine Alternative zum dort bereits vorherrschenden Fensterparadigma stellten *Ken Perlin* und *David Fox* mit dem sog. *Pad* vor [PER+ 93]. Das Pad besteht aus einer Informationsfläche, die zu den Seiten hin nicht begrenzt ist. Es enthält Informationsobjekte, wie zum Beispiel Kalendereinträge oder Textdokumente, mit denen der Benutzer interagieren kann. Es wurde für den Betrieb auf Standard-PC-Hardware entwickelt und ermöglichte dem Anwender sog. *semantisches Zooming*.

Beim Zooming denkt man in der Regel zuerst an etwas, was man als *geometrisches Zooming* bezeichnet: Objekte werden unter Beibehaltung ihrer Proportionen entweder verkleinert oder vergrößert dargestellt. Abgesehen von ihrer Größe bleibt ihr Aussehen erhalten. Beim *semantischen Zooming* ist das nicht der Fall: Je nachdem, wie viel Platz auf dem Bildschirm für ein semantisch gezoomtes Objekt verfügbar ist, ändert es seine Gestalt, um unterschiedlich viele Detailinformationen zu präsentieren.

2.17.3. Overview and Detail

Overview-and-Detail-Oberflächen zeichnen sich dadurch aus, dass sie aus mindestens zwei Fenstern mit unterschiedlichen Ansichten bestehen. Das *Overview*-Fenster zeigt einen Überblick der gesamten Daten, das *Detail*-Fenster einen Detailausschnitt. In der Regel ist das Overview-Fenster deutlich kleiner als das Detail-Fenster. Durch eine rechteckige Markierung im Overview-Fenster weiß der Betrachter sofort, wo der Ausschnitt des Detail-Fensters im gesamten Datenraum anzusiedeln ist. Im Idealfall bietet ihm das Overview-Fenster auch

[45] Selbst in den meisten Kinos war Mehrkanalton zu dieser Zeit ein Fremdwort.

Abbildung 2.44.: Das Spatial Data Management System sieht auf den ersten Blick zwar aus wie ein Heimkino, diente aber in Wirklichkeit der Erforschung neuartiger Bedienkonzepte und zoombarer Bedienoberflächen.

die Möglichkeit, durch einen Klick unmittelbar zu einer anderen Position zu springen, ohne lange durch das Detail-Fenster scrollen zu müssen.

Diese beiden Fenster müssen sich nicht zwangsläufig nebeneinander befinden. Um leeren Raum auf dem Bildschirm zu minimieren, ist eine Einbettung des Overview-Fensters in das Detail-Fenster möglich. In diesem Fall sollte es verschiebbar sein, um nicht immer dieselbe Stelle des Hintergrunds zu verdecken. Durch transparente Farben bleibt der überdeckte Hintergrund sichtbar. Es ist jedoch Geschmackssache, was übersichtlicher erscheint. Bereits Anfang der 1990er Jahre setzte der Softwarehersteller *Maxis* eine Overview-and-Detail-Ansicht in der Neuauflage seines Wirtschaftssimulationsspiel *SimCity 2000* ein (siehe Abbildung 2.45).

Um verschiebbare Fenster in Graphical User Interfaces (GUIs) ressourcenschonend zu implementieren, entschieden sich viele Entwickler, nicht den kompletten Fensterinhalt synchron zur Mausbewegung zu verschieben, sondern lediglich die neue Position des Fensterrahmens anzudeuten. Erst beim Loslassen der Maustaste wurde das Fenster inklusive Inhalt an der neuen Stelle des Bildschirms gezeichnet. Als die zum Verschieben von Fensterinhalten notwendigen Bit-boundary block transfer (BITBLT)-Funktionen nicht mehr von der Central Processing Unit (CPU) durchgeführt werden mussten, sondern von dedizierten Beschleuniger-Chips wie der Graphics Processing Unit (GPU) übernommen wurden, war das ressourcenschonende Andeuten des Fensterrahmens nicht mehr notwendig. Der Grundstein für BITBLT wurde bereits Mitte der 1970er Jahre am Xerox Palo Alto Research Center (PARC) für die Xerox Alto Workstation gelegt, einem der ersten leistungsstarken Einzelplatzrechner mit grafischer Bedienoberfläche und What you see is what you get (WYSIWYG) [ING 75].

Abbildung 2.45.: Das Wirtschaftssimulationsspiel SimCity 2000 deutet das gesamte Spielareal in einem Overview-Fenster an. In die Übersichtskarte lassen sich zusätzliche Infos wie hier die Kriminalitätsrate einblenden. Letztere ist in Graustufen kodiert.

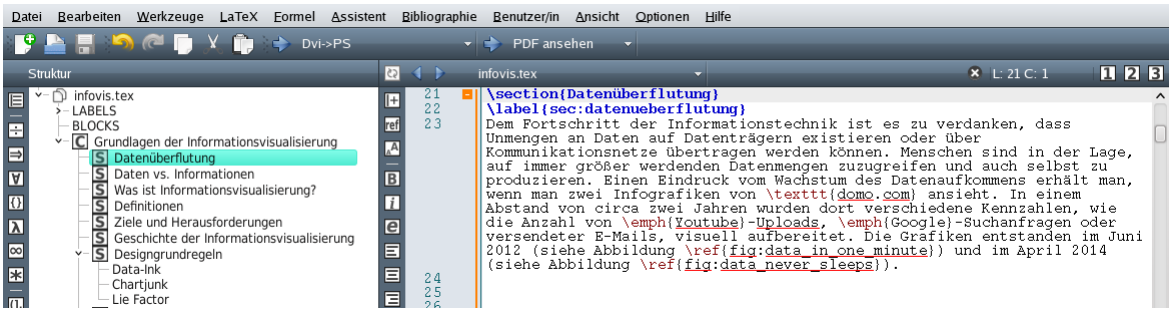


Abbildung 2.46.: Overview and Detail im Textverarbeitungsprogramm Texmaker.

Overview and Detail ist nicht auf grafische Inhalte beschränkt, auch Textverarbeitungsprogramme wie zum Beispiel die Open-Source-Anwendung *Texmaker* setzen dieses Präsentationskonzept um. Neben dem Hauptbildschirmbereich, der zur Texteingabe dient, gibt es noch eine parallel dazu eingeblendete Strukturübersicht, die zur Verbesserung der Orientierung die Gliederungspunkte und die aktuelle Position im Dokument beinhaltet. Abbildung 2.46 zeigt einen Bildausschnitt von Texmaker, in dem die beiden Elemente zu sehen sind.

2.17.4. Focus plus Context

Bei *Focus plus Context* handelt es sich um eine Präsentationsform, die eine Gesamtübersicht aller darzustellenden Daten mit geringem Detailgrad und eine Detailansicht mit hoher Informationsdichte in einer einzigen Visualisierung vereint. Im Gegensatz zu Overview and Detail muss der Betrachter so nicht ständig zwischen zwei Fenstern mit unterschiedlichem Detailgrad und Maßstab hin und her schauen.

Die Gesamtübersicht nimmt einen Großteil der Bildfläche ein und dient dem Betrachter zur Orientierung. Der Detailgrad ist dort gerade hoch genug, damit er entscheiden kann, welchen Bereich der Daten er als nächstes genauer ansehen möchte. Dazu dient der Fokus-Bereich. Er beansprucht nur einen Teil des Bildschirms und funktioniert wie eine Brille. Dort wo der Fokus-Bereich positioniert wird, werden detaillierte Informationen angezeigt.

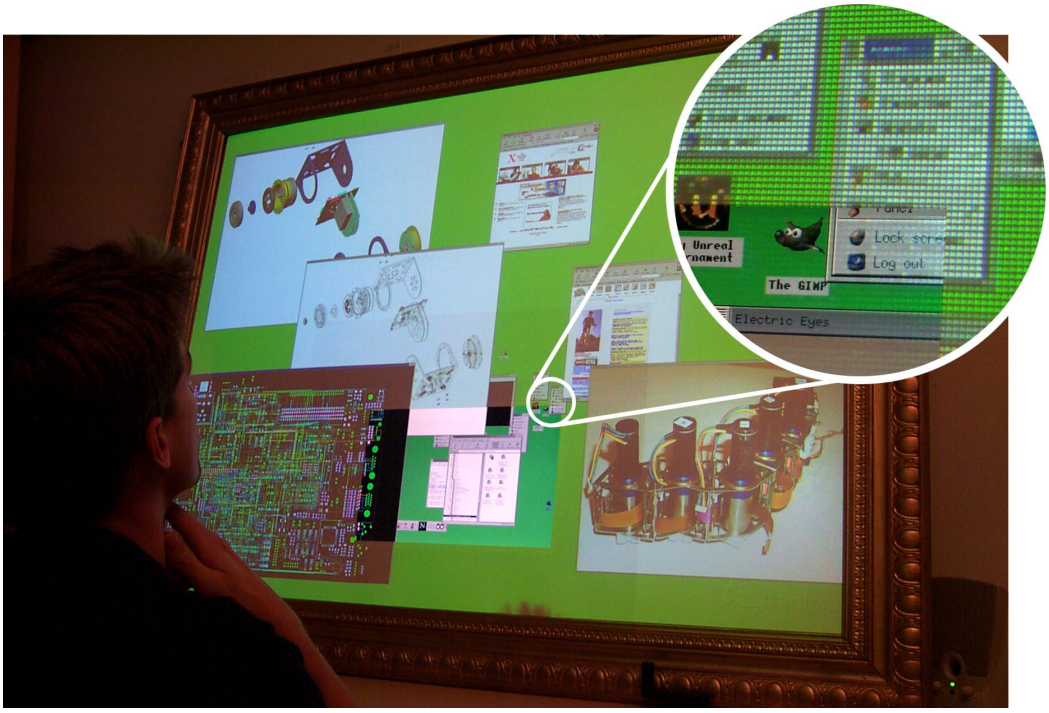


Abbildung 2.47.: Grafischer Linux-Desktop auf einem Focus-plus-Context-Bildschirm.[46]

Abbildung 2.47 zeigt die Implementierung dieses Konzepts anhand eines Computerbildschirms, der nahtlos in eine Leinwand integriert wurde. Der Monitor dient zur Detaildarstellung (Focus) während die Projektion auf die deutlich größere Leinwand die peripheren Bildinhalte mit geringem Detailgrad beinhaltet (Context). Konstruktionsbedingt ist der Focus-Bereich hier nicht verschiebbar. Stattdessen verschiebt der Anwender Fenster in den Focus-Bereich, um sie scharf sichtbar zu machen. Dieser Versuchsaufbau wurde Anfang des Jahrtausends von *Patrick Baudisch et al.* vorgestellt [BAU+ 01].

Eine frühere Variante von Focus plus Context stellte *George W. Furnas* im Jahr 1986 vor [FUR 86]. Seine Idee war es, Details abhängig von der Distanz zum Betrachter zu machen. Je größer die Entfernung, desto geringer der Detailgrad. Um den Detailgrad der darzustellenden Objekte zu bestimmen, führte er eine sog. *Degree-of-Interest*-Funktion ein. Nur eine bestimmte Anzahl der wichtigsten Objekte soll dargestellt werden. Daraus resultiert der sog. *Fisheye-Baum*. In seinem Paper *Generalized Fisheye Views* lieferte Furnas ein Konzept und eine Beispielimplementierung der essentiellen Algorithmen in der Programmiersprache *C*.

Im Jahr 1992 veröffentlichten *Manojit Sarkar* und *Marc H. Brown* ein Paper, in dem ein Verfahren vorgestellt wird, das das Betrachten großer Graphen deutlich vereinfachen soll [SAR+ 92]. Auch hier kommt ein Fisheye-ähnlicher Ansatz zur Verwendung. Vom Anwender fokussierte Knoten bieten Platz für einen hohen Detailgrad während die Zusammenhänge zu den benachbarten Knoten erhalten bleiben. Das Verfahren besteht aus einer geometrischen Transformation des ursprünglichen Graphen, die die Knoten neu anordnet und die Umrisse der Knoten skaliert. Die endgültige Größe der Knoten wird durch ihren sog. *A priori importance (API)*-Wert bestimmt. Die Abbildungen 2.48 und 2.49 stammen aus Sarkars Paper und zeigen einen Beispielgraphen vor und nach der Fisheye-Umwandlung.

[46] Bildquelle: `http://www.patrickbaudisch.com/projects/focuspluscontextscreens/applications/images/FullsizeOperatingSystemLinux.jpg`

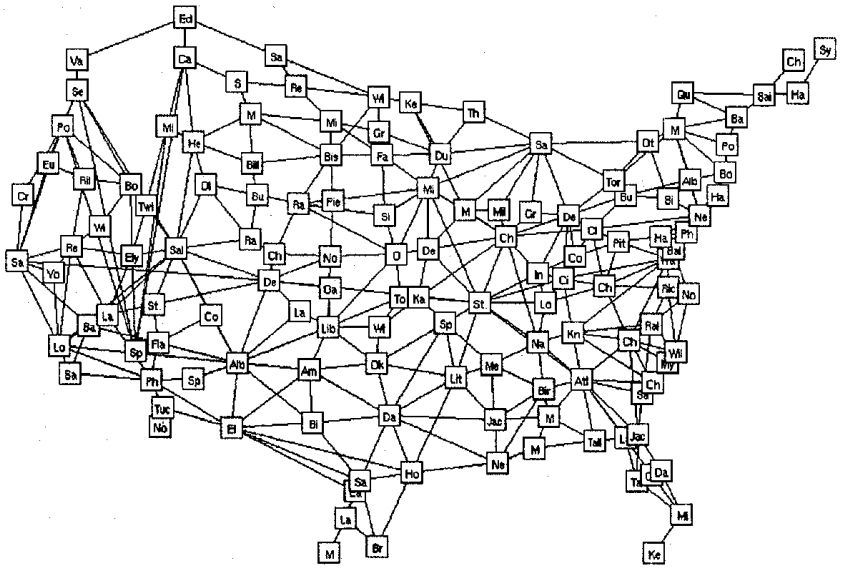


Abbildung 2.48.: Das ursprüngliche Layout eines Graphen mit Orten der USA. Er enthält insgesamt 134 Knoten und 338 Kanten.

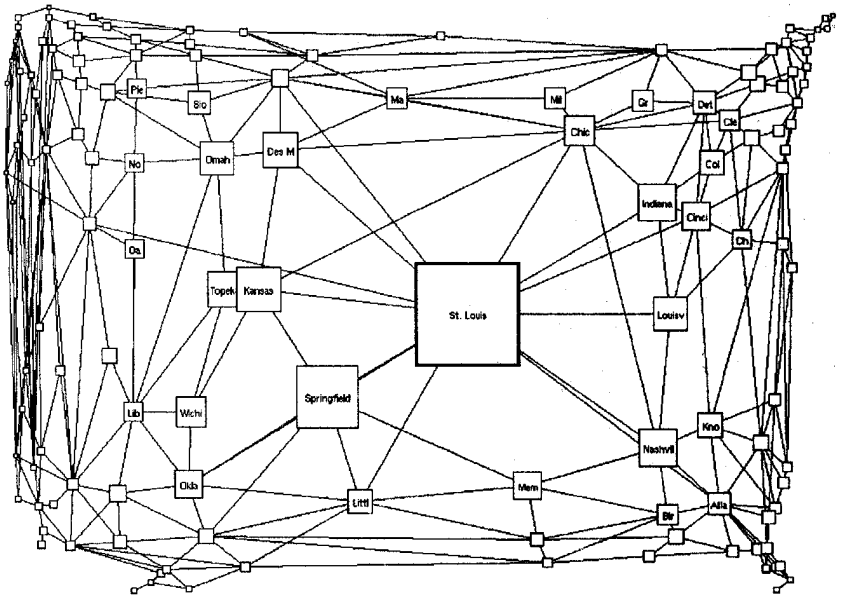


Abbildung 2.49.: Eine Fisheye-Ansicht des vorherigen Graphen mit der Stadt St. Louis im Fokus.

2.17.5. Ambient Information Visualization

Die bisher vorgestellten Techniken der Informationsvisualisierung setzen allesamt voraus, dass der Betrachter sie bewusst wahrnehmen muss, damit sie ihren eigentlichen Zweck erfüllen, nämlich komplexe Daten mit möglichst wenig Informationsverlust für den Betrachter darzustellen. In der Regel werden Systeme, die derartige Techniken einsetzen, nicht nur bewusst, sondern auch für eine längere Zeitspanne eingesetzt. Das dauert in der Regel solange, bis das System dem Benutzer zur Beantwortung einer konkreten Fragestellung geholfen hat. Oft setzen solche Systeme auch voraus, dass sich der Benutzer an einen bestimmten Arbeitsplatz setzt.

Im Gegensatz dazu existiert die sog. *Ambient Information Visualization*, die im Folgenden als *Umgebungsin-*

formationsvisualisierung bezeichnet wird. Systeme, die auf diesem Ansatz beruhen, zeichnen sich im Allgemeinen durch folgende Merkmale aus:

- Es sind nur wenige oder überhaupt keine Benutzerinteraktionen vorgesehen.
- Als Quelle dienen oft weniger kritische Daten, die in Echtzeit bezogen werden.
- Diese Daten werden zusammengefasst und in abstrakter Form dargestellt.
- Die Visualisierung wird in die reale oder eine virtuelle Umgebung eingebettet.
- Sie hat nur peripheren Charakter, damit sich Anwender ihren eigentlichen Aufgaben widmen können.
- Aktualisierungen der Visualisierung laufen sanft ab, ohne den Anwender abzulenken.
- Wichtige Informationen sollen präattentiv (siehe Abschnitt 2.11) wahrgenommen werden.
- Non-Data-Ink wird in Kauf genommen, da die Visualisierung ästhetischen oder künstlerischen Charakter hat.

Die Bandbreite von Umgebungsinformationsvisualisierung reicht von oft künstlerischen Objekten, die vollkommen ohne Interaktion auskommen (sog. *Calm Technology*) bis zur bildschirmbasierten peripheren Präsentation von Ereignismeldungen (“Second Screen”). Dies sind zwei sehr unterschiedliche Ausprägungen, die je nach der geplanten Anwendung mehr oder weniger in den Vordergrund treten. Da hier Kompromisse eingegangen werden müssen, befassten sich im Jahr 2003 *Tobias Skog et al.* mit geeigneten Designprinzipien und wie die Platzierung von Informationen und des Bildschirms selbst die Wahrnehmung der Visualisierung beeinflussen [SKOG+ 03].

Abbildung 2.50.: Der Dangling String visualisiert durch seine synchronen Bewegungen die Auslastung eines Rechnernetzes.

Als einer der ersten dokumentierten Vertreter von Calm Technology gilt der sog. *Dangling String* der Künstlerin und Ingenieurin *Natalie Jeremijenko* aus dem Jahr 1995. Ein 2,5 Meter langer Kunststofffaden wurde an der Achse eines kleinen Elektromotors befestigt, der an der Zimmerdecke hing. Der Elektromotor bewegte sich synchron zu dem Netzwerktraffic, der über ein nahegelegenes Netzwerkkabel ging. Je nach der Auslastung der Leitung zuckte und drehte sich der Faden mehr oder weniger schnell und visualisierte so die Auslastung des Netzes [WEIS+ 96]. Abbildung 2.50 zeigt ein Foto vom Dangling String im Einsatz.

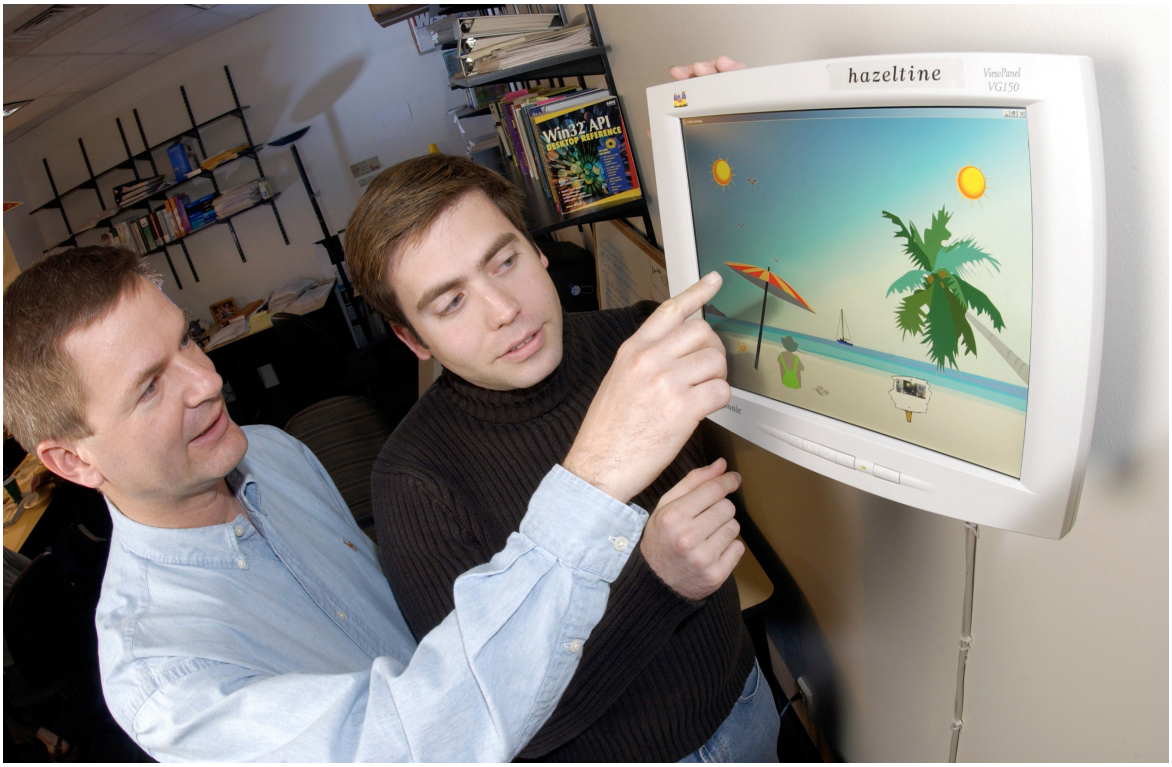


Abbildung 2.51.: John Stasko (links) und Todd Miller (rechts) stellen InfoCanvas vor. Jedes Objekt in der Cartoon-Szene verkörpert eine andere Information.[47]

Eine Studie zur bildschirmbasierten peripheren Präsentation von Ereignissen stellten *Todd Miller* und *John Stasko* im Jahr 2003 mit dem sog. *InfoCanvas* vor [MIL+ 03]. Dabei handelte es sich um ein dynamisch generiertes Bild mit einer Cartoon-Szene, das an der Wand hing oder auf einem Zweitbildschirm auf dem Schreibtisch dargestellt wurde.

Verschiedene Informationsquellen konnten benutzerspezifisch auf einzelne Bildelemente der Cartoon-Grafik gemappt werden, zum Beispiel auf den Stand der Sonne oder die Kleidungsstücke auf einer Wäscheleine. Diese Personalisierung sorgte nicht nur für einen erhöhten individuellen Nutzen von InfoCanvas, sondern trug auch zum Datenschutz bei, da aus dem Bild nicht hervorgeht, welche Bedeutung die dargestellten Elemente tatsächlich haben. In Abbildung 2.51 zeigen Todd Miller und John Stasko InfoCanvas in Aktion.

2.18. Zusammenfassung

In diesem Kapitel wurden wichtige Grundlagen der Informationsvisualisierung vorgestellt. Diese Grundlagensammlung dient einerseits als Maßstab für die Evaluierung existierender Administrationswerkzeuge für Linux-Server in Kapitel 5. Andererseits stellt sie ein Regelwerk dar, auf das bei der Anfertigung von Lösungsmöglichkeiten für ausgewählte Anwendungsfälle in Kapitel 6 zurückgegriffen wird. Dort werden Vorschläge genannt, wie Methoden der Informationsvisualisierung besser eingesetzt werden können, als es in bereits existierenden Administrationstools der Fall ist. Des Weiteren werden Lösungsmöglichkeiten für spezielle Anwendungsfälle vorgestellt, die von den evaluierten Anwendungen nicht abgedeckt werden.

Die in diesem Kapitel vorgestellten Grundlagen sollen auch in den zu entwickelnden Demonstrator einfließen. Wichtige Aspekte aus diesem Kapitel sind hierfür unter anderem die Designgrundregeln von Edward Tufte (siehe Abschnitt 2.7), präattentive Wahrnehmung (siehe Abschnitt 2.11) und die Visualisierung multivariater Daten (siehe Abschnitt 2.12). Der richtige Umgang mit Farben (siehe Abschnitt 2.13) spielt sowohl bei den

[47] Bildquelle: `http://gtresearchnews.gatech.edu/reshor/rh-ss03/info-art1_b.jpg`

dynamisch generierten Visualisierungen als auch bei der Gestaltung der Bedienoberfläche des Prototyps eine wichtige Rolle. Um große Datenmengen effizient auf dem Bildschirm darzustellen, wird auf Methoden des Abschnitts 2.17 (Präsentation) zurückgegriffen.

3. Grundlagen der Informationssicherheit

Das Ziel dieser Arbeit ist es, sicherheitsrelevante Daten von Linux-Servern zu visualisieren, um deren Administration zu erleichtern. Das vorherige Kapitel lieferte bereits eine Einführung in die Informationsvisualisierung. Um den Grundlagenteil dieser Ausarbeitung abzuschließen, befasst sich dieses Kapitel mit den Grundlagen der Informationssicherheit und geht auf die potenziellen Bedrohungen ein, denen Server ausgesetzt sind. Als Basis für diese Grundlagenübersicht dient unter anderem das Buch *Sichere Netzwerkkommunikation* von *Roland Bless et al.* [BLE+ 05]. Die Grundlagen zur Serversicherheit stammen aus der Studie *ISi-Server* vom Bundesamt für Sicherheit in der Informationstechnik (BSI) [BSI 13].

Praktisch alle privaten und öffentlichen Institutionen sind heutzutage von funktionsbereiter Informationstechnik (IT) abhängig. Während Rechnernetze eine Grundvoraussetzung für die IT darstellen, sind vor allem Server nicht wegzudenken, denn sie erbringen die Basisdienste. Dazu zählen beispielsweise der Austausch von E-Mails, das Hosten von Webseiten oder Netzwerkdienste wie Domain Name System (DNS) oder Dynamic Host Configuration Protocol (DHCP). Alleine am LRZ sind hunderte physikalischer und virtueller Server im Einsatz. Angreifer können großen Schaden anrichten, wenn sie in Server eindringen. Sie können sowohl Daten abgreifen oder manipulieren als auch die Verfügbarkeit der Systeme beeinträchtigen.

Abschnitt 3.1 stellt zunächst Bedrohungen und Angriffe vor, denen vernetzte Computersysteme im Allgemeinen ausgesetzt sein können. Der darauf folgende Abschnitt 3.2 geht auf konkrete Bedrohungsszenarien ein, die speziell am LRZ denkbar sind bzw. bereits zur Realität wurden.

Zur Abwehr von Bedrohungen existiert eine Reihe von Sicherheitszielen (siehe Abschnitt 3.3), die durch konkrete Maßnahmen und Kombinationen davon in die Realität umgesetzt werden. Die wichtigsten Maßnahmen zum Schutz von Vertraulichkeit, Integrität und Verfügbarkeit werden in Abschnitt 3.4 vorgestellt.

3.1. Bedrohungen und Angriffe

In diesem Abschnitt werden potenzielle Bedrohungen und Angriffsszenarien angesprochen, mit denen Kommunikationspartner in vernetzten Systemen konfrontiert werden. Folgende Bedrohungen sind im Allgemeinen denkbar:

- Abhören
- Einfügen, Löschen oder Verändern von Daten
- Verzögern und Wiedereinspielen von Daten
- Maskerade
- Autorisierungsverletzung
- Abstreiten von Ereignissen
- Sabotage
- Kombination von Angriffen

Beim *Abhören* liest ein Angreifer Kommunikationsdaten mit, die nicht für ihn vorgesehen sind. Die Schwierigkeit eines solchen *passiven* Angriffs hängt im Allgemeinen von der Art und der Zugänglichkeit des Mediums ab. So sind Funknetze prinzipiell leichter abhörbar als kabelgebundene Netze. Angreifer können nicht nur Nutzdaten ausspähen, sondern auch Zugangsdaten oder Chiffrierschlüssel. Dadurch ermöglichen sich zusätzliche – auch *aktive* – Angriffe. Selbst aus verschlüsselten Nutzdaten können Angreifer nützliche Informationen gewinnen. In manchen Fällen ist es ausreichend, die Zeitpunkte der Kommunikation und die Identität der

Kommunikationspartner zu ermitteln. Zu diesem Zweck muss die Transportschicht, wo die Verschlüsselung in der Regel beginnt, nicht betrachtet werden. Durch Traffic-Analyse kann bei verschlüsselten Inhalten bestimmt werden, um welche Art von Nutzdaten es sich handelt. Bereits im Jahr 1998 stellten *Heyning Cheng et al.* von der Universität Berkeley ein Verfahren vor, um trotz Secure Sockets Layer (SSL)-Verschlüsselung die von einem Client aufgerufenen Webseiten zu identifizieren [CHEN+ 98].

Beim *Einfügen, Löschen oder Verändern* erzeugt ein Angreifer selbst neue Daten und speist sie in den Kommunikationskanal ein bzw. fängt abgesendete Daten ab und vernichtet oder verändert sie. Es handelt sich dabei um einen *aktiven* Angriff. Ein sich zwischen den Kommunikationspartnern befindlicher Angreifer wird auch als *Man-in-the-Middle* bezeichnet.

Angreifer können dem Empfänger durch *Verzögerung* Daten des Absenders für eine gewisse Zeit vorenthalten, um zum Beispiel Kommunikationsvorgänge zu stören, die zeitlichen Abhängigkeiten unterliegen. Beim *Wiedereinspielen* sendet der Angreifer zuvor übertragene Daten zu einem späteren Zeitpunkt wiederholt zum Kommunikationssystem mit dem Zweck, Anwendungen ohne einen wirksamen Schutz gegen *Replay-Attacken* zu überlisten.

Bei der sog. *Maskerade* täuscht ein Angreifer die Identität eines Anderen vor. Sie dient in der Regel zur Vorbereitung weiterer Angriffe, indem zum Beispiel Zugangsberechtigungen beschafft werden. Von Betrügern erstellte *Phishing*-Webseiten, die zur Eingabe von Bankdaten nichtsahnender Kunden dienen, zählen auch zu dieser Kategorie. Ein anderes Beispiel sind sog. *Verschlüsselungs-* und *Erpressungstrojaner*, die Betroffenen die offizielle Handlung einer staatlichen Behörde vorgaukeln und Lösegeld für ihre unzugänglich gemachten Daten verlangen. Abbildung 3.1 zeigt einen Screenshot eines typischen Exemplars.

Abbildung 3.1.: Ein Erpressungstrojaner aus dem Jahr 2013, der nach dem Windows-Start den PC blockiert. Dem Betroffenen wird das volle Programm an Internetkriminalität vorgeworfen und mit Strafverfolgung gedroht, falls er nicht rechtzeitig bezahlt.[1]

Von *Autorisierungsverletzung* ist die Rede, wenn eine Instanz Dienste oder Ressourcen nutzt, ohne dazu eine offizielle Berechtigung zu besitzen. Das ist beispielsweise der Fall, wenn sich ein Angreifer durch das Aus-

[1] Bildquelle: `http://1.f.ix.de/imgs/18/1/0/5/1/1/5/9/gvutrojaner-hauptbild-8e4362a16beaa2df.jpeg`

nutzen einer Schwachstelle Administratorrechte eines Betriebssystems verschafft und auf die Daten anderer Nutzer des Systems zugreift.

Beim *Abstreiten von Ereignissen* möchte sich eine Instanz einer bestimmten Verantwortung entziehen, zum Beispiel der Bezahlung einer Dienstleistung, deren Bezug zwar abgestritten wird, in Wirklichkeit aber doch stattgefunden hat.

Mit *Sabotage* bezwecken Angreifer, die korrekte Funktion oder Verfügbarkeit von Computersystemen zu stören oder ganz außer Kraft zu setzen. Wenn die Verfügbarkeit von Diensten gestört wird, spricht man im Allgemeinen von *Denial of Service (DoS)-Angriffen*. Das kann zum Beispiel ein *physikalischer* Angriff sein, bei dem die für die Diensterbringung erforderliche Hardware beschädigt wird oder Störsignale eine Kommunikation unterbinden. Ebenso haben Angreifer die Möglichkeit, *Implementierungsschwächen* der eingesetzten Programme oder *Protokollschwächen* auszunutzen, wie zum Beispiel bei einem *Smurf-Angriff*[2]. Sabotage kann auch durch sog. *Buffer Overflows* erzielt werden. Besonders Anwendungen, die in maschinennahen Sprachen geschrieben wurden, sind dafür anfällig: Wenn ein solches Programm die Eingabe von beliebigen Nutzdaten erlaubt und nicht prüft, ob die Nutzdaten zu groß für den dafür vorgesehenen Speicher sind, werden darauf folgende Speicherbereiche überschrieben. Gezielt in den Nutzdaten untergebrachter Schadcode kann so auf einem Server zur Ausführung gebracht werden.

Die zuvor genannten Bedrohungen und Angriffstechniken werden in den seltensten Fällen alleine eingesetzt. Moderne Angreifer nutzen stattdessen alle Möglichkeiten aus, die ihnen zur Verfügung stehen. Um an Firmendaten zu gelangen nutzen sie beispielsweise Arbeitsplatzrechner als Zwischenstation, indem sie Lücken von darauf installierter Anwendungssoftware oder die Schwachstelle Mensch ausnutzen (“Social Engineering”). Sobald sie Zugang zu einem Arbeitsplatzrechner erlangt haben, können sie sich weiter zu den Servern vorarbeiten.

Angreifer können mit Schadsoftware präparierte Universal Serial Bus (USB)-Speichersticks oder andere sog. *BadUSB*-Devices [SCHM 14C] bewusst an bestimmten Orten liegenlassen, damit nichtsahnende Mitarbeiter sie am PC einstecken, um nachzusehen, was darauf gespeichert ist. Es ist nicht mehr ausreichend, Bedrohungen aus einem Unternehmen durch technische Maßnahmen abzublocken, sondern es muss auch unternehmensintern nach Bedrohungen gesucht werden [SCHIN+ 13].

Webseiten sind in der heutigen Zeit das Zugangstor zu zahlreichen Diensten und Angeboten wie E-Mail, Nachrichten und Online Shops. Webserver, die Webanwendungen bereitstellen, sind ein beliebtes Angriffsziel von Hackern. Präparierte Webseiten können *Drive-by-Downloads* im Browser der Besucher auslösen und so Schadcode auf eine große Anzahl an Computern platzieren. Eine Liste mit den zehn größten Risiken für Webanwendungen hat das *Open Web Application Security Project (OWASP)* online gestellt und zuletzt im Jahr 2013 aktualisiert [OWA 13]. Folgende Risiken sollten Webentwickler laut OWASP aktuell besonders ernst nehmen (Platzierung des Jahres 2010 in Klammern):

1. Injection (1)
2. Broken Authentication and Session Management (3)
3. Cross-Site Scripting (XSS) (2)
4. Insecure Direct Object References (4)
5. Security Misconfiguration (6)
6. Sensitive Data Exposure (7/9)
7. Missing Function Level Access Control (8)
8. Cross-Site Request Forgery (CSRF) (5)
9. Using Known Vulnerable Components (-)
10. Unvalidated Redirects and Forwards (10)

[2]Quelle: `http://www.cert.org/historical/advisories/ca-1998-01.cfm?`

Die *Injektion* von eigenem Code stellt laut der Statistik das derzeit größte Problem dar. Sie wird durch Programmierfehler ermöglicht, die Entwicklern immer wieder unterlaufen. Injections haben eine ähnliche Ursache wie Buffer Overflows: Vom Benutzer eingegebene und an den Server übertragene Daten werden nicht ausreichend oder überhaupt nicht überprüft. Dies ist zum Beispiel der Fall, wenn ein Server-Skript die Eingabe eines Formularfelds unverändert in einen *Structured Query Language (SQL) Query String* einfügt und diesen an die angebundene Datenbank überträgt. Wenn der Angreifer dort geschickt Skript- und SQL-spezifische Metazeichen wie etwa Anführungszeichen oder Strichpunkte einsetzt, kann er zusätzliche Anfragen in den Programmcode einschleusen, ohne Zugriff auf die eigentlichen Skriptdateien besitzen zu müssen.

Der eingeschleuste Code wird dadurch zwar nicht persistent, aber er kann bereits bei einmaliger Ausführung fatale Folgen haben: Der Angreifer kann vertrauliche Informationen aus der Datenbank abrufen, deren Zugreifbarkeit über das Webformular gar nicht vorgesehen war. Wenn das Skript noch dazu einen Datenbankbenutzer mit unzureichender Einschränkung der Zugriffsrechte verwendet, können Angreifer weitaus größeren Schaden anrichten und Daten manipulieren oder zerstören. Die Prüfung der vom Nutzer eingegebenen Daten und die Maskierung von Metazeichen schaffen Abhilfe.

Um derartige Fehler auszuschließen, müssen Entwickler das Rad nicht neu erfinden. Es gibt Frameworks, die bei korrekter Anwendung einen gewissen Grundschutz bieten. Das *Python*-basierte Web Framework *Django* bietet beispielsweise Mechanismen, die Webanwendungen u. a. vor XSS, SQL Injection und CSRF schützen sollen[3]. Wer den *objektrelationalen Mapper* des Frameworks nutzt, kommt gar nicht erst in Kontakt mit Query Strings, die gegen Injection abgesichert werden müssten[4]. Da auch der im Rahmen dieser Arbeit zu entwickelnde Prototyp eine Webanwendung ist, werden diese Gefahren dort ebenfalls berücksichtigt und geeignete Schutzmaßnahmen ergriffen.

3.2. Potenzielle Bedrohungen der Server am LRZ

Das LRZ stellt einer großen Anzahl akademischer Nutzer IT-Dienste bereit. Um welche es sich konkret handelt, wird im Kapitel 4.1 kurz beschrieben. Jeder Dienst, der über eine nach außen zugängliche Schnittstelle angeboten wird, ist potenziellen Bedrohungen ausgesetzt.

Ein mögliches Angriffsziel sind zum Beispiel die Webserver. Angreifer könnten unberechtigt Inhalte von Startseiten oder komplette Websites ändern, um auf ihre Verwundbarkeit öffentlichkeitswirksam hinzuweisen oder um sich selbst zu profilieren. Man spricht in diesem Fall von einem *Defacement*. Ein anderes Motiv ist das Verbreiten von Falschmeldungen, um dem Angegriffenen (finanziellen) Schaden zuzufügen und ggf. selbst Profit daraus zu schlagen.[5]

Das LRZ ist bereits in der Vergangenheit Opfer eines Defacements geworden, wie in Abbildung 3.2 zu sehen ist. Hier wurde lediglich auf eine Sicherheitslücke hingewiesen. Je nach Kontext können die Auswirkungen eines Defacements oder anderer Hackerangriffe schwerwiegend sein, zum Beispiel durch den Reputationsverlust einer Firma, die sich auf IT-Sicherheit spezialisiert hat. Im Juni 2015 wurde zum Beispiel bekannt, dass das Netz des Security-Software-Herstellers *Kaspersky* von einem hochentwickelten Trojaner befallen war, der monatelang vertrauliche Informationen abgezogen hat [SCHIR 15].

Denkbar sind neben Defacements auch Angriffe, die am LRZ deutlich mehr Schaden anrichten. Dazu zählen DoS-Angriffe auf Server, die von kritischer Bedeutung sind. Doch nicht nur Server können durch DoS-Angriffe beeinträchtigt werden, Netzkomponenten wie Router zählen ebenfalls dazu. Dies wirkt sich ebenfalls nachteilig auf die Dienstbereitschaft des Rechenzentrums aus.

Da die Virtualisierung von Netzkomponenten in den letzten Jahren immer mehr an Bedeutung gewonnen hat, werden einzelne Hardware-Appliances mehr und mehr durch Virtual Machines (VMs) ersetzt. Sofern keine Failover-Systeme vorhanden sind, genügt es bereits einen einzigen Software-defined networking (SDN)-

[3] Quelle: `https://docs.djangoproject.com/en/1.7/topics/security/`

[4] Quelle: `https://docs.djangoproject.com/en/1.7/topics/db/`

[5] Auch Rechtsextreme nutzen Defacements, um unter dem Schutzmantel der Anonymität Nazipropaganda zu verbreiten oder Hinterbliebene von Weltkriegsopfern zu diskriminieren. Ein aktuelles Beispiel stellt die Website der österreichischen Konzentrationslagergedenkstätte Mauthausen dar. Angreifer hatten dort anlässlich des 70. Gedenktags zur Beendigung des zweiten Weltkriegs kinderpornographisches Material platziert [APA 15].

[6] Bildquelle: `https://www.lrz.de/fun/leibniz-keks-zentrum.html`

Abbildung 3.2.: Defacement am Beispiel der Startseite des LRZ.[6]

oder Network Functions Virtualization (NFV)-Server zu hacken, um schwerwiegende Auswirkungen auf die Rechnernetze zu verursachen. Durch die Bündelung der gesamten Forwarding-Logik in einer zentralen Control Plane wirken sich Angriffe bei unzureichender Absicherung auf das gesamte Netz aus.

Zu den potenziellen Bedrohungen zählt auch das Abgreifen großer Datenmengen. Da das LRZ zehntausende Kunden besitzt, sind dort auch viele personenbezogene Daten vorhanden. Diese könnten entwendet und zum Nachteil der Kunden missbraucht werden. Es existieren nicht nur personenbezogene Daten, die von den Anwendern bewusst erzeugt wurden. Auch Logdateien von Webservern können für Datendiebe von Interesse sein, da sie geeignet sind, um personenbezogene Nutzungsprofile zu erstellen.

Die breitbandige Anbindung des Rechenzentrums ans Deutsches Wissenschaftsnetz (WiN) stellt für Angreifer ein attraktives Merkmal dar: Sollte es ihnen gelingen, Server des Rechenzentrums mit Schadsoftware zu infizieren, könnten sie unter Ausnutzung der schnellen Anbindung DoS-Angriffe auf andere Institutionen durchführen.

Stets aktuelle und mit Sicherheitspatches versehene Paketversionen sind für den sicheren Betrieb der Linux-Server am LRZ unerlässlich. Andernfalls könnten Angreifer Sicherheitslücken ausnutzen, um einzelne Systeme zu übernehmen und als Ausgangspunkt für weitere Taten zu missbrauchen. Um die Aktualität der Pakete hunderter Server im Blick zu behalten, gibt es Mittel der Informationsvisualisierung, die in Abschnitt 6.1 vorgestellt werden.

Damit nicht jeder der hunderten Server seine Updates individuell aus dem Internet laden muss, setzen Rechenzentren eigene Spiegelserver ein. Anstelle einzelne Server zu hacken, ist es naheliegend, dass ein Angreifer auf dem Spiegelserver Schadcode oder Viren platziert. Updates, die nicht ausgiebig getestet und stattdessen von den Servern automatisch installiert werden, können sich innerhalb von kurzer Zeit inklusive des Schadcodes auf eine beträchtliche Anzahl an Servern ausbreiten.

Um ein Sicherheitskonzept nach dem *Defense-in-Depth*-Prinzip [NSA 01] zu realisieren, kommen am Rechenzentrum neben dedizierten Firewall Appliances an Netzübergängen auch zusätzliche Personal Firewalls auf den Servern zum Einsatz. Wenn die Integrität dieser Personal Firewalls nicht ständig durch geeignete Maßnahmen überwacht wird, kann ein Angreifer oder Innentäter mit ausreichenden Rechten die Firewall-Konfigurationen nach seinen Bedürfnissen ändern oder den Zusatzschutz komplett deaktivieren. Der Angreifer schafft sich dadurch die Voraussetzung für nachfolgende Angriffsschritte. Wie mit Hilfe von Methoden der Informationsvisualisierung der Zustand der Personal Firewalls überwacht werden kann, wird in Abschnitt 6.2 vorgestellt.

Zum Schutz von Vertraulichkeit und Integrität setzt das LRZ auf seinen Servern das Transportverschlüsselungsverfahren Transport Layer Security (TLS) ein. Das Kryptosystem setzt voraus, dass private Schlüssel besonders geheim zu halten sind. Durch Unachtsamkeiten der Administratoren können Angreifer mit geringem Aufwand an private Schlüssel gelangen und diese für ihre Zwecke missbrauchen. In Abschnitt 6.3 wird ein Ansatz beschrieben, der Administratoren dabei hilft, die Sicherheit der Private Keys im Auge zu behalten.

Eng mit der Transportverschlüsselung verbunden ist neben den privaten Schlüsseln auch die TLS-Konfiguration der Server. Wenn die TLS-Konfiguration eines Systems nicht restriktiv genug ist, kann der Angreifer beim Verbindungsaufbau die Verwendung unsicherer Protokollversionen oder Cipher Suites erzwingen und die darin befindlichen Sicherheitslücken zu seinem Vorteil ausnutzen. Ein Konzept zur Überwachung der Sicherheit der TLS-Konfigurationen wird in Abschnitt 6.4 vorgestellt.

3.3. Sicherheitsziele bei Servern

Den zuvor genannten potenziellen Bedrohungen kann man entgegenwirken. Zu diesem Zweck müssen die folgenden technischen Sicherheitsziele sichergestellt werden:

- Vertraulichkeit
- Datenintegrität
- Verfügbarkeit

Unter *Vertraulichkeit* im Kontext von Informationssicherheit versteht man, dass gespeicherte oder übertragene Daten nur für berechtigte Personen oder Instanzen zugänglich sind. Als vertraulich können die verschiedensten Informationen eingestuft werden, die aus Datenschutz- oder wirtschaftlichen Gründen nur für einen bestimmten Personenkreis bestimmt sind. Dazu zählen Gehaltstabellen, Arzttermine, Krankenakten, Forschungsergebnisse, Notenlisten, Zugangsdaten, Browserverläufe, Geschäfts-E-Mails und Liebesbriefe, um ein paar alltägliche Beispiele zu nennen. Unbefugte können an vertraulichen Daten interessiert sein, um gewinnbringende Informationen zu erhalten und der ausgespähten Instanz wirtschaftlichen Schaden zuzufügen.

Ein aktuelles Beispiel für Vertraulichkeitsverletzung ist der Hack des Filmstudios *Sony Pictures* im November 2014. Angreifer, die sich selbst als Guardians of Peace (GOP) bezeichneten, sind dort in das Firmennetz eingedrungen und an große Mengen vertraulicher Daten gelangt. Die Hacker stellten den Konzern bloß und veröffentlichten firmeninterne Daten wie E-Mails, Passwörter und Informationen über Schauspieler und kommende Kinofilme im Netz. Wie sich herausstellte, begünstigte ein fahrlässiger Umgang mit IT-Sicherheit den Hack des Unternehmens [SCHE 14]. Mit juristischen Drohungen wollte Sony der Ausbreitung der internen Dokumente entgegenwirken. Brisant wurde die Angelegenheit, als GOP Terrordrohungen verbreitete, sollte Sony Pictures seine aktuelle Produktion *The Interview* in die Kinos bringen (Abbildung 3.3). Der Kinostart wurde daraufhin verschoben. Nach heftiger öffentlicher Kritik wurde der Film aber online per Videostream veröffentlicht. In Deutschland kam der Film erst im Februar 2015 in die Kinos.

Es gibt aber auch Fälle, in denen es einem Angreifer bereits ausreicht, wenn er feststellt, wer die Kommunikationspartner sind, ohne den Inhalt der Kommunikation zu kennen. Das kann zum Beispiel bei Verbindungsdaten der Fall sein. Aus den erhaltenen Zeitstempeln und Internet Protocol (IP)-Adressen im Datenstrom kann ein Angreifer ableiten, wer wann einen Dienst in Anspruch genommen hat.

[7] Bildquelle: `http://www.theinterview-movie.com/images/the_interview_movie_header.jpg` (aufgerufen am 29.12.2014)

Abbildung 3.3.: Mitglieder der Hackergruppe "Guardians of Peace" kündigten Terroranschläge an, falls Sony Pictures den Nordkorea-kritischen Film "The Interview" in die Kinos bringt. In dem Film sollen ein amerikanischer Fernsehreporter und sein Produzent im Auftrag der CIA den nordkoreanischen Staatschef Kim Jong-un bei einem Interview umbringen.[7]

Wenn *Integrität* gewährleistet ist, können Empfänger eindeutig feststellen, ob empfangene Daten während der Übertragung beschädigt oder von Angreifern manipuliert wurden. Zu den denkbaren Manipulationsverfahren zählen das Ersetzen, Einfügen oder Entfernen von Teilen des Nachrichtenverkehrs.

Der Ausfall bestimmter Dienste vernetzter Systeme kann schwerwiegende Konsequenzen nach sich ziehen. Der Sinn eines Notrufsystems ist es beispielsweise, unterbrechungsfrei zur Verfügung stehen. Dasselbe gilt nicht nur für Dienste, sondern auch für Nutzdaten. Ihre unterbrechungsfreie *Verfügbarkeit* soll gewährleistet werden.

Neben den drei zuvor genannten zentralen Sicherheitszielen gibt es noch weitere Ziele, die hier der Vollständigkeit halber angesprochen werden:

- Authentizität
- Privatsphäre
- Verbindlichkeit/Nichtabstreitbarkeit
- Zugangs- und Zugriffskontrolle
- Zurechenbarkeit

Authentizität bedeutet, dass eine Kommunikationsinstanz einer anderen ihre Identität zweifelsfrei nachweisen kann. *Authentizität der Daten* ist gewährleistet, wenn der Absender eindeutig identifiziert werden kann und die Integrität der Übertragung sichergestellt ist.

Als *Privatsphäre* wird der "[...] Bereich einer Person bezeichnet, der ohne ihre Einwilligung nicht zugänglich ist und in dem die betreffende Person ihr Recht auf freie Entfaltung ihrer Persönlichkeit wahrnimmt, ohne dass sie dabei von äußeren Einflüssen behelligt wird [...]"[8]. Sie kann durch *Anonymität* gewährleistet werden. Das bedeutet, dass die wahre Identität einer Kommunikationsinstanz geheim bleibt. Wenn mehrere Instanzen einen Kommunikationskanal gemeinsam benutzen, müssen sie dennoch unterschieden werden können, damit es zu keiner Vermischung der Nutzdaten kommt. Durch Gewährleistung von *Pseudoanonymität* bleiben die Instanzen zwar identifizierbar, ihre tatsächliche Identität wird aber nicht offengelegt.

Verbindlichkeit bedeutet, dass eine Instanz eine zuvor stattgefundene Kommunikation oder die Inanspruchnahme eines Dienstes gegenüber einer dritten Instanz *nicht abgestritten* werden kann. Es soll also nachvollziehbar sein, dass der Absender eine Nachricht tatsächlich abgeschickt hat (*Nichtabstreitbarkeit der Herkunft*) und der Empfänger sie tatsächlich erhalten hat (*Nichtabstreitbarkeit des Erhalts*).

[8] Quelle: `http://www.juraforum.de/lexikon/schutz-der-privatsphaere`

Zugangs- und Zugriffskontrolle stellen sicher, dass nur autorisierte Instanzen den Zugriff auf bestimmte Daten oder Dienste erhalten. Zusätzlich können Objekte durch bestimmte Zugriffsarten wie das Lesen, Schreiben oder Ausführen feingranularer geschützt werden.

Die Inanspruchnahme eines Dienstes muss einem Dienstnutzer eindeutig zugeordnet werden können. *Zurechenbarkeit* ist besonders bei der Abrechnung von Diensten erforderlich und setzt die zuvor genannte *Nichtabstreitbarkeit* voraus.

3.4. Maßnahmen zur Umsetzung der Sicherheitsziele

Es gibt verschiedene technische Maßnahmen, die dazu dienen, die zuvor erwähnten Sicherheitsziele in die Tat umzusetzen. Essentielle Maßnahmen aus dem Bereich der Server-Sicherheit werden in den folgenden Abschnitten angesprochen. Da eine scharfe Trennung zwischen Vertraulichkeit, Integrität und Verfügbarkeit nicht immer möglich ist, sind Überschneidungen zwischen diesen Themengebieten möglich.

3.4.1. Vertraulichkeit

In diesem Abschnitt werden die wichtigsten Techniken und Vorgehensweisen vorgestellt, die Vertraulichkeit sicherstellen. Dazu zählen die Folgenden:

- Authentifizierung
- Datensparsamkeit
- Trennung von Sicherheitszonen
- Passwortrichtlinien
- Räumliche Zugangsbeschränkung
- Datenträgerverschlüsselung
- Transportverschlüsselung
- Firewalls

Authentifizierung

Damit unautorisierte Personen nicht auf fremde Daten zugreifen können, werden Verfahren zur Authentifizierung und Autorisierung eingesetzt. Anwender müssen sich gegenüber einem System authentisieren, damit das System weiß, wer gerade darauf zugreift. Durch Autorisierungsmaßnahmen werden ihre Handlungsmöglichkeiten bei Dateien, Programmen und Geräten eingeschränkt. Zur Authentisierung werden häufig Kombinationen aus einem Benutzernamen und einem Passwort verwendet. Dieses Wissen kann leicht aus fahrlässigen Gründen weitergegeben oder ausgespäht werden und vertrauliche Informationen werden für einen Angreifer einsehbar. Diese Gefahr kann durch die Ergänzung der Authentisierung durch zusätzliche Faktoren verringert werden, z. B. indem der Anwender eine persönliche Smartcard verwendet (Faktor *Besitz*) und einen Personal Identification Number (PIN)-Code (Faktor *Wissen*) eingibt.

Nicht nur Personen müssen sich gegenüber Servern authentisieren, sondern unter Umständen auch Geräte. *Media Access Control (MAC)-Adressen-basierte* Authentifizierung kommt beispielsweise als Fallback-Lösung von *Institute of Electrical and Electronics Engineers (IEEE) 802.1X* zum Einsatz (*MAC Authentication Bypass*), um netzwerkfähige Geräte automatisch dem richtigen Virtual Local Area Network (VLAN) zuzuweisen. Der Netzwerk-Switch fragt beim zuständigen *Remote Authentication Dial-In User Service (RADIUS)-Server* nach dem VLAN, das zu der vom angeschlossenen Gerät mitgeteilten MAC-Adresse passt und schaltet erst dann den Ethernet Port für das entsprechende Netz frei. Bei fehlgeschlagener Authentifizierung ist die automatische Verbindung mit einem Quarantäne-Netz denkbar. Authentifizierungsmethoden, die lediglich auf IP-

oder MAC-Adressen basieren, müssen durch sicherere Verfahren ausgetauscht werden, da diese Adressen von Angreifern ohne großen Aufwand gefälscht werden können.

Datensparsamkeit

Zur Unterstützung des vertraulichen Umgangs mit Informationen muss stets das sog. *Need-to-know*-Prinzip (Kenntnis nur bei Bedarf) eingehalten werden. Autorisierte Personen dürfen demnach nur auf Informationen zugreifen, die sie für das Erfüllen ihrer eigenen Aufgaben benötigen. Der Administrator eines Dateiservers wäre zwar aus technischer Sicht in der Lage, sämtliche Dateien aller Mitarbeiter anzusehen (sofern sie nicht von den Anwendern verschlüsselt wurden). Zur Erledigung seiner Arbeit ist das aber nicht notwendig. Außerdem wird bei diesem Extrembeispiel der Datenschutz mit Füßen getreten.

Ein ähnliches Sparsamkeitsprinzip lässt sich auch auf die Anwendungen übertragen, die auf dem Server ausgeführt werden. Programme dürfen nur mit den minimalen Rechten laufen, die sie zur Erfüllung ihrer Aufgabe benötigen. Ein Webserverprozess sollte zum Beispiel mit den Rechten eines eingeschränkten Benutzers betrieben werden. Er darf nur Schreibrechte innerhalb des eigenen Document-Root-Verzeichnisses besitzen, falls für die gehosteten Seiten überhaupt Schreibzugriffe benötigt werden.

Private-Key-Dateien müssen ebenfalls mit entsprechenden Dateisystemrechten gesichert werden, sodass nur diejenigen Programme Lesezugriff bekommen, die TLS-Verbindungen anbieten sollen. Feingranulare Zugriffskontrollen können neben *Access Control Lists (ACLs)* auch durch Kernel-Erweiterungen wie *Security-Enhanced Linux (SELinux)* oder *AppArmor* festgelegt werden. Beide implementieren *Mandatory Access Control (MAC)* und lassen sich über sog. *Policies* bzw. *Profile* konfigurieren. Diese bestimmen, welche Zugriffsrechte einzelne Prozesse bekommen. Dadurch bieten beide Kernel-Erweiterungen einen gewissen Schutz gegen *Zero Day Exploits*.

Das Sparsamkeitsprinzip hat nicht nur bei Benutzerrechten in der IT, sondern auch im Geheimschutz eine große Bedeutung. Projektbezogene Dokumente, die einer bestimmten Einstufung wie zum Beispiel *Verschlusssache - Nur für den Dienstgebrauch (VS-NfD)* unterliegen, dürfen nur von Personen eingesehen werden, die unmittelbar mit dem Projekt zu tun haben und die darin enthaltenen Informationen zur Erfüllung ihrer Aufgaben unbedingt benötigen. Es ist notwendig, aber nicht ausreichend, vom Sicherheitsbevollmächtigten (SiBe) eine Sicherheitsbelehrung über den Umgang mit VS-NfD-Dokumenten erhalten zu haben, um auf derart eingestufte Dokumente zugreifen zu dürfen.[9]

Trennung von Sicherheitszonen

Nicht nur innerhalb eines einzigen Servers müssen Zugriffe durch User und Anwendungsprozesse auf ein Minimum beschränkt werden. Dies gilt auch für die Server untereinander, wenn zum Beispiel eine Virtualisierungslösung eingesetzt wird. Moderne Hypervisor wie zum Beispiel *VMware ESXi* sind in der Lage, ihre VMs an mehrere unterschiedliche Netze anzubinden. Zu diesem Zweck muss der Virtual Machine Host über Erweiterungskarten mit einer entsprechenden Anzahl an Ethernet-Schnittstellen ausgestattet werden. Alternativ können einzelne Interfaces auch für *VLAN Tagging* nach *IEEE 802.1Q* konfiguriert werden, um mehrere Netze über ein einziges Kabel anzubinden, sofern am anderen Ende ein Switch Port zur Verfügung steht, der ebenfalls 802.1Q unterstützt.

Die Anbindung eines Virtual Machine Hosts an mehrere Netze ermöglicht prinzipiell auch die Anbindung an unterschiedliche Sicherheitszonen. Neben organisationsinternen Servernetzen könnte man auch sicherheitskritische *DMZ-Netze* auf den selben Host bringen. Die einzelnen VMs sind zwar durch virtuelle Switche an nur eines dieser Netze angebunden, ein Angreifer von außen könnte aber durch das Ausnutzen einer Lücke theoretisch Zugriff auf eine VM erhalten, die mit einem internen Netz verbunden ist. Aus diesem Grund empfiehlt das BSI den Einsatz separater Virtual Machine Hosts für unterschiedliche Sicherheitsdomänen.[10] Abbildung 3.4 zeigt ein Beispiel, wo dies offenbar noch nicht der Fall ist.

[9] Das Bundesministerium für Wirtschaft und Energie (BMWi) stellt ein Merkblatt für den korrekten Umgang mit VS-NfD-Dokumenten auf seinem Sicherheitsserver zum Download bereit: `https://bmwi-sicherheitsforum.de/ghb/formulare/317,0,0,1,0.html`

[10] Die Bereitstellung separater Virtual Machine Hosts für die einzelnen Sicherheitszonen kann sehr kostspielig werden, da auch redundante Systeme für den Failover oder zur Überbrückung von Wartungsarbeiten vorhanden sein sollten.

Nicht immer muss ein Hacker im Spiel sein, der die Isolation zwischen einzelnen VMs aushebelt. In gut begründeten Fällen kann dies auch aus Performancegründen möglich sein. Das von VMware bereitgestellte *Virtual Machine Communication Interface (VMCI)* ermöglicht eine Hochgeschwindigkeitsverbindung zwischen zwei VMs auf demselben Host, indem die verhältnismäßig langsame Netzwerkschicht umgangen wird[11]. Zwei Anwendungen auf unterschiedlichen Servern, die die spezielle Application Programming Interface (API) *VMCI-Sockets* nutzen, können so unmittelbar Daten über einen gemeinsamen Speicherbereich austauschen. Da hier die Isolation zwischen einzelnen VMs bewusst aufgegeben wird, ist der Einsatz von VMCI in sicherheitskritischen Umgebungen nicht empfehlenswert. Es müssen dort andere Lösungen gefunden werden, um mangelnde Performance zu optimieren, zum Beispiel der Einsatz dedizierter Server anstatt von virtuellen Maschinen oder die Verwendung spezieller Hochgeschwindigkeitsbusse. Die Gast-zu-Gast-Kommunikation wurde aus *VMware vSphere 5.1* wieder entfernt. Host-zu-Gast-Kommunikation soll laut Hersteller weiterhin möglich sein.

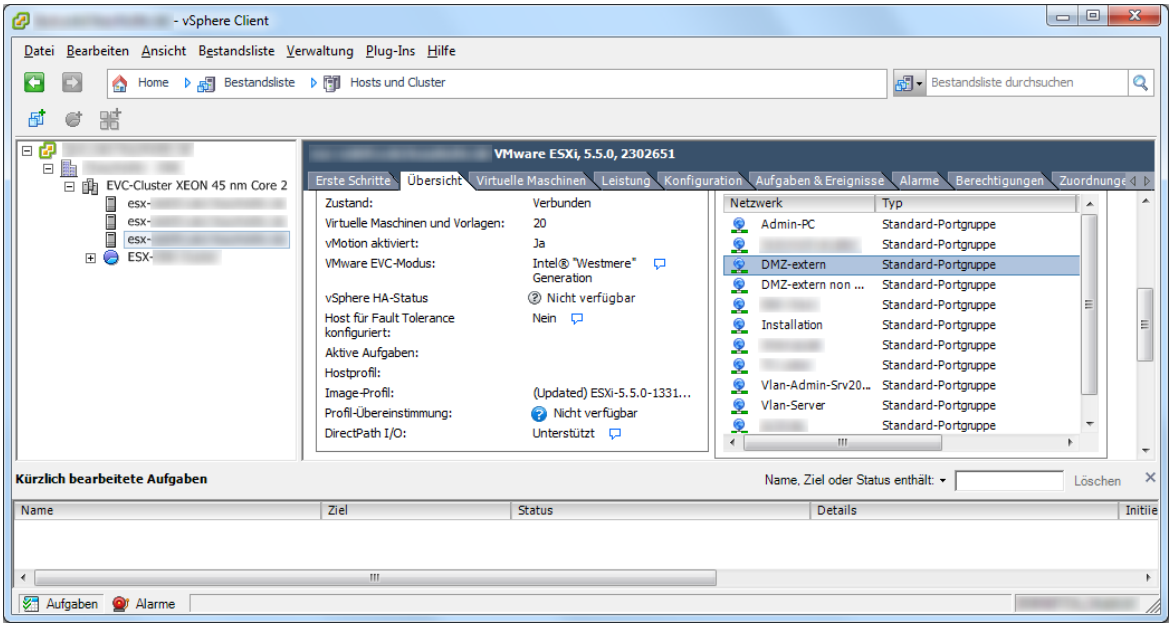


Abbildung 3.4.: An diesen ESXi Host sind sowohl DMZ-Netze als auch interne Netze angebunden (rechts). Eine physikalische Trennung der Sicherheitszonen ist in diesem Fall empfehlenswert.

Passwortrichtlinien

Der Einsatz einer Passwort-Richtlinie ist generell empfehlenswert, um ein bestimmtes Sicherheitsniveau bei der Authentifizierung der Anwender zu erhalten. Denn je länger und zufälliger ein Passwort ist, desto schwieriger ist es zu erraten oder durch einen *Brute-Force-Angriff* herauszufinden. Passwortrichtlinien gelten nicht nur für Mitarbeiter, die an ihren Workstations arbeiten, sondern auch für die Administratoren der Server. Auch Administrator-Passwörter müssen in regelmäßigen Abständen erneuert werden. Das Ausscheiden eines Administrators sollte je nach den Ursachen dafür auch als Auslöser für den Passwortwechsel betrachtet werden. Im schlimmsten Fall könnte er sich beim alten Arbeitgeber rächen und mit den weiterhin bekannten Zugangsdaten Schaden anrichten.

Bei all diesen Sicherheitsmaßnahmen muss berücksichtigt werden, dass die Benutzbarkeit der IT nicht darunter leidet. Komplexe Passwörter sind im Allgemeinen zwar als sicher anzusehen, können ohne geeignete Eselsbrücke aber auch schlecht von Anwendern eingeprägt werden. Das Resultat sind Passwort-Zettel, die für jeden sichtbar am Monitor kleben und der ursprüngliche Zweck des sicheren Passworts geht verloren.

[11] Quelle: `https://www.vmware.com/support/developer/vmci-sdk/VMCI-510-Relnotes.html`

Räumliche Zugangsbeschränkung

Unautorisierten Personen muss der physikalische Zugang zu Servern verweigert werden, indem die Geräte ausschließlich in abgesperrten Serverräumen betrieben werden. Reinigungspersonal zählt ebenfalls zur unautorisierten Personengruppe. Die Reinigung des Serverraums zählt zu den Aufgaben der Administratoren oder darf nur unter deren Aufsicht durchgeführt werden.

Schlüssel oder Transponder für das Türschloss dürfen nur an Personen ausgegeben werden, die feste Mitarbeiter der IT-Administration sind. Studenten oder Praktikanten ist der Zugang zu verweigern. Falls ein Generalschlüssel vorhanden ist, muss seine Ausgabe streng reglementiert sein und in jedem Fall in einer Ausgabeliste dokumentiert und vom Empfänger unterschrieben werden. Die Ausgabe des Schlüssels darf nur im Notfall und ausschließlich an feste Mitarbeiter erfolgen. Sollte der Generalschlüssel verloren gehen, entsteht ein großer logistischer und organisatorischer Aufwand und damit verbunden hohe Kosten. Es müssen schließlich unverzüglich sämtliche Türschlösser ausgetauscht und neue Schlüssel an die Mitarbeiter ausgehändigt werden. Die Zutrittsbeschränkung der Serverräume hat in diesem Fall eine hohe Priorität.

Datenträgerverschlüsselung

Um Vertraulichkeit der Daten auf einem Server zu gewährleisten, kann Verschlüsselung eingesetzt werden. Es können einzelne Dateien, Partitionen oder ganze Laufwerke verschlüsselt werden. Verschlüsselung ist ein wirksames Mittel gegen den Verlust von Datenträgern durch Diebstahl, denn gerade bei Servern mit *hotswap*-fähigen Laufwerken, die an der Gerätevorderseite ohne zusätzliches Werkzeug zugänglich sind, sind Laufwerke bei unzureichender Zutrittsbeschränkung einfach zu entwenden.

Es gibt verschiedene kommerzielle Produkte zur Festplattenverschlüsselung auf dem Markt. Welches Produkt auf einem Server installiert werden kann, hängt auch von rechtlichen Aspekten ab. So gibt es Szenarien, in denen Projektpartner einer Behörde vertraglich dazu verpflichtet werden, ein vom BSI zertifiziertes Verschlüsselungsprodukt wie zum Beispiel *Sirrix TrustedDisk* einzusetzen [SIR 13].

Transportverschlüsselung

Da Anwender nicht direkt auf dem Server arbeiten, sondern über ein Rechnernetz auf den dort angebotenen Dienst zugreifen, muss auch auf dem Übertragungsweg Vertraulichkeit durch Verschlüsselung gewährleistet sein. In der Praxis setzt man die folgenden Verschlüsselungsverfahren ein:

- Symmetrische Verschlüsselung
- Asymmetrische Verschlüsselung
- Hybride Verschlüsselung

Bei den *symmetrischen Verschlüsselungsverfahren* wird sowohl für die Ver- als auch die Entschlüsselung der gleiche Schlüssel verwendet. Dieser darf nur den beiden Kommunikationspartnern bekannt sein. Die Entschlüsselung kann man sich im einfachsten Fall als eine Umkehrfunktion der Verschlüsselung vorstellen, mit dem symmetrischen Schlüssel als gemeinsamen Parameter.

In der Einfachheit dieses Verfahrens liegt auch ein Problem: Da beide Instanzen den symmetrischen Schlüssel kennen müssen, müssen sie ihn zuvor über einen sicheren Kanal austauschen. Der gleiche Übertragungsweg wie bei der eigentlichen Kommunikation kommt hierfür nicht in Frage, da die Gefahr des Abhörens besteht. Ein weiterer Nachteil ist die Erfordernis, für größere bzw. längere Übertragungen in regelmäßigen Abständen neue Schlüssel generieren, austauschen und verwenden zu müssen. Dies liegt darin begründet, dass eine Kryptoanalyse immer einfacher wird, je öfter der gleiche Schlüssel für unterschiedliche Inhalte verwendet wurde. Ein Angreifer könnte so irgendwann den Schlüssel berechnen und sowohl die aufgezeichnete als auch die zukünftige Kommunikation der beiden Parteien entschlüsseln.

Im Gegensatz zur symmetrischen Verschlüsselung kommen bei *asymmetrischen* Verfahren zwei unterschiedliche Schlüssel zur Ver- und Entschlüsselung zum Einsatz. Jeder Kommunikationspartner ist im Besitz eines Schlüsselpaars, das aus dem öffentlich hinterlegten *Public Key* und dem nur einem selbst bekannten *Private*

Key besteht. Damit nur der Empfänger eine Nachricht entschlüsseln kann, verschlüsselt sie ein beliebiger Absender mit dem Public Key des Empfängers. Da nur der Empfänger im Besitz seines eigenen Private Keys ist, kann nur er die an ihn adressierte Nachricht entschlüsseln.

Der große Vorteil asymmetrischer Verfahren ist, dass nicht ständig Schlüssel zwischen den Instanzen über einen gesicherten Kanal ausgetauscht werden müssen. Man besorgt sich lediglich ein einziges Mal den Publik Key des Kommunikationspartners. Leider sind asymmetrische Verschlüsselungsverfahren deutlich rechenaufwendiger als ihre symmetrischen Pendants. Aus Performancegründen setzt man zum Beispiel bei der Transportverschlüsselung TLS in der Praxis auf ein *hybrides Verfahren*. Für die eigentliche Kommunikation wird ein performantes symmetrisches Verfahren benutzt. Für den relativ geringen Traffic, der beim regelmäßigen Schlüsselaustausch stattfindet, wird ein asymmetrisches Verfahren angewandt. Dort sind die Performance-Einbußen vor allem für Server, die mehrere Clients gleichzeitig bedienen müssen, tolerierbar.

Seit einiger Zeit sind Server mit Mainboards erhältlich, die ein sog. *Trusted Platform Module (TPM)* besitzen. Ein TPM ist in der Lage, verschiedene Sicherheitsfunktionen auszuführen. Dazu zählen die Generierung von Schlüsseln und Zufallszahlen sowie die Speicherung von Private Keys. Private Schlüssel verlassen das TPM unter keinen Umständen. Festplattenverschlüsselungssysteme können beispielsweise auf ein TPM zurückgreifen.

Transportverschlüsselung ermöglicht die sichere Übertragung von Informationen aus der Anwendungsschicht, indem neben der Vertraulichkeit auch die Integrität und die Authentizität der übertragenen Daten sichergestellt werden können. Sie wird in der Regel durch das Protokoll SSL bzw. seinen zu bevorzugenden Nachfolger TLS gewährleistet.

Die bloße Aktivierung der Transportverschlüsselung auf einem Server reicht nicht aus, um die Sicherheit zu optimieren. Das BSI empfiehlt beispielsweise, SSL überhaupt nicht mehr zu verwenden und stattdessen konsequent auf TLS Version 1.2 zu setzen.[12] Ebenso ist darauf zu achten, dass der Server nur möglichst sichere Cipher Suites gestattet und der Server diejenige Instanz sein muss, die die Sicherheitsmesslatte beim Verbindungsaufbau festlegt. Detaillierte Empfehlungen sind ein einer technischen Richtlinie vom BSI nachzulesen [BSI 15].

Angreifer auf das Firmennetz und Ausspäher von Firmengeheimnissen sind laut dem *Bundesamt für Verfassungsschutz (BfV)* in 70% aller Fälle Innentäter [BFV 14]. Erfolgt die Administration von Servern über ein Netz, auf das auch der Innentäter Zugriff hat, kann er beispielsweise an Administratorpasswörter gelangen, wenn ein unverschlüsseltes Protokoll verwendet wird. Aus diesem Grund empfiehlt sich für die Administration der Server die Einrichtung eines eigenen Management-Netzes. In diesem Fall spricht man von sog. *Out-of-Band Management*.

Generell sollen verschlüsselte Kommunikationsprotokolle verwendet werden, wo Benutzerdaten ausgetauscht werden. Das ist zum Beispiel der Fall, wenn ein zentraler Verzeichnisdienst zur Authentifizierung der Mitarbeiter benutzt wird. In diesem Fall bietet sich zum Beispiel das verschlüsselte Protokoll *Lightweight Directory Access Protocol over Secure Sockets Layer (LDAPS)* an. Ob vereinzelte Systeme weiterhin die unverschlüsselte Protokollvariante verwenden, könnte anhand einer dynamisch generierten Visualisierung überprüft werden.

Etwas unglücklich ist die Situation bei Dateiaustauschprotokollen. Das in Windows-Landschaften vorherrschende Protokoll *Server Message Block (SMB)* unterstützt Verschlüsselung erst seit Version 3.0, welche mit Windows 8 und Windows Server 2012 eingeführt wurde. In vielen Firmen wird auf den Clients aber trotz demnächst auslaufendem “grundlegendem Support” weiterhin Windows 7 eingesetzt [KNO 14]. Um die Vertraulichkeit bei der Nutzung unsicherer Protokolle dennoch zu gewährleisten, können *Tunnel* eingesetzt werden wie zum Beispiel *Secure Shell (SSH)-Tunnel*. Der Einsatz von *Internet Protocol Security (IPsec)* zur verschlüsselten Datenübertragung zwischen unterschiedlichen Standorten ist ebenfalls denkbar.

[12] Das BSI verbietet in seiner technischen Richtlinie TR-02102-2 die Verwendung von TLS 1.0 und empfiehlt TLS 1.1 nur, falls der Server nicht die aktuelle Version 1.2 unterstützt [BSI 15]. TLS 1.2 bringt zwar nützliche Verbesserungen mit, durch das Deaktivieren von TLS 1.1 und 1.0 auf Serverseite ergeben sich aber andererseits neue Probleme. Der Administrator sperrt durch diese vom BSI empfohlene Maßnahme eine Menge Clients aus, wie beispielsweise auf heise.de nachzulesen ist [SCHM 15].

Firewalls

Eine *Firewall* ist ein Sicherheitssystem, das einzelne Rechner oder ganze Rechnernetze vor unerwünschten Zugriffen über das Netz schützt. Im Grunde besteht eine Firewall aus einer Software, die auf einer speziellen Appliance, einem Server oder einem anderen vernetzten Computer ausgeführt wird und anhand bestimmter Regeln Pakete weiterleitet oder nicht. Die gezielte Manipulation von Paketen vor dem Weiterleiten ist ebenfalls möglich.

Bei der Konfiguration von Firewall-Regeln werden im Allgemeinen zwei grundlegende Ansätze unterschieden:

- Blacklist
- Whitelist

Eine *Blacklist* weist die Firewall-Software an, prinzipiell jede Art von Paketen weiterzuleiten, mit Ausnahme derjenigen Pakete, für die es eine explizite Filterregel in der Blacklist gibt. Auf diese Weise können einzelne Dienste ohne großen Aufwand gesperrt und bei Bedarf wieder freigegeben werden. Ein Nachteil reiner Blacklist-Ansätze ist, dass nicht von Anfang an klar ist, wer oder was alles ausgesperrt werden muss, um maximale Sicherheit zu gewährleisten.

Anwendung finden Blacklists zum Beispiel, um auf Servern einzelne Hosts auszusperren, die Brute-Force-Angriffe auf einen dort laufenden Dienst durchführen. In diesem Fall kann die Blacklist auch vom System dynamisch generiert werden. Dynamisch generierte Blacklists werden auch zur Bekämpfung von Spam eingesetzt. *Sinha et al.* haben vor einigen Jahren eine Studie über die Effektivität solcher sog. *reputationsbasierten* Firewalls durchgeführt [SIN+ 08]. Das Ergebnis der Studie war, dass die getesteten Systeme eine hohe False-Negative-Rate und darüber hinaus eine unerwartet hohe False-Positive-Rate aufwiesen. Einen Verbesserungsvorschlag für reputationsbasierte Verfahren veröffentlichte das Team um Sinha zwei Jahre später [SIN+ 10].

Als Alternative zur Blacklist gibt es die *Whitelist*. Im Gegensatz zur Blacklist weist sie die Firewall-Software an, standardmäßig überhaupt keine Pakete weiterzuleiten. Lediglich Traffic, für den ein Filterkriterium in der Whitelist vorhanden ist, darf passieren. Whitelists bieten im Allgemeinen ein hohes Maß an Sicherheit, da der Administrator sukzessive Regeln definiert, die nur absolut notwendige Verbindungen zulassen. Dies ist gleichzeitig auch ein Nachteil, da der Zeitaufwand auch höher ist. Noch dazu müssen Whitelists im Idealfall regelmäßig nach Karteileichen durchsucht und diese entfernt werden. In einem Unternehmensnetz kann es schließlich immer wieder vorkommen, dass für einen Mitarbeiter bestimmte Regeln hinzugefügt werden, damit er zum Beispiel eine Verbindung zu einem Testsystem eines Projektpartners herstellen kann. Spätestens nach der Beendigung des Projekts wird diese Regel nicht mehr benötigt, verweilt aber dennoch sehr lange in der Firewall-Konfiguration.

Es wurde bereits erwähnt, dass ein nicht zu vernachlässigender Anteil von Firmen-Hacks von den eigenen Mitarbeitern durchgeführt wird, um an vertrauliche Informationen zu gelangen. Aus diesem Grund müssen *Paketfilter-Firewalls* nicht nur an kritischen Netzübergängen wie etwa hinter dem Perimeter-Router, sondern auch auf den Servern selbst eingesetzt werden. Selbst wenn diese Server über eine zusätzliche (möglicherweise virtuelle) Ethernet-Schnittstelle für ein Management-Netz verfügt, macht es Sinn, bestimmte Dienste nur für wenige Administrator-PCs zuzulassen, denn ein Server könnte durch einen Konfigurationsfehler im falschen Netz landen oder der Angreifer platziert einen seiner Rechner im Management-Netz.

Zu den Diensten, die per lokalem Paketfilter auf die IP-Adressen der Admin-PCs beschränkt werden müssen, zählen beispielsweise SSH oder die Management-Oberfläche eines *Application Servers*. Das Öffnen von *Telnet*-Ports ist heutzutage aber nicht mehr zu empfehlen, auch wenn aktuell noch an der "Sicherheit" von Telnet-Implementierungen gearbeitet wird [EIK 15]: Die fehlende Verschlüsselung macht dieses Protokoll obsolet.

Einige Server-Prozesse werden möglicherweise nur auf dem Server selbst benötigt. Dabei handelt es sich zum Beispiel um einen Datenbank-Dienst, der nur von einer einzigen Webanwendung benutzt wird, die auf demselben Server betrieben wird. In diesem Fall sollte der Dienst so konfiguriert werden, dass er gar nicht erst auf den Ethernet-Schnittstellen nach Verbindungsversuchen lauscht, sondern nur auf dem internen *Loopback Interface*. Sicherheitskritische Abweichungen in Firewall-Konfigurationen mehrerer Linux-Server festzustellen wäre ein weiterer möglicher Anwendungsfall zur Betrachtung in dieser Ausarbeitung.

3.4.2. Integrität

Zur Gewährleistung der Integrität von Daten im Kontext von Linux-Servern eignen sich die folgenden Maßnahmen, die im Anschluss näher beschrieben werden:

- Zugriffsbeschränkungen
- Speicherrandomisierung
- Integrity Checker
- Minimalsystem mit Ausführungskontrolle

Zugriffsbeschränkungen

Die naheliegendste Methode, um Manipulationen von Dateien durch Unbefugte zu verhindern, besteht in eingeschränkten Zugriffsrechten für bestimmte Personengruppen. Dies wirkt sich auch auf die Vertraulichkeit des Systems aus. Anstelle individueller Rechte ist es ratsam, Benutzergruppen oder Rollen zu verwenden, um die Übersicht nicht zu verlieren. Abgesehen von lokalen Administratorkonten empfiehlt es sich darüber hinaus eine zentrale Benutzerverwaltung zu verwenden. Das verringert den Verwaltungsaufwand und wirkt vergessenen Benutzerkonten entgegen. Um solche verwaisten Accounts schnell aufspüren zu können, würde sich eine dynamisch generierte Visualisierung gut eignen. Da die hier genannten Methoden zur Zugriffsbeschränkung weitverbreitet sind und einen wichtigen Beitrag zur Sicherheit leisten, soll auch der zu entwickelnde Prototyp diese idealerweise bereitstellen.

Speicherrandomisierung

Die Datenintegrität auf Servern wird auch durch Schadprogramme gefährdet. Je nach Ausprägung können sie Dateien manipulieren oder sogar löschen und sich selbst auf andere Server ausbreiten. Um Schadprogramme aufzuspüren und zu beseitigen, sind Virenschutzprogramme sowohl auf den Servern als auch den Clients unerlässlich. Ein Angreifer kann Dateien auf einem Server aber auch manuell beschädigen oder austauschen, ohne sich auf einen Virus verlassen zu müssen. Der Angreifer könnte einen sog. *Buffer Overflow* herbeiführen, um das System zu kompromittieren. Neben einer sorgfältigen Prüfung der Speicheradressen und Bereichsgrenzen im Programmcode können Buffer Overflows zur Laufzeit erschwert werden, wenn das Betriebssystem *Address Space Layout Randomization (ASLR)* unterstützt. Durch ASLR erhalten Programme zufällige Adressbereiche zugeteilt, sodass Angreifer nicht mehr deterministisch herausfinden können, wo manipulierte Nutzdaten im Arbeitsspeicher tatsächlich abgelegt werden. Die ASLR-Technologie wird von vielen modernen Betriebssystemen unterstützt. Dazu zählen Microsoft Windows ab Vista, Linux und Mac OS X. Eine ASLR-Implementierung ist auch bei den weitverbreiteten mobilen Betriebssystemen *iOS* und *Android* vorhanden.

Obwohl der Performanceverlust von Programmen durch ASLR marginal ist, setzen Softwarehersteller diese Schutztechnik nicht konsequent ein. Ende des Jahres 2014 stellte das Antivirentestlabor *AV-Test* fest, dass nur zwei von insgesamt 32 untersuchten Internet-Security-Produkten ASLR und *Data Execution Prevention (DEP)* optimal beherrschen [SCHM 14B]. Ähnliches gilt laut einem Bericht von *golem.de* auch für die Programme populärer Linux-Distributionen. Grund sei der Verzicht auf bestimmte Compiler-Optionen [BOE 14].

Stack Canaries

Um Buffer Overflows zu erkennen, können auch sog. *Stack Canaries* eingesetzt werden. Ein Compiler reserviert dazu im Stack zwischen der Rücksprungadresse und den lokalen Variablen Platz für eine Zufallszahl, die bei jedem Programmstart unterschiedlich ist. Diese Zahl wird bei jedem Unterprogrammaufruf in den reservierten Bereich geschrieben. Vor dem Verlassen des Unterprogramms wird die Zufallszahl überprüft. Wenn sie einen abweichenden Wert besitzt, wurde mit großer Wahrscheinlichkeit auch die Rücksprungadresse von einem Angreifer überschrieben und ihr kann nicht mehr vertraut werden. Dieses Verfahren ist deshalb möglich,

da bei Buffer Overflows Speicherbereiche von niedrigen zu höheren Adressen überschrieben werden. Um eine Rücksprungadresse zu manipulieren, wird also zwingend auch die Zufallszahl überschrieben. Durch die Ausnutzung einer bekannten Lücke könnte ein Angreifer auch wichtige Systemprogramme austauschen, zum Beispiel den SSH-Client, der dann eingegebene Passwörter an den Angreifer weiterleitet.

Integrity Checker

Um die Integrität von Programmen und Konfigurationsdateien sicherzustellen, setzt man auf Servern sog. *Integrity Checker* ein. Der Integrity Checker *Samhain*[13] nutzt beispielsweise kryptographische Prüfsummen, um Änderungen an Dateien festzustellen und spürt außerdem Programme mit unerwartet gesetztem *Set User ID (Setuid) Flag* auf. Das Programm unterstützt verschiedene Techniken, um seine eigene Existenz auf dem System vor Angreifern zu verbergen [SPE 05].

Einige Betriebssysteme bieten die Möglichkeit an, Dateien mit sog. *Dateisignaturen* zu versehen. Je nach Ausprägung findet die Integritätsprüfung der Dateien vor dem Öffnen oder manuell statt. Manche Hersteller statten bestimmte Dateien bereits ab Werk mit einer Signatur aus, dazu zählen beispielsweise Gerätetreiber.

Minimalsystem mit Ausführungskontrolle

Um den Aufwand der Integritätsprüfung gering zu halten und die möglichen Angriffsflächen zu verkleinern, bietet es sich bei der Installation des Servers bereits an, nur die für den Betrieb erforderlichen Komponenten zu installieren. Falls das nicht möglich ist, müssen nach der Installation nicht benötigte Komponenten nachträglich entfernt werden. Man spricht in diesem Zusammenhang von einem *Minimalsystem*. Zur Sicherstellung der Integrität dürfen generell nur Programme auf dem System gestartet werden, die für den Betrieb erforderlich sind. Eine *Ausführungskontrolle* verhindert den Start von Fremd- oder Schadsoftware. Auf Linux-Systemen müssen Benutzerverzeichnisse derart konfiguriert werden, dass eine Ausführung darin enthaltener Programme unterbunden wird. Durch geeignete Visualisierungen könnte die konsequente Einhaltung dieser integritätssichernden Maßnahmen überwacht werden.

Die *Secure-Boot*-Funktion aktueller Hauptplatinen mit einem sog. *Unified Extensible Firmware Interface (UEFI)*-Chip ermöglicht es, nur Bootloader beim Computerstart zu akzeptieren, die eine gültige digitale Signatur besitzen. Selbst vor der Erstinstallation eines Servers kann man einen Beitrag zur Integritätssicherung leisten, indem nur Installationsmedien aus vertrauenswürdigen Quellen verwendet werden. Für aktuelle Linux-Versionen gibt es allerdings kaum Anbieter von Datenträgern. Hier bietet sich eine Integritätsprüfung der aus einer offiziellen Distributions-Homepage heruntergeladenen Image-Datei anhand der online einsehbaren *Message-Digest Algorithm 5 (MD5)*- oder *Secure Hash Algorithm (SHA)-Prüfsummen* an.

3.4.3. Verfügbarkeit

Im Idealfall funktioniert ein Server bis zu seiner Stilllegung fehlerfrei, kommt mit jedem beliebigen Lastaufkommen zurecht und reagiert augenblicklich auf Anfragen durch die Benutzer. Das ist leider ein Wunschdenken, denn Kostenfaktoren wie Rechenleistung, Übertragungsraten, Speicherplatz, Stellplätze und vorhandene Backup-Systeme müssen immer berücksichtigt werden. Auch Angriffe können zu Ausfällen führen. Je nach angebotenem Dienst und den *Service-Level Agreements (SLAs)* dürfen bestimmte Ausfallzeiten nicht überschritten werden. Die Verfügbarkeit eines Systems in Prozent berechnet sich wie folgt:

$$\textit{Verfügbarkeit} = \frac{\textit{Gesamtzeit} - \textit{Ausfallzeit}}{Gesamtzeit} \cdot 100 \tag{3.1}$$

Zur Steigerung der Verfügbarkeit von Servern bieten sich im Allgemeinen folgende Maßnahmen an:

- Redundanz
- Failover

[13] Samhain-Website: `http://www.la-samhna.de/samhain/`

- Loadbalancing

Unter *Redundanz* versteht man das mehrfache Vorhandensein von gleichen oder gleichartigen Bestandteilen eines Systems. Im Bezug auf Server kann Redundanz seine Komponenten umfassen und/oder das gesamte Gerät. Im Fall von Einzelkomponenten sind beispielsweise Festplatten mehrfach vorhanden. Angeschlossen an einen *Redundant Array of Independent Disks (RAID) Controller* toleriert das System je nach gewähltem RAID Level den Ausfall einzelner Laufwerke. Dies ersetzt zwar kein Backup, verhindert aber den Ausfall des Systems während die Platte getauscht wird. Zu den weiteren Bestandteilen, die häufig redundant ausgelegt sind, zählen Netzteile, Netzwerkkarten und Arbeitsspeicher. Durch ihre sog. *Hot-Swap*-Fähigkeit ist der Austausch der ausgefallenen Komponente im laufenden Betrieb möglich.

Multipathing

Die Ausfallsicherheit von Servern in *Storage Area Networks (SANs)* kann durch sog. *Multipathing* erhöht werden. Zu diesem Zweck besitzt ein Server mehrere *Host Bus Adapter (HBA)*, die über unterschiedliche Pfade das Speichersystem anbinden. Die SAN-Technologie *Fibre Channel* setzt spezielle HBAs ein, eine redundante Anbindung von Speichersystemen über preiswertere Ethernet-Schnittstellen ist auch denkbar.[14]

Redundante Hosts

Ganze Server redundant zu betreiben bietet sich zur Kompensation von Totalausfällen an. Idealerweise befinden sich die Systeme an unterschiedlichen Standorten, um zum Beispiel im Fall eines Brands oder einer Naturkatastrophe noch betriebsbereit zu sein. Redundante Systeme können aber während der regulären Diensterbringung auch anderweitig verwendet werden. Der Einsatz als Testsystem, zum Beispiel zum Einspielen neuer Software-Updates, ist denkbar. Wer eine geeignete Virtualisierungsplattform wie VMware ESXi einsetzt, kann VMs sogar im laufenden Betrieb auf einen anderen Host verschieben, um danach auf dem vorübergehend leergeräumten Server Updates einzuspielen. In diesem Zusammenhang spricht man auch von *Live Migration*.

Failover setzt die zuvor erwähnte Redundanz voraus. Beim Ausfall eines Systems übernimmt ein redundantes System automatisch seine Aufgabe. Die Reaktionszeit ist idealerweise so gering, dass Anwender keine Beeinträchtigung wahrnehmen. Wer redundante Systeme betreibt, die für einen Failover konfiguriert sind, muss zur Erhöhung der Verfügbarkeit den Ernstfall einmal durchspielen und testen, ob der Failover auch tatsächlich klappt.

Redundante Hosts sind nur dann nützlich, wenn auch aus elektrischer Sicht ein sicherer Betrieb sichergestellt ist. Bei der National Security Agency (NSA) entstanden beispielsweise Schäden in der Höhe mehrerer hunderttausend Dollar, da es Probleme mit der Spannungsversorgung in einem neuen Rechenzentrum gab und zahlreiche Geräte dadurch beschädigt wurden [KAH 13].

Lastverteilung

Je mehr Client-Anfragen ein Server bearbeiten muss, desto eher stößt er an seine Leistungsgrenzen. Zunächst verringert sich seine Performance, im schlimmsten Fall verweigert er den Dienst bei zu hohem Lastaufkommen vollständig. Betreiber stark frequentierter Websites wie *Google* oder *Facebook* sind besonders hohen Gesamtlasten und auch Lastschwankungen ausgesetzt, mit denen die Firmen zurechtkommen müssen. Wenn ein einziger Server die Anfragen nicht mehr alleine beantworten kann, verteilt man das Lastaufkommen durch *Loadbalancing* auf mehrere Systeme. Es muss sich dabei um gleichartige Systeme handeln, die den gleichen Dienst erbringen können. Die Gesamtmenge solcher Systeme wird in der Praxis als *Cluster* bezeichnet. Ein sog. *Loadbalancer* wird den Servern vorgeschaltet und verteilt die Anfragen nach bestimmten Kriterien auf das Cluster.

Die Verfügbarkeit von Systemen, die öffentlich zugänglich am Internet hängen, wird besonders durch potenzielle DoS-Angriffe bedroht. Angreifer nutzen oft die Phase des Verbindungsaufbaus aus, da dann noch keine

[14] Genau genommen zählen Ethernet-Erweiterungskarten auch zur Klasse der HBAs. Der Begriff HBA wird in der Praxis aber häufig mit Fibre Channel und ähnlichen Technologien in Verbindung gebracht.

Sicherheitsbeziehung zwischen Client und Server aufgebaut ist. Der Angreifer kann in diesem Fall sehr viele Verbindungsanfragen an den Server senden, bis der Server-Anwendung dafür notwendige Ressourcen ausgehen. Einen vollständigen Schutz vor solchen Angriffen gibt es derzeit nicht. Als Gegenmaßnahme bietet sich aber an, die Phase des Verbindungsaufbaus deutlich restriktiver zu gestalten, indem man zum Beispiel mit *Sperrlisten* für bösartige Clients oder *Synchronize (SYN)-Cookies* arbeitet. Plötzlichen Lastaufkommen kann – wenn auch teuer – durch Serverlastverteilung und Virtualisierung entgegengewirkt werden.

Installation von Patches

Viele Angriffe auf Server nutzen Sicherheitslücken in Software aus. Besonders bei den Linux-Derivaten ist aufgrund der breit aufgestellten Open-Source-Gemeinschaft die Reaktionszeit von der Entdeckung einer Lücke und der Bereitstellung des ersten Patches sehr gering. Updates stehen oft schon am selben Tag zum Download bereit. Einen essentiellen Beitrag zur Verfügbarkeit ihrer Systeme leisten Administratoren aber nur dann, wenn Updates zeitnah auf den betroffenen Servern installiert werden. Generell dürfen Updates nur aus vertrauenswürdigen Quellen bezogen werden und unter keinen Umständen irgendwelche Programme ausgeführt werden, die von zweifelhafter Herkunft sind. Paketverwaltungssysteme arbeiten mit digitalen Signaturen, um den Ursprung eines Pakets nachvollziehen zu können.

Auch trotz aktuellem Softwarestand kann eine geschlossene Lücke noch zeitraubende Nacharbeit zur Folge haben. Als Beispiel sei an dieser Stelle der im April 2014 gefundene *Heartbleed Bug* in der Kryptographie-Bibliothek *OpenSSL* genannt [SCHM 14]. Das Programm implementierte eine sog. *Heartbeat*-Funktion, mit dem ein Rechner die Erreichbarkeit der Gegenstelle überprüfen kann. Aufgrund eines fehlerhaft programmierten Speicherzugriffs konnten Angreifer durch wiederholtes Senden manipulierter Heartbeat-Anfragen bis zu 64 KByte große Speicherinhalte des Servers abrufen und so mit ein wenig Glück an geheime Daten wie etwa den privaten Schlüssel des Servers gelangen. Durch den Besitz des privaten Schlüssels lassen sich alle bereits zuvor aufgezeichnete Daten und auch der Verkehr aktueller Verbindungen entschlüsseln, sofern der Server nicht für *Perfect Forward Secrecy (PFS)* konfiguriert ist. Da die Private Keys der betroffenen Server potenziell öffentlich zugänglich waren, mussten die Sicherheitszertifikate etlicher Systeme ausgetauscht werden – mit Auswirkungen auf ihre Verfügbarkeit.

3.5. Zusammenfassung

In verteilten und vernetzten Systemen ist die Informationssicherheit ständig Bedrohungen ausgesetzt. Dies gilt auch für das Linux-Server-Netz am LRZ. In diesem Kapitel wurden potenzielle Bedrohungen für Server vorgestellt (siehe Abschnitt 3.1 und 3.2) und welche Sicherheitsziele daraus resultieren (siehe Abschnitt 3.3). Die konkrete Umsetzung der Sicherheitsziele erfolgt durch ein Zusammenspiel ausgewählter Maßnahmen, von denen die wichtigsten in Abschnitt 3.4 vorgestellt wurden. Der im Rahmen dieser Masterarbeit anzufertigende Prototyp soll einige der aufgezählten Maßnahmen implementieren. Er soll unter anderem einen Beitrag dazu leisten, bei den Anwendern ein Bewusstsein für Informationssicherheit zu schaffen.

Um die Sicherheitsziele Vertraulichkeit, Integrität und Verfügbarkeit umzusetzen, müssen Maßnahmen ergriffen werden, die in Abschnitt 3.4 genannt wurden. Mit der Einführung einiger Maßnahmen kann ein hoher Aufwand verbunden sein. Hier muss sinnvollerweise eine Kosten-/Nutzenrechnung durchgeführt werden.

Es gibt auch Maßnahmen, die nur mit verhältnismäßig geringem Aufwand verbunden sind. Dazu zählt beispielsweise der Betrieb von Minimalsystemen, auf denen exakt nur die Software installiert ist, die für den täglichen Betrieb benötigt wird. Sie machen nicht nur auf Servern, sondern auch im privaten Umfeld Sinn, da sie kostengünstig umzusetzen sind und eine geringe Angriffsfläche besitzen.

Administratoren müssen sich allerdings über mögliche Konsequenzen bewusst sein, die auftreten können, wenn eine große Menge von Minimalsystemen für einen erweiterten Kundenkreis angeboten wird: Je geringer die Anzahl an vorinstallierten Paketen ist, desto wahrscheinlicher sind Sonderwünsche von den Nutzern. Dies ist beispielsweise bei Webhostern der Fall. Wenn das Angebot vorinstallierter Software zu beschränkt ist, müssen ggf. spezielle Wünsche einzelner Kunden berücksichtigt werden. Dadurch steigt der administrative Aufwand. Andererseits kann man nicht im Voraus alle möglichen Bedürfnisse der Kunden erfüllen, da

hierdurch die Angriffsfläche der Systeme wieder vergrößert wird. Eine Kosten-/Nutzenrechnung ist bei der Zusammenstellung der Minimalsysteme ratsam.

Datensparsamkeit sollte auch außerhalb eines Rechenzentrums praktiziert werden und kostet unter dem Strich weniger Zeit und Geld, als mit den Konsequenzen von Datenmissbrauch konfrontiert zu werden. Unabhängig davon, welche Maßnahmen in Betracht gezogen werden, muss stets berücksichtigt werden, dass sie einerseits nicht zu liberal ausgelegt werden, da sie sonst ihre Wirkung verlieren, und andererseits nicht zu streng sind, um die Bedienbarkeit von IT-Systemen nicht unnötig zu erschweren.

4. Anforderungsanalyse

Dieses Kapitel befasst sich mit der Analyse der Anforderungen des LRZ, die an grafische Administrationswerkzeuge gestellt werden, um eine effiziente Administration der Linux-Server zu gewährleisten und einen Beitrag zur Informationssicherheit zu leisten. Zu diesem Zweck wird zunächst auf die einzelnen Dienste und die Server-Infrastruktur des LRZ eingegangen (siehe Abschnitt 4.1). Im Anschluss erfolgt in Abschnitt 4.2 eine Übersicht der konkreten Anforderungen an ein Administrationstool zur Überwachung einer großen Zahl an Linux-Servern.

4.1. Dienstleistungskatalog des LRZ

Dieser Abschnitt verschafft einen Überblick der Ausgangslage des technischen Umfelds am LRZ. Die Informationen sind öffentlich zugänglich und stammen aus der zur Zeit der Anfertigung dieser Arbeit aktuellen Ausgabe des offiziellen Jahresberichts von 2013 [LRZ 14].

4.1.1. IT-Basisdienste

Im Jahr 2013 waren im Münchner Wissenschaftsnetz (MWN) knapp 500 E-Mail-Domains im Einsatz, die im Laufe der Zeit auf eine *Postfix*-Infrastruktur migriert wurden. *Prequeue*-Filterung stellt sicher, dass Mail Server potentielle Spam- und Viren-Nachrichten anhand der Header-Informationen bereits vor dem Empfang der eigentlichen E-Mail ablehnen können. Nach dem Empfang werden die Nachrichten in der Postqueue-Filterung ein weiteres Mal untersucht und ggf. für den Anwender sichtbar als Spam gekennzeichnet. Dies ist erforderlich, da ein Server aus rechtlichen Gründen eine E-Mail, deren Empfang er bereits quittiert hat, nicht mehr nachträglich löschen darf. Die Filterung findet hauptsächlich an den *Relay Servern* des LRZ statt, da nur sie in der Lage sind, E-Mails direkt aus dem Internet entgegenzunehmen. Forschungseinrichtungen, die keinen eigenen Mail Server betreiben möchten, bietet das LRZ virtuelles Mailhosting an.

Neben den Linux-Mail-Systemen wird auch das auf Windows Server basierende Groupware-System *Microsoft Exchange* angeboten. Dieser Dienst erfreut sich laut der Statistiken des Jahresberichts wachsender Beliebtheit. Als Ergänzung dazu stellt das LRZ auch die webbasierte Kollaborations- und Intranet-Plattform *Microsoft SharePoint* zur Verfügung.

Ein weiterer wichtiger Basisdienst, den das LRZ seit rund 20 Jahren anbietet, ist das *Webhosting*. Dem historischen Wachstum ist es geschuldet, dass viel Arbeit in die Neuorganisation der damit verbundenen Datenspeicherung und der Verwaltung der gehosteten Seiten gesteckt werden musste.

Zum Management der zahlreichen Windows-Desktops setzt das LRZ *Microsoft System Center Configuration Manager (SCCM)* ein. Es ermöglicht die Installation des Betriebssystems über bereits vorhandene Systeme und auf noch leeren Rechnern. Ein zentrales Software Management der verwalteten Clients ist damit ebenfalls möglich. SCCM wird am LRZ sowohl zur Verwaltung von Mitarbeiter-Rechnern als auch Server-Systemen und VMs eingesetzt.

Windows-Arbeitsplätze werden vom LRZ in unterschiedlichen Ausprägungen bereitgestellt. Zum Portfolio zählen vollwertige Workstations für Mitarbeiter, Terminalserverlösungen und virtuelle Umgebungen für Testzwecke. Rechnerpools und auf Terminalserver basierende Pools werden für Studenten zur Verfügung gestellt. Das LRZ betreut die Installation, kümmert sich um die Aktualität der Software und führt auch Monitoring-Maßnahmen durch.

4.1.2. Netzdienste für Endanwender

Die vom LRZ angebotenen Netzdienste orientieren sich an den Anforderungen der Endanwender, die sich hauptsächlich aus Studenten und Institutsmitarbeitern zusammensetzen. Neben einer stetig wachsenden Benutzerzahl konnte auch ein Anstieg bei der Nutzung mobiler Dienste verzeichnet werden.

Der Zugang zum *MWN* wird für die Endanwender über das *Deutsche Wissenschaftsnetz (WiN)* angeboten. Das *Deutsche Forschungsnetz (DFN)* hat das MWN direkt an seinen sog. *Super-Core* angebunden, um den hohen Bandbreitenanforderungen der Anwender gerecht zu werden. Er verbindet die Standorte Berlin, Erlangen, Hannover und Frankfurt mit 100 GBit/s.

Um Hörsäle, Seminarräume, Uni-Lounges, Bibliotheken und Foyers mit *Wireless Local Area Network (WLAN)* zu versorgen, waren Ende des Jahres 2013 fast 2400 Access Points in Betrieb. Aufgrund der zunehmenden Verbreitung von mobilen Geräten mit WLAN Interface konnte eine ständig zunehmende Nutzung der WLAN-Infrastruktur festgestellt werden. Ende des Jahres 2013 konnten in Stoßzeiten beinahe 20.000 gleichzeitige Verbindungen gemessen werden. Über 300.000 verschiedene Endgeräte konnten beobachtet werden. Seit 2005 nimmt das LRZ an *Education Roaming (eduroam)* teil, einer Initiative, die Studenten und Mitarbeitern der teilnehmenden Forschungsstätten WLAN-Zugang über einen zentralen Authentifizierungsdienst anbietet. Dadurch müssen die Nutzer bei einem Standortwechsel keine neuen Zugangsdaten beantragen und können sich sofort mit einem Funknetz mit der einheitlichen *Service Set Identifier (SSID)* `eduroam` verbinden.[1] Für Gäste können spezielle eduroam-Gastkennungen eingerichtet werden, die bis zu sieben Tage lang gültig sind. Für einzelne Veranstaltungen richtet das LRZ bei Bedarf auch WLANs ohne besondere Authentifizierung ein.

Zu den Netzdiensten für Endanwender zählen auch *Virtual Private Networks (VPNs)*. Mit Hilfe eines vorkonfigurierten VPN-Routers, den Mitarbeiter für Heimarbeit erhalten, können Telearbeitsplätze realisiert und gleichzeitig Informationssicherheit sichergestellt werden. Der sichere Zugang zu internen Diensten des MWN über öffentliche Ethernet Ports ist durch VPN ebenfalls möglich.

4.1.3. Benutzerverwaltung und Verzeichnisdienste

Studenten der LMU und der Technischen Universität München (TUM) erhalten Benutzerkennungen direkt von ihren Hochschulen. An Studenten anderer Münchener Hochschulen vergibt das LRZ direkt Benutzerkonten. Ende des Jahres 2013 zählte das LRZ fast 92.000 Accounts. Für das Identitätsmanagementsystem *Leibniz-Rechenzentrum Secure Identity Management (LRZ-SIM)* nutzt das Rechenzentrum ein Cluster aus *OpenLDAP*- und *NetIQ*-Servern. Im Jahr 2007 begann das LRZ mit dem Aufbau des mandantenfähigen *MWN Active Directory (AD)*. Damit können nicht nur Windows-, sondern auch Linux- und Mac-OS-Systeme von einer zentralen Benutzerverwaltung profitieren. Auf Datenschutz wird besonderer Wert gelegt. Diesbezüglich wurde im Lauf der Zeit die Transparenz gegenüber den Anwendern verbessert. Personenbezogene Daten werden fristgerecht gelöscht.

4.1.4. Informationssicherheit

Informationssicherheit ist nicht als etwas Isoliertes zu betrachten, da sämtliche Dienste und Abteilungen des LRZ davon betroffen sind. Es betreibt eine Service-Infrastruktur zur automatischen Verteilung von Virensignaturen der Firma *Sophos* für die Benutzer des MWN. Zur Aktualisierung der zahlreichen Windows-Installationen und anderer Microsoft-Programme bietet das Rechenzentrum einen zentralen *Windows Server Update Service (WSUS)* an.

Zur Authentifizierung von Servern und Benutzern nach X.509 betreibt das LRZ eine eigene *Certificate Authority (CA)* mit je einer *Registration Authority (RA)* für die Grid-CA der DFN-Policy Certification Authority (PCA) und für die CAs der beiden Münchener Universitäten. Die zuletzt genannte CA wird aktuell nur für Server verwendet, die direkt am LRZ gehostet werden.

[1] Eine automatische Verbindung zu WLANs mit der SSID `eduroam` sollte nur nach einer erfolgreichen Prüfung des Zertifikats stattfinden, um auszuschließen, im Netz eines Angreifers zu landen.

Auf den Backbone-Routern des MWN kommen Virtual Firewalls (VFWs) der Firma *Cisco* zum Einsatz, um mehr als 120 Kunden damit zu bedienen (Stand 2013). Zum Schutz vor Bedrohungen aus dem Internet kommen außerdem verschiedene Sicherheitswerkzeuge zum Einsatz. Eines davon ist ein automatisches proaktives *Intrusion Detection System (IDS)* mit dem Namen *Secomat*, das aus einem Cluster mit mehreren Nodes besteht, von denen jeder theoretisch $10GBit/s$ bewältigen kann. Ein weiterer Sicherheitsbestandteil ist das *Open Source Security Information Management (OSSIM)* mit seinen automatischen Auswertungs- und Alarmierungsfunktionen. Das Management-Werkzeug *Nyx* dient dazu, Switch Ports und die Bezeichnung von Netzwerkdosen anhand einer IP- oder MAC-Adresse zu ermitteln, um den Standort einzelner Rechner herauszufinden. Zudem gibt es eine zentrale Sperrliste mit einem webbasierten Self-Service-Portal.

4.1.5. Storage

Der Zugriff auf die *Network Attached Storage (NAS)*-Systeme erfolgt im *Local Area Network (LAN)* bzw. *Wide Area Network (WAN)* über die Protokolle *Common Internet File System (CIFS)* und *Network File System (NFS)*. Die Gesamt-Brutto-Kapazität für Online-Speicher lag im Jahr 2013 bei 18.161 TByte. An *Nearline*-Speicher waren 76.516 TByte vorhanden. Darunter versteht man Speicher wie z. B. Bänder, der nicht unmittelbar zugreifbar ist.

4.1.6. Serverbetrieb

Das *LRZ* stellt Kapazitäten seines Höchstleistungsrechners für Benutzer aus deutschen Hochschulen und europäischen Nutzern aus dem *Partnership for Advanced Computing in Europe (PRACE)*-Projekt zur Verfügung. Zu diesem Zweck besitzt es 18 Thin-Node-Inseln mit insgesamt 1032 Intel-Sandy-Bridge-Prozessoren vom Typ Xeon E5-2680 und fast 300.000 GByte Hauptspeicher. Eine weitere Fat-Node-Insel bestehend aus 205 IBM Blade x3850 greift auf insgesamt 820 Intel-Xeon-E7-4870-Prozessoren und 52.480 GByte Random Access Memory (RAM) zurück. Zu den Höchstleistungsrechnern gesellt sich noch ein Testsystem aus 32 Knoten. Jeder dieser Knoten besitzt zwei Intel-Ivy-Bridge-Prozessoren mit je acht Prozessorkernen und zwei Knights-Corner-Many-Core-Prozessoren von Intel mit jeweils 60 Cores.

Neben den Höchstleistungsrechnern verfügt das LRZ auch über ein heterogenes Linux-Cluster für Hochleistungsrechenaufgaben. Eine detaillierte Aufschlüsselung dieses Linux-Clusters ist dem Jahresbericht [LRZ 14] zu entnehmen. Zusammenfassend ist zu sagen, dass er sich aus insgesamt 1100 Komponenten mit 23,5 TByte Hauptspeicher zusammensetzt. Die einzelnen Komponenten dieses heterogenen Clusters sind mit unterschiedlichen Technologien wie zum Beispiel *10-GBit-Ethernet*, *NUMAlink*, und *InfiniBand* miteinander verbunden. Der Cluster dient zur Ausführung herkömmlicher Linux-Anwendungen und von Programmen, die durch *Message Passing Interface (MPI)* oder *OpenMP* parallelisierbar sind.

Für die aufwendigen Grafikberechnungen großformatiger und dreidimensionaler Projektionssysteme stellt das LRZ zusätzliche Hochleistungs-Cluster bereit: Eines besteht aus zwölf SGI-Altix-XE500-Servern und ein weiteres aus drei selbst konfigurierten Systemen. Ergänzt werden diese Cluster durch einen SGI-Altix-UV10-Server.

Am LRZ werden 80 Blade Server betrieben, um eine Virtualisierungsinfrastruktur zur Verfügung zu stellen. Die Server verfügen über insgesamt 640 *CPU*-Kerne und greifen auf 100 TByte Festplattenspeicher sowie 7,6 TByte *RAM* zurück. Von den über 900 VMs basiert ein Großteil auf dem freien Betriebssystemkern Linux. Nur knapp 100 der Systeme besitzen eine Windows-Server-Installation.

Um den Betrieb der Serversysteme für die zahlreichen Internetdienste aufrecht zu erhalten, fallen für die Administration eine Reihe von Aufgaben an. Dabei macht es in der Praxis kaum einen Unterschied, ob es sich um eine VM oder einen dedizierten Server handelt. Zu den regelmäßigen Aufgaben in Bezug auf Linux-Server zählen unter anderem die Folgenden:

- Das Betriebssystem muss installiert und die für den Betrieb des Servers erforderlichen Programme hinzugefügt und konfiguriert werden.
- Die Systeme müssen durch geeignete Sicherheitsmaßnahmen wie *Operating System (OS) Hardening*, Sicherheits-Updates und Zugangsbeschränkungen abgesichert werden.

- Monitoring der Verfügbarkeit und der Performance sowie der Funktion des Systems und der laufenden Anwendungen.
- Gemessene Systemzustände werden unter Berücksichtigung der jeweiligen Anwendung aufgezeichnet und für zukünftige Ressourcenplanungen genutzt.
- Betriebsanleitungen für Anwender und Notfallpläne müssen erstellt und auf dem laufenden Stand gehalten werden.
- Im Störungsfall müssen Fehler aufgespürt und beseitigt werden.
- Für eine optimale Benutzung des Systems werden Anwender unterstützt und erhalten Beratung.

Die von den Administratoren betreuten Server lassen sich nach ihren Funktionen kategorisieren. Zu den exemplarischen Kategorien zählen unter anderem folgende:

- Application Management bzw. Backend Server
- Cache Server
- Content Management Server
- Datenbankserver
- Development Server für Kunden
- Gatewaysysteme
- Service und Performance Monitoring Server
- Web Server

Im Laufe der Zeit hat sich herausgestellt, dass Virtualisierung in Einzelfällen nicht sinnvoll ist, da Performance-Engpässe eintreten können. Dies trifft zum Beispiel auf Systeme zu, die von einer direkten Anbindung an die vorhandenen Storage-Systeme profitieren. Dazu zählen unter anderem die Mailbox Server des LRZ.

4.2. Anforderungen

Dieser Abschnitt befasst sich mit der Schnittmenge konkreter Anforderungen, die das LRZ an ein Administrationstool für Linux-Server sowie an den anzufertigenden Prototyp stellt. Dabei wird zwischen funktionalen und nichtfunktionalen Anforderungen unterschieden. Da Administrationstools den unterschiedlichsten Aufgaben dienlich sind, werden in diesem Kapitel nur generische Anforderungen betrachtet, die unabhängig vom Einsatzzweck gelten.

Da sich diese Masterarbeit schwerpunktmäßig mit Informationsvisualisierung im Kontext der Linux-Server-Administration befasst, werden die visuellen Anforderungen an ein Administrationstool bzw. an den Prototyp separat betrachtet. Sie spielen später bei der Bewertung der vorhandenen Lösungen in Kapitel 5, bei der Entwicklung neuer Lösungsmöglichkeiten in Kapitel 6, bei der Implementierung des Prototyps in Kapitel 7 und bei seiner Evaluation in Kapitel 8 eine wichtige Rolle.

Im Folgenden werden die einzelnen Anforderungen zunächst beschrieben und jeweils im Anschluss zu einzelnen Punkten zusammengefasst. Zur Unterscheidung von funktionalen, nichtfunktionalen und visuellen Anforderungen werden die Präfixe *F*, *NF* und *V* gefolgt von einer aufsteigenden Zahl verwendet.

4.2.1. Funktionale Anforderungen

An einem Rechenzentrum von der Größenordnung des LRZ fallen immense Datenmengen an. Dabei handelt es sich nicht nur um Nutzdaten, die von den Anwendern der dort angebotenen Dienste generiert werden, sondern auch um Daten aus dem Bereich der Systemadministration, die im laufenden Betrieb anfallen. Jeder Dienst, der auf einem Server läuft, sollte so konfiguriert sein, dass er wichtige Systemmeldungen protokolliert.

Protokolleinträge können Administratoren im Fehlerfall oder nach einem Hacker-Angriff wichtige Informationen liefern, um in kurzer Zeit die Gründe festzustellen oder Sicherheitsschlupflöcher ausfindig zu machen. Auch weniger kritische Loglevel können zu diesem Zweck nützlich sein. Doch je mehr feingranulare Loglevel mitprotokolliert werden, desto schneller wachsen die Protokolldateien und werden immer unüberschaubarer.

Die Protokollierung von Systemereignissen ist kein Selbstzweck. Irgendwann müssen die Dateien von Menschen gelesen und interpretiert werden. Wenn Administratoren viel Zeit damit verbringen, Logdateien zu durchforsten, aber dennoch wichtige Details übersehen, ist das verschwendete Arbeitszeit. Das gleiche gilt auch für Informationen, die der Administrator manuell von den Servern anfordern muss. Bei der Kontrolle von Firewall-Konfigurationen hunderter Systeme verliert man schnell den Überblick. Das gilt auch für Paketlisten, die den Softwarestand der Server widerspiegeln.

Aus den genannten Gründen ist das Beherrschbarmachen der großen Datenmengen ein zentraler Anforderungspunkt an Administrationstools bzw. an den Demonstrator. Das Tool soll die notwendigen Informationen vieler Server konsolidieren und dem Anwender mit Hilfe geeigneter visueller Darstellungsformen die essentiellen Informationen vermitteln. Der Anwender soll nur mit den absolut notwendigen Informationen versorgt und Textkolonnen sollen vor ihm verborgen werden. Das Administrationstool soll ihn in die Lage versetzen, anhand der dort dargestellten Visualisierungen entscheiden zu können, ob einzelne Systeme einen manuellen Eingriff erfordern oder nicht.

F1: Beherrschbarmachen großer Datenmengen

Große Datenmengen sollen automatisch beschafft, konsolidiert und durch geeignete Methoden der Informationsvisualisierung beherrschbar gemacht werden.

Auf jedem Linux-Server sind neben dem eigentlichen Betriebssystem auch noch spezifische Pakete installiert, die zur Diensterbringung benötigt werden. Mit der einmaligen Installation eines Servers ist es nicht getan: Während Konfigurationsdateien oft nur einmalig editiert und danach in Ruhe gelassen werden, müssen die Software-Pakete regelmäßig aktualisiert werden, um bekanntgewordene Sicherheitslücken zu schließen. Einige Distributionen bieten zwar die Möglichkeit automatischer Updates an, dieser Mechanismus kommt aber nicht auf allen Servern in Frage, da Updates vor dem Einspielen sicherheitshalber getestet werden müssen. Manuelle Updates sind deshalb auf vielen Servern unerlässlich.

Bevor Updates auf einem Server installiert werden, muss der Administrator wissen, ob für ihn überhaupt Updates bereitstehen. Es ist auch wünschenswert zu wissen, ob installierte Software besonders sicherheitskritische Lücken besitzt, um diese Systeme zuerst zu aktualisieren oder zur Not vorübergehend vom Netz zu nehmen. Damit der Administrator nicht bei jedem Server auf Verdacht den Softwarestand prüfen muss und wertvolle Zeit vergeudet, soll ein Administrationstool zur Vereinfachung der Linux-Server-Administration eine zentrale Übersicht der Softwarestände aller Server bereitstellen. Es soll den Administrator auf den ersten Blick nur mit den notwendigsten Informationen versorgen und Systeme, bei denen Handlungsbedarf besteht, durch geeignete Methoden hervorheben, um die Aufmerksamkeit des Administrators darauf zu lenken. Zur Bewertung des Sicherheitszustands soll das Tool auf eine zentrale Paketreferenzliste zurückgreifen, die stets aktuell zu halten ist. Paketlisten der einzelnen Server mit den konkret installierten Paketen sollen bei Bedarf abrufbar sein.

F2: Softwarestände überwachen

Auf den Servern installierte Paketversionen sollen zentral abrufbar sein. Anhand einer Referenzliste soll die Aktualität und die Sicherheit der Pakete bestimmt werden.

Um das Sicherheitskonzept *Defense in Depth* umzusetzen, befinden sich auf vielen Linux-Servern Personal Firewalls mit individuellen Konfigurationen. Die Anforderungen an die Firewall-Konfigurationen können sich im Lauf der Zeit ändern. Ebenso ist es möglich, dass einzelne Konfigurationen lückenhaft oder fehlerhaft erfolgt sind. Im schlimmsten Fall ist die Firewall auf einem Linux-Server komplett abgeschaltet, zum Beispiel weil nach einer vorübergehenden Deaktivierung vergessen wurde, sie wieder einzuschalten.

Ein Tool zur Verwaltung von Linux-Servern sollte deshalb auch eine Übersicht über die Zustände aller darauf befindlicher Firewalls bereitstellen. Es muss selbstständig den Sicherheitszustand einer Firewall-Konfiguration bewerten und kategorisieren und den Administrator alarmieren, falls Handlungsbedarf herrscht. Detaillierte Konfigurationsdateien einzelner Systeme sollen auf Wunsch des Administrators abrufbar sein.

F3: Firewall-Konfigurationen überwachen

Die Wirksamkeit und Sicherheit der Firewall-Konfigurationen auf den Servern soll an zentraler Stelle ersichtlich sein. Einzelne Konfigurationsdateien sollen bei Bedarf abrufbar sein.

Viele Server unterstützen den Aufbau verschlüsselter Verbindungen durch die Clients. Dazu zählen zum Beispiel Web- und Datenbankserver. Dazu nutzen sie X.509-Zertifikate und private Schlüssel. Private Keys dürfen nicht in falsche Hände geraten, da ein Angreifer damit aufgezeichnete Ciphertexte entschlüsseln und so an vertrauliche Informationen geraten könnte. Andere Missbrauchsszenarien sind auch denkbar. Um die Wahrscheinlichkeit zu minimieren, dass ein Angreifer an private Schlüssel gelangt, müssen auf den Servern private Schlüssel durch restriktive Zugriffsrechte geschützt sein. Sie dürfen nur an einer Stelle im Dateisystem liegen, alte private Schlüssel müssen bei einem Austausch des Zertifikats umgehend vom Server gelöscht werden.

Um einen Beitrag zur Informationssicherheit zu leisten, soll ein Administrationstool zur Verwaltung von Linux-Servern die Sicherheit der darauf befindlichen privaten Schlüssel im Auge behalten. Es muss Informationen über die Anzahl, den Speicherort und die Wirksamkeit der Dateizugriffsrechte liefern und den Administrator ggf. über Missstände informieren. Damit das Administrationstool selbst nicht von Angreifern zum Stehlen privater Schlüssel missbraucht werden kann, müssen geeignete Sicherheitsmaßnahmen durchgesetzt werden. Aus diesem Grund darf das Administrationstool weder Kopien der privaten Schlüssel in einer zentralen Datenbank speichern, noch den Zugriff auf den Inhalt der Schlüsseldateien aus der Ferne ermöglichen. Eingriffe müssen aus Sicherheitsgründen weiterhin direkt an den Servern erfolgen.

F4: Sicherheit privater X.509-Schlüssel überwachen

Die Sicherheit privater X.509-Schlüssel aller Server soll bewertet und an zentraler Stelle ersichtlich sein. Ein Zugriff auf die Schlüssel ist nicht erlaubt.

Der verschlüsselte Verbindungsaufbau zwischen Clients und Linux-Servern geschieht in der Regel über TLS. Serverseitige Dienste, die TLS unterstützen, stellen vielfältige Konfigurationsmöglichkeiten bereit, die sich unmittelbar auf die Sicherheit auswirken. Vorgefertigte Standardkonfigurationen sind mit Vorsicht zu genießen, da sie ggf. Kryptografiealgorithmen aktivieren, die nicht mehr als sicher gelten und von Angreifern ausgenutzt werden könnten. Algorithmen, die vor kurzem noch als sicher galten, können schnell obsolet werden. Aus diesem Grund müssen die TLS-Konfigurationen der Server hin und wieder auf den aktuellen Sicherheitsstand gebracht werden.

Damit Administratoren nicht auf Verdacht die TLS-Konfigurationsdateien der Linux-Server überprüfen müssen und so wertvolle Zeit verschwenden, soll ein Administrationstool die Aufgabe der Überprüfung übernehmen. Es muss die Konfigurationsdateien abfragen und anhand eines zentralen Regelwerks, das ebenfalls aktuell zu halten ist, die Sicherheit der Konfigurationen ermitteln. Optionale Portscans können zur Prüfung der Wirksamkeit der TLS-Einstellungen durchgeführt werden.

F5: Sicherheit von TLS-Konfigurationen überwachen

Die Sicherheit und Wirksamkeit der TLS-Konfigurationen aller Server soll bewertet und zentral angezeigt werden.

Das Bundesdatenschutzgesetz (BDSG) regelt den Umgang personenbezogener Daten, die in Informations- und Kommunikationssystemen oder manuell verarbeitet werden [BDSG 15]. Aufgrund des sog. *Verbotsprinzips mit Erlaubnisvorbehalt* ist die Erhebung, Verarbeitung und Nutzung von personenbezogenen Daten nur dann erlaubt, wenn dies entweder ein spezielles Gesetz erlaubt oder die i. d. R. schriftliche Einverständniserklärung der betreffenden Person vorliegt. Zudem ruft das BDSG zur Datensparsamkeit auf.

Ein Administrationstool darf keine unnötigen personenbezogenen Daten speichern oder weiterverarbeiten. Sollte es Dateien mit Personenbezug als Datenquelle verwenden, sind diese zuvor durch Maßnahmen wie Anonymisierung oder Pseudonymisierung unkenntlich zu machen.

F6: Datenschutzrechtliche Aspekte

Das Tool muss Datenschutzrechtliche Regelungen umsetzen. Datensparsamkeit ist essentiell.

4.2.2. Nichtfunktionale Anforderungen

Wenn einem Administrator keine anderen Werkzeuge zur Überwachung der Linux-Server zur Verfügung stehen, bleibt ihm nichts anderes übrig, als verschiedene Kommandozeilentools zu verwenden. Ein Anwendungsbeispiel hierfür wäre die Anonymisierung von anwenderbezogenen Daten wie IP-Adressen.

Kurze Logdateien können mit dem Programm `cat` ausgegeben werden, bei längeren Dateien sind Pager wie `less` oder `more` unentbehrlich. In manchen Fällen muss das Programm `find` verwendet werden, um mehrere Quelldateien auszuwählen. In Kombination mit `find` wird oft auf das Programm `xargs` zurückgegriffen, das die Suchergebnisse aus der Standardausgabe verwendet, um neue Befehlszeilen daraus zu erzeugen und diese nacheinander auszuführen. Um unüberschaubare Dateien nach bestimmten Inhalten zu durchsuchen, kann das Programm `grep` eingesetzt werden. In manchen Fällen macht es Sinn, die Ausgabe des Programms noch in einer eigenen Datei zu speichern. Der Stream-Editor `sed` dient schließlich dazu, textuelle Änderungen automatisch durchzuführen und, wie im Beispiel genannt, IP-Adressen zu anonymisieren.

Die erwähnten Kommandozeilen-Tools sind mächtige Werkzeuge und können aufgrund der vielen Kommandozeilenparameter und der Kaskadierbarkeit durch Pipes auf vielfältige Art und Weise eingesetzt werden. Das setzt allerdings Erfahrung voraus. Oft ist eine sukzessive Erweiterung von Kommandozeilenbefehlen erforderlich, um das gewünschte Endergebnis zu erhalten. Nicht selten müssen bei dieser Arbeit Manpages gelesen und das WWW nach Tutorials durchsucht werden.

Auf wenigen Servern mag dieses Vorgehen noch zielführend sein. In Umgebungen wie dem LRZ, wo über 900 Linux-Server administriert werden, sieht die Sache anders aus. Wenn sich Administratoren auf jedem Server erst einloggen müssen, um dann ein und dieselben Arbeiten auf der Kommandozeile durchzuführen, kann von Bedienfreundlichkeit nicht die Rede sein. Noch dazu besteht die Gefahr, dass Menschen dabei Fehler unterlaufen, die sie nicht bemerken.

Aufgrund dieser Umstände soll ein Administrationstool die Arbeit von Administratoren bedienfreundlicher gestalten. Administratoren sollen weniger Zeit auf der Kommandozeile für Routineaufgaben verbringen und stattdessen einen zentralen Anlaufpunkt verwenden, um von dort unkompliziert per Mausklick notwendige Informationen zu erhalten.

Anschauliche visuelle Darstellungen sollen den Administrator beim Treffen von Entscheidungen unterstützen. Arbeiten auf der Kommandozeile der einzelnen Server sollen dadurch nur dann erforderlich sein, wenn es unbedingt nötig ist. Eine intuitive Oberfläche soll ihren Beitrag zur Bedienfreundlichkeit leisten und die Einarbeitungszeit minimieren. Ein einziger Log-in muss genügen, um den Zustand vieler Linux-Systeme abrufen zu können, auch wenn dort unterschiedliche Passwörter notwendig wären.

Als webbasierte Anwendung soll insbesondere der Prototyp an beliebigen PCs und anderen Geräten aus dem Administratornetz verwendbar sein, ohne eine clientseitige Software-Installation vorauszusetzen. Damit trägt er zu einer einheitlichen Benutzbarkeit auf unterschiedlichen Plattformen bei. Um die Bedienfreundlichkeit auch für anderssprachige Mitarbeiter des LRZ zu erhöhen, sollte der Prototyp so konzipiert sein, dass er bei Bedarf ohne großen Aufwand um andere Sprachen erweitert werden kann.

> **NF1: Bedienfreundlichkeit**
>
> Bedienfreundlichkeit soll erreicht werden, indem das Tool dem Administrator das häufige Wechseln zu anderen Tools oder der Kommandozeile erspart. Es muss einfach und geräteübergreifend zu bedienen sein. Mehrsprachigkeit ist erwünscht.

Wenn eine neue Sicherheitslücke in einem Programmpaket in der Fachpresse bekannt wird, sind Administratoren zwar bemüht, schnell Sicherheitsupdates einzuspielen, sie müssen aber auch wissen, welche der zahlreichen Server von ihr betroffen sind. Ansonsten wird kostbare Zeit mit den nicht verwundbaren Systemen verschwendet, die Angreifer zur Ausnutzung der Sicherheitslücke ausnutzen könnten. Methoden der Informationsvisualisierung können hier gewinnbringend eingesetzt werden.

Um die Reaktionszeit der Administratoren zu minimieren, muss die Überwachung der vielen Linux-Server über eine zentrale Oberfläche ermöglicht werden. Die Prüfung der Aktualität der Software-Installationen der vielen Server bietet sich als Anwendungsfall besonders an. Einerseits gibt es eine große Anzahl an Servern

mit einer unterschiedlichen Menge an installierten Paketen, andererseits ist ein schneller Überblick über die Aktualität der Software sicherheitsrelevant.

Das gleiche gilt für Firewall-Konfigurationen. Ein zentraler Überblick über die geöffneten Ports hunderter Server ermöglicht es dem Administrator unmittelbar zu entscheiden, auf welchen der Systeme Handlungsbedarf besteht.

NF2: Reaktionszeit

Als zentraler Anlaufpunkt soll das Tool sicherheitsrelevante Daten durch Informationsvisualisierung augenblicklich begreifbar machen und so eine geringe Reaktionszeit ermöglichen.

Plattformunabhängigkeit wirkt sich auch positiv auf Bedienfreundlichkeit aus. Aus diesem Grund ist der Verzicht auf proprietäre Rich-Internet-Technologien wie *Adobe Flash* oder *Microsoft Silverlight* wünschenswert. Stattdessen sollte auf die Fähigkeiten von Hypertext Markup Language (HTML)5 und JavaScript gesetzt werden. Dadurch erübrigt sich die Installation zusätzlicher Plug-ins im Browser und es können nahezu beliebige Endgeräte und Plattformen zur Arbeit mit dem Administrationstool verwendet werden. Insbesondere der Prototyp sollte dieser Anforderung gerecht werden.

Auch aus Sicht der Informationssicherheit macht es Sinn, auf die genannten proprietären Technologien zu verzichten: Im Adobe Flash Player wurden im Lauf der Zeit immer wieder gravierende Sicherheitslücken entdeckt. Es gibt Browser, die bei Bekanntwerden einer solchen Sicherheitslücke das entsprechende Plug-in automatisch deaktivieren. Das macht einerseits Sinn, andererseits können Anwendungen, die dieses Plug-in zwingend benötigen, dann auch nicht mehr benutzt werden. Im Fall von Adobe Flash wurden erst kurz bevor diese Zeilen entstanden neue Sicherheitslücken bekannt. Es wurde kurzfristig ein Patch bereitgestellt, der allerdings eine kritische Lücke offen lies, die bereits von Internetkriminellen ausgenutzt wurde [EIK 15B].

NF3: Basistechnologien

Durch den Verzicht auf proprietäre Basistechnologien sollen Unabhängigkeit geschaffen, Einstiegsbarrieren gesenkt und die Wartbarkeit der Anwendung erhöht werden.

Das Administrationstool soll in einem Umfeld betrieben werden, in dem Informationssicherheit berücksichtigt werden muss. Aus diesem Grund werden auch an den Demonstrator bestimmte Anforderungen gestellt. Dazu zählt die Einhaltung des Need-to-know-Prinzips. Dies setzt Mechanismen zur Authentifizierung und Autorisierung voraus. Ein einfach gehaltenes, gruppenbasiertes Rechtesystem soll den Zugriff auf bestimmte Funktionen des Programms einschränken.

Das Tool sollte einerseits lokale Benutzerkonten und andererseits die Benutzerauthentifizierung über einen angebundenen Verzeichnisdienst unterstützen. Durch die Anbindung an einen AD- oder Lightweight Directory Access Protocol (LDAP)-Server können sich Anwender nach der Änderung ihres Domänenpassworts weiterhin am Demonstrator anmelden, ohne dort auch ihr Kennwort ändern zu müssen. Die Kommunikation zwischen dem Demonstrator und dem Authentifizierungsserver hat verschlüsselt zu erfolgen, z. B. durch LDAPS.

Die Unterstützung eines zentralen Benutzerkontos macht auf den ersten Blick Sinn: Falls das Passwort einem Angreifer bekannt wird, kann es an einer zentralen Stelle durch ein neues ersetzt werden. Unmittelbar danach tritt die Passwortänderung bei allen angebundenen Systemen in Kraft. Dadurch wird keine wertvolle Zeit verschwendet, um mehrere lokal gespeicherte Passwörter nacheinander zu ändern. In manchen Fällen werden Systeme auch einfach im Laufe der Zeit vergessen und Anwender wissen nicht mehr, wo sie überall ein Benutzerkonto besitzen oder welches Passwort dort eingestellt wurde. Auch hier schafft eine zentrales System zur Authentifizierung Abhilfe.

Auf einem Authentifizierungsserver zentral hinterlegte Benutzerkonten bergen aber auch ein Risiko: Sobald ein Angreifer die Zugangsdaten eines Administrators ausspäht, hat er Zugriff auf eine große Anzahl von Servern. Der Verzicht auf zentrale Benutzerkonten ist aber nicht praktikabel, da aus Sicht der Informationssicherheit im Idealfall für jeden Server unterschiedliche Zugangsdaten zu verwenden sind. Das Ergebnis sind ausgedruckte Passwortlisten, die von Unbefugten entwendet werden können. Um beim Einsatz zentral hinterlegter Benutzerkonten dennoch das Sicherheitsniveau hoch zu halten, müssen Passwörter regelmäßig geändert werden.[2]

[2]Bevor sich Administratoren Passwort-Policies für die Mitarbeiter ausdenken, sollten sie für sich selbst Passwortrichtlinien einführen,

Im Kapitel 3.1 wurden Angriffstechniken aufgezählt, die vor allem für Webanwendungen von großer Bedeutung sind. Dazu zählen unter anderem Injections, XSSs und CSRFs. Da auch der zu entwickelnde Prototyp zur Klasse der Webanwendungen zählt, wird Wert darauf gelegt, dass geeignete Maßnahmen dagegen getroffen werden. Außerdem müssen Datenübertragungen zwischen der Anwendung und dem Browser oder anderen Diensten nach Möglichkeit über verschlüsselte Kommunikationsprotokolle stattfinden. Der Webserver, der die Anwendung bereitstellt, muss sicherstellen, dass beim TLS Handshake keine unsicheren Cipher Suites ausgehandelt werden. Im Zweifelsfall kommt stattdessen überhaupt keine Verbindung zustande. Sollte die an den Prototyp angebundene Datenbank auf demselben Host ausgeführt werden, sollen Verbindungsversuche von außen abgelehnt werden.

Viele Protokoll- und einige Konfigurationsdateien beinhalten anwenderbezogene Daten. Datenschutz muss deshalb ebenfalls berücksichtigt werden. Aus diesem Grund müssen diese Daten in einem angemessenen Umfang anonymisiert werden, bevor sie durch ein Administrationstool weiterverarbeitet und visualisiert werden.

Sollte insbesondere der Prototyp im Laufe der Zeit zu einem Produktivsystem weiterentwickelt und konsequent am LRZ eingesetzt werden, müssen in jedem Fall wirksame Vorkehrungen getroffen werden, um den unbefugten Zugang zu dem System oder zumindest zu dessen Datenbank zu verhindern. Im Idealfall dient die Anwendung dazu, dass Administratoren sofort auf Unstimmigkeiten in der Linux-Serverlandschaft hingewiesen werden, um dann unmittelbar auf den betroffenen Servern die Fehler zu beheben.

Die Sicherheit und Wirksamkeit der TLS-Konfigurationen aller Server soll bewertet und zentral angezeigt werden. Ein Angreifer könnte das System aber auch für seine eigenen Zwecke missbrauchen, um verwundbare Linux-Server ausfindig zu machen und diese gezielt zu manipulieren. Er könnte dort Schadsoftware installieren und dem zentralen System schließlich falsche Werte vorgaukeln, um unentdeckt zu bleiben. Der Angriffsvektor für dieses Szenario ist für einen Außentäter gewiss sehr komplex. Es darf aber nicht vergessen werden, dass ein Großteil der Angreifer Innentäter sind [BFV 14].

Die Daten, mit denen der Demonstrator arbeitet, müssen auf jeden Fall besonders geschützt werden. Vor der Überführung des Programms in den Wirkbetrieb ist es notwendig, eine Risikobewertung und eine Kosten-/Nutzenrechnung durchzuführen.

NF4: Informationssicherheit

Das Tool soll grundlegende Sicherheitsfunktionen wie ein differenziertes Rechtesystem und LDAP-Authentifizierung implementieren. Webbasierte Anwendungen sollen insbesondere Schutzmaßnahmen gegen häufig durchgeführte Angriffstechniken bieten. Die Sicherheit der gespeicherten Daten ist zu gewährleisten.

Die Aufgabe des Prototyps ist es zu zeigen, dass die Linux-Server-Administration durch Datenvisualisierung verbessert werden kann. Das Wort "Linux-Server" ist ein sehr allgemeiner Begriff: Es existieren unzählige Distributionen mit unterschiedlichen Schwerpunkten und Zielgruppen. Zudem gleicht eine Linux-Installation auf einem Server kaum einer anderen, da Linux sehr viel Freiraum für benutzerspezifische Anpassungen lässt.

Linux-Distributionen besitzen in den meisten Fällen eine vergleichbare Verzeichnisstruktur, die auf dem Filesystem Hierarchy Standard (FHS) beruht, dennoch gibt es oft Abweichungen im Detail.[3] Bei Paketverwaltungs- und Init-Systemen unterscheiden sich die Distributionen ebenfalls. Während die bewährte Server-Distribution Debian zum Beispiel *Advanced Packaging Tool (APT)* und *SysVinit* nutzt, verwendet openSUSE stattdessen *RPM Package Manager (RPM)* und *systemd*. Je nachdem, welche Aufgaben Linux-Server erfüllen, sind sie anders konfiguriert und verfügen über unterschiedliche Softwarepakete.

Auf den knapp 1000 Linux-Servern am LRZ wird größtenteils SUSE Linux Enterprise Server (SLES) als Betriebssystem eingesetzt. Der im Rahmen dieser Masterarbeit anzufertigende Prototyp soll darauf Rücksicht nehmen, indem zum Beispiel Bildschirmausgaben distributionsspezifischer Admin-Tools ohne zusätz-

die mindestens genauso streng formuliert sind. Erstens sind ihre Zugangsdaten im Allgemeinen kritischer als die von Mitarbeitern, zweitens erfahren sie so selbst, wie praktikabel oder wirksam die Richtlinien sind.

[3] Die Linux-Distribution *GoboLinux* (`http://gobolinux.org/`) stellt einen Sonderfall dar, denn sie weicht bewusst vom FHS ab. Bei GoboLinux befindet sich jedes Programm und jede Version davon in einem eigenen Verzeichnis unterhalb von `/Programs/`. Da hierdurch das Dateisystem die Aufgabe einer Paketdatenbank erfüllt, kann auf eine herkömmliche Paketdatenbank verzichtet werden. Einen ähnlichen Weg geht der Linux-Distributor Canonical, der mit *Snappy Apps* eine Alternative zum Debian-Paketsystem entwickelt hat [DIE 15]. Dort bringt jede Paketversion ihre eigenen Abhängigkeiten mit. Paketkonflikte sollen so komplett ausgeschlossen werden, allerdings steigt der Speicherbedarf und die Wartung fehlerhafter Abhängigkeiten wird dadurch erschwert.

liche Konvertierung akzeptiert werden. Dies ist zum Beispiel beim Format der Paketlisten relevant, die das Kommandozeilentool `zypper` ausgibt. Die Unterstützung der auf den Linux-Servern seltener eingesetzten Distribution Debian durch den Prototyp ist optional.

Die Datenbank des Prototyps soll bei Bedarf auf einen bereits bestehenden Datenbankserver ausgelagert werden können. Zu diesem Zweck soll der Prototyp Schnittstellen zu den am LRZ eingesetzten Database Management Systems (DBMSs) bereitstellen. Die Auslagerung der Datenbank zu einem bestehenden Datenbankserver bietet den Vorteil, dass Datensicherungen zentral durchgeführt werden können. Außerdem verringert sich dadurch die Anzahl der Server, die administrativen Beistand benötigen.

Am Rechenzentrum existiert zwar bereits ein selbst entwickeltes, webbasiertes Tool, das Auskunft über den Status der überwachten Server gibt. Durch Informationsvisualisierung und neue Ideen bezüglich der Use Cases könnten sowohl die Administration erleichtert als auch das Sicherheitsniveau erhöht werden. Im Rahmen dieser Masterarbeit sollen Lösungsmöglichkeiten vorgestellt werden, die in Zukunft am LRZ eingesetzt werden können.

NF5: Integrierbarkeit in bestehende Umgebung

Das Tool soll die am häufigsten verwendeten Linux-Server-Distributionen unterstützen. Schnittstellen zu vorhandenen DBMSs müssen vorhanden sein. Die Integrierbarkeit bzw. Kompatibilität zu vorhandenen Tools ist erwünscht.

4.2.3. Visuelle Anforderungen

Die Gestaltung der Bedienoberfläche ist ausschlaggebend für die Bedienfreundlichkeit und Verwendbarkeit von Anwendungsprogrammen. Dies gilt auch für Administrationstools bzw. den im Rahmen dieser Masterarbeit anzufertigenden Prototyp. Die visuellen Anforderungen in diesem Abschnitt beziehen sich auf die Grundlagen aus Kapitel 2.

Beim Betrachten einer grafischen Bedienoberfläche sticht sofort ins Auge, ob sie einen aufgeräumten oder überfrachteten Eindruck hinterlässt. Überfrachtet werden Oberflächen beispielsweise durch den Einsatz von unnötigem Eye Candy (siehe Abschnitt 2.13.3). Dazu zählen zum Beispiel Schatten oder Farbverläufe, die nur der optischen Verschönerung dienen. Sie vermitteln keine Information und erfüllen deshalb keinen Zweck. Indem eine Bedienoberfläche zu sehr mit unnützen Elementen vollgepackt wird, verliert sie an Übersichtlichkeit. Dem Anwender fällt es schwerer, die gewünschte Information zu erhalten, da er von Eye Candy abgelenkt wird.

V1: Übersichtlichkeit

Die Bedienoberfläche darf nicht überfrachtet sein, da sonst die Übersichtlichkeit darunter leidet.

Wie in Abschnitt 2.13 gezeigt wurde, sind Farben ein wichtiger Bestandteil in der Informationsvisualisierung. In Bedienoberflächen dienen sie zur Differenzierung und zum Hervorheben von Elementen (siehe Abschnitt 2.13.3). Einige Farben sind unabhängig vom Anwendungszweck bei vielen Anwendern mit denselben Assoziationen vorbelegt. So wirkt die Farbe Rot alarmierend, während die Farbe Grün beruhigenden Charakter hat. Da die Farben in den Visualisierungen eine wichtige Rolle spielen, sollten sie in der Bedienoberfläche und drum herum sparsam eingesetzt werden, um den Anwender nicht von den Visualisierungen der essentiellen Informationen abzulenken. Große Farbflächen sollten ebenso vermieden werden. Um von Anfang an nicht in Versuchung zu geraten, Farben beim Oberflächendesign inflationär zu verwenden, sollte Ben Shneidermans Rat befolgt werden, Bedienoberflächen zunächst so zu gestalten, als würde das Anzeigegerät keine Farben beherrschen.

V2: Farben

In der Bedienoberfläche müssen Farben sparsam und zielgerichtet eingesetzt werden.

Unabhängig davon, ob Oberflächenelemente farbig sind oder nicht, müssen sie gut erkennbar sein. Dies geschieht durch einen ausreichend großen Kontrast zwischen Vordergrund und Hintergrund. Die Kontrastverhältnisse sollen sich in erster Linie an die Anforderungen eines Computerbildschirm richten. Auch bei starkem Umgebungslicht sollen Bildschirminhalte noch gut zu unterscheiden sein.

V3: Kontrast

In der Bedienoberfläche müssen die Kontrastverhältnisse hoch sein.

Neben dem Kontrast spielt auch die Größe von Oberflächenelementen eine Rolle. Kleine Piktogramme und Texte sparen zwar Platz, sind aber schwer zu entziffern und wirken sich nachteilig auf die Bedienfreundlichkeit aus. Insbesondere Oberflächenelemente, die Events auslösen können, müssen ausreichend großflächig sein, damit der Anwender nicht versehentlich daneben oder gar auf den falschen Button klickt.

V4: Objektgröße

In der Bedienoberfläche müssen Oberflächenelemente eine ausreichende Größe besitzen.

Neben der Bedienoberfläche stellt ein grafisches Administrationstool dynamisch generierte Visualisierungen bereit. Für die Gestaltung von Visualisierungen gelten ähnliche Anforderungen wie auch bei der Bedienoberfläche. Zunächst ist aber ausschlaggebend, ob für die aktuell gewählten Informationen auch passende Präsentationsformen gewählt werden. Es macht zum Beispiel keinen Sinn, für n-Dimensionale Daten eine Visualisierungsform zu wählen, die für die Darstellung von $(n + 1)$ Dimensionen oder mehr ausgelegt ist, da sonst ein visuelles Attribut ungenutzt bleibt.[4]

V5: Visualisierungsformen

Die Arten der Visualisierungsformen müssen zu den zugrundeliegenden Daten und Dimensionalitäten passen.

Unabhängig von ihrer Gestalt sind Visualisierungen nur dann aussagekräftig, wenn sie ausreichend beschriftet sind. Ohne eine aussagekräftige Beschriftung kann es zu Fehlinterpretationen kommen, die im sicherheitsrelevanten Kontext der Linux-Server-Administration nicht auftreten dürfen. Zu obligatorischen Beschriftungen zählen zum Beispiel Achsbeschriftungen und Legendentexte. Einheiten dürfen dem Betrachter auch nicht vorenthalten werden, da sie zur Interpretation von Visualisierungen unverzichtbar sind. Länderspezifische Besonderheiten wie die Verwendung bestimmter Tausender- und Dezimaltrennzeichen müssen berücksichtigt werden, um Verwirrung zu vermeiden.

V6: Beschriftungen

Visualisierungen müssen mit den notwendigen Beschriftungen versehen sein, damit sie nicht fehlinterpretiert werden können.

In Kapitel 2 wurde der Begriff *Visual Clutter* erwähnt. Er entsteht zum Beispiel durch die Überfrachtung einer Visualisierung durch zu viele Elemente und wirkt sich negativ auf die Übersichtlichkeit und Lesbarkeit aus. In Abschnitt 2.12.7 wurden Maßnahmen vorgestellt, um Visual Clutter zu vermeiden.

V7: Visual Clutter

Visualisierungen müssen frei von Visual Clutter sein, um die Übersichtlichkeit zu verbessern.

Bei den Anforderungen an die Bedienoberfläche wurde bereits genannt, dass einige Farben beim Menschen als Signalfarbe vorbelegt sind und bestimmte Assoziationen hervorrufen. Diese Tatsache muss auch bei Visualisierungen berücksichtigt werden. Visualisierungen setzen häufig Farben ein, damit der Betrachter die zugrundeliegenden Daten besser unterscheiden kann. Dabei muss darauf geachtet werden, dass die Menge an unterschiedlichen Farben nicht zu groß ist. Bei einer zweistelligen Anzahl ist eine gute Unterscheidbarkeit der einzelnen Farben nicht mehr gewährleistet. Die Übersichtlichkeit und die Lesbarkeit der Visualisierung leiden darunter.

V8: Farbskalen

Farben in Visualisierungen müssen sparsam und zielgerichtet eingesetzt werden. Farbige Datenreihen müssen gut voneinander unterscheidbar sein.

Wie auch bei der Bedienoberfläche gilt bei den Visualisierungen, dass die Lesbarkeit und Übersichtlichkeit vom Kontrast zwischen Vordergrundfarbe und Hintergrundfarbe abhängen. Daraus folgt diese visuelle Anforderung:

[4] Dass man für jede Arbeit das richtige Werkzeug verwenden soll, sagte auch Montgomery Scott, der Chefingenieur der USS Enterprise im Film *Star Trek: Am Rande des Universums*, zu seinen Technikern.

V9: Kontraste in Visualisierungen

Kontrastverhältnisse in Visualisierungen müssen hoch sein, um die Lesbarkeit zu optimieren.

Schaubilder werden oft mit visuellen Inhalten versehen, die lediglich künstlerischen Zwecken dienen und keine Zusatzinformation bereitstellen. Für diese Elemente hat sich im Lauf der Zeit der abschätzige Begriff *Chartjunk* (siehe Kapitel 2.7.2) etabliert. In einem Administrationstool ist Chartjunk nicht erwünscht, da sich der Administrator auf die Aussage der Visualisierungen verlassen muss und keine Ablenkung duldet. Die Übersichtlichkeit in Diagrammen steigt weiter, indem beispielsweise auf Hintergrundfarben verzichtet wird. Pixel auf dem Bildschirm bzw. Tinte oder Toner auf dem Papier, die selbst keine Zusatzinformation ausdrücken, werden als *Non-Data-Ink* (siehe Abschnitt 2.7.1) bezeichnet. Die Minimierung von Non-Data-Ink ist aus Gründen der Übersichtlichkeit anzustreben. Nebenbei verringern sich dadurch die Druckkosten. Diese fallen zum Beispiel bei Dokumentationen im Rahmen eines Audits ins Gewicht.

V10: Chartjunk und Non-Data-Ink

Chartjunk lenkt den Betrachter ab und ist unerwünscht. Non-Data-Ink muss auf ein Minimum reduziert werden.

Manche Visualisierungen werden dazu missbraucht, dem Betrachter falsche Tatsachen zu suggerieren oder die Differenz zwischen zwei Werten viel größer zu vermitteln als sie in Wirklichkeit ist. Dies geschieht zum Beispiel durch Verschiebungen oder Verzerrungen innerhalb der Grafik. Der Grad der Abweichung der Visualisierung von den zugrundeliegenden Daten wird mit Hilfe des sog. *Lie Factor* ermittelt, der im Idealfall den Wert 1 besitzt (siehe Abschnitt 2.7.3). In einem Administrationstool muss der Lie Factor optimal sein, damit der Administrator die Visualisierungen nicht falsch interpretiert.

V11: Lie Factor

Der Lie Factor muss optimal sein, um die Aussage der Visualisierungen nicht zu verzerren und Fehlinterpretationen zu verhindern.

4.2.4. Zusammenfassung der Anforderungen

In Tabelle 4.1 werden die zuvor genannten Anforderungen noch einmal zusammengefasst.

4.3. Zusammenfassung

Das LRZ bietet – wie in Abschnitt 4.1 gezeigt wurde – ein breit gefächertes Portfolio an Diensten an und ist zur Diensterbringung auf den reibungslosen Betrieb seiner Linux-Server angewiesen. Um die Administration zu erleichtern, setzt das Rechenzentrum bereits diverse grafische Administrationstools ein. Da im Rahmen dieser Masterarbeit eine prototypische Anwendung entstehen soll, die den gewinnbringenden Einsatz von Informationsvisualisierung in der Linux-Server-Administration untersuchen soll, erfolgte in diesem Kapitel eine Analyse der Anforderungen an ein grafisches Administrationstool am LRZ. Die visuellen Anforderungen basieren auf dem Grundlagenteil aus Kapitel 2.

Nr.	Kurzbeschreibung
F1	**Beherrschbarmachen großer Datenmengen:** Große Datenmengen sollen automatisch beschafft, konsolidiert und durch geeignete Methoden der Informationsvisualisierung beherrschbar gemacht werden.
F2	**Softwarestände überwachen** Auf den Servern installierte Paketversionen sollen zentral abrufbar sein. Anhand einer Referenzliste soll die Aktualität und die Sicherheit der Pakete bestimmt werden.
F3	**Firewall-Konfigurationen überwachen:** Die Wirksamkeit und Sicherheit der Firewall-Konfigurationen auf den Servern soll an zentraler Stelle ersichtlich sein. Einzelne Konfigurationsdateien sollen bei Bedarf abrufbar sein.
F4	**Sicherheit privater X.509-Schlüssel überwachen:** Die Sicherheit privater X.509-Schlüssel aller Server soll bewertet und an zentraler Stelle ersichtlich sein. Ein Zugriff auf die Schlüssel ist nicht erlaubt.
F5	**Sicherheit von TLS-Konfigurationen überwachen:** Die Sicherheit und Wirksamkeit der TLS-Konfigurationen aller Server soll bewertet und zentral angezeigt werden.
F6	**Datenschutzrechtliche Aspekte:** Das Tool muss Datenschutzrechtliche Regelungen umsetzen. Datensparsamkeit ist essentiell.
NF1	**Bedienfreundlichkeit:** Bedienfreundlichkeit soll erreicht werden, indem das Tool dem Administrator das häufige Wechseln zu anderen Tools oder der Kommandozeile erspart. Es muss einfach und geräteübergreifend zu bedienen sein. Mehrsprachigkeit ist erwünscht.
NF2	**Reaktionszeit:** Als zentraler Anlaufpunkt soll das Tool sicherheitsrelevante Daten durch Informationsvisualisierung augenblicklich begreifbar machen und so eine geringe Reaktionszeit ermöglichen.
NF3	**Basistechnologien:** Durch den Verzicht auf proprietäre Basistechnologien sollen Unabhängigkeit geschaffen, Einstiegsbarrieren gesenkt und die Wartbarkeit der Anwendung erhöht werden.
NF4	**Informationssicherheit:** Das Tool soll grundlegende Sicherheitsfunktionen wie ein differenziertes Rechtesystem und LDAP-Authentifizierung implementieren. Webbasierte Anwendungen sollen insbesondere Schutzmaßnahmen gegen häufig durchgeführte Angriffstechniken bieten. Die Sicherheit der gespeicherten Daten ist zu gewährleisten.
NF5	**Integrierbarkeit in bestehende Umgebung:** Das Tool soll die am häufigsten verwendeten Linux-Server-Distributionen unterstützen. Schnittstellen zu vorhandenen DBMSs müssen vorhanden sein. Die Integrierbarkeit bzw. Kompatibilität zu vorhandenen Tools ist erwünscht.
V1	**Übersichtlichkeit:** Die Bedienoberfläche darf nicht überfrachtet sein, da sonst die Übersichtlichkeit darunter leidet.
V2	**Farben:** In der Bedienoberfläche müssen Farben sparsam und zielgerichtet eingesetzt werden.
V3	**Kontrast:** In der Bedienoberfläche müssen die Kontrastverhältnisse hoch sein.
V4	**Objektgröße:** In der Bedienoberfläche müssen Oberflächenelemente eine ausreichende Größe besitzen.
V5	**Visualisierungsformen:** Die Arten der Visualisierungen müssen zu den zugrundeliegenden Daten und Dimensionalitäten passen.
V6	**Beschriftungen:** Visualisierungen müssen mit den notwendigen Beschriftungen versehen sein, damit sie nicht fehlinterpretiert werden können.
V7	**Visual Clutter:** Visualisierungen müssen frei von Visual Clutter sein, um die Übersichtlichkeit zu verbessern.
V8	**Farbskalen:** Farben in Visualisierungen müssen sparsam und zielgerichtet eingesetzt werden. Farbige Datenreihen müssen gut voneinander unterscheidbar sein.
V9	**Kontraste in Visualisierungen:** Kontrastverhältnisse in Visualisierungen müssen hoch sein, um die Lesbarkeit zu optimieren.
V10	**Chartjunk und Non-Data-Ink:** Chartjunk lenkt den Betrachter ab und ist unerwünscht. Non-Data-Ink muss auf ein Minimum reduziert werden.
V11	**Lie Factor:** Der Lie Factor muss optimal sein, um die Aussage der Visualisierungen nicht zu verzerren und Fehlinterpretationen zu verhindern.

Tabelle 4.1.: Zusammenfassung der Anforderungen.

5. Evaluierung vorhandener Lösungen

Es existieren bereits zahlreiche Anwendungen, die Administratoren beim Aufrechterhalten der Sicherheit ihrer Server unterstützen. Zu diesen Werkzeugen zählen beispielsweise *Security Information and Event Managements (SIEMs)*, die sicherheitsrelevante Echtzeitdaten analysieren oder spezialisierte Logdatenanalyseprogramme. Diese Anwendungen setzen Methoden der Informationsvisualisierung ein, um Bedrohungen auf einen Blick sichtbar zu machen und eine schnelle Reaktion zu ermöglichen.

Sog. *Vulnerability Scanner* tragen proaktiv zur Sicherheit der Server bei, indem sie Sicherheitslücken und veraltete Software aufspüren, die von Angreifern ausgenutzt werden könnten. Administratoren können so vorsorglich Lücken schließen, bevor es zu einem Sicherheitsvorfall kommt. Auch diese Werkzeuge nutzen zum Teil Techniken der Informationsvisualisierung, um ihre Scanergebnisse zu verdeutlichen.

Tools für das Konfigurationsmanagement sind an Rechenzentren ebenfalls unverzichtbar. Gegenüber der Einzelkonfiguration von Systemen ermöglichen sie die Konfiguration vieler Systeme über eine zentrale Oberfläche. Es bleibt dem Administrator damit auch erspart, sich auf den einzelnen Servern einloggen zu müssen. Dies führt zu einer deutlichen Zeitersparnis. Auch diese Klasse von Tools greift zum Teil auf Methoden der Informationsvisualisierung zurück.

In diesem Kapitel wird eine repräsentative Auswahl von Administrationstools vorgestellt, die zu den zuvor genannten Klassen zählen. Einige davon finden bereits am LRZ Verwendung. Da sich diese Arbeit in erster Linie mit Informationsvisualisierung befasst, geschieht die Bewertung der ausgewählten Tools ebenfalls hauptsächlich anhand der visuellen Attribute, die aus der Anforderungsanalyse in Abschnitt 4.2.3 hervorgingen. Eine Bewertung der funktionalen und nichtfunktionalen Eigenschaften würde einen für jede Toolkategorie speziellen Katalog an funktionalen und nichtfunktionalen Anforderungen voraussetzen. Dies ist nicht Bestandteil der Aufgabenstellung, würde sich aber möglicherweise als Thema für eine andere Abschlussarbeit eignen.

Als Bewertungsgrundlage für die Umsetzung der visuellen Anforderungen der ausgewählten Tools wurden zum Teil Produktivsysteme herangezogen, die während der Anfertigung dieser Arbeit zugänglich waren. Teilweise wurde auch auf Testsysteme zurückgegriffen. Die Bewertungen der restlichen Tools erfolgte durch die Auswertung von Bildschirmfotos. Zunächst wird in diesem Kapitel jedes Tool einzeln betrachtet, am Ende erfolgt schließlich eine Gegenüberstellung der Bewertungsergebnisse.

Der Grad der Einhaltung einer bestimmten Anforderung wird durch das bekannte Notenschema ausgedrückt. Es umfasst in qualitativ absteigender Reihenfolge die Noten *sehr gut*, *gut*, *befriedigend*, *ausreichend* und *mangelhaft*. Auf die Note *ungenügend* wird verzichtet, da die Note *mangelhaft* bereits ein ausreichender Indikator für besonders schlechte Qualität ist. Bei der Benotung der visuellen Anforderungen ist zu betonen, dass die zugrundeliegenden Eigenschaften mehr oder weniger schwer quantifizierbar sind. Dies liegt unter anderem am Zusammenspiel vieler unterschiedlicher Faktoren, die in den einzelnen Bedienoberflächen und Visualisierungen anzutreffen sind. Subjektive Tendenzen können deshalb bei der Benotung nicht ausgeschlossen werden. Absolut objektive Bewertungsalgorithmen für die unterschiedlichen visuellen Anforderungen zu entwickeln würde sich als mögliches Thema für eine andere Abschlussarbeit eignen.

5.1. Vulnerability Scanner

Zur Kategorie *Vulnerability Scanner* zählen Programme, die auf Computern nach bekannten Sicherheitslücken suchen. Die dazu notwendigen Informationen beziehen Vulnerability Scanner aus einer Datenbank. Diese Programme dienen unter anderem dazu, offene Ports, Dateifreigaben, veraltete Paketversionen und von Servern bereitgestellte Dienste aufzuspüren.

Besonders bei einer großen Anzahl von Servern, wie es am LRZ der Fall ist, sind Vulnerability Scanner unverzichtbare Wertzeuge, um die Informationssicherheit der Systeme bewerten zu können. Dieser Abschnitt betrachtet zwei häufig genutzte Vulnerability Scanner und bewertet ihre Umsetzung von Informationsvisualisierung zur Unterstützung der Informationssicherheit.

5.1.1. Nessus

Dieser Abschnitt befasst sich mit dem Netzwerk- und Vulnerability Scanner *Nessus*. Das Programm wird seit Version 3.0 unter einer proprietären Lizenz vertrieben. Davor stand das Projekt unter der GNU General Public License (GPL).

Beschreibung

Der Vulnerability Scanner Nessus verfolgt das Client-Server-Prinzip: Die eigentliche Anwendung wird auf einem Server installiert, der Zugriff darauf erfolgt über verschlüsselte TLS-Verbindungen von beliebigen Clients aus. Um nach verschiedenen Sicherheitslücken scannen zu können, unterstützt Nessus das Laden von Plug-ins, die in der eigenen Skriptsprache Nessus Attack Scripting Language (NASL) erstellt wurden. An Betriebssystemen unterstützt Nessus Linux, Unix, Windows und Mac OS X.

Um einen Host nach Sicherheitslücken zu untersuchen, wird mit Hilfe des Clients eine sog. *Session* gestartet. Dort werden der Zielhost, die zu verwendenden Plug-ins und weitere Einstellungen festgelegt. Im Anschluss zeigt das Programm Informationen zu den offenen Ports und gefundenen Sicherheitslücken an.

Visualisierung

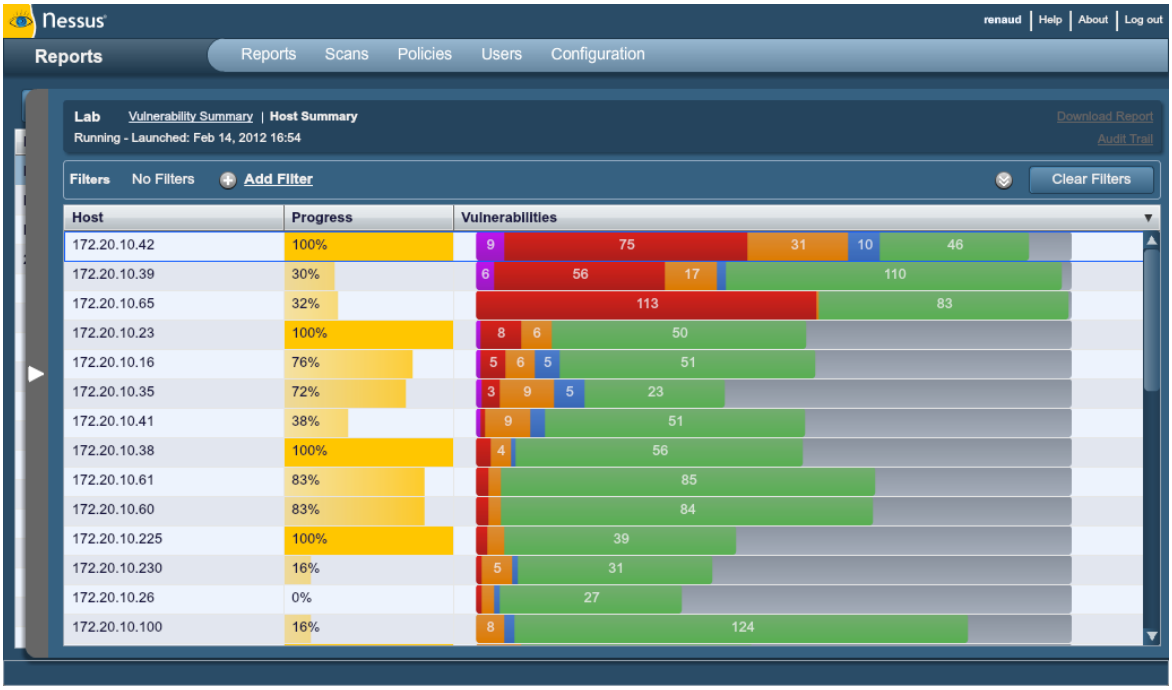


Abbildung 5.1.: Übersicht des Scanfortschritts verschiedener Hosts in Nessus (Bildausschnitt).[1]

[1] Bildquelle: http://4.bp.blogspot.com/-JrhBTouq5VE/T0ZDDJmXCDI/AAAAAAAAAgU/48QYOEyJWSU/s1600/1_Nessus5_HostSummary.png

In der Bedienoberfläche von Nessus werden durchwegs dunkelblaue Farbtöne für Hintergründe eingesetzt. Dies ist etwas ungeschickt, da diese Farbe dann bereits verbraucht ist und nicht mehr aussagekräftig in Visualisierungen verwendet werden kann. Um einen optimalen Kontrast zu gewährleisten, wäre entweder ein weißer oder ein schwarzer Hintergrund empfehlenswert.

Verbesserungswürdig ist der Kontrast auch bei Eingabeformularen. Während sich die hellen Eingabefelder selbst gut vom blauen Hintergrund abheben, gilt das nicht für die Formulare selbst, da sich ihre Hintergrundfarbe kaum vom Hintergrund der Anwendung selbst unterscheidet (siehe Abbildung 5.2). Anstatt das Kontrastproblem direkt zu lösen, greift die Anwendung auf Schattenverläufe zurück, um die Formulare etwas vom Hintergrund abzuheben.

In Abbildung 5.1 ist eine Tabelle zu sehen, die den Scanfortschritt einzelner Systeme und die Anzahl der gefundenen Sicherheitslücken kategorisiert in Form von Balkendiagrammen visualisiert. Es ist unverständlich, weshalb bei den zuletzt genannten Diagrammen im Gegensatz zu den Fortschrittsbalken eine deutlich dunklere Hintergrundfarbe verwendet wird. Sie wirkt sich nachteilig auf den Kontrast zu den Balken aus, die selbst keine besonders hellen Farben besitzen. Eine fehlende Skala erschwert die Lesbarkeit. Es fällt noch dazu auf, dass die einzelnen Balken überhaupt keinen gemeinsamen Maßstab besitzen: Das rote Segment in der ersten Zeile besitzt einen Wert von 75 und ist fast genauso breit wie das grüne Segment in der vierten Zeile mit dem Wert 50, um ein Beispiel zu nennen. Da kein gemeinsamer Maßstab vorhanden ist, erschwert dies den Vergleich der gefundenen Sicherheitslücken der einzelnen Hosts deutlich. Immerhin besitzen die Fortschrittsbalken links daneben eine einheitliche Skala.

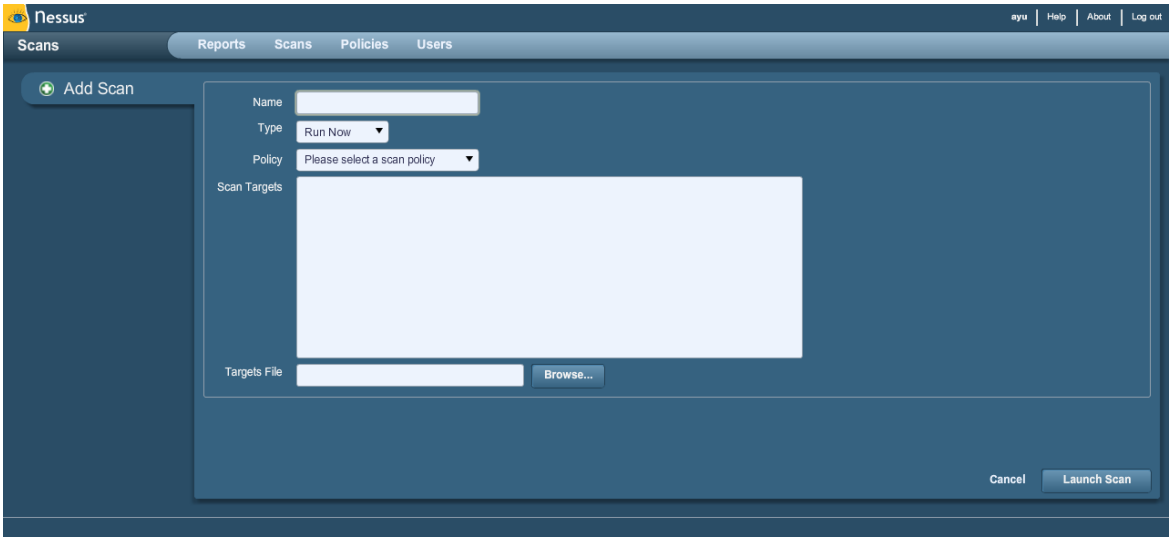


Abbildung 5.2.: Eingabeformular zur Definition eines neuen Scan-Auftrags in Nessus (Bildausschnitt).[2]

Die Textgröße oberhalb der Tabelle und in der Titelleiste ist grenzwertig. Besonders schlecht erkennbar sind die beiden dunkelgrauen Links “Download Report” und “Audit Trail” auf dunkelblauem Hintergrund im ersten Quadranten des Bildschirmfotos. Es ist nicht nachvollziehbar, warum ausgerechnet diese beiden Links zu wichtigen Programmfunktionen mit einem besonders schlechten Kontrast versehen wurden. Die Bildlaufleiste im rechten Bildbereich besitzt aufgrund der ähnlichen Farbe ebenfalls einen schlechten Kontrast zum Hintergrund. Durch eine geschicktere Farbwahl wäre auch die Größe des Bildausschnitts schneller auf einen Blick abschätzbar. Der Farbverlauf in der Bildlaufleiste ist überflüssig.

Unterschiedliche Farbabstufungen sollten nur verwendet werden, wenn Informationen darin abgebildet werden. Dies ist zum Beispiel bei Farbskalen der Fall (siehe Abschnitt 2.13.1). Nessus weicht von diesem Designprinzip ab und verwendet Farbverläufe hauptsächlich zur Verschönerung der Bedienoberfläche. So ist es zum Beispiel bei sämtlichen Balkendiagrammen der Fall (siehe Abbildung 5.1). Unter Berücksichtigung der Grundlagen der Informationsvisualisierung wären dort gleichfarbige Flächen besser geeignet.

[2]Bildquelle: `http://1.bp.blogspot.com/-9d9NVFssfa8/UHFCx0GHKuI/AAAAAAAAASs/xHweNw4f1Q0/s1600/installing%20nessus%20on%20backtrack%205%20r3%201.png`

Besser schlägt sich das Programm bei der Wahl der Visualisierungsformen. Abgesehen vom Kontrast und dem Maßstab sind die Diagramme selbst in einem akzeptablen Maß beschriftet. Als positiv ist zu bewerten, dass die Visualisierungen frei von Visual Clutter sind.

Bewertung der visuellen Anforderungen

Nr.	Kurzbeschreibung	Bewertung
V1	**Übersichtlichkeit:** Die Bedienoberfläche darf nicht überfrachtet sein, da sonst die Übersichtlichkeit darunter leidet.	ausreichend
V2	**Farben:** In der Bedienoberfläche müssen Farben sparsam und zielgerichtet eingesetzt werden.	ausreichend
V3	**Kontrast:** In der Bedienoberfläche müssen die Kontrastverhältnisse hoch sein.	ausreichend
V4	**Objektgröße:** In der Bedienoberfläche müssen Oberflächenelemente eine ausreichende Größe besitzen.	befriedigend
V5	**Visualisierungsformen:** Die Arten der Visualisierungen müssen zu den zugrundeliegenden Daten und Dimensionalitäten passen.	gut
V6	**Beschriftungen:** Visualisierungen müssen mit den notwendigen Beschriftungen versehen sein, damit sie nicht fehlinterpretiert werden können.	befriedigend
V7	**Visual Clutter:** Visualisierungen müssen frei von Visual Clutter sein, um die Übersichtlichkeit zu verbessern.	sehr gut
V8	**Farbskalen:** Farben in Visualisierungen müssen sparsam und zielgerichtet eingesetzt werden. Farbige Datenreihen müssen gut voneinander unterscheidbar sein.	gut
V9	**Kontraste in Visualisierungen:** Kontrastverhältnisse in Visualisierungen müssen hoch sein, um die Lesbarkeit zu optimieren.	ausreichend
V10	**Chartjunk und Non-Data-Ink:** Chartjunk lenkt den Betrachter ab und ist unerwünscht. Non-Data-Ink muss auf ein Minimum reduziert werden.	befriedigend
V11	**Lie Factor:** Der Lie Factor muss optimal sein, um die Aussage der Visualisierungen nicht zu verzerren und Fehlinterpretationen zu verhindern.	ausreichend
	Gesamtbewertung	**befriedigend**

Tabelle 5.1.: Bewertung der visuellen Anforderungen bei Nessus.

Fazit

Der Vulnerability Scanner Nessus ist dank des Client-Server-Prinzips auch in heterogenen Netzen flexibel einsetzbar. Mit Hilfe des Plug-In-Systems ist der Scanner an die Anforderungen des zu Scannenden Netzes anpassbar. Durch die eigene Skriptsprache lässt sich das System um neue Scantechniken erweitern.

Verbesserungswürdig fallen bei Nessus sowohl die Bedienoberfläche als auch die dynamisch generierten Visualisierungen aus. Die blaue Hintergrundfarbe der Bedienoberfläche und die zum Teil starken Kontrastprobleme wirken sich negativ auf die Übersichtlichkeit aus. Schatten und Farbverläufe laden die Bedienoberfläche mit unnötigem Ballast auf. Die Aussagekraft der Diagramme in Nessus leidet unter dem fehlenden gemeinsamen Maßstab.

Aufgrund seines Funktionsumfangs und der flexiblen Konfigurierbarkeit eignet sich Nessus prinzipiell für den Einsatz am LRZ, um die Sicherheit der Linux-Server über das Netz zu überprüfen. Die Bedienoberfläche und die Visualisierungen halten sich aber nicht konsequent an Grundregeln der Informationsvisualisierung. Administratoren könnten das Tool effizienter einsetzen, wenn diese Defizite nicht vorhanden wären.

5.1.2. Qualys SSL Server Test

Unter Betreibern von Webservern mit Hypertext Transfer Protocol Secure (HTTPS)-Zugang ist der kostenlose *SSL Server Test* der IT-Sicherheitsfirma *Qualys* ein beliebtes Werkzeug, um die Sicherheit der TLS-Konfiguration aus der Sicht eines Clients zu ermitteln.

Beschreibung

Die webbasierte Anwendung SSL Server Test kann kostenlos eingesetzt werden und prüft unterschiedliche Sicherheitsmerkmale von TLS-gesicherten Webservern wie die Vertrauenswürdigkeit des Serverzertifikats und die unterstützten Protokollversionen und Cipher Suites. Zu diesem Zweck führt sie eine Vielzahl von Handshakes durch, um alle Kombinationsmöglichkeiten gebräuchlicher Kryptographie-Algorithmen zu testen.

Für jede zu testende Kategorie gibt das Programm ein einzelnes Testergebnis in Form einer Zahl zwischen 0 und 100 Punkten aus. Die einzelnen Testergebnisse fließen in eine Gesamtbewertung in Form amerikanischer Schulnoten ein. Neben einer Bewertung der Serversicherheit gibt das Tool auch detaillierte Informationen zu den durchgeführten Einzeltests aus. Darüber hinaus erteilt es Ratschläge, wie die Sicherheit der TLS-Konfiguration optimiert werden kann.

Visualisierung

In Abbildung 5.3 ist ein Screenshot zu sehen, der einen Ausschnitt eines Testergebnisses zeigt. In diesem Beispiel wurde ein Server der Suchmaschine Google gescannt. Der Bildausschnitt zeigt die Zusammenfassung des Scanergebnisses. Lediglich an dieser Stelle wird ein Balkendiagramm verwendet, um die Bewertungen der einzelnen Kategorien *Certificate*, *Protocol Support*, *Key Exchange* und *Cipher Strength* zu visualisieren.

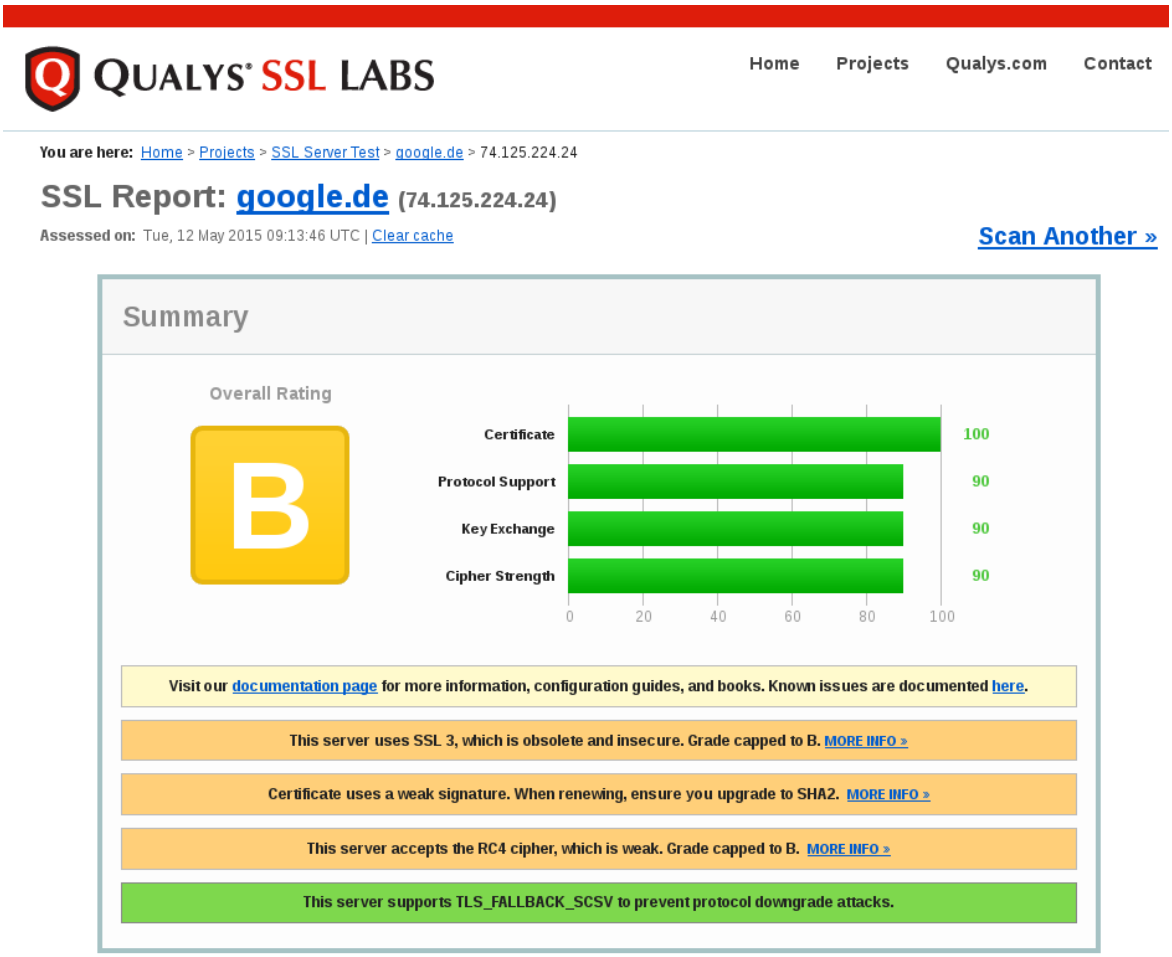


Abbildung 5.3.: Ein Testergebnis im SSL Server Test.[3]

[3]Bildquelle: `https://www.ssllabs.com/ssltest/analyze.html?d=google.de&s=74.125.224.24`
[4]Bildquelle: `http://goo.gl/km1Wvq`

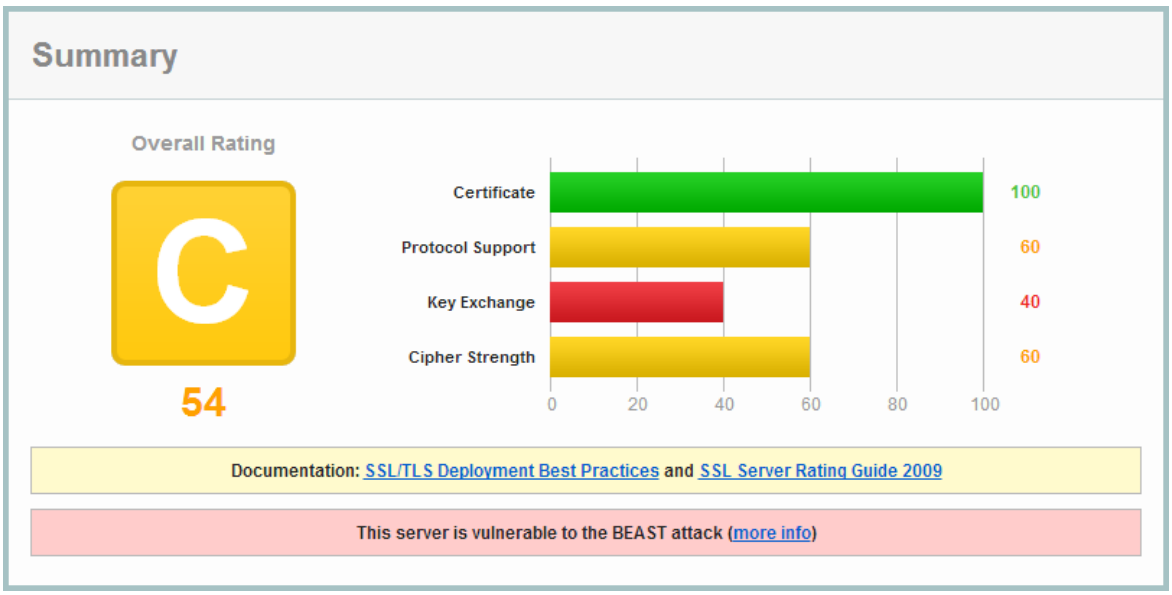


Abbildung 5.4.: Ein weiteres Testergebnis im SSL Server Test (Bildausschnitt).[4]

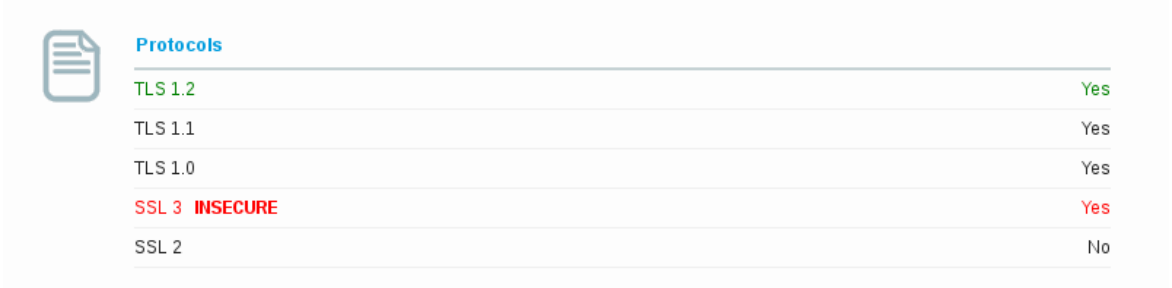


Abbildung 5.5.: Tabelle mit Prüfdetails zu SSL-Protokollen im SSL Server Test.[5]

Bei den Balken dieses Diagramms und den dazugehörigen Bewertungspunkten kommen Signalfarben zum Einsatz. Gute Testergebnisse werden grün, mittelmäßige gelb und schlechte rot eingefärbt. Im Beispielbild mit dem Google-Server fallen die Tests der Einzelkategorien durchweg positiv aus. Abbildung 5.4 zeigt ein Negativbeispiel, das besonders beim Schlüsselaustausch schlecht abschneidet.

Das Diagramm kommt ohne ablenkenden Chartjunk aus und macht einen aufgeräumten Eindruck. Über den leichten Farbverlauf der einzelnen Balken lässt sich streiten, da er keine Zusatzinformation verkörpert und lediglich die Optik des Diagramms aufhübschen soll. Aus Sicht der Informationsvisualisierung sind in diesem Fall Flächen mit einer einheitlichen Farbe zu bevorzugen.

Die Visualisierung der Gesamtbewertung anhand einer amerikanischen Schulnote erfolgt durch ein Piktogramm auf der linken Bildschirmseite unter der Überschrift *Overall Rating*, das ebenfalls mit Signalfarben arbeitet. Der Kontrast zwischen der grauen Überschrift und dem weißen Hintergrund ist nicht optimal, das Piktogramm selbst ist aber gut erkennbar.

Unterhalb der beiden Visualisierungen listet der SSL Server Test Ratschläge zur Verbesserung der TLS-Konfiguration auf. Besonders am Beispiel in Abbildung 5.3 sticht ins Auge, dass es einen inhaltlichen Widerspruch zwischen den Einzelwertungen und den gefundenen Sicherheitslücken gibt:

Der dort getestete Webserver unterstützt immer noch die veraltete und als unsicher geltende SSL-Version 3. Das Tool bezeichnet diese Protokollversion selbst als “obsolet” und “unsicher”, dennoch vergibt es in der Kategorie “Protocol Support” nahezu optimale 90 Punkte. Trotz aktivierter Ron’s Code 4 (RC4)-Unterstützung

[5]Bildquelle: `https://www.ssllabs.com/ssltest/analyze.html?d=google.de&s=74.125.224.24`

gibt es noch dazu in der Kategorie “Cipher Strength” ebenfalls 90 Punkte. Diese Wertungen sind irreführend, da dringender Handlungsbedarf besteht und SSL 3 deaktiviert werden muss. Immerhin erfolgt aufgrund dieser beiden Sicherheitslücken unabhängig von der Punktzahl eine Herabstufung der Gesamtnote auf “B”.

Ebenfalls unverständlich ist die volle Punktzahl für die Sicherheit des Serverzertifikats, obwohl ein schwacher Signatur-Hash-Algorithmus ermittelt wurde. Insgesamt ist eine andere Gewichtung der gefunden Sicherheitslücken dringend anzuraten. Diese beiden Missstände wirken sich besonders unvorteilhaft auf das Bewertungskriterium Lie Factor aus.

Unterhalb der Änderungsempfehlungen zeigt das Tool Detailinformationen zu den durchgeführten Tests in tabellarischer Form an. Bei den Überschriften der Tabellen verwendet die Webseite erstmals eine hellblaue Farbe. Im Gesamtkontext ist das nicht einheitlich, da die anderen Überschriften in Grautönen gehalten sind. Eine Zusatzinformation wird durch die hellblaue Farbe auch nicht vermittelt. Eine Verwechslungsgefahr mit den anderen auf der Webseite verwendeten Signalfarben besteht allerdings nicht.

Neben den Tabellenüberschriften befinden sich zur Verdeutlichung noch einfarbige Piktogramme. Der Informationsgehalt dieser Piktogramme ist allerdings gering, da sie aufgrund ihrer Abstrakten Form alle sehr ähnlich aussehen. Die Piktogramme tragen deshalb nicht besonders gut zur Übersichtlichkeit bei und sind überflüssig.

Abbildung 5.5 zeigt ein Beispiel für eine solche Tabelle. Innerhalb der Tabelle kommen die bekannten Signalfarben zum Einsatz, zum Beispiel die Farbe Rot beim unsicheren Protokoll SSL 3 oder die Farbe Grün beim zu bevorzugenden TLS 1.2. Während der rote Text ins Auge sticht, ist der Kontrast zwischen dem grünen Text und dem weißen Hintergrund etwas verbesserungswürdig.

Die Diagramme, die das Tool zeichnet, sind zwar nicht besonders spektakulär, dafür aber vollständig beschriftet. In den Diagrammen werden im Gegensatz zu den Kommentaren darunter ausschließlich gut zu unterscheidende Primärfarben verwendet. Die Objektgröße ist als gut zu bezeichnen.

Bewertung der visuellen Anforderungen

Nr.	Kurzbeschreibung	Bewertung
V1	**Übersichtlichkeit:** Die Bedienoberfläche darf nicht überfrachtet sein, da sonst die Übersichtlichkeit darunter leidet.	gut
V2	**Farben:** In der Bedienoberfläche müssen Farben sparsam und zielgerichtet eingesetzt werden.	gut
V3	**Kontrast:** In der Bedienoberfläche müssen die Kontrastverhältnisse hoch sein.	befriedigend
V4	**Objektgröße:** In der Bedienoberfläche müssen Oberflächenelemente eine ausreichende Größe besitzen.	gut
V5	**Visualisierungsformen:** Die Arten der Visualisierungen müssen zu den zugrundeliegenden Daten und Dimensionalitäten passen.	gut
V6	**Beschriftungen:** Visualisierungen müssen mit den notwendigen Beschriftungen versehen sein, damit sie nicht fehlinterpretiert werden können.	sehr gut
V7	**Visual Clutter:** Visualisierungen müssen frei von Visual Clutter sein, um die Übersichtlichkeit zu verbessern.	gut
V8	**Farbskalen:** Farben in Visualisierungen müssen sparsam und zielgerichtet eingesetzt werden. Farbige Datenreihen müssen gut voneinander unterscheidbar sein.	gut
V9	**Kontraste in Visualisierungen:** Kontrastverhältnisse in Visualisierungen müssen hoch sein, um die Lesbarkeit zu optimieren.	gut
V10	**Chartjunk und Non-Data-Ink:** Chartjunk lenkt den Betrachter ab und ist unerwünscht. Non-Data-Ink muss auf ein Minimum reduziert werden.	gut
V11	**Lie Factor:** Der Lie Factor muss optimal sein, um die Aussage der Visualisierungen nicht zu verzerren und Fehlinterpretationen zu verhindern.	mangelhaft
	Gesamtbewertung	**gut**

Tabelle 5.2.: Bewertung der visuellen Anforderungen beim SSL Server Test.

Fazit

Der SSL Server Test ist ein nützliches Tool, um die Sicherheit eines HTTPS-Webservers aus Sicht der Clients zu ermitteln. Am informativsten sind die tabellarischen Details zu den unterschiedlichen Einzeltests. Da das Tool imstande ist, unterschiedliche Browser und Plattformen zu simulieren, erspart es dem Webseitenbetreiber viel Zeit, die sonst in manuelle Tests einfließen müsste.

Ein paar Abstriche gibt es in visueller Hinsicht. Der Einsatz von Oberflächenfarben und Piktogrammen ist teilweise verbesserungswürdig. Als gravierend anzusehen ist die Verzerrung der Testergebnisse durch die Diagramme: Auch wenn sicherheitsrelevante Lücken gefunden wurden, die sich zum Teil ohne großen Aufwand schließen lassen, signalisiert das Tool durch eine hohe Punktzahl und die beruhigende Farbe Grün falsche Tatsachen. Trotz der genannten Defizite fällt die Gesamtbewertung der visuellen Eigenschaften gut aus.

Neben den visuellen Defiziten gibt es funktionale Punkte, die zwar nicht in die Bewertung eingehen, aber der Vollständigkeit halber erwähnt werden müssen: Die Scanergebnisse werden zentral gespeichert und sind von beliebige Personen online abrufbar. Es ist nicht auszuschließen, dass Hacker die kompletten Datenbankinhalte des Betreibers herunterladen und so an Informationen über die Sicherheitslücken unzähliger Webserver gelangen. Zudem ist der SSL Server Test nur zum Testen von Systemen geeignet, die über das Internet erreichbar sind. Für interne Systeme müssen andere Vulnerability Scanner eingesetzt werden.

Da am LRZ eine große Anzahl an Servern betrieben wird, ist zudem eine Lösung erforderlich, die die Sicherheit der TLS-Konfiguration aller Systeme auf einen Blick gewährleistet. Der SSL Server Test ist dazu nicht geeignet, da er immer nur einen einzelnen Server betrachtet.

5.2. Konfigurationsmanagement-Tools

Zur Familie der *Konfigurationsmanagement-Tools* gehören Anwendungen, die die Durchführung wiederkehrender Aufgaben auf einer großen Anzahl von Servern erleichtern. Die Bereitstellung virtueller Server geschieht ebenfalls über diese Art von Tools. Je größer die Anzahl an Servern ist, desto mehr Zeitersparnis gewähren Konfigurationsmanagement-Tools den Administratoren. Das LRZ mit seiner sehr hohen Anzahl an Linux-Servern ist ein Paradebeispiel hierfür.

In diesem Abschnitt wird eine repräsentative Auswahl von Konfigurationsmanagement-Tools vorgestellt, die zur Gewährleistung und Verbesserung der Informationssicherheit eingesetzt werden. Auch die Bewertung dieser Tools geschieht schwerpunktmäßig anhand des Einsatzes von Methoden der Informationsvisualisierung.

5.2.1. Ansible Tower

Ansible Tower[6] ist eine grafische Bedienoberfläche für die Konfigurationsmanagement-Lösung Ansible des gleichnamigen Herstellers. Dieser Abschnitt widmet sich der Bewertung dieses Tools.

Beschreibung

Ansible Tower dient in erster Linie dazu, sog. *Jobs* wie zum Beispiel Software-Installationen oder Konfigurationsänderungen auf den Rechnern in der Domäne zu starten und zu überwachen. Der Fokus der Anwendung liegt laut Herstellerangaben auf Minimalismus, Sicherheit und geringem Einarbeitungsaufwand. Zudem besitzt Ansible Tower eine auf Representational State Transfer (REST) basierende API, um bestimmte Programmfunktionen in eigenen Anwendungen nutzen zu können.

Die Installation eines speziellen Hintergrunddienstes auf den verwalteten Systemen ist nicht erforderlich, da Ansible sämtliche Eingriffe über SSH-Verbindungen durchführt. Dadurch entfällt die Notwendigkeit, auch

[6] Ansible-Website: `http://www.ansible.com/`
[7] Bildquelle: `http://goo.gl/zNsFnr`

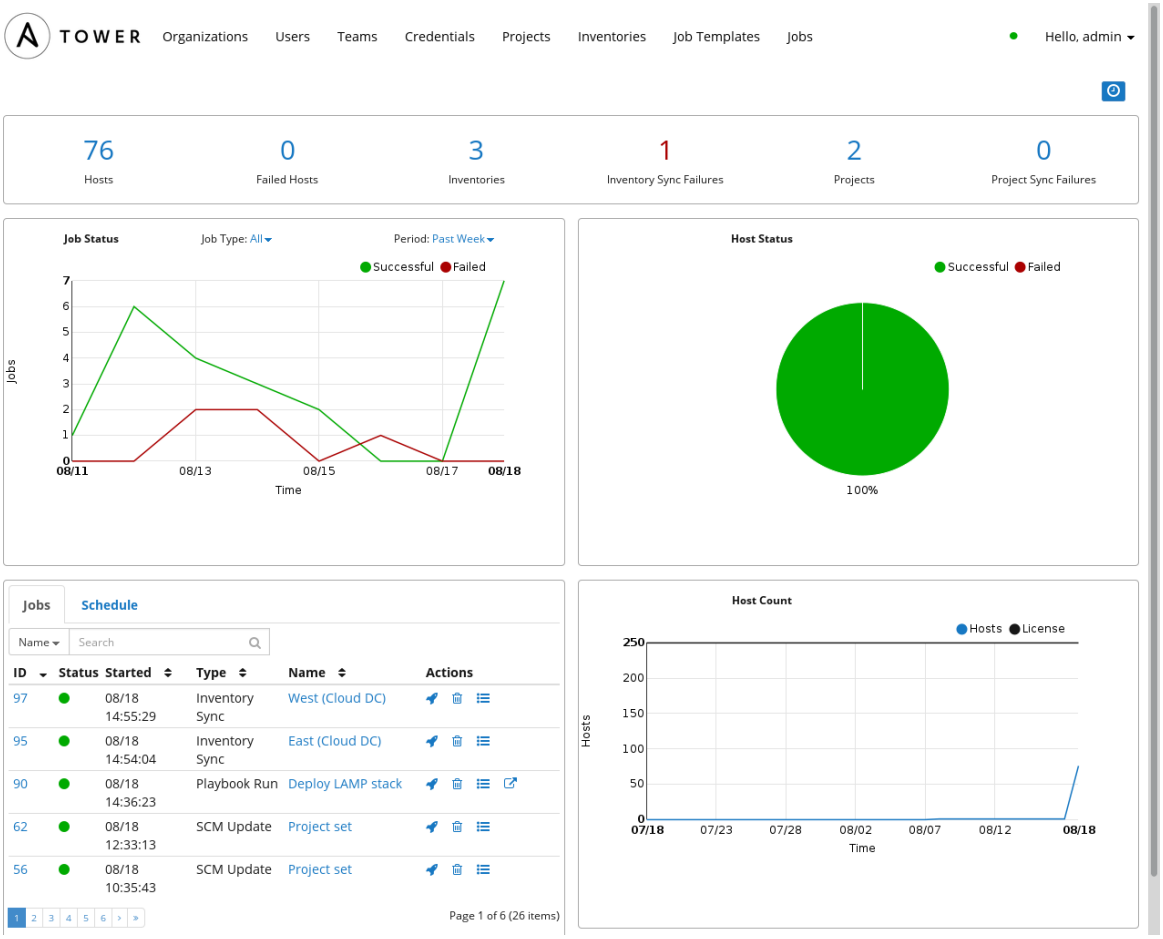


Abbildung 5.6.: Startbildschirm von Ansible Tower mit Informationen zu Hosts und Aufgaben.[7]

den Dienst regelmäßig aktuell zu halten. Das spart nicht nur Zeit, sondern schließt auch Probleme aus, die bei einem fehlerhaft konfigurierten oder gar beendeten Hintergrunddienst auftreten könnten.

Für die SSH-Verbindungen greift die Anwendung auf individuell vom Administrator eingerichtete Benutzerkonten zurück, die nur über eingeschränkte Rechte verfügen. Ein genereller Root-Zugriff ist nicht erforderlich, stattdessen verwendet das Tool `sudo`, um temporär Administratorrechte zu erhalten. Das funktioniert auch nur dann, wenn dies vom Administrator auf dem Server explizit so konfiguriert wurde.

Das Ressourcenmodell von Ansible ist zustandsorientiert. Es beschreibt die Zustände, in denen sich ein Dienst oder ein kompletter Server befinden kann. Konfigurationsänderungen erfolgen dadurch, dass der Administrator den gewünschten Zielzustand auswählt. Dies unterscheidet Ansible von skriptbasierten Lösungen, die explizite Aktionen und keine Endzustände beschreiben. Ein optionaler Probelauf zeigt ihm vor dem Eingriff ins Produktivsystem die konkreten Konfigurationsänderungen an, die Ansible zum Erreichen des Ziels durchführen will. Das Programm speichert Informationen zu jeder Zustandsänderung in seiner Datenbank ab und macht auch lückenlos nachvollziehbar, wer eine bestimmte Änderung angestoßen hat.

Visualisierung

Die Bedienoberfläche von Ansible Tower macht einen sehr aufgeräumten Eindruck. Farben werden für Schaubilder reserviert, die Bedienoberfläche selbst setzt lediglich einen Blauton als Akzentfarbe ein. Verwechslungsgefahren bei der Bedeutung der Farben sind allerdings nicht vollständig ausgeschlossen, da die Farbe

[8] Bildquelle: `http://goo.gl/bjgZFn`

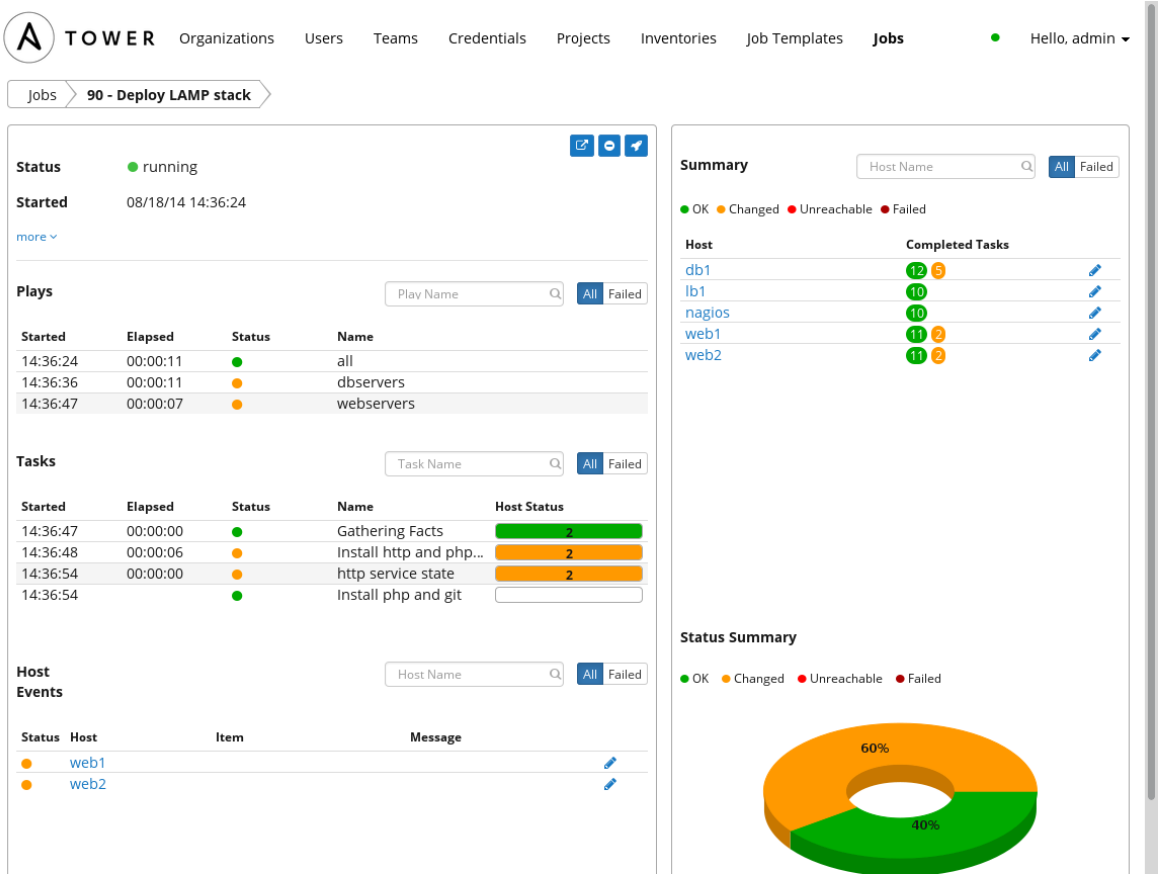


Abbildung 5.7.: Übersicht laufender Aufgaben in Ansible Tower.[8]

Blau auch in dem Liniendiagramm unten rechts auf der Startseite vorkommt (siehe Abbildung 5.6).

Ansible Tower greift auf etablierte Techniken der Informationsvisualisierung zurück. Der Startbildschirm bietet Administratoren eine Statusübersicht in Form von Tabellen und Diagrammen (siehe Abbildung 5.6). Er enthält ein Liniendiagramm, das die Anzahl der aktiven Jobs und ihren Zustand im Lauf der Zeit darstellt. Ein weiteres Liniendiagramm stellt die Anzahl der Hosts und der verfügbaren Lizenzen zeitlich dar. Der allgemeine Zustand der Hosts wird anhand eines Kreisdiagramms visualisiert.

Eine Tabelle listet die konkreten Jobs mit Beschreibungstexten, aktuellem Zustand und möglichen Interaktionen auf. Zur Kennzeichnung der beiden Zustände *Successful* und *Failed* verwendet Ansible Tower die naheliegenden Signalfarben Grün und Rot.

Trotz der sehr aufgeräumten Bedienoberfläche gibt es Abstriche, die sich negativ auf die Bedienfreundlichkeit auswirken: Die Schriftgröße ist besonders bei der Statusübersicht in der Kopfzeile und bei den Diagrammen zu klein. Dasselbe gilt für die Schaltflächen wie zum Beispiel beim Paginator ganz unten links. Dort muss der Anwender genau zielen, um nicht daneben zu klicken.

Abbildung 5.7 zeigt die Detailansicht eines Jobs, der die Software für eine Linux Apache MySQL PHP (LAMP)-Umgebung installieren soll. Die Fortschritte der Aufgabenabarbeitung visualisiert Ansible Tower durch farbige Fortschrittsbalken. In einem Kreisdiagramm zeigt die Anwendung eine relative Übersicht der anteiligen Konfigurationsschritte an.

Ansible Tower greift auf eine nachvollziehbare Auswahl von Signalfarben zurück: Schritte, bei denen keine Änderungen am Server notwendig waren, werden grün gekennzeichnet, Schritte mit Änderungsbedarf in der Farbe Orange. Fehlgeschlagene Schritte werden dunkelrot gekennzeichnet. Falls ein Server nicht erreichbar war, wird dieser Zustand rot markiert. Neben allen Informationen kann sich der Administrator bei Bedarf auch nur Fehlschläge einblenden lassen. Dadurch kann er sich besser auf die Systeme konzentrieren, bei denen ein

manueller Eingriff notwendig ist.

Bei dem Kreisdiagramm mit der Beschriftung “Status Summary” macht Ansible Tower den ersten nennenswerten Fehler anhand der Wahl des Diagrammtyps: Die dritte Dimension in der Visualisierung dient keiner Zusatzinformation und ist daher überflüssig. Als Alternative zu dem dreidimensionalen Kreisdiagramm würde sich ein zweidimensionales Pendant besser anbieten. Um Platz in der Senkrechten einzusparen, könnte alternativ auch ein gestapeltes Balkendiagramm verwendet werden.

Inkonsequent ist Ansible Tower auch bei der Farbe der Diagrammbeschriftungen. Obwohl in Abbildung 5.7 sowohl bei den kleinen Kreisen in der Spalte “Completed Tasks” als auch bei den Balken- und dem Kreisdiagramm die gleichen Hintergrundfarben verwendet werden, ist die Textfarbe in den Kreisen weiß und in den Diagrammen schwarz.

Der geringe Kontrast der schwarzen Textfarbe in den grünen Diagrammsegmenten wirkt sich nachteilig auf die Lesbarkeit aus. Die Schriftgröße der Diagrammlegenden auf der rechten Seite in Abbildung 5.7 ist grenzwertig. Dasselbe gilt auch hier für die Schaltflächen.

Als besonders positiv ist hervorzuheben, dass die Diagramme weder Chartjunk noch bedeutungslose Farbverläufe besitzen. Es sind auch keine Verzerrungen der Daten festzustellen. Die Konzentration auf die zugrundeliegenden Daten fällt so leichter.

Bewertung der visuellen Anforderungen

Nr.	Kurzbeschreibung	Bewertung
V1	**Übersichtlichkeit:** Die Bedienoberfläche darf nicht überfrachtet sein, da sonst die Übersichtlichkeit darunter leidet.	gut
V2	**Farben:** In der Bedienoberfläche müssen Farben sparsam und zielgerichtet eingesetzt werden.	gut
V3	**Kontrast:** In der Bedienoberfläche müssen die Kontrastverhältnisse hoch sein.	gut
V4	**Objektgröße:** In der Bedienoberfläche müssen Oberflächenelemente eine ausreichende Größe besitzen.	befriedigend
V5	**Visualisierungsformen:** Die Arten der Visualisierungen müssen zu den zugrundeliegenden Daten und Dimensionalitäten passen.	befriedigend
V6	**Beschriftungen:** Visualisierungen müssen mit den notwendigen Beschriftungen versehen sein, damit sie nicht fehlinterpretiert werden können.	befriedigend
V7	**Visual Clutter:** Visualisierungen müssen frei von Visual Clutter sein, um die Übersichtlichkeit zu verbessern.	gut
V8	**Farbskalen:** Farben in Visualisierungen müssen sparsam und zielgerichtet eingesetzt werden. Farbige Datenreihen müssen gut voneinander unterscheidbar sein.	sehr gut
V9	**Kontraste in Visualisierungen:** Kontrastverhältnisse in Visualisierungen müssen hoch sein, um die Lesbarkeit zu optimieren.	gut
V10	**Chartjunk und Non-Data-Ink:** Chartjunk lenkt den Betrachter ab und ist unerwünscht. Non-Data-Ink muss auf ein Minimum reduziert werden.	sehr gut
V11	**Lie Factor:** Der Lie Factor muss optimal sein, um die Aussage der Visualisierungen nicht zu verzerren und Fehlinterpretationen zu verhindern.	sehr gut
	Gesamtbewertung	**gut**

Tabelle 5.3.: Bewertung der visuellen Anforderungen bei Ansible Tower.

Fazit

Die Visualisierungen von Ansible Tower konzentrieren sich auf den Anwendungsfall Konfigurationsmanagement. Echtzeitdaten werden nicht visualisiert, da hierzu die Installation eines Agenten auf den Servern erforderlich wäre. Ansonsten müssten laufend SSH- Verbindungen aufrechterhalten oder in kurzen Abständen hergestellt werden. Durch den Fokus auf das Konfigurationsmanagement sind auch keine Visualisierungen

vorhanden, die zur Überwachung sicherheitskritischer Systeme wie etwa der Personal Firewalls dienen. Für solche Zwecke müssen Administratoren auf ein zusätzliches Tool zurückgreifen.

Das Tool hält sich größtenteils an Grundlagen der Informationsvisualisierung. Die Bedienoberfläche kommt ohne unnötigen Ballast aus, jedoch haben es die Entwickler bei den Schriftgrößen mit dem Minimalismus ein wenig übertrieben. Dies wirkt sich nachteilig auf die Bedienfreundlichkeit aus. Außerdem gibt es bei einigen Details wie der unnötigen Dreidimensionalität und der uneinheitlichen Beschriftungsfarben Nachbesserungsbedarf. Insgesamt fällt die Bewertung der visuellen Eigenschaften des Tools gut aus.

5.2.2. BlueCat Proteus

Proteus ist ein IP Address Management (IPAM)-System der Firma *BlueCat Networks*.[9] Es dient zur zentralen Verwaltung unternehmenseigener Netzressourcen wie PCs, Servern, IP-Adressen, MAC-Adressen, Rechnernetzen, Hostnamen und vielen mehr.

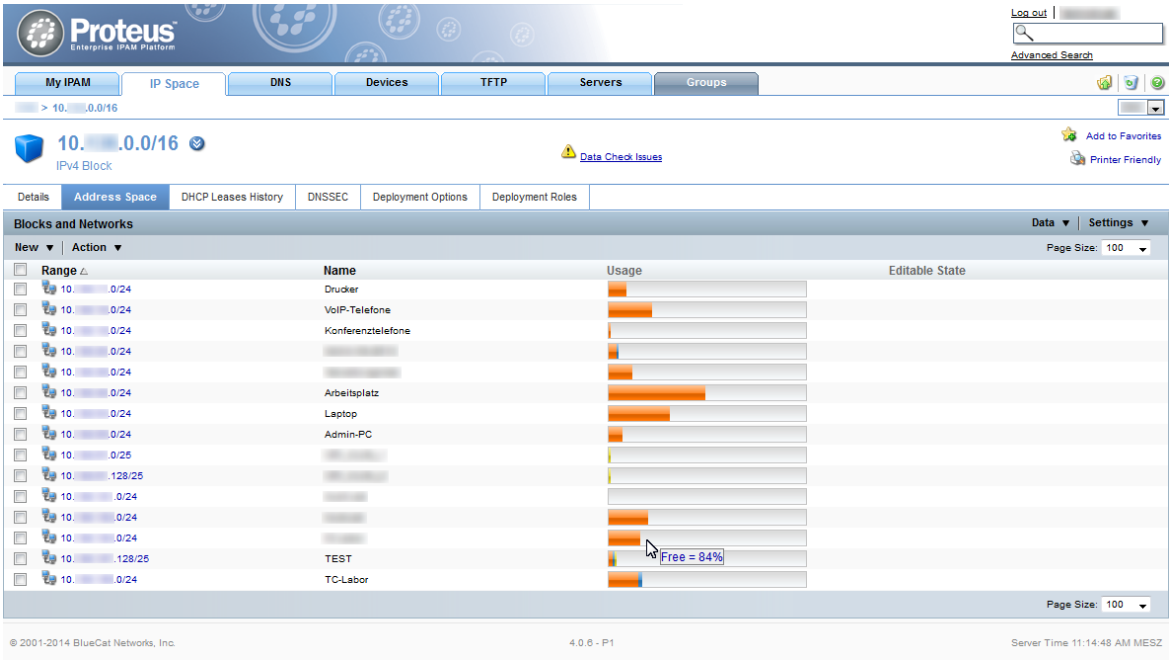


Abbildung 5.8.: Übersicht von Rechnernetzen und der Anteile der belegten IP-Adressen in Proteus.

Beschreibung

Beim Begriff *IPAM* denkt man zunächst nicht in erster Linie an die Sicherheit von Linux-Servern, doch es gibt Berührungspunkte mit diesem Thema: Durch ein einheitliches, webbasiertes Management erfolgt die Konfiguration der IPAM-Parameter hunderter Server auf eine ebenso schnelle wie auch syntaktisch korrekte Art und Weise. Konfigurationsfehler, die ein Angreifer ausnutzen könnte, werden so ausgeschlossen. Als Vertreter von IPAM-Werkzeugen unterstützt Proteus den Administrator außerdem bei der Absicherung der verwalteten DNS-Zonendaten mittels Domain Name System Security Extensions (DNSSEC), um zum Beispiel *Cache-Poisoning*-Angriffen entgegenzuwirken.

Neben der Asset-Management-Funktionalität dient Proteus zur vereinfachten Konfiguration der DNS-, DHCP- und Network Time Protocol (NTP)-Appliance *Adonis* desselben Herstellers. Der Administrator führt zunächst die gewünschten Konfigurationsänderungen an der webbasierten Oberfläche durch und stößt danach ein Deployment auf seine Adonis-Appliance an, um die Änderungen wirksam werden zu lassen.

[9] BlueCat-Networks-Website: `https://www.bluecatnetworks.com/`

Aufgrund seiner Mandantenfähigkeit kann eine Proteus-Instanz von mehreren Unternehmensstandorten oder Instituten gleichzeitig verwendet werden. Zudem bietet Proteus eine Simple Object Access Protocol (SOAP)-Schnittstelle an, die von Entwicklern genutzt werden kann, um aus ihren eigenen Anwendungen heraus auf Funktionen von Proteus zurückzugreifen.

Visualisierung

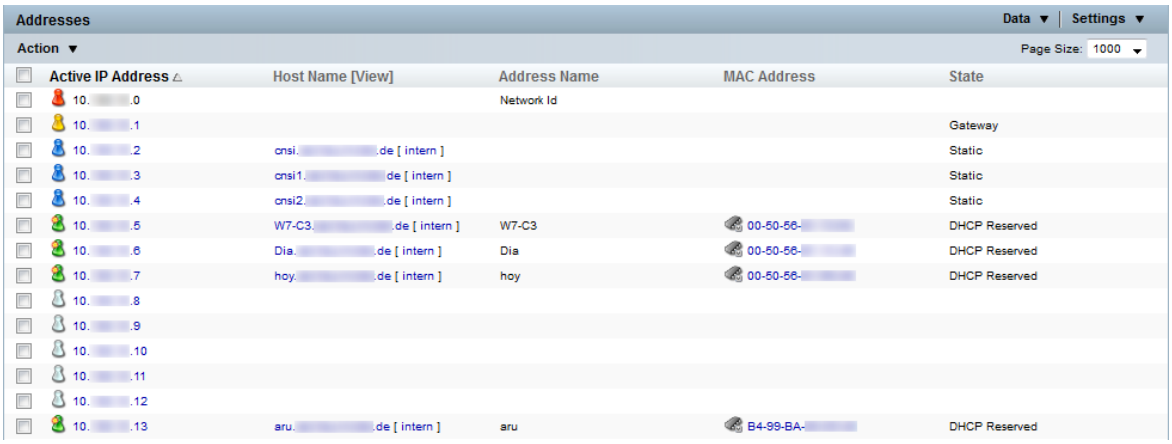


Abbildung 5.9.: Ausschnitt der Detailansicht eines Rechnernetzes. Die Zustände der IP-Adressen sind durch Piktogramme unterschiedlicher Farbe gekennzeichnet.

In der Bedienoberfläche von Proteus dominieren blaue bzw. blaugraue Farbtöne. Farbverläufe sollen die Oberfläche scheinbar attraktiver machen, tragen aber nicht dazu bei. Die hohe Anzahl unterschiedlicher Blautöne macht diese Farbe in Visualisierungen unwirksam. Die Oberfläche hinterlässt auf den Anwender insgesamt den Eindruck, als wären ihre Bestandteile von unterschiedlichen Personen entworfen und anschließend zusammengefügt worden. Die hellblauen Tabs passen nicht so recht in das Gesamtbild. Die Übersichtlichkeit leidet unter dem Patchwork-Charakter der Anwendung.

Proteus stellt dem Administrator unter anderem Übersichtstabellen der verfügbaren Rechnernetze zur Auswahl bereit. Den Füllstand der einzelnen Netze visualisiert Proteus durch gestapelte Balkendiagramme. Unterschiedliche Farben kennzeichnen die möglichen Zustände, die eine IP-Adresse im Netz haben kann. Die Bedeutung der Farben ist mangels Legende leider nicht sofort ersichtlich. Erst Tooltip-Texte geben Auskunft über die Bedeutung. Ein Beispiel für eine Netzübersicht in Proteus ist in Abbildung 5.8 dargestellt.

Aus diesen Visualisierungen geht leider nur der relative Füllstand hervor. Die absolute Anzahl freier und belegter IP-Adressen ist nicht sichtbar. Die Prozentsätze sind standardmäßig ausgeblendet und nur in Form von Tooltip-Texten erhältlich. Anwender von Touchbildschirmen können diese aus technischen Gründen nicht ohne weitere Hilfsmittel einblenden. Die Farbverläufe in den Diagrammen dienen nicht zur Anzeige zusätzlicher Informationen und sind deshalb fehl am Platz. Abgesehen von den ungenügenden Diagrammbeschriftungen ist der optimale Lie Factor der Diagramme als sehr gut zu bewerten.

Um die absoluten Zahlen zu erhalten, muss der Anwender erst umständlich die Größe des Netzes anhand seiner Adresse in Classless Inter-Domain Routing (CIDR)-Notation berechnen und dann den entsprechenden Prozentsatz darauf anwenden. Zusätzliche Tabellenspalten mit den absoluten Zahlen würden dieses Problem lösen. Schließlich unterstützt Proteus die Auswahl angezeigter Spalten durch den Benutzer und es wäre noch genügend Platz vorhanden.

Als zusätzliche Methode der Informationsvisualisierung setzt Proteus kleine Piktogramme ein. Sie finden hauptsächlich in den tabellarischen Übersichten der verwalteten Ressourcen Verwendung. Sie liefern nicht immer einen Mehrwert, da sie in einigen Übersichten redundant sind. In einer Tabelle, die ausschließlich Rechnernetze beinhaltet, taucht am Anfang jeder Zeile immer dasselbe Piktogramm auf (siehe Abbildung 5.8). Diese überflüssigen Informationen verringern die Übersichtlichkeit der Anwendung.

Sinnvoller setzt Proteus Piktogramme in den Detailansichten der Rechnernetze ein. Dort kennzeichnen kleine Spielhütchen durch ihre unterschiedlichen Farben die Zustände der dazugehörigen IP-Adresse. Proteus setzt hier gut unterscheidbare, kanonische Farben zur Kodierung ein. Die Bezeichnung des Zustands ist in der Spalte *State* zusätzlich angegeben.

Nicht nachvollziehbar ist ein Widerspruch bei den Farben zwischen den beiden zuvor genannten Ansichten: Während in der Gesamtübersicht der Netze in Abbildung 5.8 die Farbe Orange für den Zustand “DHCP Reserved” verwendet wird, kommen in den Detailansichten der Netze grüne Piktogramme hierfür vor. Die anderen Farben stimmen jedoch schon überein.

Auch in der zuletzt genannten Ansicht werden überflüssige Piktogramme verwendet: Jede MAC-Adresse besitzt das gleiche Symbol. Abbildung 5.9 zeigt den Ausschnitt aus der Detailansicht eines IP-Netzes.

Die restlichen Visualisierungen, die Proteus bereitstellt, befassen sich ausschließlich mit dem Zustand der angebundenen Adonis-Appliances. Der Anwender kann sich unterschiedliche Diagramme mit Statusinformationen anzeigen lassen. Die folgenden Metriken stehen dabei zur Auswahl:

- DNS Queries pro Sekunde
- DHCP Leases pro Sekunde
- Netzwerkverkehr
- Prozentuale CPU-Auslastung
- Prozentuale RAM-Auslastung
- Prozentuale Nutzung der Festplatte

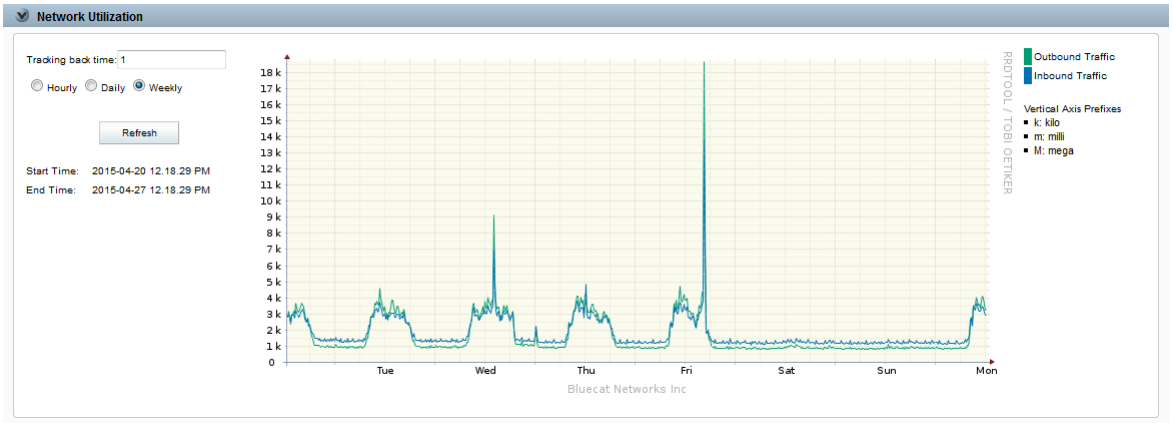


Abbildung 5.10.: Visualisierung der Netzwerkaktivität in Proteus.

Abbildung 5.10 zeigt eine Visualisierung, die aus dem Bereich der Systemzustände stammt. Dargestellt wird dort der Traffic, der über die Ethernet-Schnittstelle des Servers geht. Es wird zwischen eingehenden und ausgehenden Daten unterschieden. In dem Diagramm könnten die beiden Datenlinien durch die Wahl anderer Farben voneinander besser unterscheidbar sein. Hier ist der Kontrast zwischen den Farben zu gering.

Die zahlreichen Hilfslinien im Hintergrund der Grafik stellen unnötigen Ballast dar. Bei der Hintergrundfarbe handelt es sich um Non-Data-Ink und sie ist deshalb überflüssig (siehe Abschnitte 2.7.1 und 2.7.2). Auch in der sog. “druckerfreundlichen” Ansicht ist die Hintergrundfarbe noch vorhanden, obwohl dort unnötigerweise Tinte bzw. Toner verschwendet wird und auch die Lesbarkeit darunter leidet. Der Copyright-Hinweis in dem Diagramm ist zwar gerechtfertigt, aber er stört. Er beinhaltet keine für die Visualisierung relevante Information und wiederholt sich in jedem weiteren Diagramm, das auf der Webseite angezeigt wird.

Bewertung der visuellen Anforderungen

Nr.	Kurzbeschreibung	Bewertung
V1	**Übersichtlichkeit:** Die Bedienoberfläche darf nicht überfrachtet sein, da sonst die Übersichtlichkeit darunter leidet.	befriedigend
V2	**Farben:** In der Bedienoberfläche müssen Farben sparsam und zielgerichtet eingesetzt werden.	befriedigend
V3	**Kontrast:** In der Bedienoberfläche müssen die Kontrastverhältnisse hoch sein.	befriedigend
V4	**Objektgröße:** In der Bedienoberfläche müssen Oberflächenelemente eine ausreichende Größe besitzen.	gut
V5	**Visualisierungsformen:** Die Arten der Visualisierungen müssen zu den zugrundeliegenden Daten und Dimensionalitäten passen.	befriedigend
V6	**Beschriftungen:** Visualisierungen müssen mit den notwendigen Beschriftungen versehen sein, damit sie nicht fehlinterpretiert werden können.	ausreichend
V7	**Visual Clutter:** Visualisierungen müssen frei von Visual Clutter sein, um die Übersichtlichkeit zu verbessern.	befriedigend
V8	**Farbskalen:** Farben in Visualisierungen müssen sparsam und zielgerichtet eingesetzt werden. Farbige Datenreihen müssen gut voneinander unterscheidbar sein.	befriedigend
V9	**Kontraste in Visualisierungen:** Kontrastverhältnisse in Visualisierungen müssen hoch sein, um die Lesbarkeit zu optimieren.	befriedigend
V10	**Chartjunk und Non-Data-Ink:** Chartjunk lenkt den Betrachter ab und ist unerwünscht. Non-Data-Ink muss auf ein Minimum reduziert werden.	ausreichend
V11	**Lie Factor:** Der Lie Factor muss optimal sein, um die Aussage der Visualisierungen nicht zu verzerren und Fehlinterpretationen zu verhindern.	sehr gut
	Gesamtbewertung	**befriedigend**

Tabelle 5.4.: Bewertung der visuellen Anforderungen bei Proteus.

Fazit

Proteus setzt Methoden der Informationsvisualisierung nicht optimal ein. Die Füllstandanzeigen besitzen keine dauerhaft sichtbare Beschriftung. Fehlende einheitliche Skalierungen verhindern die Vergleichbarkeit dieser Diagramme. Das Tool setzt zwar hauptsächlich kanonische Farben ein, es gibt aber vereinzelt Widersprüche bei der einheitlichen Verwendung. Zahlreiche redundante Piktogramme liefern keinen Mehrwert und sind überflüssig. Der Patchwork-Charakter der Bedienoberfläche wirkt sich nachteilig auf die Übersichtlichkeit aus. Die Durchschnittsbewertung der visuellen Eigenschaften fällt lediglich befriedigend aus.

Proteus speichert sämtliche Informationen über die verwalteten Assets in einer zentralen Datenbank. Dieser Datenbestand ließe sich für Visualisierungen nutzen, die Administratoren sinnvolle Zusatzinformationen bereitstellen: Eine Visualisierung der Veränderung der DHCP Leases im Lauf der Zeit wäre zum Beispiel denkbar. Ebenso vorstellbar ist eine Visualisierung der Entwicklung des Füllstands einzelner Netze. Anhand einer solchen Grafik könnten Trends beobachtet werden und Maßnahmen eingeleitet werden, bevor die verfügbaren Adressen zur Neige gehen.

5.2.3. Spacewalk

Bei *Spacewalk*[10] handelt es sich um eine freie System-Management-Anwendung, die in erster Linie Derivate von Red Hat Enterprise Linux (RHEL)[11] wie Fedora[12], CentOS[13] und Scientific Linux[14] sowie SLES und

[10] Spacewalk-Website: `http://www.spacewalkproject.org/`
[11] Red-Hat-Website: `http://www.redhat.com/de/global/germany`
[12] Fedora-Website: `https://getfedora.org/de/`
[13] CentOS-Website: `https://www.centos.org/`
[14] Scientific-Linux-Website: `https://www.scientificlinux.org/`

Debian unterstützt. Es stellt eine communitybasierte Alternative zum kommerziellen Produkt *Satellite*[15] dar, das von Red Hat vertrieben und supportet wird.

Spacewalk kommt zum Einsatz, um die Aktualität der Paketversionen einer großen Anzahl von Linux-Servern zu überwachen und zu gewährleisten. Die regelmäßige Aktualisierung der Softwarestände ist essentiell, um Sicherheitslücken nach Bekanntwerden zu schließen. Da Spacewalk mehrere Distributionen unterstützt und frei verfügbar ist, leistet es einen wichtigen Beitrag zur Serversicherheit.

Beschreibung

Spacewalk ermöglicht eine Inventarisierung der Computersysteme, die sowohl Hardware- als auch Softwareinformationen umfasst. Das Programm bietet dem Anwender einen tabellarischen Überblick der Softwarestände der überwachten Systeme und ermöglicht ein zentralisiertes Deployment von Paketupdates. Selbst zusammengestellte Softwarepakete werden ebenfalls unterstützt. Zudem können Anwender Konfigurationsdateien auf mehreren Systemen gleichzeitig aktualisieren.

Alternativ zur webbasierten Oberfläche bringt eine Spacewalk-Installation auch Kommandozeilenbefehle mit. Diese lassen sich per Cronjob automatisch ausführen, um beispielsweise regelmäßige Berichte erstellen zu lassen. Informationen über kritische Sicherheitslücken nach Common Vulnerabilities and Exposures (CVE)-Standard können dort ebenfalls einbezogen werden.

Visualisierung

Auf dem Startbildschirm versorgt Spacewalk den Administrator mit sicherheitsrelevanten Statusinformationen. Dazu zählen eine Auflistung der Systeme mit dem ältesten und dem kritischsten Softwarestand, Anwendungen, die aktualisiert werden müssen und die Server, die kürzlich in die Spacewalk-Domäne aufgenommen wurden.

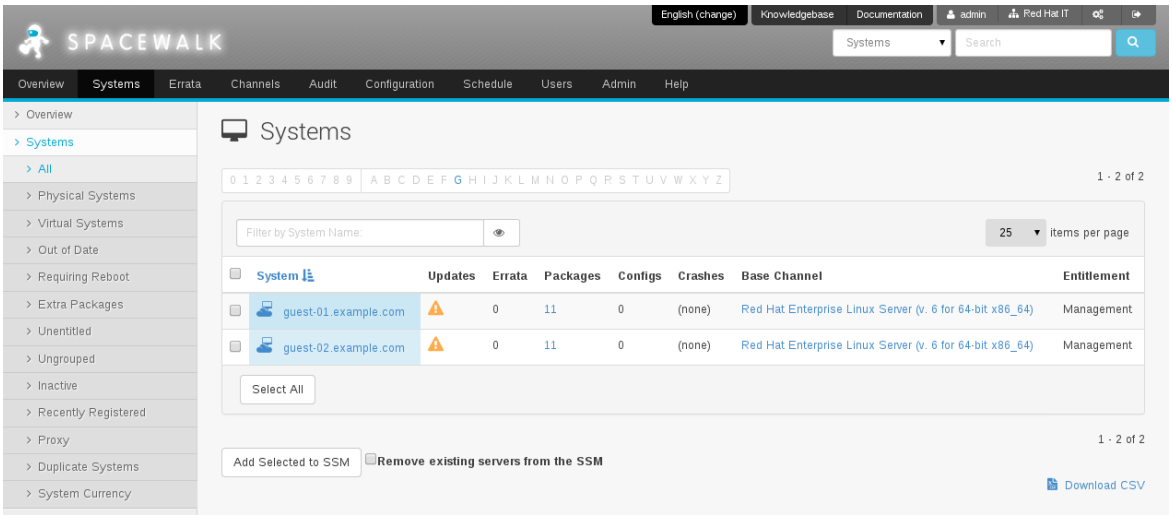


Abbildung 5.11.: Auflistung von Servern mit Update-Bedarf in Spacewalk (Bildausschnitt).[16]

Die Bedienoberfläche der webbasierten Anwendung hinterlässt einen aufgeräumten Eindruck und geht sparsam, aber gezielt mit Farben um. Das User Interface (UI) setzt, wie Ben Shneiderman es vorschlägt (siehe Abschnitt 2.13.3), primär auf Grautöne und verwendet nur einen hellen Blauton als Akzentfarbe. Lediglich

[15] Satellite-Website: `https://access.redhat.com/products/red-hat-satellite`
[16] Bildquelle: `http://www.spacewalkproject.org/img/screenshots/system-overview.png`
[17] Bildquelle: `http://www.spacewalkproject.org/img/screenshots/your_spacewalk.png`

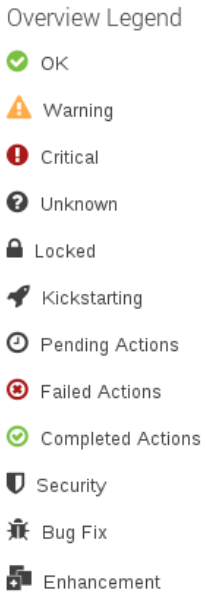


Abbildung 5.12.: Spacewalk nutzt verschiedene Piktogramme zur Kennzeichnung von Ereignissen und Zuständen (Bildausschnitt).[17]

auf weißem Hintergrund ist ihr Kontrast etwas zu schwach (siehe Seitenmenü in Abbildung 5.11). Der Kontrast der Schaltflächentexte im linken Seitenmenü könnte ebenfalls besser ausfallen. Das gilt auch für die Ziffern und Buchstaben unter der Überschrift “Systems” und die Standardtexte in den Textfeldern.

Die meisten Informationen zeigt Spacewalk in Form von tabellarischen Übersichten an. Der Einsatz von Methoden der Informationsvisualisierung beschränkt sich auf ein Minimum, nämlich auf kleine Piktogramme. Diese werden sparsam und an sinnvollen Stellen anstelle von Texten benutzt. Da das Programm auch für Werte, die sich als Grundlage für Visualisierungen eignen würden, keine Diagramme zeichnet, fällt die Bewertung der Visualisierungsformen nur ausreichend aus.

Da abgesehen von Piktogrammen keine anderen Darstellungstypen vorhanden sind, beschränkt sich die Bewertung der Diagrammbeschriftung auf die Legende für die Piktogramme. Sie sagt alles Wissenswerte kurz und knapp aus, lediglich die uneinheitlichen Einrückungen der Texte und Bilder sind für Punktabzug verantwortlich. Die Abwesenheit von Visual Clutter führt zu einer sehr guten Bewertung in dieser Kategorie. Dasselbe gilt für Chartjunk und den Lie Factor.

Da sich die Visualisierungen auf Piktogramme beschränken, wird zur Bewertung der Kategorie “Kontraste in Visualisierungen” der Kontrast zwischen den Piktogrammen und dem Hintergrund herangezogen. Die signalfarbenen Piktogramme sind untereinander sehr gut zu unterscheiden, der Kontrast zwischen den gelben und grünen Symbolen und dem Hintergrund ist lediglich einen Tick zu gering.

In Abbildung 5.11 ist eine Tabelle mit Servern zu sehen, für die Spacewalk ermittelt hat, dass jeweils elf Updates für sie bereitstehen. Wie in der Legende in Abbildung 5.12 zu sehen ist, unterscheidet das Programm unter anderem drei Zustände für den Softwarestand (*OK*, *Warning* und *Critical*). Die hierfür verwendeten Signalfarben Grün, Gelb und Rot sind intuitiv nachvollziehbar und sehr gut unterscheidbar.

Bei einer großen Anzahl verwalteter Server wird die Tabelle sehr groß und verbirgt je nach eingestellter Sortierung und ausgewählter Seite ggf. Server mit kritischem Status. Je nach eingestellter Anzahl der angezeigten Zeilen pro Seite muss der Anwender ggf. mehrere Seiten durchklicken, um sich einen Eindruck über den Gesamtzustand aller Systeme zu verschaffen.

Bewertung der visuellen Anforderungen

Nr.	Kurzbeschreibung	Bewertung
V1	**Übersichtlichkeit:** Die Bedienoberfläche darf nicht überfrachtet sein, da sonst die Übersichtlichkeit darunter leidet.	gut
V2	**Farben:** In der Bedienoberfläche müssen Farben sparsam und zielgerichtet eingesetzt werden.	gut
V3	**Kontrast:** In der Bedienoberfläche müssen die Kontrastverhältnisse hoch sein.	befriedigend
V4	**Objektgröße:** In der Bedienoberfläche müssen Oberflächenelemente eine ausreichende Größe besitzen.	gut
V5	**Visualisierungsformen:** Die Arten der Visualisierungen müssen zu den zugrundeliegenden Daten und Dimensionalitäten passen.	ausreichend
V6	**Beschriftungen:** Visualisierungen müssen mit den notwendigen Beschriftungen versehen sein, damit sie nicht fehlinterpretiert werden können.	gut
V7	**Visual Clutter:** Visualisierungen müssen frei von Visual Clutter sein, um die Übersichtlichkeit zu verbessern.	gut
V8	**Farbskalen:** Farben in Visualisierungen müssen sparsam und zielgerichtet eingesetzt werden. Farbige Datenreihen müssen gut voneinander unterscheidbar sein.	sehr gut
V9	**Kontraste in Visualisierungen:** Kontrastverhältnisse in Visualisierungen müssen hoch sein, um die Lesbarkeit zu optimieren.	gut
V10	**Chartjunk und Non-Data-Ink:** Chartjunk lenkt den Betrachter ab und ist unerwünscht. Non-Data-Ink muss auf ein Minimum reduziert werden.	sehr gut
V11	**Lie Factor:** Der Lie Factor muss optimal sein, um die Aussage der Visualisierungen nicht zu verzerren und Fehlinterpretationen zu verhindern.	sehr gut
	Gesamtbewertung	**gut**

Tabelle 5.5.: Bewertung der visuellen Anforderungen bei Spacewalk.

Fazit

Spacewalk setzt Techniken der Informationsvisualisierung nur sehr sparsam ein und greift hauptsächlich auf Piktogramme zurück, um unterschiedliche Systemzustände zu kennzeichnen. Die Bedienoberfläche hinterlässt aufgrund des sehr sparsamen und zielgerichteten Einsatzes von Farben einen aufgeräumten Eindruck. Das ein oder andere Kontrastproblem wirkt sich aber negativ auf die Übersichtlichkeit aus. Insgesamt ist die Umsetzung der visuellen Anforderungen aber als gut anzusehen.

Kurz vor der Fertigstellung dieser Arbeit hat das Spacewalk-Entwicklerteam eine neue Version des Programms mit einer leicht überarbeiteten Bedienoberfläche veröffentlicht. Es fällt auf, dass der Kontrast der Bedienoberfläche teilweise verbessert wurde. Auch einige Beschriftungen wurden zugunsten der Lesbarkeit vergrößert. Auf der anderen Seite wurde aber unnötiger Eye Candy in Form von Farbverläufen bei den Menüschaltflächen und schattierten Buttons hinzugefügt (siehe Abbildung 5.13). Man sieht an diesem Beispiel, dass bei einigen Entwicklern zwar die Absicht existiert, Bedienoberflächen übersichtlicher zu gestalten, die Optimierungen beruhen aber offensichtlich auf dem eigenen Geschmack und nicht immer auf Grundlagen der Informationsvisualisierung. Aktuelle Bildschirmfotos sind auf der offiziellen Spacewalk-Website zu finden.[18]

Durch die konsequente Umsetzung der Grundlagen der Informationsvisualisierung ließe sich eine Bedienoberfläche mit sehr hoher Qualität erzielen. Beim Prototyp, der Bestandteil der Aufgabenstellung ist, wird großer Wert auf diese Grundlagen gelegt.

Das Programm verfügt eigentlich über genügend Informationen über die Server in der Spacewalk-Domäne, um zusätzliche Informationen in Form von Diagrammen oder anderen Visualisierungen zu vermitteln. Interessant wäre zum Beispiel eine grafische Gegenüberstellung gepatchter und ungepatchter Systeme oder wie viele Pakete darauf insgesamt installiert sind. Dieses Potenzial wird bei Spacewalk verschenkt.

[18] Spacewalk-Website: `http://spacewalk.redhat.com/`
[19] Bildquelle: `http://spacewalk.redhat.com/img/screenshots/system-overview.png`

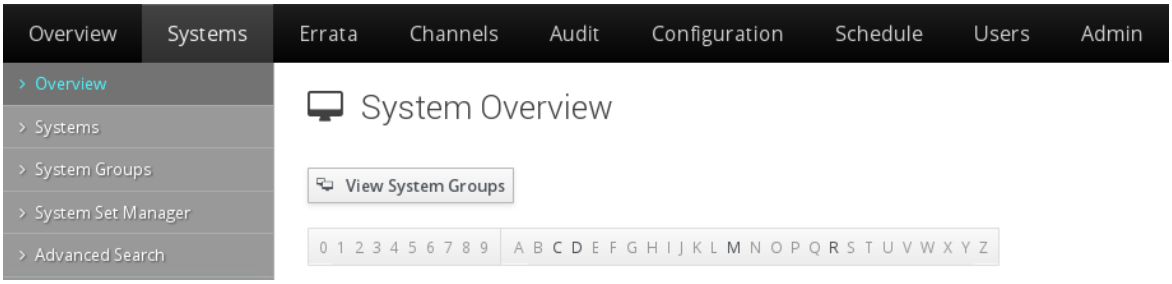


Abbildung 5.13.: Verbesserter Kontrast aber auch zusätzlicher Eye Candy in der neuen Spacewalk-Version (Bildausschnitt).[19]

5.2.4. VMware vSphere Client

Bei zahlreichen Linux-Servern, die in Rechenzentren und anderen Institutionen betrieben werden, handelt es sich um VMs, die auf leistungsfähigen Virtual Machine Hosts ausgeführt werden. Zur Bereitstellung, Verwaltung und Überwachung virtueller Maschinen nutzen Administratoren spezielle Management-Anwendungen, die auf den Hypervisor zugreifen. Ein prominenter Vertreter solcher Anwendungen in VMware-Umgebungen ist der *vSphere Client*, der bereits in Abschnitt 3.4.1 erwähnt wurde.

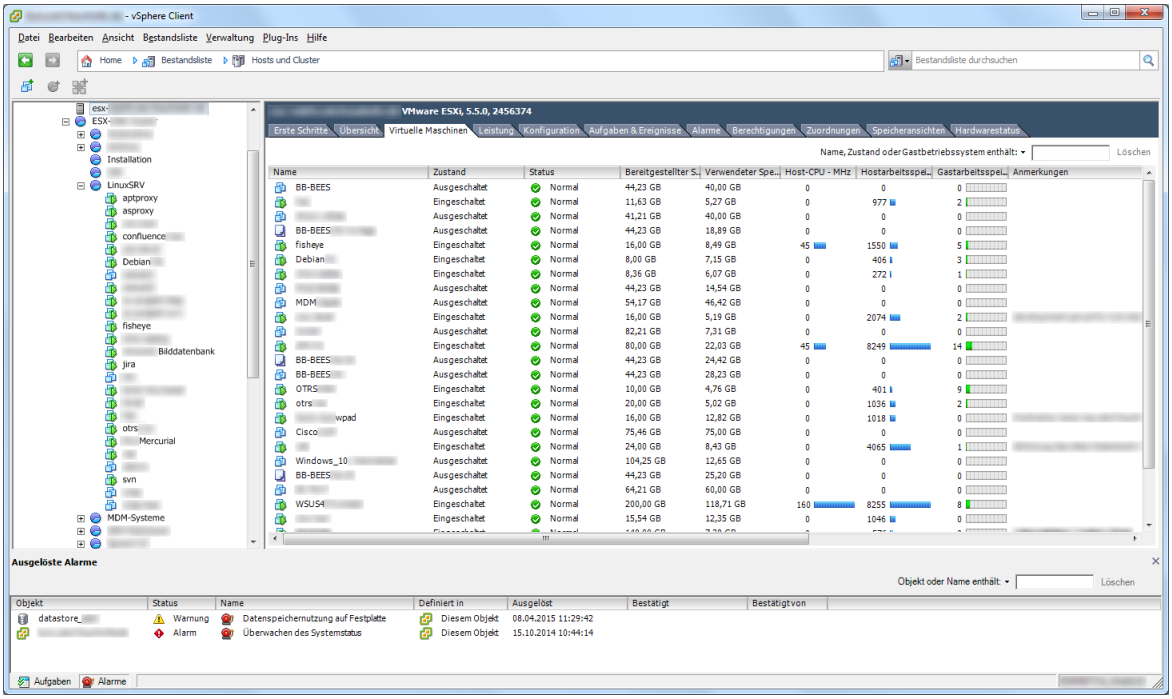


Abbildung 5.14.: Überblick des Zustands und des Ressourcenbedarfs virtueller Maschinen im vSphere Client.

Beschreibung

Der vSphere Client hat unmittelbar etwas mit der Sicherheit der Linux-VMs zu tun, da über ihn die Anbindung der Maschinen an die Rechnernetze oder an andere physikalische Schnittstellen erfolgt. Er ermöglicht über eine virtuelle Konsole die Interaktion mit dem Server, als würde man vor einem dort angeschlossenen Monitor sitzen. Dies dient als Fallbacklösung für den Fall, dass Remote-Desktop- oder Terminal-Sitzungen nicht möglich oder nicht erwünscht sind.

Visualisierung

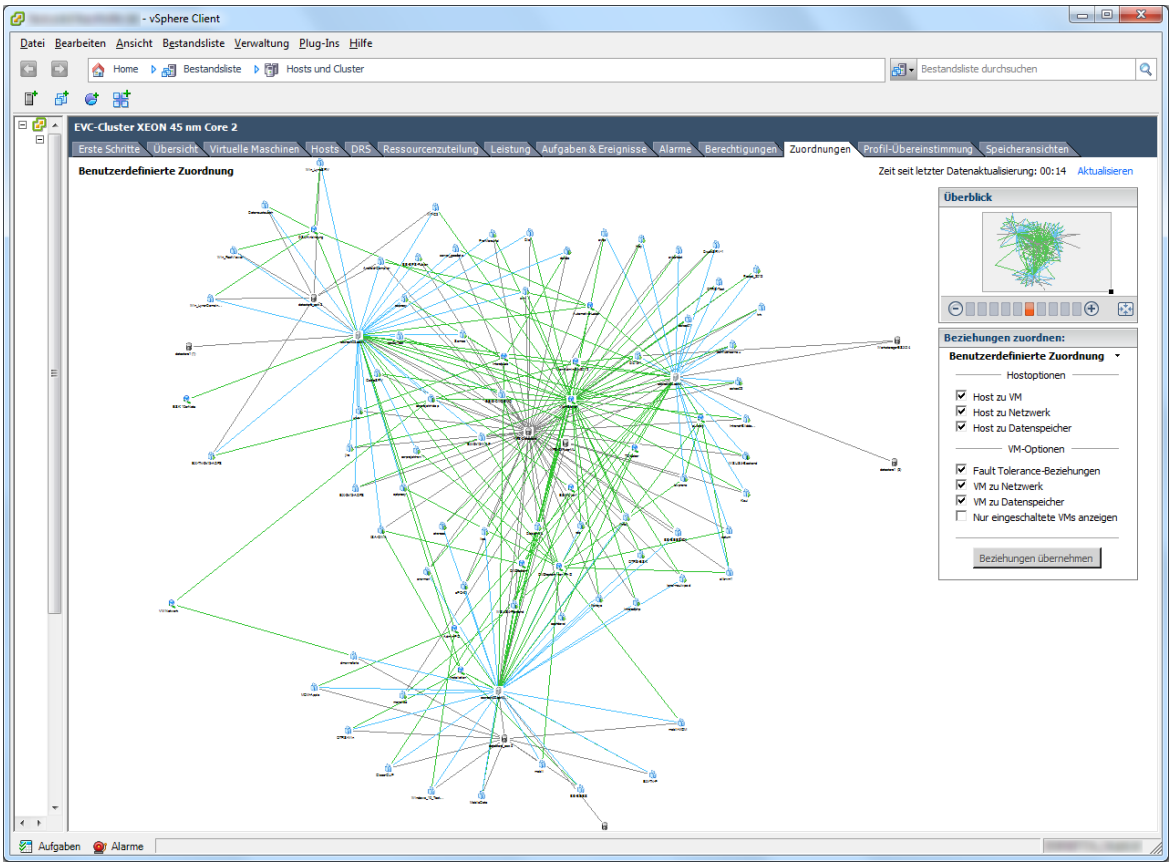


Abbildung 5.15.: Virtuelle Maschinen und ihre Beziehungen zu Ressourcen im vSphere Client.

Der vSphere Client dient nicht nur zur Installation und Inbetriebnahme von virtuellen Servern, sondern liefert auch Zustandsinformationen an den Administrator. Zunächst werden die einzelnen VMs über eine Baumstruktur angezeigt. Auch die Suche nach einzelnen Maschinen anhand von Suchbegriffen ist möglich. Der allgemeine Betriebszustand (eingeschaltet, ausgeschaltet, Wartungsmodus etc.) eines Servers wird durch ein farbiges Piktogramm neben dem Servernamen gekennzeichnet.

Die Bedienoberfläche des vSphere Clients sieht aus wie bei vielen anderen typischen Windows-Anwendungen auch, die auf von Microsoft vorgefertigte Oberflächenelemente zurückgreifen. Aus Sicht der Wiederverwendbarkeit ist das eine lobenswerte Sache, allerdings erkennt man anhand des angewandten Baukastenprinzips auch, dass sich die Entwickler keine großartigen Gedanken gemacht haben, wie man die Bedienoberfläche möglichst bedienfreundlich machen kann. Immerhin ist durch das Standard-Farbschema ein akzeptabler Kontrast gewährleistet.

Das Programm bedient sich einfacher Mittel der Informationsvisualisierung. Die Verhältnisse zwischen den verfügbaren und den benötigten Ressourcen veranschaulicht das Programm durch den Einsatz von Fortschrittsbalken, wie sie zum Beispiel in Abbildung 5.14 zu sehen sind. Die Übersichtlichkeit der Fortschrittsbalken leidet unter Non-Data-Ink. Bei den blauen Fortschrittsbalken, die keine prozentualen Werte anzeigen, fehlt eine einheitliche Endskala, um die einzelnen Werte besser miteinander vergleichen zu können.

Symbolfarben tragen im vSphere Client dazu bei, Warnungen und Alarmierungen besser wahrzunehmen. In der unteren Bildhälfte werden zum Beispiel zwei Benachrichtigungen angezeigt: Die Farbe Gelb warnt den Administrator vor knapp werdendem Festplattenspeicher auf einem angebundenen Storage-System, ein rotes Alarmsymbol weist auf den kritischen Zustand eines anderen Systems hin. Grüne Piktogramme signalisieren ordnungsgemäße Zustände. Die signalfarbenen Piktogramme stellen eine Ergänzung zu Oberflächentexten dar und werden immer zusätzlich neben ihnen eingeblendet.

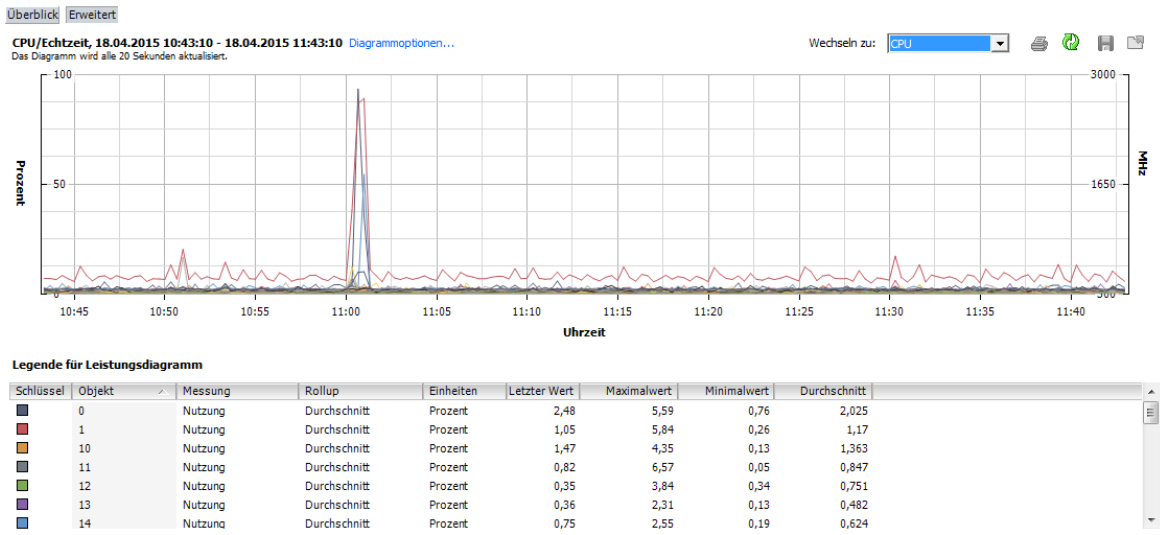


Abbildung 5.16.: Visualisierung der Prozessorauslastungen im vSphere Client.

Da die Piktogramme nur maximal $16x16$ Pixel groß sind und in verschiedenen funktionalen Bereichen der Bedienoberfläche auftauchen können, ist eine erhöhte Aufmerksamkeit gefordert, um nichts Wichtiges zu übersehen. Darüber hinaus zeigt die Anwendung oft sehr viele Informationen gleichzeitig auf dem Bildschirm an. Dies hat zur Folge, dass der Anwender sehr oft scrollen muss, um verdeckte Informationen angezeigt zu bekommen.

Der vSphere Client ist in der Lage, Beziehungen zwischen VMs und anderen Ressourcen untereinander anhand eines Graphen zu visualisieren. Abbildung 5.15 zeigt ein Beispiel für die Visualisierung der Umgebung virtueller Maschinen. Der Anwender hat die Möglichkeit, Beziehungen zwischen Hosts und VMs, Hosts und Netzen sowie Hosts und Datenspeichern einblenden zu lassen. Durch das Overview-and-Detail-Konzept kann sich der Anwender auch trotz hoher Zoomstufe gut in dem komplexen Graphen orientieren. Das ist auch erforderlich, da in der Gesamtübersicht je nach Anzahl der zur Verfügung stehenden Ressourcen viel Visual Clutter auftreten kann. Das Zoomen in die Visualisierung erfolgt nicht stufenlos und wirkt ein wenig abgehackt. Um nicht in die Mitte des Bilds, sondern direkt zu einem Objekt unter dem Mauszeiger zu zoomen, muss der Anwender während er das Mausrad dreht gleichzeitig die linke Maustaste gedrückt halten. Das ist etwas unintuitiv und wird beim Prototyp, der im Rahmen dieser Arbeit entwickelt werden soll, besser gelöst.

Der Anwender ist in der Lage, Echtzeitdaten wie Prozessorlast oder Arbeitsspeichernutzung als Diagramm anzeigen zu lassen. Ein Beispiel hierfür ist in Abbildung 5.16 zu sehen. Da das Programm die Werte kontinuierlich aufzeichnet, ist auch der Zugriff auf historische Daten möglich. Der Inhalt der Diagramme kann an die Bedürfnisse des Anwenders angepasst werden.

Die in Abbildung 5.16 von der Anwendung vorgeschlagenen Farbkodierungen sind etwas ungünstig gewählt, da sie alle etwas dunkel sind. Der Kontrast zwischen den Objekten mit den Ziffern 0, 11 und 14 fällt zu gering aus. Die Anzahl der Hilfslinien im Diagramm ließe sich noch etwas reduzieren. Trotz der etwas ungünstig gewählten Farben ist ein hoher Kontrast zwischen Vordergrund und Hintergrund in den Visualisierungen gewährleistet.

Bei den erfassten Werten handelt es sich hauptsächlich um die Auslastungen bestimmter Komponenten bzw. die Beanspruchung vorhandener Ressourcen auf dem Host. Sie können den Administratoren behilflich sein, um Leistungsengpässe zu erkennen und zu beheben. Aussagen über die internen Zustände der virtualisierten Server wie zum Beispiel ihrer Softwarestände, den Firewall-Konfigurationen oder die Sicherheit ihrer privaten Schlüssel sind anhand dieser Daten bestenfalls indirekt erkennbar. Hierfür sind zusätzliche Administrationstools erforderlich.

Bewertung der visuellen Anforderungen

Nr.	Kurzbeschreibung	Bewertung
V1	**Übersichtlichkeit:** Die Bedienoberfläche darf nicht überfrachtet sein, da sonst die Übersichtlichkeit darunter leidet.	ausreichend
V2	**Farben:** In der Bedienoberfläche müssen Farben sparsam und zielgerichtet eingesetzt werden.	befriedigend
V3	**Kontrast:** In der Bedienoberfläche müssen die Kontrastverhältnisse hoch sein.	gut
V4	**Objektgröße:** In der Bedienoberfläche müssen Oberflächenelemente eine ausreichende Größe besitzen.	befriedigend
V5	**Visualisierungsformen:** Die Arten der Visualisierungen müssen zu den zugrundeliegenden Daten und Dimensionalitäten passen.	gut
V6	**Beschriftungen:** Visualisierungen müssen mit den notwendigen Beschriftungen versehen sein, damit sie nicht fehlinterpretiert werden können.	befriedigend
V7	**Visual Clutter:** Visualisierungen müssen frei von Visual Clutter sein, um die Übersichtlichkeit zu verbessern.	befriedigend
V8	**Farbskalen:** Farben in Visualisierungen müssen sparsam und zielgerichtet eingesetzt werden. Farbige Datenreihen müssen gut voneinander unterscheidbar sein.	befriedigend
V9	**Kontraste in Visualisierungen:** Kontrastverhältnisse in Visualisierungen müssen hoch sein, um die Lesbarkeit zu optimieren.	gut
V10	**Chartjunk und Non-Data-Ink:** Chartjunk lenkt den Betrachter ab und ist unerwünscht. Non-Data-Ink muss auf ein Minimum reduziert werden.	befriedigend
V11	**Lie Factor:** Der Lie Factor muss optimal sein, um die Aussage der Visualisierungen nicht zu verzerren und Fehlinterpretationen zu verhindern.	sehr gut
	Gesamtbewertung	**befriedigend**

Tabelle 5.6.: Bewertung der visuellen Anforderungen beim VMware vSphere Client.

Fazit

Aufgrund der einfallslosen Gestaltung der Bedienoberfläche und einiger Defizite bei der Qualität der Visualisierungen ist der vSphere Client aus visueller Sicht nur Mittelmaß. Oft leidet die Übersichtlichkeit aufgrund überfrachteter Bildschirmseiten. Immerhin gibt es mit der zoombaren Zuordnungs-Ansicht mit Overview-and-Detail fortgeschrittenere Techniken der Informationsvisualisierung in der Anwendung. Insgesamt fällt die Bewertung der visuellen Attribute des vSphere Clients befriedigend aus.

Der vSphere Client ist unverzichtbar, um VMs zu installieren, bereitzustellen und Lastparameter der Hosts zu überwachen. Um die internen Zustände der laufenden VMs zu visualisieren, ist er allerdings nicht geeignet. Hierfür sind zusätzliche Anwendungen erforderlich. Es ist bestenfalls eine Sicht auf die Beziehungen zwischen VMs und anderen Ressourcen untereinander möglich. Die Visualisierungen in dem Programm beschränken sich auf diese Anwendungsbereiche.

5.3. Security-Information- und Event-Management-Systeme

Unter *SIEMs* versteht man Anwendungen, die Echtzeitanalysen sicherheitsrelevanter Ereignisse in einem Rechnernetz ermöglichen. Sie beziehen Daten aus einer Vielzahl unterschiedlicher Quellen wie zum Beispiel Logdateien, filtern und korrelieren diese Daten und weisen Administratoren auf sicherheitsrelevante Ereignisse hin. Durch die Langzeitspeicherung dieser Daten sind Analysen über einen längeren Zeitraum möglich.

SIEMs stellen nützliche Werkzeuge dar, um einerseits einen Beitrag zur Informationssicherheit der überwachten Server zu gewährleisten. Andererseits liefern sie aufgrund der Langzeitspeicherung eine Datengrundlage für Sicherheitsaudits.

Dieser Abschnitt stellt eine repräsentative Auswahl von SIEM-Tools vor, die in Rechnernetzen eingesetzt werden, um die Informationssicherheit sicherzustellen und zu optimieren. Auch diese Klasse von Administrationstools wird anhand ihrer Umsetzung informationsvisualisierender Techniken bewertet.

5.3.1. AlienVault USM

Bei *AlienVault Unified Security Management (USM)* handelt es sich um eine kommerzielle Version der freien SIEM-Anwendung OSSIM. Die Sicherheitssoftware kann als vorkonfigurierte Appliance bezogen werden und steht seit kurzem auch auf der Plattform Amazon Web Services (AWS) als Cloud-basierter Dienst zur Verfügung [ALI 15].

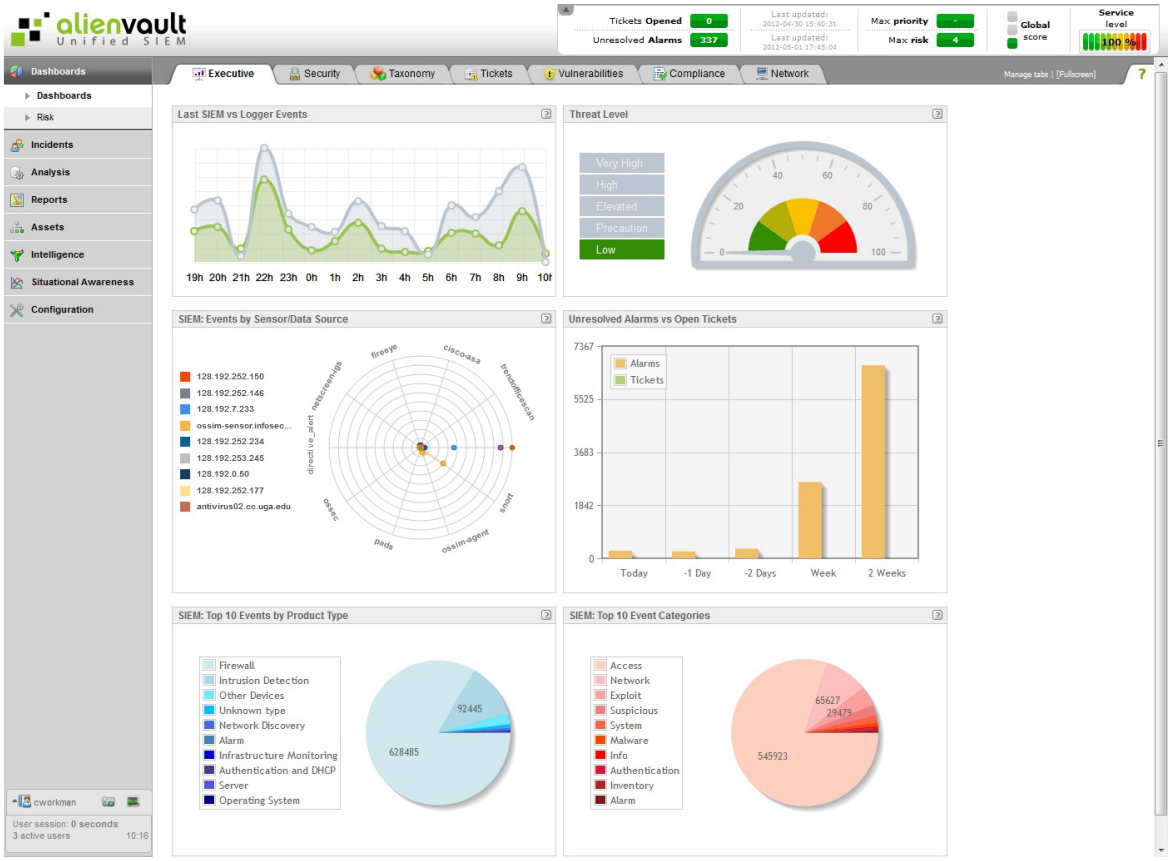


Abbildung 5.17.: Executive-Übersicht von AlienVault 4.0.[20]

Beschreibung

Der Haupteinsatzzweck von AlienVault USM gliedert sich wie folgt auf:

- Bestandsaufnahme vorhandener Assets
- Schwachstellenprüfung
- Bedrohungserkennung
- Verhaltensüberwachung
- Austausch von sicherheitsbezogenen Informationen

[20] Bildquelle: `http://2.bp.blogspot.com/-JUW2V2Z2o20/UR-2q5rW4ZI/AAAAAAAAPQQ/drDkfXfy6zM/s1600/AlienVault.png`

AlienVault USM ist nicht nur zur Überwachung sicherheitskritischer Parameter von Linux-Servern geeignet. Administratoren können die Anwendung auch zum Monitoring des gesamten Firmen- oder Institutsnetzes inklusive der darin enthaltenen Clients und Server verwenden.

Das Tool unterstützt das frei erhältliche Host-based Intrusion Detection System (HIDS) *OSSEC* und kann so Logdaten von Systemen beziehen, auf denen ein OSSEC Client installiert ist. Falls kein Agent für eine bestimmte Plattform zur Verfügung steht, versucht AlienVault USM durch wahlweise aktive oder passive Scans die Sicherheit der Geräte in einem Netz zu prüfen.

AlienVault USM ist in der Lage, Benutzer mit Systemen zu assoziieren und nach dem Aufspüren einer Sicherheitslücke automatisch ein Service Ticket zu erstellen. Außerdem kann das System Sicherheitsinformationen anonymisiert mit einer zentralen Datenbank austauschen. Dies dient einer kontinuierlichen Verbesserung des Produkts.

Visualisierung

Abbildung 5.18.: Executive-Übersicht von AlienVault 4.5.[21]

Die Anwendung greift auf verschiedene Techniken der Informationsvisualisierung zurück. Auf dem Startbildschirm sind diverse Linien-, Kreis-, Säulen- und Netzdiagramme zu sehen, die den Administrator zum Beispiel über die allgemeine Bedrohungslage oder die Anzahl nicht geschlossener Sicherheitslücken im Allgemeinen informiert.

In den Abbildungen 5.17 und 5.18 sind Bildschirmfotos von den beiden Programmversionen 4.0 und 4.5 zu sehen. Beim Vergleich der beiden Bilder fällt auf, dass die Bedienoberfläche der neueren Version von Chartjunk und Non-Data-Ink (siehe Abschnitt 2.7.2) befreit und die Übersichtlichkeit dadurch deutlich verbessert

[21] Bildquelle: `https://goo.gl/NgSQr1`

wurde. Die Entwickler von AlienVault sind scheinbar bestrebt, die Bedienfreundlichkeit durch Umsetzung von Grundlagen der Informationsvisualisierung zu verbessern. Das ist lobenswert.

Um die Aufmerksamkeit des Benutzers auf die sicherheitsrelevanten Visualisierungen zu lenken, wurde der Einsatz von Farben in der gesamten Bedienoberfläche auf ein Minimum reduziert. Neben einer Handvoll Graustufen kommt lediglich ein hellblauer Farbton als Akzentfarbe zum Einsatz. Auf Farbverläufe, die keine Zusatzinformationen beinhalten, wurde zugunsten des klaren Designs auch verzichtet. In der Bedienoberfläche kommen vereinzelt Piktogramme zum Einsatz, die trotz ihrer Farblosigkeit gut voneinander unterschieden werden können.

Die Schatten hinter den Diagrammflächen vermitteln in der alten version keine zusätzlichen Informationen und wurden deshalb entfernt. Die Bedrohungssituation wurde bei der alten Version in der Ansicht “Thread Level” noch mit Hilfe eines Tachometers angezeigt. Die neue Version verzichtet auf diesen Skeuomorphismus (siehe Abschnitt 2.14.2) und optimiert stattdessen das Data-Ink-Verhältnis (siehe Abschnitt 2.7.1).

Die Skala des Bedrohungslage-Diagramms nutzt intuitiv nachvollziehbare Signalfarben als kategorische Skala, damit der Administrator auf einen Blick den Sicherheitszustand der Systeme grob einschätzen kann. Ergänzt wird diese Skala durch eine textuelle Beschriftung und einen numerischen Wert.

In den beiden Kreisdiagramme in der Mitte von Abbildung 5.18 werden zwar nicht in erster Linie Primärfarben, aber dennoch gut voneinander unterscheidbare Farben verwendet. Vermutlich wurde dort bewusst auf die Farben Rot und Grün verzichtet, um eine Verwechslungsgefahr mit der Bedeutung dieser Farben im Bedrohungslage-Diagramm auszuschließen. Hier handelt es sich um einen weiteren Fall, wo sich die Entwickler vorzüglich an Grundlagen der Informationsvisualisierung gehalten haben.

Visual Clutter ist in den Visualisierungen kaum vorhanden. Bei den Diagrammen ließe sich die Anzahl eingeblendeter Hilfslinien noch etwas reduzieren. In den Kategorien Chartjunk, Non-Data-Ink und Lie Factor gibt es Bestnoten.

Aus Sicht der Informationsvisualisierung macht die neue Version von AlienVault USM vieles richtig. Besonders im direkten Vergleich der Screenshots der beiden Versionen in den Abbildungen 5.17 und 5.18 wird sichtbar, dass viel Arbeit in die Optimierung der Bedienoberfläche und der Visualisierungen gesteckt wurde. Die Informationsquellen, die dort angezeigt werden, sind identisch. Das erleichtert den Vergleich der beiden Screenshots. Die Diagramme wurden sichtbar aufgeräumt.

Dennoch gibt es einige verbesserungswürdige Details: Im Gegensatz zu den Legendentexten ist der Kontrast zwischen den Achsbeschriftungen der Diagramme und dem weißen Hintergrund grenzwertig. Bei dem Diagramm oben links in Abbildung 5.18 fehlt die Achsbeschriftung der Ordinate komplett.

Dass es die Oberflächen-Designer auch bei diesem Administrationstool mit dem Minimalismus eine Spur übertrieben haben, spiegelt sich in den grenzwertigen Schriftgrößen wieder. Einige Piktogramm-Schaltflächen, wie sie zum Beispiel in den Kopfzeilen der Visualisierungen anzutreffen sind, sind ebenfalls zu klein geraten und daher schwer anzuklicken. Auf einem Touchscreen stellen diese kleinen Schaltflächen ein noch größeres Problem dar.

Bewertung der visuellen Anforderungen

Nr.	Kurzbeschreibung	Bewertung
V1	**Übersichtlichkeit:** Die Bedienoberfläche darf nicht überfrachtet sein, da sonst die Übersichtlichkeit darunter leidet.	sehr gut
V2	**Farben:** In der Bedienoberfläche müssen Farben sparsam und zielgerichtet eingesetzt werden.	sehr gut
V3	**Kontrast:** In der Bedienoberfläche müssen die Kontrastverhältnisse hoch sein.	sehr gut
V4	**Objektgröße:** In der Bedienoberfläche müssen Oberflächenelemente eine ausreichende Größe besitzen.	ausreichend
V5	**Visualisierungsformen:** Die Arten der Visualisierungen müssen zu den zugrundeliegenden Daten und Dimensionalitäten passen.	gut
V6	**Beschriftungen:** Visualisierungen müssen mit den notwendigen Beschriftungen versehen sein, damit sie nicht fehlinterpretiert werden können.	befriedigend
V7	**Visual Clutter:** Visualisierungen müssen frei von Visual Clutter sein, um die Übersichtlichkeit zu verbessern.	gut
V8	**Farbskalen:** Farben in Visualisierungen müssen sparsam und zielgerichtet eingesetzt werden. Farbige Datenreihen müssen gut voneinander unterscheidbar sein.	sehr gut
V9	**Kontraste in Visualisierungen:** Kontrastverhältnisse in Visualisierungen müssen hoch sein, um die Lesbarkeit zu optimieren.	befriedigend
V10	**Chartjunk und Non-Data-Ink:** Chartjunk lenkt den Betrachter ab und ist unerwünscht. Non-Data-Ink muss auf ein Minimum reduziert werden.	sehr gut
V11	**Lie Factor:** Der Lie Factor muss optimal sein, um die Aussage der Visualisierungen nicht zu verzerren und Fehlinterpretationen zu verhindern.	sehr gut
	Gesamtbewertung	**gut**

Tabelle 5.7.: Bewertung der visuellen Anforderungen bei AlienVault USM.

Fazit

Es ist lobenswert, dass sich die Entwickler von AlienVault größtenteils an die aus Kapitel 2 bekannten Designgrundregeln für Visualisierungen und Bedienoberflächen halten. Besonders beim Vergleich der beiden Programmversionen ist festzustellen, dass sie sich Mühe gegeben haben, um die Oberfläche übersichtlicher zu gestaltet. Der übertriebene Minimalismus führte aber auch zu verbesserungswürdigen Diagrammbeschriftungen und zu Bildschirmelementen, die zu klein sind. Besonders bei Schaltflächen ist das ungünstig, da der Anwender oft daneben klickt. Insgesamt führt die Bewertung der visuellen Anforderungen bei AlienVault zu einer guten Note mit der Tendenz zu sehr gut.

Speziell im Bereich der Linux-Server-Sicherheit gibt es funktionale Defizite: Selbst wenn ein OSSEC Agent auf dem Server installiert ist, beschränken sich die Informationen, die das zentrale Sicherheitssystem erhält, auf den Bereich Intrusion Detection. Das System ist nicht in der Lage, einzelne Softwarestände von den Servern abzufragen, sondern ist auf einen Scan der offenen Ports angewiesen, um anhand gefundener Schwachstellen Rückschlüsse zu veralteter Software zu ermöglichen. Aussagen über die Sicherheit privater Schlüssel oder der TLS-Konfiguration sind auch nicht unmittelbar möglich.

5.3.2. LogRhythm

Bei *LogRhythm* handelt es sich um ein SIEM-System, das File Integrity Monitoring (FIM), Log Management sowie Host- und Netzwerkforensikfunktionen bereitstellt. Der gleichnamige Hersteller bietet sein Produkt sowohl als Hardware Appliance als auch als reine Softwarelösung zur Installation in einer virtualisierten Umgebung an.

[22] Bildquelle: `http://www.comguard.cz/fileadmin/user_upload/reseni/LogRhythm/LGR-gui-dashboard-1200.png`

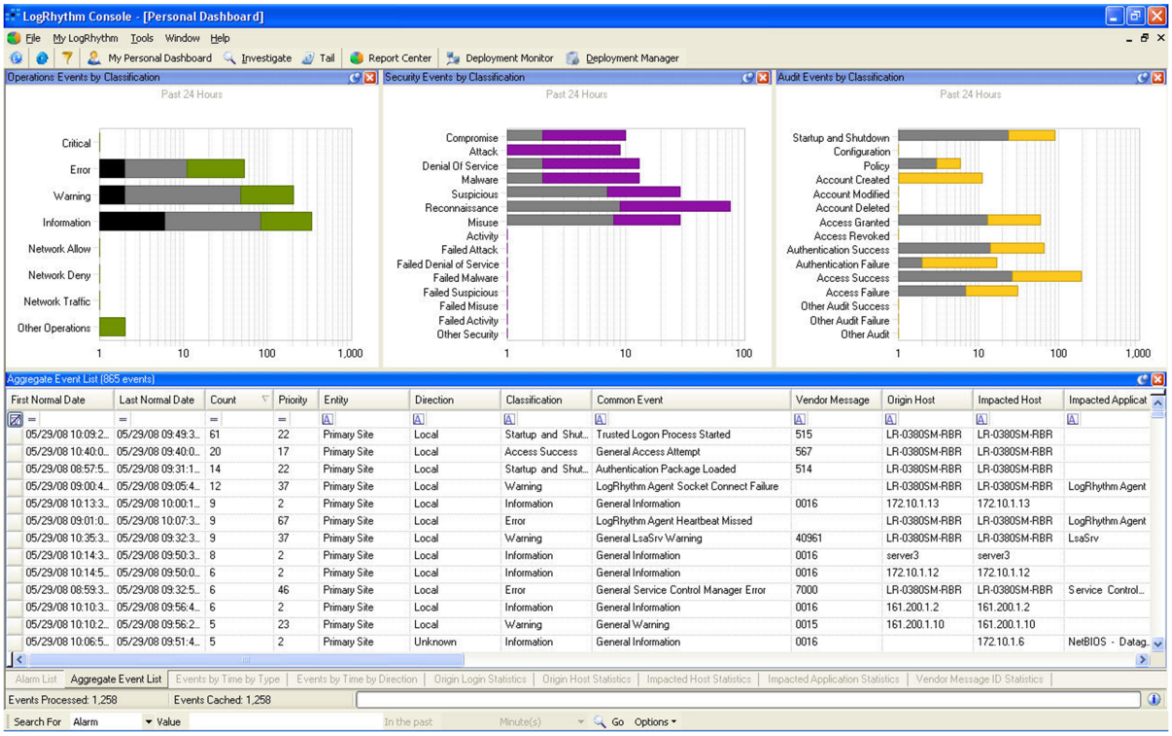


Abbildung 5.19.: Ein benutzerspezifisch zusammengestelltes Dashboard in LogRhythm.[22]

Beschreibung

LogRhythm ermöglicht Bedrohungsanalysen in Echtzeit, indem es auf eine Vielzahl von Datenquellen im ganzen Unternehmen zurückgreift und sie analysiert. Bei der Korrelation von Daten und zur Durchführung verhaltensbasierter Analysen kommt eine sog. *Advanced Intelligence Engine* zum Einsatz. Sie weist den Administrator auf sicherheitsrelevante Anomalien hin, die bei Computern, Programmen oder ihren Anwendern aufgespürt wurden. Indem LogRhythm kontextabhängige Informationen wie etwa Verwundbarkeitsdaten bei der Analyse eingetroffener Logdaten miteinbezieht, kann das Programm den Administrator auch auf potenzielle Bedrohungen hinweisen, die möglicherweise eintreten könnten.

Als primäre Informationsquelle dient dem Administrator bei LogRhythm die sog. *LogRhythm Console*. Im Gegensatz zu vielen webbasierten Konkurrenten handelt es sich hier um einen native Client-Anwendung für Windows. Erst letztes Jahr wurde zwar mit *LogRhythm UX* eine browserbasierte Alternative vorgestellt, sie besitzt aber (noch) nicht den vollen Funktionsumfang der Client-Anwendung.[23]

Visualisierung

Sowohl vorgegebene als auch benutzerdefinierte Zusammenstellungen von Visualisierungen und Tabellen fasst das Programm in sog. *Dashboards* zusammen (siehe Abbildung 5.19). Dort wird Ben Shneidermans Grundregel “detail on demand” (siehe Abschnitt 6.1.2) umgesetzt: Sobald der Anwender doppelt auf ein Diagrammelement klickt, öffnet sich eine Detailansicht mit den zugrundeliegenden Daten.

Der prinzipielle Aufbau der Bedienoberfläche entspricht einer typischen Windows-Anwendung, die mit Hilfe altbekannter Klassenbibliotheken entwickelt wurde. Dazu zählen zum Beispiel eine Menüleiste, ein Schnellzugriffsmenü mit farbigen Piktogrammen, frei platzierbare und skalierbare Fenster und Reiter. Die Entwickler

[23] Zum Zeitpunkt der Produktbewertung war im Internet eine Stellenausschreibung von LogRhythm zu finden, um Webentwickler für die neue Version zu suchen.
[24] Bildquelle: `https://www.youtube.com/watch?v=6KmfxgeIAJo`
[25] Bildquelle: `http://complexdatavisualized.com/wp-content/uploads/2013/02/LogRhythm.png`

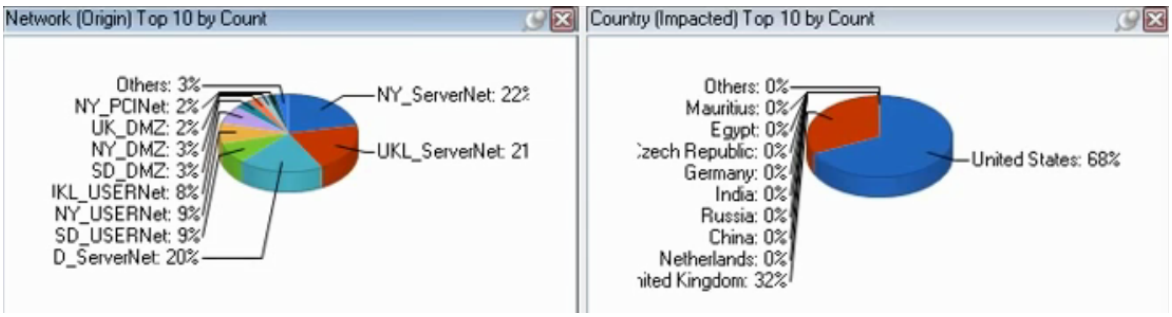


Abbildung 5.20.: Visual Clutter erschwert die Interpretation von Kreisdiagrammen in LogRhythm (Bildausschnitt).[24]

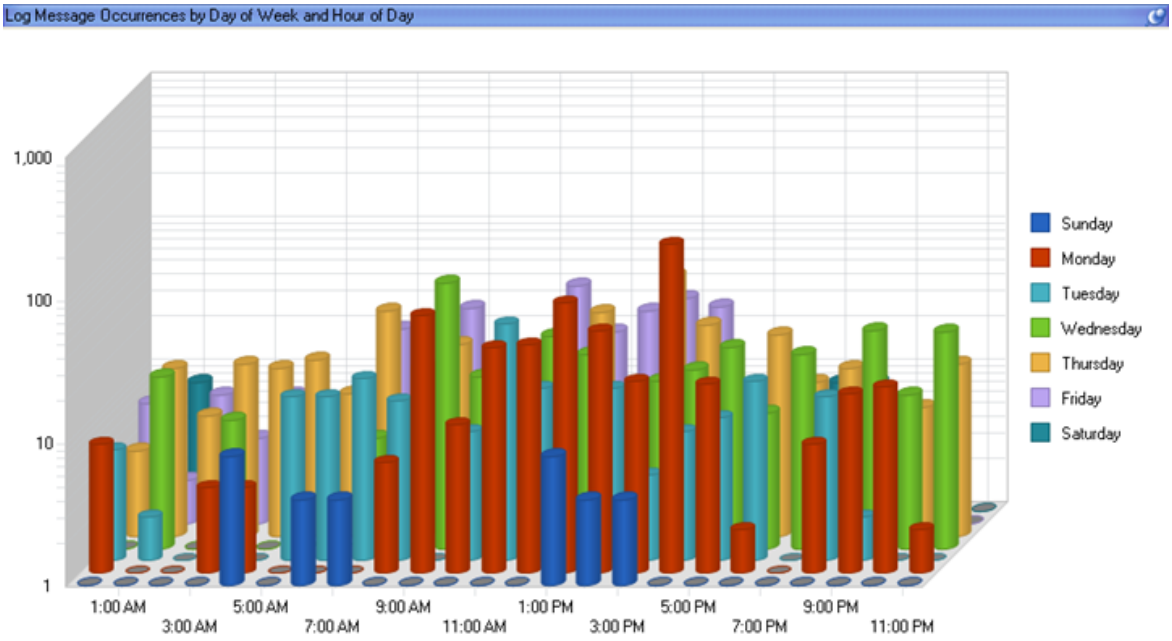


Abbildung 5.21.: Visual Clutter erschwert die Interpretation eines Diagramms in LogRhythm (Bildausschnitt).[25]

der Client-Anwendung haben sich demnach keine besonderen Gedanken gemacht, wie man das Bedienkonzept für den speziellen Anwendungsfall von LogRhythm optimieren kann, sondern größtenteils auf Baukastenkomponenten zurückgegriffen. Immerhin ist der Kontrast, den das Standard-Farbschema liefert, akzeptabel.

Je nach der Zusammenstellung eines Dashboards beinhalten die Tabellen sehr viele Informationen. Dies führt dazu, dass der Anwender viel Scrollen muss, um nichts zu übersehen. Zur Erhöhung der Informationsdichte tragen auch die zahlreichen Reiter im unteren Bildbereich von Abbildung 5.19 bei. Insgesamt wirkt die Bedienoberfläche etwas überfrachtet.

In den in Abbildung 5.19 dargestellten Stapeldiagrammen verwendet das Programm sowohl Graustufen als auch Farben. Es ist nicht nachvollziehbar, weshalb dort nicht vollständig auf Farben gesetzt wird, da sie besser voneinander unterscheidbar sind als es bei Graustufen der Fall ist. Wie bereits in Abschnitt 2.13.2 berichtet wurde, sollte in den Diagrammen komplett auf Schwarz verzichtet werden, da es zu viel Aufmerksamkeit auf sich zieht und die anders eingefärbten Diagrammsegmente übertönt. Eine Legende mit den Bedeutungen der einzelnen Farben fehlt. Der Kontrast zwischen den Diagrammüberschriften und dem weißen Hintergrund ist zu gering.

Bei den gerade erwähnten Diagrammen auf dem Dashboard fällt auch auf, dass dort logarithmische Skalen verwendet werden. Diese stellen prinzipiell ein geeignetes Mittel dar, um sowohl sehr kleine als auch sehr

große Werte gemeinsam in einem Diagramm abbilden zu können. Gleichzeitig erschweren sie aber das Ablesen des genauen Werts. Aus diesem Grund sollten auch die konkreten Werte der einzelnen Diagrammsegmente angezeigt werden, was bei LogRhythm aber nicht der Fall ist.

Negativ fällt zudem auf, dass in den vorgegebenen Dashboards dreidimensionale Diagramme auch dann verwendet werden, wenn zweidimensionale Daten die Grundlage sind. Die Dreidimensionalität liefert hier keine zusätzliche Informationen. Stattdessen werden einige Diagramme wie zum Beispiel Kreisdiagramme mit vielen kleinen Tortenstücken schlecht lesbar. Beispiele hierfür sind in Abbildung 5.20 zu sehen. Die Hilfslinien der Beschriftungen sind dermaßen dicht aneinandergedrängt, dass es sehr mühsam ist, die dazugehörigen Tortensegmente zu identifizieren, auch unter Zuhilfenahme einer Bildschirmlupe.

Einige Diagramme visualisieren zeitabhängige Daten, die sowohl in Stunden als auch in Wochentagen aufgeteilt sind. Bei diesen trivariaten Daten macht eine dreidimensionale Visualisierung zwar prinzipiell Sinn, LogRhythm gelingt es aber nicht, diese Daten frei von Visual Clutter grafisch darzustellen. Durch einfache Interaktionsmöglichkeiten wie das Drehen der Grafik würde der Anwender sie bereits etwas besser betrachten können, was allerdings nicht möglich ist.

Ein Beispiel für eine extrem unübersichtliche Visualisierung ist in Abbildung 5.21 zu sehen. Je fortgeschrittener der Wochentag, desto schwieriger ist es, die Werte der dazugehörigen Datenreihe abzulesen, da die Säulen von den sich davor befindlichen verdeckt werden. Das Ablesen der einzelnen Werte aus dem dreidimensionalen Säulendiagramm wird zusätzlich erschwert, da die senkrechte Bildachse eine logarithmische Skala besitzt. Logarithmische Skalen eignen sich zwar prinzipiell hervorragend, um sowohl sehr große als auch sehr kleine Werte zusammen in einem Diagramm unterzubringen, in der dreidimensionalen Visualisierung, die im Beispielbild zu sehen ist, sind Ablesefehler aber vorprogrammiert: Ein sicheres Ablesen der Werte gelingt nur bei den Säulen in der letzten Reihe gut (sofern die entsprechende Säule nicht verdeckt ist). Bei den vorderen Säulen unterlaufen dem Betrachter unweigerlich sog. *Parallaxenfehler*, da er aufgrund der perspektivischen Darstellung nicht genau einordnen kann, welche Hilfslinie im Hintergrund auf der gleichen Höhe wie die Säule ist. Noch dazu führen bei einer logarithmischen Skala bereits kleine Ablesefehler zu großen Abweichungen vom tatsächlichen Wert. Damit ist dieses Diagramm nahezu unbrauchbar und ermöglicht nur qualitative Aussagen.

Es gibt aber auch Pluspunkte: Trotz ihrer Unübersichtlichkeit sind die Diagramme frei von Chartjunk. Lediglich ein paar Farbverläufe hätte man einsparen können. Die Objektgrößen innerhalb der Anwendung sind ebenfalls akzeptabel.

Abbildung 5.22.: Das neue Webinterface von LogRhythm wirkt deutlich aufgeräumter als der native Client.[26]

Bei dem in Entwicklung befindlichen Webclient LogRhythm UX haben die Entwickler deutlich mehr Arbeit in das Design der Bedienoberfläche und der Übersichtlichkeit der Visualisierungen investiert: Die neu gestaltete Bedienoberfläche kommt mit wenigen Farben und schlichten Formen aus. Durch den schwarzen Bildschirmhintergrund und die leuchtenden Farben der Bedienelemente ist ein hoher Kontrast gewährleistet.

Seinen Vorteil gegenüber einem weißen Hintergrund entfaltet das schwarze Pendant allerdings erst in dunkleren Räumen. Im Gegensatz zu einer großen, leuchtend weißen Fläche müssen sich die Augen nicht ständig an die unterschiedlichen Helligkeiten des Bildschirms und der Umgebung anpassen. Dies ermöglicht ein entspanntes Arbeiten [SCHEU 08]. Abbildung 5.22 zeigt ein Bildschirmfoto von LogRhythm UX.

Bewertung der visuellen Anforderungen

Nr.	Kurzbeschreibung	Bewertung
V1	**Übersichtlichkeit:** Die Bedienoberfläche darf nicht überfrachtet sein, da sonst die Übersichtlichkeit darunter leidet.	ausreichend
V2	**Farben:** In der Bedienoberfläche müssen Farben sparsam und zielgerichtet eingesetzt werden.	befriedigend
V3	**Kontrast:** In der Bedienoberfläche müssen die Kontrastverhältnisse hoch sein.	gut
V4	**Objektgröße:** In der Bedienoberfläche müssen Oberflächenelemente eine ausreichende Größe besitzen.	gut
V5	**Visualisierungsformen:** Die Arten der Visualisierungen müssen zu den zugrundeliegenden Daten und Dimensionalitäten passen.	ausreichend
V6	**Beschriftungen:** Visualisierungen müssen mit den notwendigen Beschriftungen versehen sein, damit sie nicht fehlinterpretiert werden können.	befriedigend
V7	**Visual Clutter:** Visualisierungen müssen frei von Visual Clutter sein, um die Übersichtlichkeit zu verbessern.	mangelhaft
V8	**Farbskalen:** Farben in Visualisierungen müssen sparsam und zielgerichtet eingesetzt werden. Farbige Datenreihen müssen gut voneinander unterscheidbar sein.	ausreichend
V9	**Kontraste in Visualisierungen:** Kontrastverhältnisse in Visualisierungen müssen hoch sein, um die Lesbarkeit zu optimieren.	befriedigend
V10	**Chartjunk und Non-Data-Ink:** Chartjunk lenkt den Betrachter ab und ist unerwünscht. Non-Data-Ink muss auf ein Minimum reduziert werden.	gut
V11	**Lie Factor:** Der Lie Factor muss optimal sein, um die Aussage der Visualisierungen nicht zu verzerren und Fehlinterpretationen zu verhindern.	gut
	Gesamtbewertung	**befriedigend**

Tabelle 5.8.: Bewertung der visuellen Anforderungen bei LogRhythm.

Fazit

Da der Webclient von LogRhythm noch recht neu ist und sich in Entwicklung befindet, findet die Evaluierung des Programms anhand des nativen Windows Clients statt. Die Entwickler des Programms bemühten sich, die Informationsdichte auf dem Bildschirm möglichst hoch zu halten. Die überfrachteten Tabellen und Diagramme sind aber alles andere als übersichtlich und verfehlen ihren Zweck. Aufgrund der ungünstig umgesetzten Visualisierungen können Administratoren wichtige Details aus den Augen verlieren und es drohen Ablesefehler, die zu Fehlentscheidungen führen können. Die Umsetzung der visuellen Anforderungen ist insgesamt nur befriedigend.

Die vom Programm generierten Visualisierungen sind zum Teil so unübersichtlich, dass sie lediglich qualitative Vergleiche zulassen. Darstellungsformen wie räumliche Kreisdiagramme ohne Informationsgewinn durch die dritte Dimension und ungünstiger Anordnung der Beschriftungen wirken sich ebenfalls nachteilig auf die Aussagekraft der Grafiken aus. Immerhin wird anhand des noch jungen Webclients deutlich, dass bei den Entwicklern im Lauf der Zeit ein Bewusstsein für elementare Designgrundregeln Einzug gehalten hat.

[26] Bildquelle: `http://www.interdata.fr/files/uploads/image/LogRhythm-Web%20Console-Interdata.png`

5.3.3. McAfee Enterprise Security Manager

Der von Intel im Jahr 2010 aufgekaufte IT-Security-Spezialist McAfee [TOP 10] bietet mit dem *Enterprise Security Manager (ESM)*[27] eine kommerzielle SIEM-Lösung für Unternehmen an.

Beschreibung

Als Hauptmerkmale des Enterprise Security Managers nennt McAfee die folgenden Punkte:

- Visualisierung von Echtzeit- und historischen Daten mit dem Ziel einer umfassenden Erkennung von und Reaktion auf Bedrohungen.
- Bereitstellung priorisierter gerichtsverwertbarer Informationen innerhalb von Minuten für eine schnelle Reaktion auf Bedrohungen.
- Gewinnung sicherheitsrelevanter Informationen durch fortschrittliche Analysefunktionen.
- Compliance Framework für mehr als 240 internationale Regelungen.

Visualisierung

Abbildung 5.23 zeigt eine Ansicht im ESM, in der die Anzahl der Quell- und Ziel-IP-Adressen von Verbindungen über einen bestimmten Zeitraum visualisiert werden. Ungefähr die Hälfte der Bildschirmfläche wird von Visualisierungen eingenommen, die Tachometer darstellen sollen. Sie werden verwendet, um numerische Werte zu visualisieren. In allen Fällen handelt es sich um die Anzahl bestimmter IP-Adressen.

Die Tachometer verfügen über eine beschriftete Skala, die jeweils in einen blauen und einen roten Bereich unterteilt ist. Der rote Bereich kennzeichnet offensichtlich bedenklich große Werte. Zusätzlich zu der Skala besitzt jeder Tachometer ein Label, das den exakten aktuellen Wert anzeigt.

Die geographische Karte in der Bildschirmmitte zeigt die Standorte von Systemen an, die verdächtige und nachweislich schädliche Datenpakete absendeten. Eine Legende mit den Bedeutungen der beiden Farben Rot und Blau ist nicht vorhanden. Es kann nur angenommen werden, dass verdächtige Absender blau und boshafte Absender rot markiert werden.

Die beiden zur Standortmarkierung verwendeten Farben haben scheinbar nichts mit den Tachometer-Anzeigen zu tun, da sich dort alle Zeiger im blauen Bereich befinden. Die Bedeutung der Farben ist in beiden Darstellungen also unterschiedlich. Dies verwirrt den Betrachter und schafft Raum für Fehlinterpretationen.

Bei den Tachometern handelt es sich um skeuomorphische Darstellungen (siehe Abschnitt 2.14.2). Die breiten Zeiger würden in einem physikalischen Anzeigegerät zwar zur mechanischen Stabilität beitragen, in einer computergenerierten Visualisierung bieten sie aber keinen Mehrwert. Stattdessen schaffen sie Visual Clutter (siehe Abschnitt 2.12.4) und erschweren das Ablesen der dahinter liegenden Skala. Dies wird besonders bei den Tachometern rechts unten deutlich. Vermutlich wurde deshalb auch das zusätzliche Label mit dem aktuellen Wert eingeführt.

[27] ESM-Website: `http://www.mcafee.com/us/products/enterprise-security-manager.aspx`

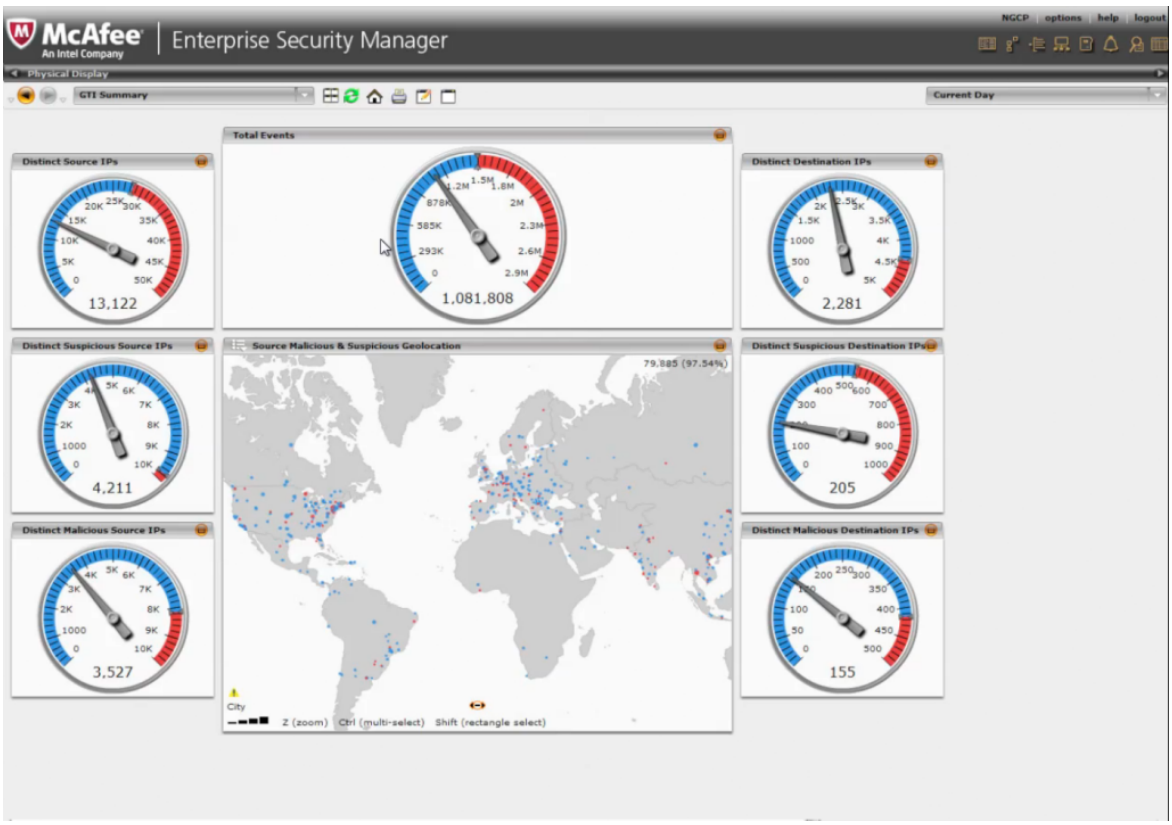


Abbildung 5.23.: Visualisierung der Anzahl von Verbindungen im McAfee Enterprise Security Manager.[28]

Abbildung 5.24.: Ereignis-Dashboard im McAfee Enterprise Security Manager.[29]

[28] Bildquelle: http://complexdatavisualized.com/wp-content/uploads/2013/02/McAfee-NitroSecurity-2-1024x716.png

Eine Vergleichbarkeit der Tachometerwerte untereinander ist aufgrund unterschiedlicher Endskalen nicht ohne weiteres möglich. Durch die Einblendung der aktuellen Kennzahl und die fehlende Vergleichbarkeit der Tachometer werden diese platzraubenden Visualisierungen im Grunde genommen überflüssig. Die Überschreitung eines Schwellwerts könnte man im einfachsten Fall auch durch die Rotfärbung der Zahl verdeutlichen. Den besten Kompromiss zwischen Informationsgehalt und Platzbedarf würden anstelle der Tachometer Fortschrittsbalken liefern. Mindestens die Hälfte der von den Tachometern benötigten Bildfläche ließe sich so einsparen, um mehr Platz für die Weltkarte zu erhalten. Dort wäre der gewonnene Platz sehr von Vorteil, denn der Anwender müsste dann auch weniger Pan- und Zoom-Operationen durchführen und käme schneller zum Ziel.

Abbildung 5.24 zeigt einen Ausschnitt vom Ereignis-Dashboard des ESMs. Die Einfallslosigkeit der Oberflächen-Designer bezüglich der Wahl geeigneter Visualisierungsformen setzt sich auch in dieser Ansicht fort: Für die gesamte Anzahl verzeichneter Ereignisse und die Anzahl vom System korrelierter Ereignisse setzen sie dort ebenfalls auf Tachometer. Hinzu kommt bei ihnen im Gegensatz zu den zuvor genannten Pendants eine rote Hintergrundfarbe. Sie soll vermutlich die besondere Wichtigkeit dieser beiden Anzeigen betonen. Der Kontrast zum roten Bereich der Skala verringert sich dadurch allerdings. Der rote Bereich der Skala geht im gleichfarbigen Hintergrund unter. Dem Betrachter fällt es so nicht mehr schnell genug auf, wenn sich einer der Zeiger im roten Bereich befindet.

Auf eine flächendeckende Verwendung der Farbe Rot sollte generell verzichtet werden, da sonst tatsächlich wichtige Hervorhebungen mit dieser Farbe im Gesamtbild untergehen [MOHR 11, S. 104 f.]. Abgesehen davon sind die Tachos auch in dieser Übersicht aus den bereits zuvor genannten Gründen ein ungeeignetes Mittel zur Visualisierung der zugrundeliegenden Informationen und nehmen unnötigen Platz weg.

In Abbildung 5.24 sind zur Abwechslung auch Balkendiagramme zu sehen. Als positiv hervorzuheben ist dort der Einsatz intuitiv nachvollziehbarer Signalfarben bzw. gut unterscheidbarer Primärfarben. Auf eine Skala wird dort zwar verzichtet, die absoluten Werte sind zur besseren Vergleichbarkeit aber als Beschriftung angegeben. Immerhin ist der Grad der Beschriftungen dort akzeptabel.

Die Bedienoberfläche an sich setzt als Designelemente zahlreiche Schatten und Farbverläufe ein. Zugunsten der Übersichtlichkeit und um den Benutzer nicht vom Wesentlichen abzulenken sollte darauf allerdings verzichtet werden. Der Kontrast zwischen den Hyperlinks ganz oben rechts und dem Farbverlauf im Hintergrund ist suboptimal.

Die Anwendung setzt Piktogramme nicht nur in unterschiedlicher Größe ein, sondern variiert auch zwischen ein- und mehrfarbigen Symbolen. Eine Vereinheitlichung von Größe und Farben der Piktogramme würde einen zusätzlichen Beitrag zur Übersichtlichkeit leisten. Die Farbgebung einiger Piktogramme ist auch nicht nachvollziehbar. So wird beispielsweise die Signalfarbe Grün für eine Reload-Schaltfläche verwendet, und zwar mit dem gleichen Farbton, der auch in den Visualisierungen vorkommt. Dies irritiert den Anwender, da dort kein direkter Zusammenhang besteht. Es ist auch nicht nachvollziehbar, weshalb die Buttons zum Vergrößern der Fenster gelb eingefärbt sind, ebenso wie die Zurück-Schaltfläche. Wer auf einen Vergrößern-Button klickt, will sich bewusst eine Sache genauer ansehen. Dies entspricht genau dem Gegenteil des Verwendungszweck einer Zurück-Schaltfläche.

Einige Piktogramme besitzen Schatten und andere wiederum nicht. Die Richtung der Schatten ist innerhalb der Bedienoberfläche nicht einheitlich. Dies trifft sogar auf Symbole zu, die sich unmittelbar nebeneinander befinden: Der Schatten der Home-Schaltfläche zeigt nach unten rechts, während die Schatten der benachbarten Symbole nach unten links zeigt. Die Schatten der gelben Buttons in der Titelleiste zeigen wiederum nach unten rechts, die Fensterschatten zur Abwechslung weder nach links noch nach rechts sondern nur nach unten. Die uneinheitlichen Schatten und Farben, die sich durch die gesamte Bedienoberfläche ziehen, hinterlassen einen unangenehmen Patchwork-Charakter. Insgesamt führt dies zu einer mangelhaften Bewertung bei der Farbwahl.

[29] Bildquelle: `https://blogs.mcafee.com/wp-content/uploads/ESM-GOZ-1.png`

Bewertung der visuellen Anforderungen

Nr.	Kurzbeschreibung	Bewertung
V1	**Übersichtlichkeit:** Die Bedienoberfläche darf nicht überfrachtet sein, da sonst die Übersichtlichkeit darunter leidet.	ausreichend
V2	**Farben:** In der Bedienoberfläche müssen Farben sparsam und zielgerichtet eingesetzt werden.	mangelhaft
V3	**Kontrast:** In der Bedienoberfläche müssen die Kontrastverhältnisse hoch sein.	befriedigend
V4	**Objektgröße:** In der Bedienoberfläche müssen Oberflächenelemente eine ausreichende Größe besitzen.	befriedigend
V5	**Visualisierungsformen:** Die Arten der Visualisierungen müssen zu den zugrundeliegenden Daten und Dimensionalitäten passen.	mangelhaft
V6	**Beschriftungen:** Visualisierungen müssen mit den notwendigen Beschriftungen versehen sein, damit sie nicht fehlinterpretiert werden können.	gut
V7	**Visual Clutter:** Visualisierungen müssen frei von Visual Clutter sein, um die Übersichtlichkeit zu verbessern.	befriedigend
V8	**Farbskalen:** Farben in Visualisierungen müssen sparsam und zielgerichtet eingesetzt werden. Farbige Datenreihen müssen gut voneinander unterscheidbar sein.	ausreichend
V9	**Kontraste in Visualisierungen:** Kontrastverhältnisse in Visualisierungen müssen hoch sein, um die Lesbarkeit zu optimieren.	ausreichend
V10	**Chartjunk und Non-Data-Ink:** Chartjunk lenkt den Betrachter ab und ist unerwünscht. Non-Data-Ink muss auf ein Minimum reduziert werden.	ausreichend
V11	**Lie Factor:** Der Lie Factor muss optimal sein, um die Aussage der Visualisierungen nicht zu verzerren und Fehlinterpretationen zu verhindern.	gut
	Gesamtbewertung	**ausreichend**

Tabelle 5.9.: Bewertung der visuellen Anforderungen beim McAfee Enterprise Security Manager.

Fazit

Bei der Auswahl der Darstellungstechniken in McAfees ESM spielten Grundlagen der Informationsvisualisierung offensichtlich nur eine untergeordnete Rolle. Das Programm hinterlässt aufgrund der unpassenden skeuomorphischen Darstellungen beim Anwender den Eindruck, als hätte sich entweder ein Autonarr oder ein Kontrollfetischist beim Oberflächendesign durchgesetzt. Anders ist die inflationäre Verwendung von Tachometern kaum nachvollziehbar.[30] Durch geeignetere Techniken der Informationsvisualisierung wäre die Aussagekraft der Grafiken in dieser Anwendung höher und das Produkt könnte von Administratoren vermutlich effizienter eingesetzt werden.

Ein weiterer Fehlgriff ist die Bedienoberfläche an sich: Sie gehört dringend aufgeräumt und vereinheitlicht. Die Art der Farbgebung ist uneinheitlich und überhaupt nicht nachvollziehbar. Noch dazu haben Farben, die in den Visualisierungen verwendet werden, in der Bedienoberfläche selbst nichts zu suchen, da dies zu Verwechslungen führt. Die Gesamtnote bei der Bewertung der visuellen Anforderungen ist beim McAfee ESM lediglich ausreichend.

5.4. Logdatenanalyse-Tools

Je mehr Server in einem Rechenzentrum betrieben werden und je stärker ihre Dienste in Anspruch genommen werden, desto mehr Informationen werden in Logdateien geschrieben. Logdatenanalyse-Tools helfen Admi-

[30] Es wahr vermutlich ein Kontrollfetischist für die Gestaltung der Oberfläche verantwortlich. Ein Autonarr hätte an seiner Stelle als Hintergrund für die Titelleiste keinen simplen Gradienten, sondern eine schicke Karbon-Textur verwendet. Bei der Weboberfläche des Linux-basierten NAS-Betriebssystems *OpenMediaVault* ist dies beispielsweise der Fall (`http://www.openmediavault.org/`).

nistratoren dabei, trotz der stetig wachsenden Informationsmenge den Überblick zu behalten. Sie durchsuchen Logdateien nach bestimmten Mustern, um Störfälle zu erkennen und den Administrator darauf hinzuweisen.

Da Logdatenanalysetools einen wichtigen Beitrag zum reibungslosen Serverbetrieb und zur Informationssicherheit leisten, sind sie auch für das LRZ von Bedeutung. Dort wird auch das unter anderem bewertete Tool *Splunk* verwendet. Dieser Abschnitt widmet sich einer repräsentativen Auswahl von Administrationstools aus dieser Kategorie. Diese Programme werden anhand der Umsetzung informationsvisualisierender Techniken bewertet.

5.4.1. AWStats

AWStats[31] ist eine quelloffene Analyse-Software für Webseitenzugriffe. Das Programm dient zur Visualisierung von Besucherzugriffen auf einen Webserver. Im Gegensatz zur häufig genannten Alternative *Webalizer*[32] verfügt AWStats über einen größeren Funktionsumfang. Darüber hinaus wurde Webalizer im Gegensatz zu AWStats schon seit fast zwei Jahren nicht mehr weiterentwickelt.[33]

Beschreibung

Als Datenquelle greift AWStats auf Logdateien des Webserver-Dienstes zurück. Aus den dort extrahierten Daten generiert das Programm webbasierte Übersichtsseiten, die anhand verschiedener Kriterien aufgegliedert sind. Das Programm beantwortet die folgenden Fragen anhand von Tabellen und einfachen Grafiken:

- Wann fanden Zugriffe statt?
- Wer hat Webseiten aufgerufen (Länder, Rechner, beglaubigte Benutzer, Bots)?
- Wie lang war die Aufenthaltsdauer auf einzelnen Webseiten?
- Welche Dateitypen wurden heruntergeladen?
- Welche Client Software wurde benutzt (Betriebssystem, Browser)?
- Von welchen Webseiten kamen die Besucher?
- Welche Suchausdrücke und Suchbegriffe wurden verwendet?
- Wie effizient wurden übertragene Dateien vom Webserver komprimiert?

Visualisierung

Neben detaillierten tabellarischen Übersichten setzt AWStats zur Visualisierung der Zahlenwerte hauptsächlich auf Balken- und Säulendiagramme. Abbildung 5.25 zeigt einen Bildausschnitt, wo dies der Fall ist. Die Diagramme haben ergänzenden Charakter, da sie keine Informationen anzeigen, die nicht bereits aus den Tabellen hervorgehen.

Charakteristisch für sämtliche Visualisierungen in AWStats ist, dass sie keine beschrifteten Achsen besitzen. Dies wirkt sich sehr nachteilig auf die Interpretierbarkeit der Infografiken aus, da sie auf diese Weise nur qualitative Vergleiche zulassen. Die Farbverläufe in den Diagrammen sind unnötig, da sie keine zusätzlichen Informationen kodieren.

Die einzelnen Diagrammteile verfügen zwar über einblendbare Tooltip-Texte, Anwender von Touchscreens können diese technisch bedingt aber nicht anzeigen. Außerdem ist es umständlich, für eine große Anzahl von Säulen oder Balken nacheinander die Maus darauf zu bewegen und jede Mal auf die Einblendung des

[31] AWStats-Website: `http://www.awstats.org/`
[32] Webalizer-Website: `http://www.webalizer.org/`
[33] Quelle: `http://www.webalizer.org/news.html`
[34] Bildquelle: `http://www.nltechno.com/awstats/awstats.pl?config=destailleur.fr` (abgerufen am 27.04.2015)

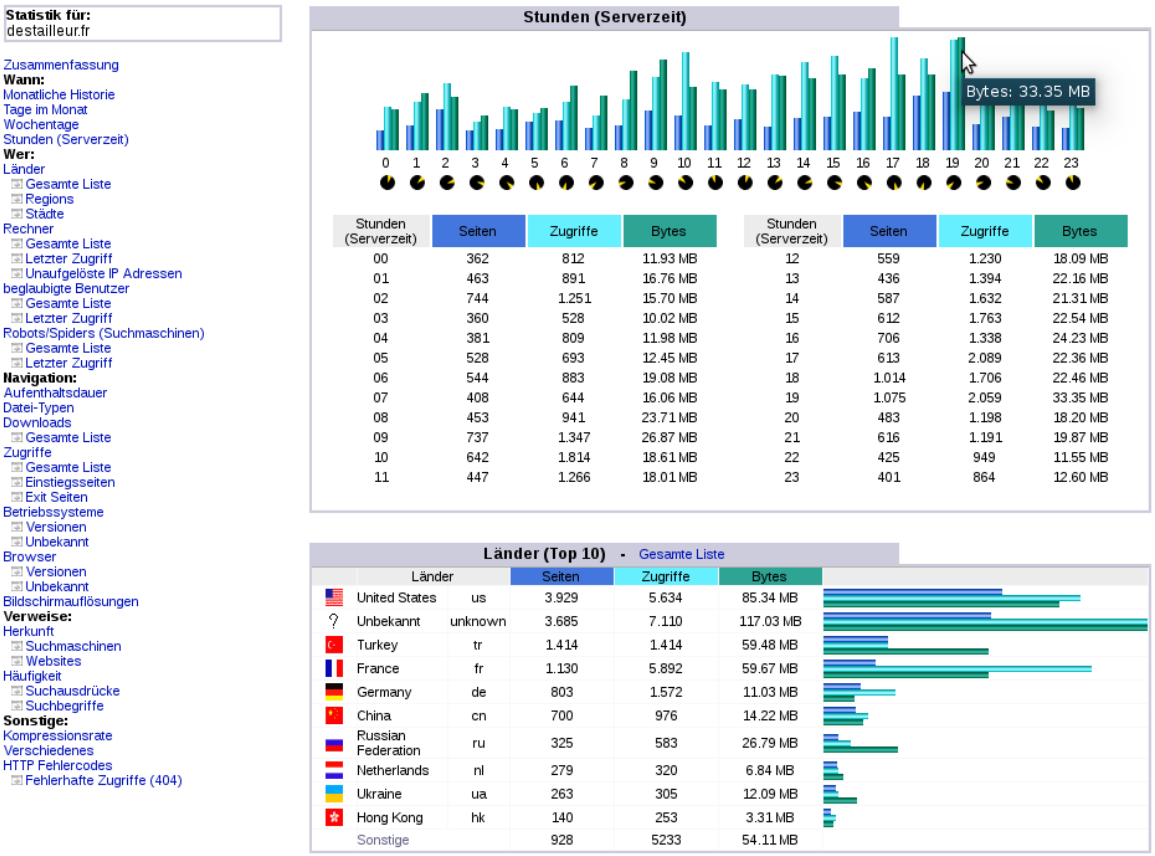


Abbildung 5.25.: Server-Statistiken in Stunden gruppiert und die Top 10 der zugreifenden Länder in AWStats.[34]

entsprechenden Textes zu warten. Die geringe Breite der Diagramme erschwert eine genaue Positionierung des Mauszeigers.

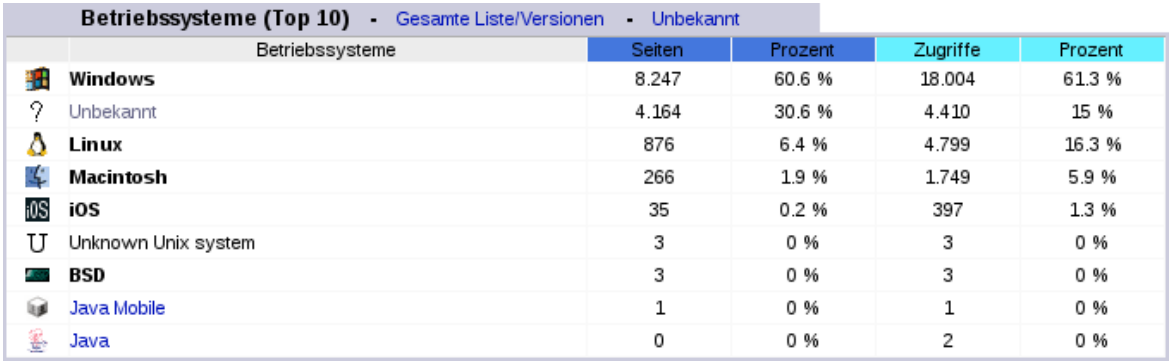
Der Farbkontrast zwischen den jeweils gruppierten Balken und Säulen ist in Abbildung 5.26 etwas zu gering. Anstelle von Hellblau wäre eine andere Primärfarbe besser geeignet. In einigen Ansichten werden Balkendiagramme tabellarisch untereinander angeordnet. Aufgrund der schlecht unterscheidbaren Farben und der räumlichen Nähe sind die einzelnen Balken für den Betrachter schlecht auseinanderzuhalten. Allerdings ist das Data-Ink-Verhältnis aufgrund der schmalen Balken ausgezeichnet. Bei den Tabellenüberschriften unter dem Stunden-Diagramm und in der Länder-Tabelle leidet die Lesbarkeit teilweise unter dem grenzwertigen Kontrast.

Neben den Balken- und Säulendiagrammen setzt AWStats auch kleine Piktogramme ein, die die Lesbarkeit der Tabellen verbessern. So gibt es beispielsweise kleine Länderflaggen in länderbezogenen Statistiken zu sehen (siehe Abbildung 5.25). In Abbildung 5.26 werden Piktogramme zur Unterscheidung von Betriebssystemen eingesetzt. Da den Anwendern die meisten der Symbole bekannt vorkommen dürften, finden sie so schneller zu den für sie interessanten Datensätzen. Der geringe Kontrast zwischen dem schwarzen Text und dem blauen Hintergrund wirkt sich auch hier nachteilig auf die Lesbarkeit der Spaltenüberschriften aus.

Die Bedienoberfläche an sich hinterlässt zwar keinen modernen, aber einen aufgeräumten Eindruck. Die Schriftgröße der Oberflächentexte ist grenzwertig. Auf große Farbflächen, die den Benutzer ablenken könnten, wird zugunsten der Übersichtlichkeit größtenteils verzichtet.

Bei genauer Betrachtung des Bildschirmausschnitts in Abbildung 5.26 fällt ein Detail auf, das für Verwirrung

[35] Bildquelle: `http://www.nltechno.com/awstats/awstats.pl?config=destailleur.fr` (abgerufen am 27.04.2015)



Betriebssysteme (Top 10) - Gesamte Liste/Versionen - Unbekannt

Betriebssysteme	Seiten	Prozent	Zugriffe	Prozent
Windows	8.247	60.6 %	18.004	61.3 %
Unbekannt	4.164	30.6 %	4.410	15 %
Linux	876	6.4 %	4.799	16.3 %
Macintosh	266	1.9 %	1.749	5.9 %
iOS	35	0.2 %	397	1.3 %
Unknown Unix system	3	0 %	3	0 %
BSD	3	0 %	3	0 %
Java Mobile	1	0 %	1	0 %
Java	0	0 %	2	0 %

Abbildung 5.26.: Verwendung von Piktogrammen zur Unterscheidung von Betriebssystemen und Browsern in AWStats.[35]

sorgen kann: In der deutschen Lokalisierung der Bedienoberfläche werden Punkte sowohl als Tausendertrennzeichen als auch als Dezimaltrennzeichen bei den Prozentangaben eingesetzt. Als Dezimaltrennzeichen ist allerdings ein Komma in Deutschland gebräuchlich.

Bewertung der visuellen Anforderungen

Nr.	Kurzbeschreibung	Bewertung
V1	**Übersichtlichkeit:** Die Bedienoberfläche darf nicht überfrachtet sein, da sonst die Übersichtlichkeit darunter leidet.	gut
V2	**Farben:** In der Bedienoberfläche müssen Farben sparsam und zielgerichtet eingesetzt werden.	gut
V3	**Kontrast:** In der Bedienoberfläche müssen die Kontrastverhältnisse hoch sein.	sehr gut
V4	**Objektgröße:** In der Bedienoberfläche müssen Oberflächenelemente eine ausreichende Größe besitzen.	befriedigend
V5	**Visualisierungsformen:** Die Arten der Visualisierungen müssen zu den zugrundeliegenden Daten und Dimensionalitäten passen.	befriedigend
V6	**Beschriftungen:** Visualisierungen müssen mit den notwendigen Beschriftungen versehen sein, damit sie nicht fehlinterpretiert werden können.	mangelhaft
V7	**Visual Clutter:** Visualisierungen müssen frei von Visual Clutter sein, um die Übersichtlichkeit zu verbessern.	gut
V8	**Farbskalen:** Farben in Visualisierungen müssen sparsam und zielgerichtet eingesetzt werden. Farbige Datenreihen müssen gut voneinander unterscheidbar sein.	ausreichend
V9	**Kontraste in Visualisierungen:** Kontrastverhältnisse in Visualisierungen müssen hoch sein, um die Lesbarkeit zu optimieren.	ausreichend
V10	**Chartjunk und Non-Data-Ink:** Chartjunk lenkt den Betrachter ab und ist unerwünscht. Non-Data-Ink muss auf ein Minimum reduziert werden.	gut
V11	**Lie Factor:** Der Lie Factor muss optimal sein, um die Aussage der Visualisierungen nicht zu verzerren und Fehlinterpretationen zu verhindern.	gut
	Gesamtbewertung	**befriedigend**

Tabelle 5.10.: Bewertung der visuellen Anforderungen bei AWStats.

Fazit

Trotz der sehr aufgeräumten Bedienoberfläche sind sowohl Qualität als auch Aussagekraft der Visualisierungen noch ausbaufähig. Die Visualisierungen besitzen selbst kaum Beschriftungen, da alle Werte schon in irgendeiner Tabelle vorkommen. Das ist allerdings nicht optimal gelöst. Das Visualisierungskonzept mit den

schmalen Balken, die in Tabellenzeilen gepackt werden, muss überarbeitet werden. Insgesamt erzielt das Tool bei der Bewertung der visuellen Anforderungen nur eine befriedigende Note.

AWStats eignet sich, um diverse Statistiken über Webseitenzugriffe seiner Server zu erhalten. Es vermittelt Administratoren eine Vorstellung davon, welche Zielgruppen wann und mit welchem Client Webseiten besuchen. Diese Informationen können beispielsweise benutzt werden, um die Auslastung der Server zu bewerten, um festzustellen, welche Webseiten besonders gefragt sind oder um auf einen Blick Bots und Crawler zu identifizieren.

Einen Rückschluss auf einzelne IP-Adressen lässt das Tool ebenfalls in einigen Übersichten zu. Aus Datenschutzgründen sollten Webseitenbetreiber diese Adressen in den Logdateien anonymisieren. Als Konsequenz verringert sich dann allerdings auch die Aussagekraft der Analysen von AWStats.

Um einen unmittelbaren Beitrag zur Gewährleistung und Verbesserung der Sicherheit der Server zu leisten, ist AWStats weniger geeignet. Einerseits aggregiert und visualisiert es nur Daten aus den vorhandenen Logdateien ohne aktiv nach irgendwelchen Angreifermustern zu suchen und den Administrator zu warnen. Andererseits visualisiert das Tool Logdateien nicht in Echtzeit. Ein Cronjob liest in zeitlichen Abständen, die der Administrator festlegt, die Logdateien ein und aktualisiert die Datenbank von AWStats. Diese Datenbank dient schließlich als Grundlage für die Visualisierungen.

5.4.2. glTail.rb

Bei *glTail.rb*[36] handelt es sich um ein Tool, das die Menge des Datenverkehrs einzelner Dienste oder Kommunikationsprotokolle in Echtzeit visualisiert. Es verwendet visuelle Metaphern und eine Physik-Engine, um die Aufmerksamkeit des Betrachters auf sich zu lenken. Plötzliche Lastspitzen, die ggf. auf einen Distributed Denial of Service (DDoS)-Angriff hinweisen, stechen sofort ins Auge.

Beschreibung

Das in in der Skriptsprache *Ruby* programmiertes Tool glTail.rb ruft per SSH Logdateien von den zu überwachenden Servern ab. Das Programm unterstützt nicht nur Apache-Logdateien, sondern verfügt auch über Parser für die folgenden Logfile-Formate:

- Rails
- Internet Information Services (IIS)
- Postfix/spamd/clamd
- Nginx
- Squid
- PostgreSQL
- PureFTPD
- MySQL
- TShark
- qmail/vmpop3d

Auf der Projekt-Website wird das Tool damit beworben, dass es alles visualisieren kann, was auch vom Kommandozeilentool `tail` verarbeitet werden kann. Das bedeutet, dass es imstande ist, neue Einträge in einer Logdatei unmittelbar festzustellen und diese als Datenquelle zu benutzen, um die Visualisierung dynamisch anzupassen.

[36] glTail-Website: `http://www.fudgie.org/`

Visualisierung

Der Anwender hat die Wahl, ob aktuell ermittelte Datenraten, Gesamtwerte oder Durchschnittswerte in visueller Form aufbereitet werden sollen. Neue Einträge in einer Logdatei werden in Form von kreisrunden Blasen verbildlicht, die sprichwörtlich in die Bildfläche geworfen werden. Durch die Integration der Physik-Engine *Chipmunk2D*[37] unterliegen die Blasen einer simulierten Schwerkraft, die sie nach unten sinken lässt. Sie stoßen sich auch realistisch voneinander ab, sobald sie aufeinanderprallen.

Wenn beispielsweise eine Webserver-Logdatei als Datenquelle gewählt wird, entspricht jeder Request an den Webserver einer eigenen Blase, die von der linken Seite in die Szene fliegt. Blasen, die von der rechten Seite kommen, repräsentieren jeweils eine angefragte Uniform Resource Locator (URL) oder einen Treffer von einem Referrer. Treffen viele Requests in kurzer Zeit auf den Server, resultiert das in einem Schwall von Blasen, die in das Bild geworfen werden. Damit der Bildschirm nicht überläuft, gibt es am unteren Bildende eine trichterförmige Öffnung, über die die Blasen wieder herausfallen können und schließlich verschwinden.

Als visuelle Attribute verwendet das Programm sowohl den Durchmesser als auch die Farbe der Blasen. So wird zum Beispiel bei Webserver-Logdateien die Größe eines Requests auf den Durchmesser einer Blase abgebildet. Die Farbe gibt Aufschluss darüber, von welcher Website der Request kam. Abbildung 5.27 zeigt ein Bildschirmfoto einer laufenden Visualisierung von Webserver-Logdateien.

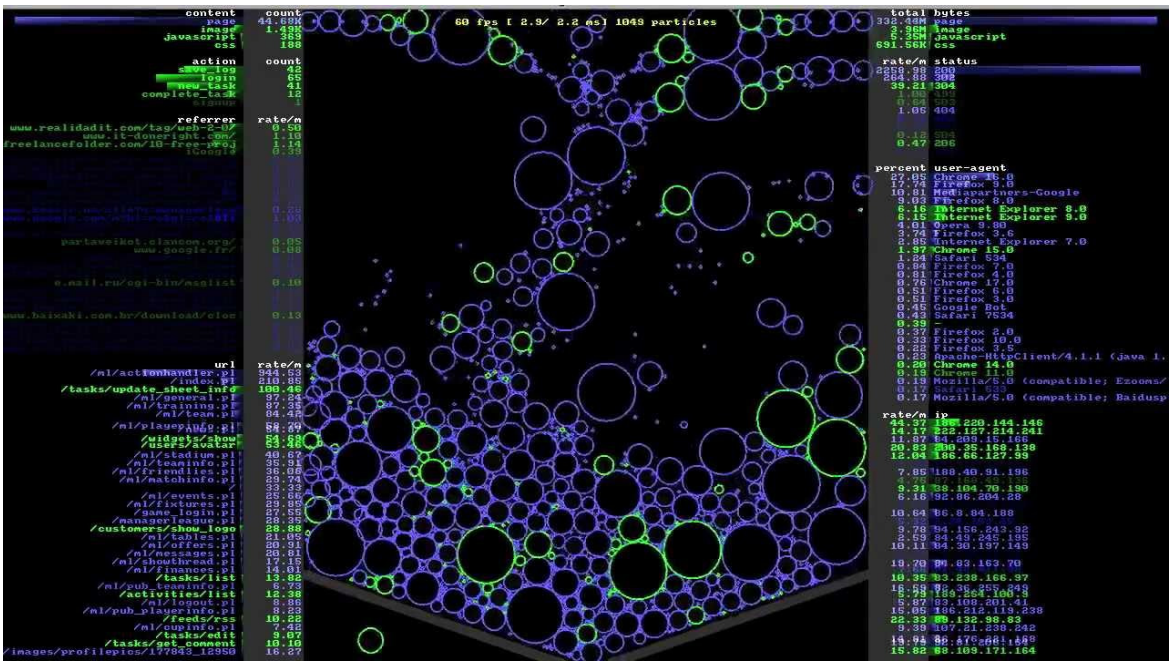


Abbildung 5.27.: Echtzeitvisualisierung von Webserver-Logs mit glTail.rb.[38]

Bedienoberfläche und Visualisierung sind bei dieser Anwendung quasi aus einem Guss, eine explizite Trennung ist dort nicht möglich. Aufgrund der leuchtenden Farben auf schwarzem Hintergrund ist der Kontrast optimal. Nicht mehr benötigte Beschriftungen am Bildschirmrand werden durch ästhetisch anmutende Animationen ausgeblendet. Das Problem bei den seitlich angezeigten Beschriftungen ist aber, dass sie für den Betrachter ihren Bezug zu den Blasen in der Visualisierung verlieren. Der eindeutige Rückschluss von einer Blase zu ihrem Ursprung ist nur über die Farbe möglich, sodass der Betrachter die aufgerufene Webseite nachvollziehen kann. Eine zeitliche Einordnung ist nur indirekt über die Position einer Blase in dem Haufen möglich. Da Eine Blase von anderen auch wieder nach oben verdrängt werden kann, ist die zeitliche Zuordnung anhand der Position aber nicht zuverlässig. Da die Blasen in der Visualisierung aufgrund der Physikberechnungen ständig verdrängt werden, fällt der Lie Factor nicht optimal aus.

[37] Chipmunk2D-Website: `http://chipmunk-physics.net/`
[38] Bildquelle: `http://i.ytimg.com/vi/A_UjjqXNlXg/maxresdefault.jpg`

Die Visualisierungen, die glTail.rb erzeugt, dienen in erster Linie dazu, eine grobe Vorstellung davon zu bekommen, wie sehr ein Server ausgelastet ist. Der Betrachter kann auch grob mit einem Blick erfassen, ob viele oder wenige unterschiedliche Clients Anfragen an den Server schicken. Durch die Metapher mit dem Trichter und den Bläschen, die ständig nachströmen und langsam wieder abfließen wird dem Betrachter unterschwellig die Tatsache vermittelt, dass die Ressourcen eines Servers begrenzt sind und bei großer Last auch erschöpfen können.

Die Visualisierung ist von qualitativer Natur, da keine Skala zur Ermittlung der Durchmesser oder der zugrundeliegenden Daten eingeblendet wird. Zudem wird aufgrund der simulierten Schwerkraft auf ein Mapping zusätzlicher Attribute auf die räumliche Position der Blasen verzichtet. Ergänzt wird die Darstellung von Kennzahlen am linken und rechten Bildschirmrand. Sie umfassen absolute und relative Werte, die aus den Logdateien gewonnen werden.

Bewertung der visuellen Anforderungen

Nr.	Kurzbeschreibung	Bewertung
V1	**Übersichtlichkeit:** Die Bedienoberfläche darf nicht überfrachtet sein, da sonst die Übersichtlichkeit darunter leidet.	befriedigend
V2	**Farben:** In der Bedienoberfläche müssen Farben sparsam und zielgerichtet eingesetzt werden.	sehr gut
V3	**Kontrast:** In der Bedienoberfläche müssen die Kontrastverhältnisse hoch sein.	sehr gut
V4	**Objektgröße:** In der Bedienoberfläche müssen Oberflächenelemente eine ausreichende Größe besitzen.	gut
V5	**Visualisierungsformen:** Die Arten der Visualisierungen müssen zu den zugrundeliegenden Daten und Dimensionalitäten passen.	sehr gut
V6	**Beschriftungen:** Visualisierungen müssen mit den notwendigen Beschriftungen versehen sein, damit sie nicht fehlinterpretiert werden können.	ausreichend
V7	**Visual Clutter:** Visualisierungen müssen frei von Visual Clutter sein, um die Übersichtlichkeit zu verbessern.	ausreichend
V8	**Farbskalen:** Farben in Visualisierungen müssen sparsam und zielgerichtet eingesetzt werden. Farbige Datenreihen müssen gut voneinander unterscheidbar sein.	sehr gut
V9	**Kontraste in Visualisierungen:** Kontrastverhältnisse in Visualisierungen müssen hoch sein, um die Lesbarkeit zu optimieren.	sehr gut
V10	**Chartjunk und Non-Data-Ink:** Chartjunk lenkt den Betrachter ab und ist unerwünscht. Non-Data-Ink muss auf ein Minimum reduziert werden.	sehr gut
V11	**Lie Factor:** Der Lie Factor muss optimal sein, um die Aussage der Visualisierungen nicht zu verzerren und Fehlinterpretationen zu verhindern.	ausreichend
	Gesamtbewertung	**gut**

Tabelle 5.11.: Bewertung der visuellen Anforderungen bei glTail.rb.

Fazit

Die Art der Visualisierung, die das Programm generiert, ist ungewohnt, aber faszinierend anzusehen. Trotz oder gerade wegen ihrer einfachen Grundbausteine macht die Visualisierung einen ästhetischen Eindruck. Da auf der Ästhetik auch der Schwerpunkt liegt, eignet sich diese Art der Visualisierung weniger für ein System, das im Alltag eines Administrators als primäre Informationsquelle dient. Stattdessen eignet sich glTail.rb gut, um rund um die Uhr im Hintergrund auf einem separaten Bildschirm zu laufen, der an einer Wand im Flur oder in einem Pausenraum hängt. Dort erfüllt es einerseits einen dekorativen Zweck, andererseits sehen Administratoren beim Vorbeigehen, wie hoch die Serverlast im Allgemeinen ist und können bei Bedarf einen näheren Blick auf ihre Systeme werfen, indem sie zu einem spezielleren Tool greifen. Dies macht glTail.rb zu einem guten Praxisbeispiel für sog. *Ambient Information Visualization* (siehe Abschnitt 2.17.5). Ein Video, das glTail.rb in Aktion zeigt, ist auf YouTube zu sehen.[39].

[39] Video von glTail.rb in Aktion: `https://www.youtube.com/watch?v=RCa2sjyrUdQ`

Die Teilbewertungen dieser Art der Visualisierung fallen sehr unterschiedlich aus. Dies liegt vermutlich daran, dass Ambient Information Visualization im Vergleich mit den Visualisierungstechniken, die in den anderen Tools zum Einsatz kommen, etwas aus dem Rahmen fällt. Insgesamt gibt es eine gute Bewertung für die Umsetzung der visuellen Anforderungen.

5.4.3. Logstalgia

Logstalgia, auch als *ApachePong* bezeichnet, visualisiert Logdateien von Webservern in Echtzeit. Ähnlich wie glTail.rb besitzt es eine effektvolle, auf Ästhetik ausgelegte Form der Visualisierung.

Beschreibung

Ähnlich wie glTail.rb dient Logstalgia in erster Linie dazu, die Frequenz von Logdateiänderungen zu visualisieren. Je höher diese Frequenz ist, desto dichter fällt die Visualisierung aus. Aus diesem Grund eignet sich das Tool hervorragend, um nebenbei das Besucheraufkommen auf einem Webserver zu überwachen und auch um Angriffe sofort zu erkennen.

Visualisierung

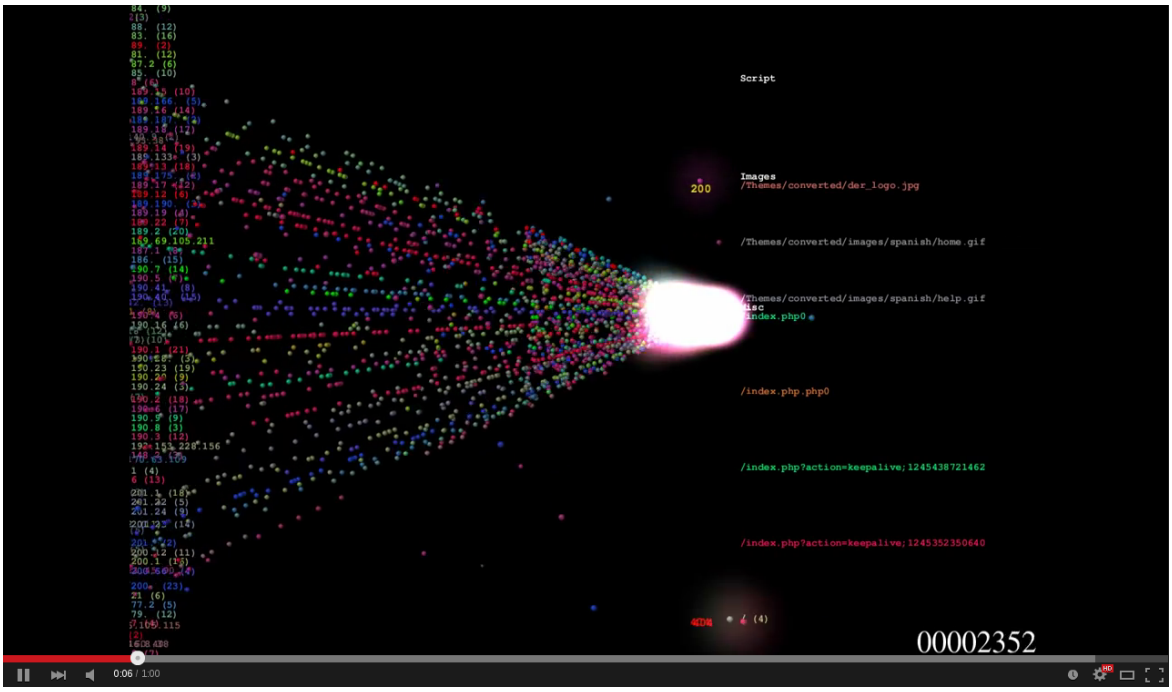


Abbildung 5.28.: Logstalgia visualisiert einen DDoS-Angriff auf einen Webserver in Echtzeit.[40]

Wie bei glTail.rb ist der Bildschirmhintergrund mit der Farbe Schwarz gefüllt. Die Vordergrundfarben der Visualisierung besitzen einen guten Kontrast zum Hintergrund und sind auch voneinander gut zu unterscheiden.

Requests von Clients werden bei Logstalgia als kleine, farbige Bälle dargestellt, die von den Absender-IP-Adressen auf der linken Seite des Bildschirms zu den Ressourcen des Webserver auf der rechten Seite abgefeuert werden. Jede angeforderte Ressource wird in Form des Dateinamens auf dem Server angezeigt und dynamisch eingeblendet.

[40]Bildquelle: `http://www.youtube.com/watch?v=_R-dm62NZ5E`

Sobald ein Request in Form eines Balls bei der angeforderten Datei auftrifft, wird er ähnlich wie beim Videospielklassiker *Pong* von einem angedeuteten Tischtennisschläger zurück zum Client reflektiert (daher kommt auch der Alternativname des Programms ApachePong). Der reflektierte Ball visualisiert die Antwort des Servers auf die Anfrage des Clients. Gleichzeitig werden an der Stelle der Reflexion die dazugehörigen Fehlercodes eingeblendet, die erst nach und nach wieder verschwinden. Durch das langsame Ausblenden und die farblichen Überlagerungen ist die Frequenz der Requests bzw. Responses intuitiv als eine Art Nachglühen auf dem Bildschirm aufzufassen.

Jeder Fehlercode besitzt eine eigene Farbskala. Codes der Fehlerklasse 400 werden beispielsweise in Rottönen, Codes der Klasse 200 in Gelbtönen dargestellt. Häufungspunkte von Requests und Responses visualisiert Logstalgia effektvoll als farbige Wolken, die sich nach links zu den Clients bewegen. Bei extremem Verkehrsaufkommen, wie es etwa bei einem DDoS der Fall ist, resultiert dies in einer Wolke aus gleißendem Licht. Ein Beispiel hierfür ist in Abbildung 5.28 zu sehen. Ein weiteres Beispiel gibt es in einem YouTube-Video[41].

Bewertung der visuellen Anforderungen

Nr.	Kurzbeschreibung	Bewertung
V1	**Übersichtlichkeit:** Die Bedienoberfläche darf nicht überfrachtet sein, da sonst die Übersichtlichkeit darunter leidet.	gut
V2	**Farben:** In der Bedienoberfläche müssen Farben sparsam und zielgerichtet eingesetzt werden.	sehr gut
V3	**Kontrast:** In der Bedienoberfläche müssen die Kontrastverhältnisse hoch sein.	sehr gut
V4	**Objektgröße:** In der Bedienoberfläche müssen Oberflächenelemente eine ausreichende Größe besitzen.	gut
V5	**Visualisierungsformen:** Die Arten der Visualisierungen müssen zu den zugrundeliegenden Daten und Dimensionalitäten passen.	sehr gut
V6	**Beschriftungen:** Visualisierungen müssen mit den notwendigen Beschriftungen versehen sein, damit sie nicht fehlinterpretiert werden können.	ausreichend
V7	**Visual Clutter:** Visualisierungen müssen frei von Visual Clutter sein, um die Übersichtlichkeit zu verbessern.	mangelhaft
V8	**Farbskalen:** Farben in Visualisierungen müssen sparsam und zielgerichtet eingesetzt werden. Farbige Datenreihen müssen gut voneinander unterscheidbar sein.	sehr gut
V9	**Kontraste in Visualisierungen:** Kontrastverhältnisse in Visualisierungen müssen hoch sein, um die Lesbarkeit zu optimieren.	sehr gut
V10	**Chartjunk und Non-Data-Ink:** Chartjunk lenkt den Betrachter ab und ist unerwünscht. Non-Data-Ink muss auf ein Minimum reduziert werden.	sehr gut
V11	**Lie Factor:** Der Lie Factor muss optimal sein, um die Aussage der Visualisierungen nicht zu verzerren und Fehlinterpretationen zu verhindern.	ausreichend
	Gesamtbewertung	**gut**

Tabelle 5.12.: Bewertung der visuellen Anforderungen bei Logstalgia.

Fazit

Logstalgia legt ähnlich wie glTail.rb den Schwerpunkt auf Ästhetik. Es fällt dem Betrachter aber im Vergleich zu glTail.rb noch schwerer, einzelne Logereignisse voneinander zu unterscheiden, sobald das Verkehrsaufkommen zunimmt. Ebenso wenig sind die einzelnen Fehlercodes bei hoher Last noch erkennbar. Durch die effektvolle Darstellung der Frequenz der Logereignisse vermittelt das Tool aber ein allgemeines Gefühl dafür, wie stark ein Webserver ausgelastet ist.

Falls eine hohe Last bemerkt wird, kann der Administrator die Situation mit Hilfe anderer Mittel genauer untersuchen, denn hierfür ist das Tool aufgrund des sprichwörtlichen Effektfeuerwerks nicht geeignet. Somit ist Logstalgia ebenfalls in die Kategorie *Ambient Information Visualization* (siehe Abschnitt 2.17.5) einzuordnen.

[41] Video von Logstalgia in Aktion: `https://www.youtube.com/watch?v=HeWfkPeDQbY`

Auch bei diesem Tool fallen die Teilbewertungen sehr unterschiedlich aus. Insgesamt gibt es aber eine gute Gesamtbewertung.

5.4.4. Splunk Enterprise

Splunk Enterprise ist ein Log-, Monitoring- und Reporting-Tool des kalifornischen Softwareherstellers Splunk, Inc.[42]. Die Hauptaufgabe von Splunk Enterprise ist das Sammeln von Logdaten verschiedenster Quellen und eine automatische Verarbeitung und Indexierung, um sie später komfortabel durchsuchen und auswerten zu können. Das Programm soll Administratoren dabei helfen, Störfälle zu erkennen und zu analysieren.

Beschreibung

Als webbasierte Anwendung kommt Splunk ohne die Installation spezieller Client-Programme aus. Um den Zugriff über mobile Endgeräte zu erleichtern, gibt es eine spezielle App für die Plattformen Android und iOS.

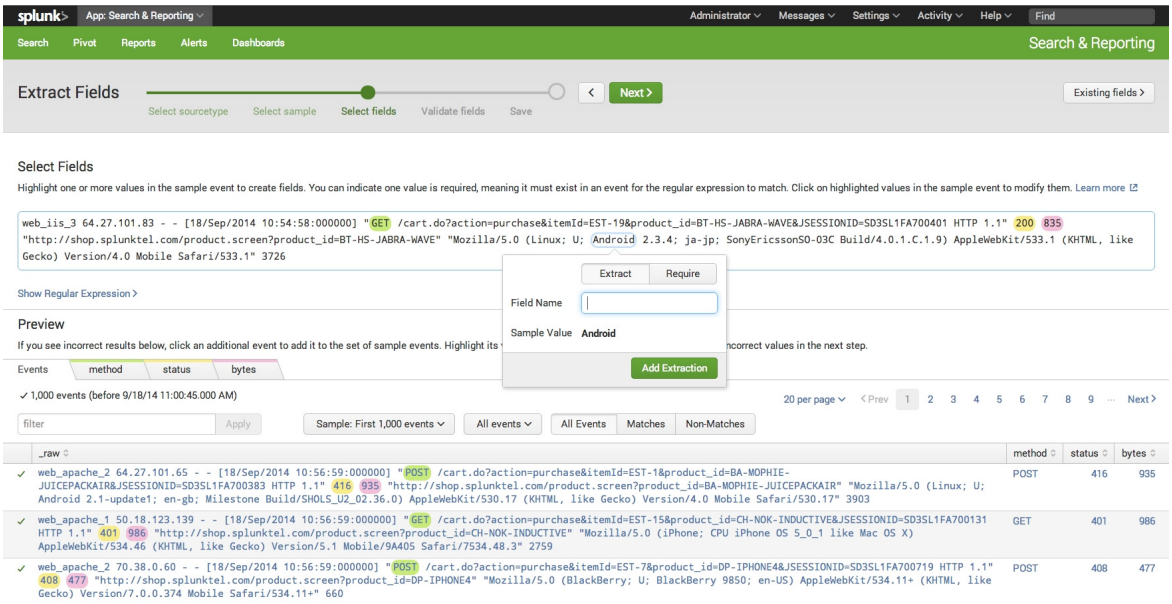


Abbildung 5.29.: Über den Advanced Field Extractor lernt Splunk den Umgang mit bisher unbekannten Logdateiformaten.[43]

Mit Ausnahme von binären Programmdateien kann Splunk beliebige Dateien als Datenquelle verwenden, in der Regel sind das Protokolldateien von Server-Prozessen. Der *Advanced Field Extractor* unterstützt Administratoren dabei, dem System neue Logdateiformate beizubringen. Farbliche Hervorhebungen helfen dem Anwender dabei, einzelne Komponenten von Logdateien besser unterscheiden zu können (siehe Abbildung 5.29).

Splunk verfügt mit Search Processing Language (SPL) über eine eigene formale Sprache, um maßgeschneiderte Auswertungen der Daten zu erstellen. Mehr als 140 Funktionen stehen dem Anwender während der Datenanalyse zur Verfügung. Darüber hinaus existieren zahlreiche Erweiterungsmöglichkeiten für unterschiedliche Anwendungsbereiche wie zum Beispiel der Überwachung von Cisco-Netzkomponenten.

Splunk ist nicht nur in der Lage, Informationen aus zahlreichen verschiedenen Datenquellen zu sammeln, sondern auch zu korrelieren. Beziehungen zwischen Ereignissen oder Aktivitäten stellt das Programm anhand von Uhrzeiten, Orten oder den Ergebnissen benutzerdefinierter Suchen her. Einzelne Ereignisse, die logisch zusammengehören, fasst Splunk zu sog. *Transaktionen* zusammen.

[42] Splunk-Website: `http://www.splunk.com/`
[43] Bildquelle: `http://mms.businesswire.com/media/20141007005275/en/435321/5/Advanced_Field_Extractor.jpg`

Visualisierung

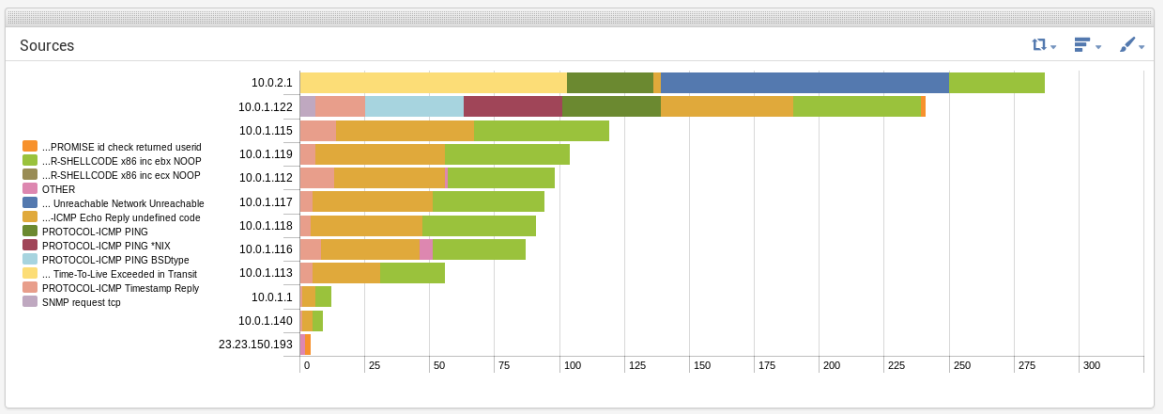


Abbildung 5.30.: Visualisierung von Netzwerkverkehr aufgegliedert nach Host-Systemen und Art der Pakete in Splunk.[44]

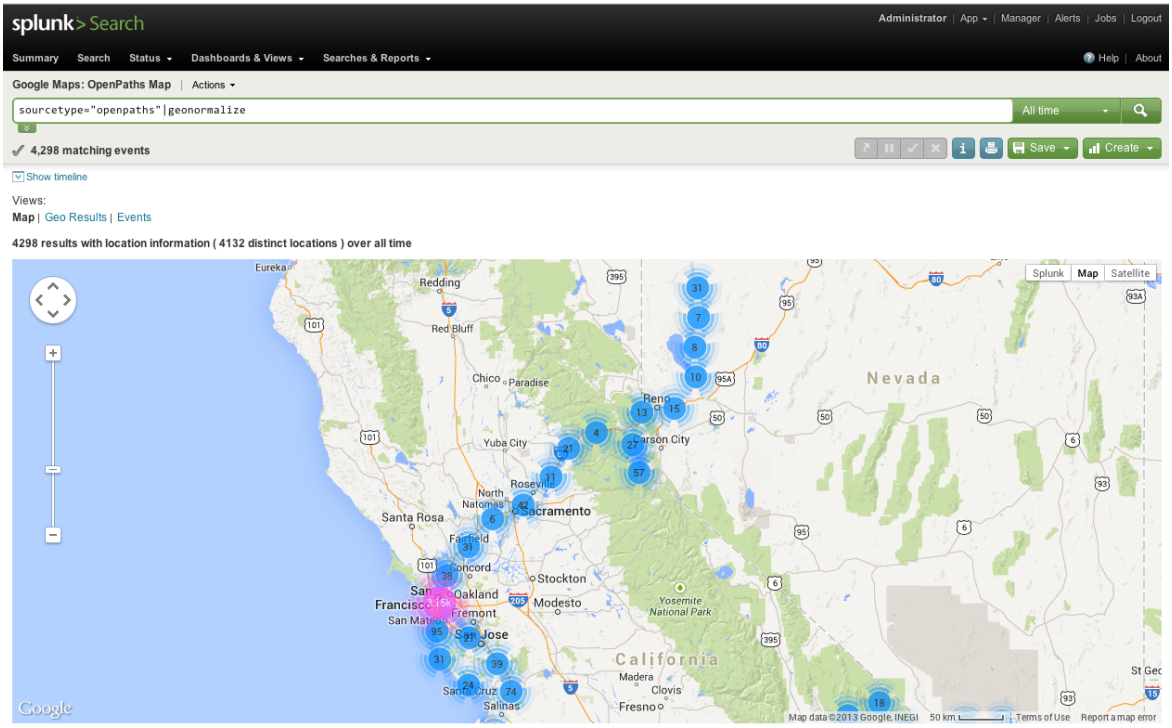


Abbildung 5.31.: Visualisierung personenbezogener Bewegungsdaten mit Splunk.[45]

In seiner Bedienoberfläche setzt Splunk hauptsächlich auf Graustufen. Abgesehen von den beiden Akzentfarbe Grün und Blau mit unterschiedlichen Sättigungsgraden bleiben Farben den Nutzinhalten der Anwendung vorbehalten. Der in Abbildung 5.29 sichtbare Advanced Field Extractor nutzt beispielsweise unterschiedliche Pastellfarben zur Hervorhebung verschiedener Bestandteile von Logeinträgen.

Abbildung 5.30 zeigt ein Beispiel für eine Visualisierung in Splunk. Das gestapelte Balkendiagramm gibt Aufschluss über den Netzwerktraffic, den mehrere betrachtete Hosts generieren. Die verschieden eingefärbten

[44] Bildquelle: `http://www.metanetivs.com/wp-content/uploads/2013/11/Selection_008.png` (abgerufen am 27.04.2015)

[45] Bildquelle: `http://www.geeked.info/wp-content/uploads/2013/10/OpenPaths-Travel-to-Burning-Man.png`

Stapel kennzeichnen unterschiedliche Arten von Traffic wie zum Beispiel Internet Control Message Protocol (ICMP) Echo Requests oder Simple Network Management Protocol (SNMP) Requests.

Die Visualisierung wirkt auf den ersten Blick aufgeräumt und kommt ohne Chartjunk aus. Der Kontrast zwischen Vordergrund und Hintergrund ist gut, allerdings gibt es innerhalb des Diagramms Kontrastprobleme: Besonders ab der dritten Zeile in Abbildung 5.30 sind die Farben der ersten beiden Stapel kaum voneinander zu unterscheiden.

Bei der Betrachtung der in der Legende vorkommenden Farben möchte man dieses Problem nicht für möglich halten. Da in den betrachteten Zeilen aber nicht alle Farben vorkommen, ist es möglich, dass ähnliche Farben nebeneinander liegen, obwohl das in der Legende nicht der Fall ist. Je mehr Farben die Legende umfasst, desto wahrscheinlicher sind derartige Konflikte. Mit insgesamt zwölf unterschiedlichen Farben ist das Diagramm jedoch überladen. Die Übersichtlichkeit leidet außerdem etwas unter der grenzwertigen Schriftgröße und den abgeschnittenen Beschreibungstexten der Legende.

Splunk ist so flexibel einsetzbar, dass damit auch personenbezogene Daten ausgewertet und visualisiert werden können. Ein Beispiel hierfür sind Bewegungsdaten. Die aufgezeichneten Aufenthaltsorte der Personen lassen sich in geographische Karten einblenden, um detaillierte Bewegungsprofile zu erstellen. Selbstverständlich können Karten auch zum Einzeichnen serverbezogener Daten verwendet werden.

Ein ehemaliger Splunk-Entwickler berichtet in einem Blog-Eintrag von einem Selbstversuch und den technischen Möglichkeiten, die positionsbestimmende und biometrische Gadgets bieten, um Splunk mit visualisierbaren Daten aus dem Alltag zu füllen [HUN 13]. Es ist bemerkenswert, welche Spuren Anwender von Internetdiensten hinterlassen. Bedenklich sind die technischen Möglichkeiten aus Sicht des Datenschutzes.

Abbildung 5.31 zeigt einen Screenshot aus diesem Blog-Eintrag. Die markierten Punkte auf der Karte der US-Westküste stellen dar, wie viele ungelesene E-Mails der Autor in seinem Posteingang hatte. Nebenbei lässt sich dabei auch erkennen, wo er sich entlang bewegt hatte.

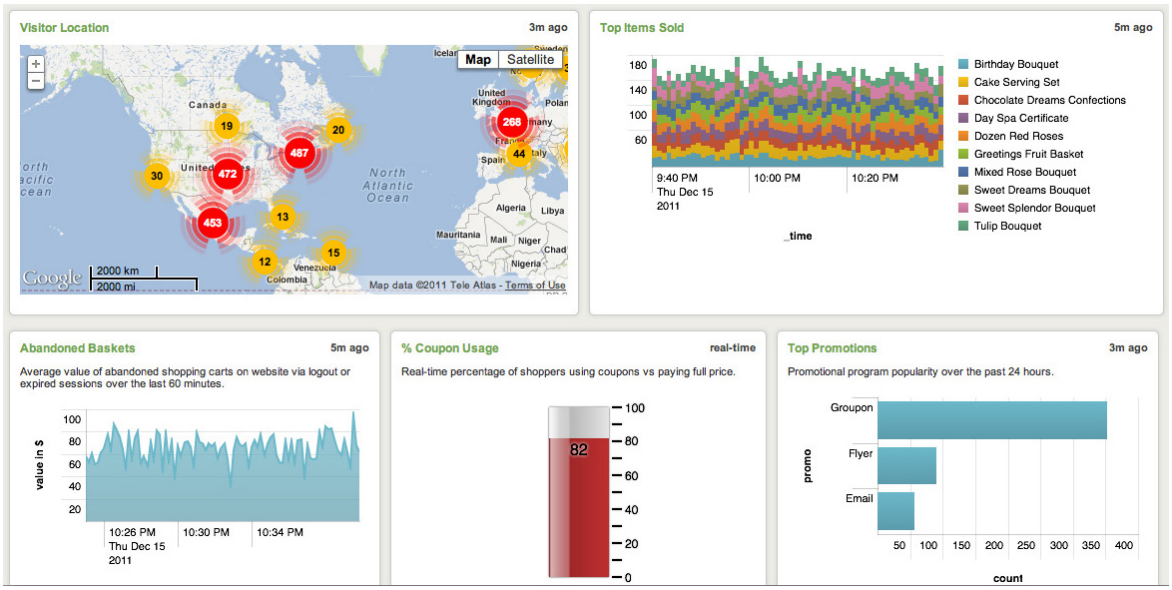


Abbildung 5.32.: Splunk unterstützt Internethändler durch die Visualisierung unternehmenskritischer Kennzahlen (Bildausschnitt).[46]

Splunk wird in der Praxis nicht mehr ausschließlich als Logdatenanalysetool eingesetzt, sondern auch in den verschiedensten Big-Data-bezogenen Anwendungsfällen [MCKE 13]. Unternehmen verwenden es beispielsweise, um die Auswirkungen eines Webseiten-Redesigns auf die Besucherzahlen zu messen. Betreiber von Online Shops greifen auf Splunk zurück, um unternehmensstrategische Entscheidungen zu treffen oder sie zumindest zu beeinflussen.

[46] Bildquelle: `http://data-informed.com/wp-content/uploads/2012/05/Dashboard.jpg`

Abbildung 5.32 demonstriert beispielsweise die Tauglichkeit von Splunk für Online Shops. Das Programm gibt Aufschluss über die Herkunft der Konsumenten, welche Artikel die aktuellen Bestseller sind und wie sich Rabattaktionen auf das Geschäft auswirken. Sogar der theoretische Gewinnverlust aufgrund von Einkaufskörben, die nicht bestellt, sondern zuvor wieder geleert wurden, ist sichtbar.

In dem Screenshot wird sichtbar, wie vielfältig Splunk Daten visualisieren kann. Neben herkömmlichen Linien- und Balkendiagrammen kommen ein gestapeltes Säulendiagramm und eine Weltkarte zum Einsatz. Letztere stellt dem Benutzer Interaktionsmöglichkeiten bereit, wie sie zum Beispiel von *Google Maps* bekannt sind und integriert farblich hervorgehobene Symbole zur Darstellung ortsbezogener Kennzahlen.

Aus Sicht der Informationsvisualisierung lässt sich der Anteil an *Eye Candy* (siehe Abschnitt 2.13.3) reduzieren, da er keinen Beitrag zur eigentlichen Funktion der Anwendung leistet. Dies trifft zum Beispiel auf die Füllstandanzeige unten in der Mitte des Bilds zu. Durch den Gradienten und den hellen Schatten hinter dem schwarzen Prozentsatz soll ein räumlicher Eindruck vermittelt werden. Dieser Effekt ist aufgrund der besseren Übersichtlichkeit entbehrlich. Die Übersichtlichkeit des Liniendiagramm unten links in Abbildung 5.32 kann ebenfalls verbessert werden, indem auf Eye Candy verzichtet wird. Der Transparenzeffekt sowie der Farbverlauf der Diagrammfläche erfüllen keinen Zweck und sind überflüssig. Dasselbe gilt für den Farbverlauf im Diagramm unten rechts.

Das gestapelte Säulendiagramm oben rechts wirkt etwas überfrachtet. Die Anzahl der Farben, die zur Kodierung der Artikel verwendet werden, ist grenzwertig und zieht Kontrastprobleme nach sich. Ein weiteres Detail wirkt sich bei der großen Anzahl unterschiedlicher Farben nachteilig auf die Lesbarkeit aus: Die Sortierung der Farben in der Legende verhält sich genau entgegengesetzt zu der Sortierung im Diagramm. Sie spiegelt zwar die Bedeutung der Top-10-Artikel wider, verwirrt aber den Betrachter.

Bewertung der visuellen Anforderungen

Nr.	Kurzbeschreibung	Bewertung
V1	**Übersichtlichkeit:** Die Bedienoberfläche darf nicht überfrachtet sein, da sonst die Übersichtlichkeit darunter leidet.	gut
V2	**Farben:** In der Bedienoberfläche müssen Farben sparsam und zielgerichtet eingesetzt werden.	gut
V3	**Kontrast:** In der Bedienoberfläche müssen die Kontrastverhältnisse hoch sein.	gut
V4	**Objektgröße:** In der Bedienoberfläche müssen Oberflächenelemente eine ausreichende Größe besitzen.	befriedigend
V5	**Visualisierungsformen:** Die Arten der Visualisierungen müssen zu den zugrundeliegenden Daten und Dimensionalitäten passen.	gut
V6	**Beschriftungen:** Visualisierungen müssen mit den notwendigen Beschriftungen versehen sein, damit sie nicht fehlinterpretiert werden können.	gut
V7	**Visual Clutter:** Visualisierungen müssen frei von Visual Clutter sein, um die Übersichtlichkeit zu verbessern.	befriedigend
V8	**Farbskalen:** Farben in Visualisierungen müssen sparsam und zielgerichtet eingesetzt werden. Farbige Datenreihen müssen gut voneinander unterscheidbar sein.	befriedigend
V9	**Kontraste in Visualisierungen:** Kontrastverhältnisse in Visualisierungen müssen hoch sein, um die Lesbarkeit zu optimieren.	befriedigend
V10	**Chartjunk und Non-Data-Ink:** Chartjunk lenkt den Betrachter ab und ist unerwünscht. Non-Data-Ink muss auf ein Minimum reduziert werden.	befriedigend
V11	**Lie Factor:** Der Lie Factor muss optimal sein, um die Aussage der Visualisierungen nicht zu verzerren und Fehlinterpretationen zu verhindern.	sehr gut
	Gesamtbewertung	**gut**

Tabelle 5.13.: Bewertung der visuellen Anforderungen bei Splunk Enterprise.

Fazit

Eine allgemeine Bewertung der Visualisierungen, die Splunk bietet, fällt etwas schwer, da das Tool häufig in Kombination mit Erweiterungen betrieben wird. Es wird deshalb davon ausgegangen, dass sich die Plug-in-Entwickler an die Style-Vorgaben von Splunk halten.

Die Bedienoberfläche wirkt aufgrund des sparsamen Einsatzes von Farben übersichtlich und lässt den Visualisierungen den Vortritt. Da das Tool die unterschiedlichsten Datenquellen als Grundlage für Visualisierungen nutzen kann, und durch Plug-ins erweiterbar ist, ist es sehr vielfältig einsetzbar, nicht nur in der Serveradministration.

Die Fülle an Funktionen und Visualisierungsmethoden führen aber auch zum Teil zu überladenen Visualisierungen. Zu viele unterschiedliche Farben in einem Diagramm resultieren in einem schlechten Farbkontrast und in verschlechterter Übersichtlichkeit. Durch die Reduzierung von Non-Data-Ink und Eye Candy ließe sich die Übersichtlichkeit erhöhen. Insgesamt fällt die Bewertung der visuellen Eigenschaften bei Splunk jedoch gut aus.

5.5. Zusammenfassung

In Tabelle 5.14 werden die einzelnen Testergebnisse dieses Kapitels noch einmal zusammengefasst. Für die Noten von *sehr gut* bis *mangelhaft* werden aus Platzgründen die Ziffern von 1 bis 5 verwendet.

Anwendungskategorie	Vuln. Scanner		Konfigurationsmanagement				SIEM			Logdatenanalyse			
Anwendung / Bewertungskriterium	Nessus	Qualys SSL Server Test	Ansible Tower	BlueCat Proteus	Spacewalk	VMware vSphere Client	AlienVault USM	LogRhythm	McAfee ESM	AWStats	glTail.rb	Logstalgia	Splunk Enterprise
Visuelle Anforderungen													
V1: Übersichtlichkeit	4	2	2	3	2	4	1	4	4	2	3	2	2
V2: Farben	4	2	2	3	2	3	1	3	5	2	1	1	2
V3: Kontrast	4	3	2	3	3	2	1	2	3	1	1	1	3
V4: Objektgröße	3	2	3	2	2	3	4	2	3	3	2	2	3
V5: Visualisierungsformen	2	2	3	3	4	2	2	4	5	3	1	1	2
V6: Beschriftungen	3	1	3	4	2	3	3	3	2	5	4	4	2
V7: Visual Clutter	1	2	2	3	2	3	2	5	3	2	4	5	3
V8: Farbskalen	2	2	1	3	1	3	1	4	4	4	1	1	3
V9: Kontraste in Visualisierungen	4	2	2	3	2	2	3	3	4	4	1	1	3
V10: Chartjunk und Non-Data-Ink	3	2	1	4	1	3	1	2	4	2	1	1	3
V11: Lie Factor	4	5	1	1	1	1	1	2	2	2	4	4	1
Gesamtbewertung:	3	2	2	3	2	3	2	3	4	3	2	2	2

Tabelle 5.14.: Gesamtbewertung der vorhandenen Lösungen.

Wie in diesem Kapitel deutlich wurde, existieren die unterschiedlichsten Tools, die die Administration von Linux-Servern erleichtern und einen Beitrag zur Informationssicherheit leisten. Die Evaluation hat gezeigt, dass die informationsvisualisierenden Komponenten in diesen Tools nicht immer sinnvoll eingesetzt werden. Es wurden einerseits Verstöße gegen grundlegende Designrichtlinien aus Kapitel 2 festgestellt. Andererseits

wurden vorhandene Potentiale zum Teil nicht ausgenutzt, um bestimmte Anwendungsfälle durch geeignete Visualisierungen zu bereichern. Dies hat zur Folge, dass kein einziges Programm im Lauf der Evaluation eine sehr gute Gesamtnote bei der Bewertung der visuellen Attribute erzielt (siehe Tabelle 5.14). Andererseits war nur ein Programm unter den Kandidaten, das lediglich eine ausreichende Benotung erhielt.

Da bei sämtlichen betrachteten Tools Defizite zu verzeichnen sind, motiviert dies zu einer eigenen Lösung, die beweist, dass es besser geht. Um die Defizite auszugleichen, werden deshalb im folgenden Kapitel zielführende Lösungsmöglichkeiten erarbeitet, die sich konsequent an Grundlagen der Informationsvisualisierung halten und auch Grundlagen der Informationssicherheit berücksichtigen. Die Realisierbarkeit der Lösungsmöglichkeiten unter Berücksichtigung von Grundlagen der Informationssicherheit wird durch den zu entwickelnden Demonstrator untersucht.

Diese Evaluation hat auch gezeigt, dass Tools aus dem Bereich Ambient Information Visualization (siehe 2.17.5) zwar stark divergierende Teilergebnisse bei der Bewertung der visuellen Attribute liefern, im Durchschnitt aber dennoch mit einer guten Note abschneiden. Administrationstools, die Ambient Information Visualization einsetzen, liefern zwar keine besonders genauen Aussagen, visualisieren dafür sicherheitsrelevante Tendenzen umso besser. Aus diesem Grund sollte der Einsatz solcher Tools am LRZ durchaus einmal überdacht werden.

6. Lösungsmöglichkeiten für verschiedene Anwendungsfälle

In diesem Kapitel werden typische Anwendungsfälle der Linux-Server-Administration besprochen und ihre Unterstützbarkeit durch Techniken der Datenvisualisierung diskutiert. Bei allen Anwendungsfällen ist ein enger Bezug zur Informationssicherheit vorhanden. Durch geeignete Visualisierungsmethoden sollen Administratoren einen schnellen Gesamtüberblick über sicherheitsrelevante Faktoren erhalten, um so unmittelbar wichtige Eingriffe in die Systeme vornehmen zu können. Neben reaktiven Aktionen eignen sich die Vorschläge, die in diesem Kapitel angesprochen werden auch für proaktive Zwecke. Es ist schließlich besser und auch günstiger, potenzielle Sicherheitslücken im Voraus zu erkennen und zu beheben, als bereits eingetretene Schäden zu reparieren.

Zu den in dieser Ausarbeitung betrachteten Use Cases zählen die Prüfung von Paketversionen (Abschnitt 6.1), der Konfigurationen von Personal Firewalls (Abschnitt 6.2), der Sicherheit privater Schlüssel (Abschnitt 6.3) und der sicheren Konfiguration von TLS am Beispiel des Web Servers Apache (Abschnitt 6.4). Die Auswahl dieser Use Cases ist einerseits durch den engen Bezug zur Informationssicherheit begründet, andererseits handelt es sich um alltägliche Themen, mit denen Administratoren konfrontiert werden. Der Bezug dieser Anwendungsfälle zum LRZ ist auch sehr stark.

Zunächst ist es denkbar, für jeden der ausgewählten Anwendungsfälle ein eigenes Grundkonzept für die Visualisierung sicherheitsrelevanter Server-Informationen zu erstellen. Es gibt bereits verschiedene Ansätze für Visualisierungsmethoden, die aus dem Bereich der IT- und Server-Sicherheit stammen. Dazu zählt zum Beispiel das Tool *PolicyVis* [TRAN+ 07], das unübersichtliche Firewall-Regelwerke durch Visualisierung begreifbarer machen soll. Ein weiterer Ansatz wurde von *Florian Mansmann et al.* im Jahr 2012 vorgestellt (siehe Abbildung 6.1) [MAN+ 12].

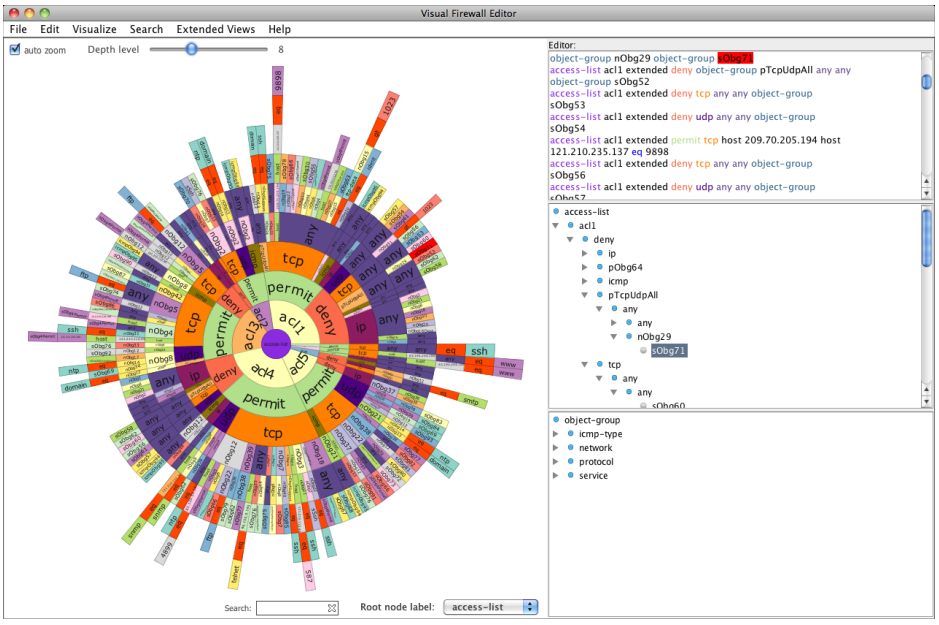


Abbildung 6.1.: Visualisierung von Firewall-Regeln im Visual Firewall Editor von Mansmann et al.

Betrachtet man neben dem Anwendungsbereich der Firewalls noch weitere, wird die Anzahl möglicher Vi-

sualisierungskonzepte unüberschaubar. Es ist daher erstrebenswert, ein gemeinsames Grundkonzept zur Visualisierung der Server-Zustände aus den vier ausgewählten Anwendungsfällen zu entwickeln. Ein gemeinsames Grundkonzept senkt in einer konkreten Implementierung die Eintrittshürden für die Administratoren, die täglich damit arbeiten müssen. Es erreicht zwar nicht den Detailgrad, den die Summe mehrerer spezieller Visualisierungskonzepte liefern, als zentrale Komponente in einer Implementierung sorgt es allerdings für eine hohe Übersichtlichkeit, da der Anwender nicht gezwungen ist, zwischen mehreren Visualisierungen, die inhaltlich vollkommen unabhängige Informationen liefern, ständig hin und her zu wechseln.

Durch ihren hohen Detailgrad werden die Use-Case-spezifischen Visualisierungen, die bereits existieren, aber nicht überflüssig. Sie können zusätzlich in das Informationssystem integriert werden und werden erst dann sichtbar, wenn der Anwender sie explizit über die für alle Anwendungsfälle einheitliche Oberfläche auswählt.

Der folgende Abschnitt betrachtet zunächst den Anwendungsfall der Software-Verwaltung auf den Servern. Dieser Anwendungsfall wird als Einstieg gewählt, da es sich um den vermutlich alltäglichsten handelt und der Sicherheitsbezug sehr hoch ist. Anhand dieses Anwendungsfalls wird basierend auf Grundlagen der Informationsvisualisierung ein Schema erarbeitet, das sich zur Integration der anderen Anwendungsfälle eignet.

6.1. Software-Verwaltung

Programmpakete auf einem Linux-System regelmäßig auf den neuesten Stand zu bringen ist in erster Linie deshalb notwendig, da Programmfehler und Sicherheitslücken dadurch beseitigt werden. Dies ist besonders bei Servern wichtig. Sie laufen in der Regel den ganzen Tag und verfügen über eine breitbandige Anbindung ans Internet. Rund um die Uhr sind sie potenziellen Angriffen ausgesetzt, die auf ungepatchte Sicherheitslücken abzielen.

6.1.1. Vorüberlegungen

Praktisch alle für den Server-Betrieb relevanten Linux-Distributionen stellen ein Paketmanagementsystem zur Verfügung, das die Installation, Aktualisierung und Verwaltung von Abhängigkeiten erleichtert. Einzelne Programme müssen so nicht als Quellcode heruntergeladen und selbst kompiliert werden. Zudem unterstützen einige der Paketwerkzeuge den Anwender bei der Beseitigung von Konflikten, die bei der Installation bestimmter Komponenten auftreten können. Zu den möglichen Konflikten zählen Folgende:

Wenn es Dateien gibt, die in mehr als einem Paket enthalten sind, dürfen nicht beide Pakete gleichzeitig installiert werden. Ohne eine Paketverwaltung, die die gleichzeitige Installation beider Pakete verhindert, werden die gemeinsamen Dateien vom zuletzt installierten Paket überschrieben und es kann zu Problemen bei der Ausführung des zuerst installierten Pakets kommen. Beim Einspielen von Updates werden nicht nur die Anwendungen selbst, sondern auch gemeinsam genutzte Dateien wie zum Beispiel Bibliotheken aktualisiert. In einigen Fällen sind Programme aber nur mit der älteren Version einer Bibliothek lauffähig. Darauf muss das Paketsystem achten.

Die Software auf einem einzigen Rechner aktuell zu halten, stellt für den Benutzer kein großes Problem dar. Anwendungsfreundliche Desktop-Distributionen bieten oft die Möglichkeit, regelmäßig selbst nach Updates zu suchen und bei Bedarf sogar automatisch zu installieren. Im einer Linux-Server-Landschaft, wie sie beim LRZ zu finden ist, ist das nicht so leicht. Die Anzahl der aktuell zu haltenden Systeme ist sehr hoch. Linux-Updates treffen zu den unterschiedlichsten Uhrzeiten ein. Es gibt auch keine langfristig angekündigten Patch Days wie etwa bei Microsoft. Um den laufenden Betrieb nicht zu gefährden, müssen Updates idealerweise vor der eigentlichen Installation auf den Produktivsystemen auf separaten Maschinen getestet werden. Das ist einer der Gründe, der gegen die automatische Installation von Updates spricht.

Bei einer derart großen Anzahl von Linux-Servern sind Administratoren regelmäßig mit der Frage konfrontiert, welche Systeme zuerst aktualisiert werden müssen. Dabei spielen kurzfristig bekannt gewordene Sicherheitslücken wie Heartbleed [SCHM 14] oder Shellshock [SCHE 14B] eine große Rolle. Das Ausspähen vertraulicher Informationen oder das unerlaubte Erlangen von Administratorrechten zu unterbinden ist schließlich wichtiger, als eine Sicherheitslücke zu stopfen, die nur unter unwahrscheinlichen oder speziellen Laborbedingungen ausgenutzt werden kann und weniger Schaden verursacht. Eine Priorisierung der Server-

Aktualisierungen können Administratoren auch nur dann sinnvoll durchführen, wenn sie genau wissen, welche verwundbare Software auf welchem Server installiert ist.

Damit Administratoren diese Entscheidungen sinnvoll treffen können, benötigen sie ein zentrales Informationssystem. Im einfachsten Fall sieht es folgendermaßen aus: Das System muss einerseits wissen, welche Programmversionen die jeweils aktuellsten sind. Diese Informationen bezieht es aus einem aktuellen Software Repository. Andererseits muss es von allen zu überwachenden Servern abfragen, welche Paketversionen dort installiert sind. Diese Informationen können direkt über die Verwaltungstools des Paketmanagementsystems abgefragt werden. Anhand eines einfachen Soll-Ist-Vergleichs kann das Informationssystem an zentraler Stelle bestimmen, ob Pakete auf einem Server veraltet sind oder nicht.

Bereits bei diesem einfach gehaltenen System stellt sich die Frage, wie es an die Paketlisten der Server gelangt. Eine Möglichkeit besteht darin, auf den Servern ein Programm oder Skript per Cronjob regelmäßig zu starten, das eine Liste der aktuell installierten Pakete zusammenstellt und zu dem zentralen Informationssystem sendet. Bei einer großen Anzahl von Systemen muss aber darauf geachtet werden, dass die Cronjobs nicht alle zur selben Zeit starten, um das zentrale System nicht plötzlich zu überfordern, sondern gleichmäßig auszulasten.

Eine Alternative dazu ist ein speicherresistenter Hintergrundprozess auf den Servern. Dieser Dienst wartet auf Anforderungen des zentralen Systems und versendet erst dann seine Paketliste, wenn ein Scheduling-Algorithmus auf dem zentralen System dies für angebracht hält. Paketlisten nur per Anforderung zu erhalten macht Sinn, um den Traffic im Netz und die Last des zentralen Systems möglichst gleichmäßig zu verteilen bzw. an einer zentralen Anlaufstelle steuern zu können. Das System fragt sinnvollerweise erst dann Informationen von den einzelnen Linux-Servern ab, sobald es Neuerungen in seiner eigenen Referenz-Paketliste gibt. Auf diese Weise kann unnötiger Traffic im Netz verhindert werden und die Linux-Server werden nur dann vom zentralen System beansprucht, wenn es unbedingt nötig ist.

Unabhängig davon, ob die Paketlisten vom zentralen Informationssystem angefordert oder aktiv von den zu überwachenden Servern versendet werden, sollte die Nachricht an den Server stets auch die Version des Skripts oder speicherresistenten Prüfprogramms selbst beinhalten. Das Prüfprogramm wird schließlich im Lauf der Zeit auch weiterentwickelt und muss deshalb auf allen Systemen auf dem aktuellen Stand sein.

Zu Test- oder Demonstrationszwecken anhand eines Prototyps genügt es aber auch, die Paketlisten einer Handvoll repräsentativer Server manuell beim zentralen Informationssystem einzuspielen. Dies erfolgt über ein spezielles Datei-Upload-Formular. Da der Schwerpunkt dieser Masterarbeit bei der Visualisierung von Serverdaten liegt, wird auf die Beschaffung der Paketdaten Daten nur theoretisch eingegangen.

Wie bereits in Abschnitt 4.2.2 erwähnt wurde, können die Daten, die von den Servern regelmäßig an das zentrale Informationssystem übertragen werden, theoretisch auch von Angreifern abgegriffen und missbraucht werden. Die Kenntnis über die Softwarestände der Server ist für Angreifer kostbar. Durch geeignete Sicherheitsmaßnahmen muss deshalb das unbefugte Ausspähen dieser Daten verhindert werden. Geeignete Maßnahmen wurden bereits in Abschnitt 3.4 angesprochen, dazu zählt die Transportverschlüsselung.

6.1.2. Mögliche Visualisierung

Die Anzahl der Soll-Ist-Vergleiche und dementsprechend der Resultate entspricht dem Produkt der Anzahl der zu überwachenden Server und der mittleren Paketanzahl. Bei mehreren hundert Servern bedeutet das viel Arbeit für den Administrator. Die Ergebnisse liegen zwar schon an zentraler Stelle vor, müssen aber durch Techniken der Informationsvisualisierung handhabbar gemacht werden. Bei diesem Vorhaben wird Ben Shneidermans *Information Seeking Mantra* [SHN 96] angewendet:

Information Seeking Mantra

> "Overview first, zoom and filter, then details on-demand"

Dieses Mantra lässt sich hervorragend anwenden, um mit den großen Datenmengen zurechtzukommen. Es beruft sich auf die Vorteile, die *Overview and Detail* (siehe Abschnitt 2.17.3), *zoombare Benutzeroberflächen* (siehe Abschnitt 2.17.2) und *Dynamic Queries* (siehe Abschnitt 2.14.3) bieten. Dem Anwender soll demnach

zunächst eine grobe Gesamtübersicht der Daten gezeigt werden. Aus dieser Gesamtheit wählt der Anwender durch Filterprozesse und das sukzessive Wechseln in immer detailliertere Ansichten die für ihn interessanten Komponenten aus der gesamten Menge aus. Schließlich lässt er sich konkrete Details einiger dieser Komponenten anzeigen, um die essentiellen Informationen zu erhalten. Dieses Prinzip lässt sich auf die Administration der Softwarestände von Linux-Servern übertragen:

Abbildung 6.2.: Ampel-Piktogramme mit allen möglichen Zuständen.

Einsatz von Signalfarben

Der Administrator soll zunächst eine grobe Übersicht über den Software-Stand der Systeme erhalten und nur mit den notwendigsten Informationen konfrontiert werden. Für den optimalen Fall, dass alle Linux-Server auf dem aktuellen Stand sind, muss ein Statusbildschirm nicht jeden einzelnen Server auflisten und als Zusatzinformation einblenden, dass alle Systeme auf dem aktuellen Stand sind. Das sind redundante Informationen, die weggelassen werden können. In diesem Fall genügt bereits ein einziges aber aussagekräftiges Piktogramm.

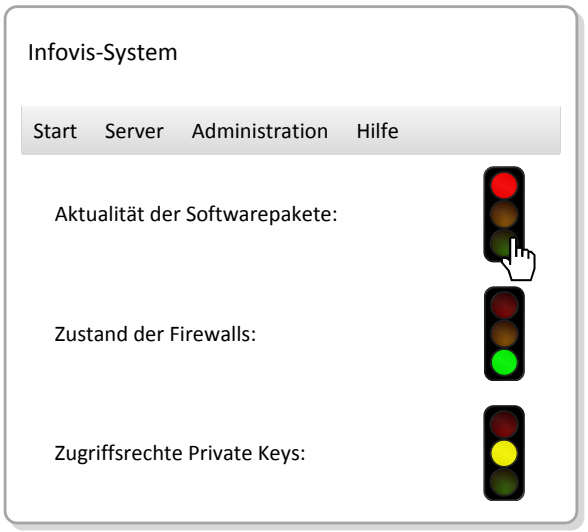


Abbildung 6.3.: Schematische Gesamtübersicht des Informationssystems.

Da die zugrundeliegenden Informationen auf einem Soll-Ist-Vergleich beruhen, ist ein Ampelsymbol (wie beispielsweise in Abbildung 6.2) ein naheliegendes Piktogramm. Es beansprucht nur wenig Platz auf dem Bildschirm und lässt somit genügend Raum für andere Komponenten der Statusübersicht, wie zum Beispiel den Zustand der Firewalls, frei.

Bereits beim ersten Kontakt mit dem Informationssystem erkennen Anwender intuitiv, welche tendenzielle Aussage hinter der Ampel steckt, da die Signallichter aus dem Alltag allseits bekannt sind. Nichtsdestotrotz ist eine Legende mit der genauen Bedeutung der Farben im Informationssystem unabdingbar, um Missverständnisse durch unterschiedliche individuelle Interpretationen auszuschließen.

Abbildung 6.3 zeigt den möglichen Aufbau einer solchen Statusübersicht. Solange die Ampel grün ist, kann der Administrator seine Aufmerksamkeit anderen Aufgaben oder Visualisierungen zukommen lassen. Andern-

falls wechselt er in eine Ansicht, die sich auf die Softwarestände beschränkt und detailliertere Informationen beinhaltet.

Im einfachsten Fall reichen die beiden Ampelfarben Rot und Grün aus, um die wichtigsten Zustände der Softwarestände darzustellen. Grün repräsentiert den optimalen Zustand, Rot eine Abweichung davon, also mindestens ein Paket ist veraltet. Durch die Hinzunahme der Farbe Gelb kann ein weiterer Zustand farblich kodiert werden. Die zusätzliche Farbe macht zum Beispiel Sinn, wenn nicht jedes veraltete Paket per se als problematisch eingestuft wird, sondern erst dann, wenn eine sicherheitskritische Schwachstelle zu dieser Paketversion bekannt geworden ist. Standardmäßig weist dann die Farbe Gelb auf die Tatsache hin, dass generell Updates vorhanden sind. Zur Beurteilung des Schweregrads einer Sicherheitslücke kann beispielsweise auf das Common Vulnerability Scoring System (CVSS) zurückgegriffen werden.

Die Farbe Rot bleibt am besten den kritischen Patches vorbehalten. Die präattentive Wahrnehmung (siehe Abschnitt 2.11) dieser Signalfarbe sorgt beim Anwender für unmittelbare Aufmerksamkeit. Es gibt nun einen konkreten Anlass, zu einer Ansicht zu wechseln, die sich auf die Softwarestände der Linux-Server beschränkt.

Man geht davon aus, dass die Farbe Rot im Lauf der Evolutionsgeschichte immer schon alarmierend auf den Betrachter wirkte. Sie darf nicht inflationär in Visualisierungen eingesetzt werden, da sie für besonders kritische Fälle reserviert ist und auf einzelne Besonderheiten hinweisen soll. Generell darf die Farbe Rot nur zu alarmierenden Zwecken verwendet werden, da sie sonst gänzlich ihre warnende Wirkung verliert [MOHR 11, S. 104 f.]. Der bedachte Einsatz von Rot sorgt zudem für einen entspannteren Umgang mit dem Informationssystem (siehe Abschnitt 2.13.3).

Server-Statistiken

Bevor der Anwender einzelne Server näher betrachtet, will er sich in der Regel allgemeine Statusinformationen ansehen, die alle beteiligten Systeme betreffen. So kann er auf einen Blick sehen, wie ernst die Gesamtsituation ist. Der IT-Sicherheitsexperte *Andrew Jaquith* empfiehlt als Grundlage hierfür unter anderem die folgenden Kennzahlen [JAQ 07]:

- Der relative Anteil der Server, die nicht auf dem aktuellen Softwarestand sind.
- Die Anzahl unterschiedlicher Pakete, die noch nicht gepatcht sind (Mittelwert und Median aller betroffenen Systeme).
- Der relative Anteil ungepatchter Pakete pro Server.

Der Anteil ungepatchter Server ist deshalb von Interesse, da er Auskunft über die allgemeine Patch-Situation Rechnernetz gibt. An ihm lässt sich ablesen, wie dringend generell Handlungsbedarf besteht, ohne sich konkrete Systeme oder Serverkategorien anzusehen. Dieser Wert kann auch als Indikator für die Leistung der Administratoren herangezogen werden. Anhand der als zweiten Punkt genannten Kennzahl können Administratoren grob abschätzen, wie groß der Aufwand ist, einen einzelnen Server zu patchen. Punkt drei ist von Interesse, wenn Server im Detail angesehen werden.

Die Kennzahlen, die relativen Charakter haben, können im einfachsten Fall als Tortendiagramme angezeigt werden. Falls aber auch die absoluten Werte von Interesse sind, sind Tortendiagramme ungeeignet und stattdessen standardmäßig Säulendiagramme im Informationssystem zu verwenden. Es spricht aber nichts dagegen, dem Anwender die Wahl zu lassen, sich optional ein Tortendiagramm einblenden zu lassen. Abbildung 6.4 zeigt ein Beispiel für beide Darstellungsformen anhand des Patch-Standes von insgesamt 25 Linux-Servern.

In den bisherigen Visualisierungen wurden nur die beiden Zustände *gepatcht* und *nicht gepatcht* berücksichtigt. Ungepatchte Systeme können aber noch zusätzlich unterschieden werden: Entweder ist mindestens ein Paket vorhanden, dessen veraltete Version als kritisch eingestuft wurde oder nicht. Die Diagramme können also um einen dritten Zustand erweitert werden.

Analog zu den zuvor genannten Ampeln bietet sich hier eine Einfärbung der entsprechenden Diagrammkomponenten in Ampelfarben an. Abbildung 6.5 zeigt die gleichen Diagramme wie zuvor unter Berücksichtigung des dritten Zustands und der Signalfarben.

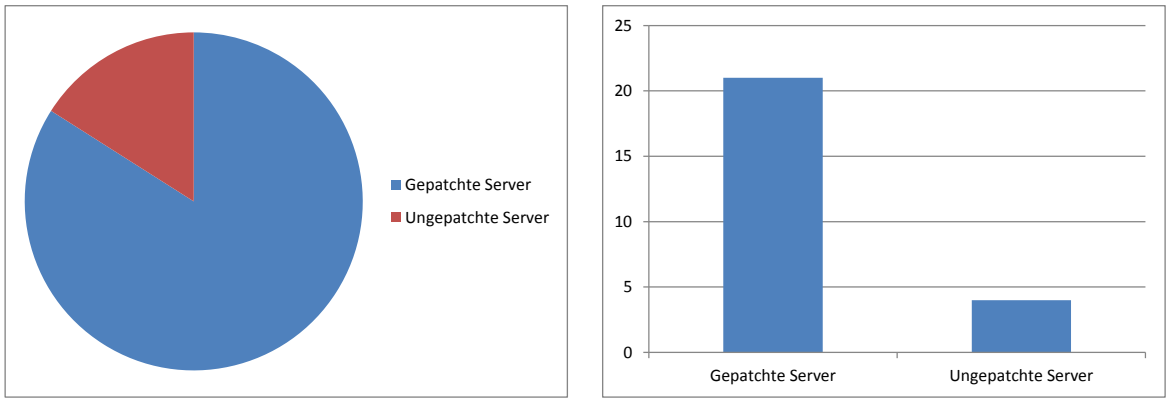


Abbildung 6.4.: Gepatchte und ungepatchte Server als Tortendiagramm (links) und als Säulendiagramm (rechts).

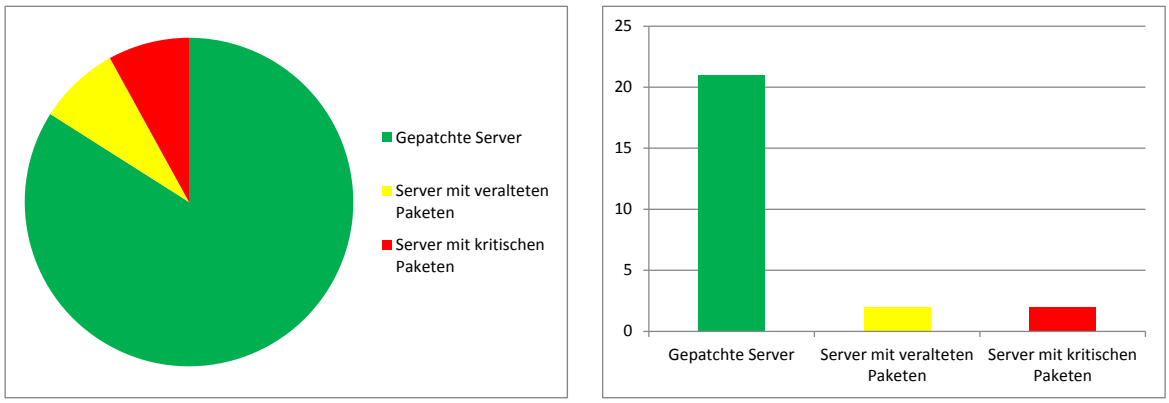


Abbildung 6.5.: Gepatchte Server, Server mit veralteten Paketen und Server mit kritischen Paketen als Tortendiagramm (links) und als Säulendiagramm (rechts).

Wenn das Informationssystem eine Kategorisierung der unterschiedlichen Servertypen (Webserver, Datenbankserver etc.) unterstützt, ist es denkbar dass auch die Server-Statistiken für einzelne Kategorien abgefragt werden können. Diese Funktion ist von Nutzen, wenn sich der Administrator zum Beispiel besonders für den allgemeinen Patch-Stand der Webserver interessiert, nachdem eine kritische Lücke bekannt wurde. Bei den Webservern muss besonders schnell auf kritische Lücken reagiert werden, da dort ein besonders hohes Bedrohungspotenzial vorhanden ist: Sie werden von einer breiten Masse genutzt und verfügen in der Regel über eine schnelle Anbindung ans Internet.

Es ist auch denkbar, dass Server bestimmter Kategorien von unterschiedlichen Administratorenteams betreut werden. Anhand einer Gegenüberstellung der Patchzustände einzelner Kategorien kann auch ein Rückschluss darauf gezogen werden, wie schnell die Teams auf Sicherheitslücken in den Programmpaketen reagieren. Wenn in der visuellen Gegenüberstellung der Patchzustand einer Kategorie besonders negativ ausfällt, weiß der Anwender sofort, welches Team er am besten auf diesen Zustand hinweisen muss.

In einem kleineren Unternehmen mit einer Handvoll Administratoren mag die Ermittlung der Personen, die für eine Serverkategorie verantwortlich sind, einfach fallen. Damit dieser Prozess auch in einem großen Unternehmen keine Schwierigkeit darstellt, kann auf die Datenbank des Informationssystems zurückgegriffen werden: Es ist sinnvoll, zu jeder Kategorie die jeweiligen Ansprechpartner abzuspeichern. Das Informationssystem muss nicht ausschließlich zu dem Zweck zu benutzt werden, um die nachlässigsten Administratoren zu bestimmen. Es kann auch einen Anreiz schaffen, auf Sicherheitslücken schneller zu reagieren, um dann einen Bonus dafür zu erhalten.

Die Statistiken der einzelnen Serverkategorien können mit den gleichen Methoden visualisiert werden wie die allgemeinen Statistiken. Um den allgemeinen Zustand mehrerer Kategorien miteinander vergleichen zu können, gibt es verschiedene Möglichkeiten: Die einzelnen Diagramme können nebeneinander oder untereinander angeordnet werden. Alternativ dazu können sie auch zu einer einzigen Grafik kombiniert werden. Dazu eignen sich zum Beispiel gestapelte Säulendiagramme.

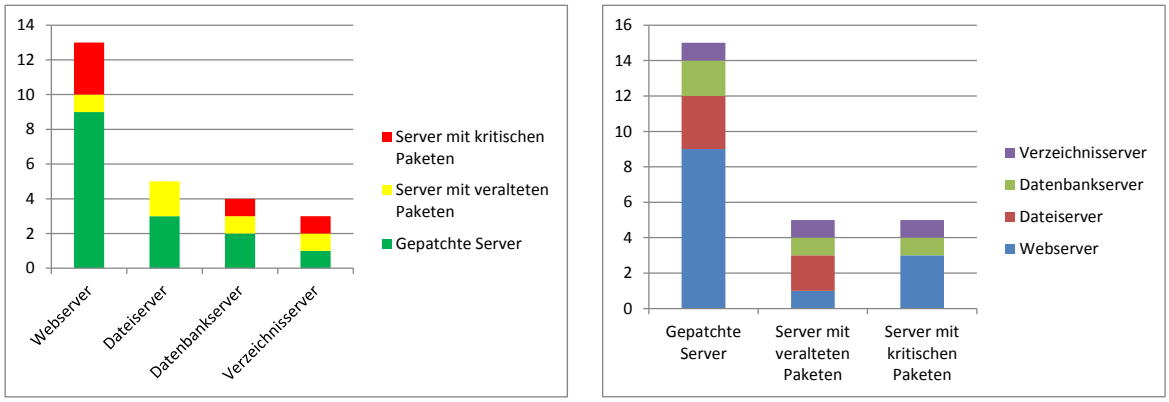


Abbildung 6.6.: Gepatchte Server, Server mit veralteten Paketen und Server mit kritischen Paketen unter Berücksichtigung von Kategorien mit unterschiedlichen Achsbeschriftungen.

Abbildung 6.6 zeigt Beispiele für die Visualisierung der Patch-Stände unter Einbeziehung der unterschiedlichen Server-Kategorien. Im linken Teilbild besitzt jede Kategorie eine eigene gestapelte Säule zur Unterscheidung der Zustände *gepatchter Server*, *Server mit veralteten Paketen* und *Server mit kritischen Paketen*. Da die Kategorien nebeneinander angeordnet sind, ist diese Darstellungsform besonders dazu geeignet, die Zustände der Kategorien miteinander zu vergleichen.

Im zweiten Teilbild sind die Werte aller Server gestapelt und es existiert je eine Säule für die drei verschiedenen Patch-Zustände. Durch die mehrfache Stapelung fällt es dem Betrachter hier etwas schwieriger, die Anzahl der Server der einzelnen Kategorien herauszulesen. Diese Form der Visualisierung bietet aber den Vorteil, dass man schneller sieht, wie viele Server generell gepatcht bzw. nicht gepatcht sind oder Software mit kritischen Lücken besitzen. Sie vereint den Vorteil der Visualisierung ohne Kategorien mit dem der Visualisierung mit Kategorien. Mit Hilfe von Tooltip-Texten kann das Problem der schlechten Erkennbarkeit der Werte einzelner Stapelelemente gelöst werden.

Die in diesem Abschnitt vorgeschlagenen Diagramme für Serverstatistiken müssen nicht auf ihren zusammenfassenden Charakter begrenzt sein. Noch nützlicher werden sie, indem der Nutzer auch die einzelnen Server betrachten kann, die sich hinter einem Balken oder Kreissegment verbergen. Es ist deshalb denkbar, Event Handler an die einzelnen Elemente zu binden, sodass der Anwender durch einen Klick ins Diagramm unmittelbar die entsprechenden Server in einer Liste oder in der zoombaren Bedienoberfläche, die im Folgenden vorgestellt wird, angezeigt bekommt. Es bleibt ihm so erspart, erneut nach ungepatchten Servern einer bestimmten Kategorie in einer anderen Ansicht zu suchen und er kann so schneller auf sicherheitskritische Ereignisse reagieren.

Gesamtübersicht

Sobald die Aufmerksamkeit des Administrators auf den Bereich der Softwarestände gelenkt wurde, bekommt er im nächsten Schritt eine Gesamtübersicht der Linux-Server zu sehen, deren Paketversionen überwacht werden (Stichwort “overview first”). Sie enthält keine Details, lässt aber durch weitere Ampelsymbole erkennen, wo Handlungsbedarf herrscht.

Im Ausgangszustand beinhaltet die Ansicht alle verfügbaren Server in Form von kleinen Piktogrammen, die auf einer zweidimensionalen Fläche angeordnet sind. Dieses Schema skaliert einerseits besser als beispielsweise eine Listenansicht. Andererseits ist eine bessere Übersichtlichkeit gegeben als zum Beispiel bei einer dreidimensionalen wolkenähnlichen Ansicht, in der die Navigation schwierig fällt. Eine Baumstruktur würde

ebenfalls bei großer Serveranzahl gut skalieren, es müssten aber alle Ordner vollständig aufgeklappt werden, um alle Kindknoten bzw. Server zu sehen. Das ist nicht im Sinne einer Gesamtübersicht.

Die Piktogramme müssen in der Gesamtübersicht optisch möglichst einfach gehalten sein, denn sie dienen nur dazu, einzelne Server voneinander zu unterscheiden. Die Verwendung von Glyphen ist an dieser Stelle denkbar, um beispielsweise die Funktion der Server oder – falls so etwas vorhanden ist – ihre Priorität sofort erkennen zu können. Die Funktion lässt sich als kleines Zusatzsymbol abbilden, zum Beispiel in Form einer Weltkugel für einen Webserver oder einem Aktenordner für einen Dateiserver.

Um das Verdecken einzelner Piktogramme zu vermeiden, ist eine weitere Vereinfachung der Piktogramme denkbar. Anstelle von Server-Piktogrammen mit einem angehängten Symbol für die Serverkategorie können im einfachsten Fall auch Kreise verwendet werden. Der Radius könnte als visuelles Attribut eingesetzt werden. Falls sich bei einer bestimmten Detail-Stufe dennoch einzelne Piktogramme von anderen verdeckt werden, könnten diese durch eine Art *Point-Displacement-Algorithmus* (siehe Abschnitt 2.12.7) so weit verschoben werden, bis die Überlappung aufgehoben ist. Praktisch wäre es auch, wenn der Anwender in Problemfällen verdeckte Piktogramme durch Verschiebeoperationen mit der Maus freilegen könnte.

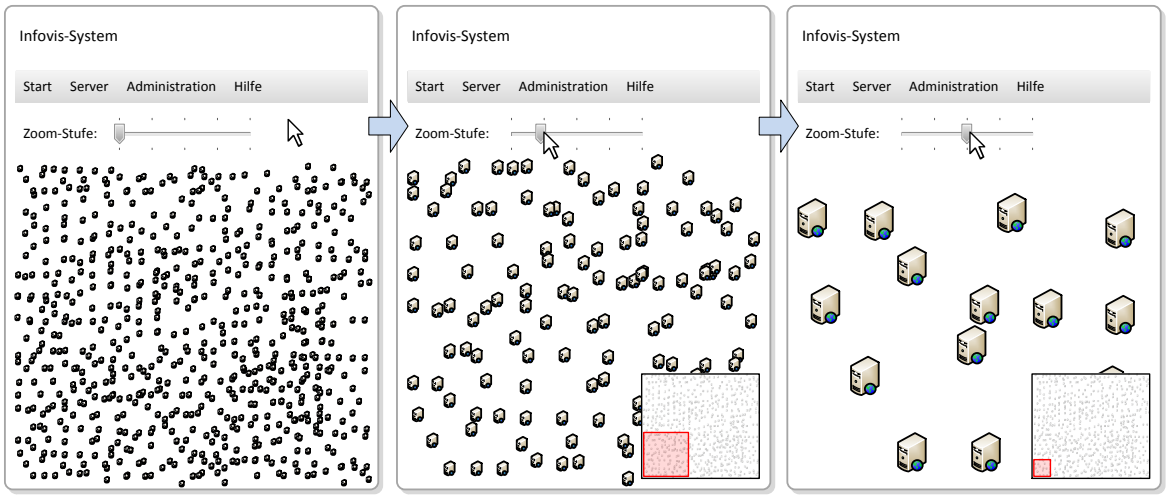


Abbildung 6.7.: Prinzip einer zoombaren Bedienoberfläche basierend auf Overview and Detail.

Zoombare Bedienoberfläche

Dem Anwender muss die Möglichkeit gegeben sein, zwischen unterschiedlichen Detailgraden zu wählen. Die wenigsten Details zeigt die Gesamtübersicht an. Je kleiner ein Ausschnitt der betrachteten Server ist, desto mehr Details können angezeigt werden. Einerseits weil dann mehr Platz auf dem Bildschirm vorhanden ist, andererseits weil der Anwender die Detailansicht bewusst gewählt hat, um mehr Informationen zu den betrachteten Servern zu erhalten.

Es gibt zwei Möglichkeiten, unterschiedliche Detailgrade in der Anwendung zur Verfügung zu stellen: Die eine besteht darin, eine feste Anzahl von Vergrößerungsgraden anzubieten. Der zweite Ansatz basiert auf einer Bedienoberfläche, die dem Anwender ein stufenloses herein- und herauszoomen erlaubt. Eine stufenlos zoombare Bedienoberfläche (siehe Abschnitt 2.17.2) ist zwar komplexer zu implementieren und hardwarehungriger, ermöglicht dem Anwender aber eine intuitivere Bedienung. Wenn er stufenlos zoomt, kann er sich besser in der Übersicht zurechtfinden, als beim abgehackten Umschalten zwischen wenigen vorgegebenen Vergrößerungsgraden. Durch die ästhetische Illusion, in die Übersicht regelrecht eintauchen zu können, nimmt der *Immersionseffekt* zu [WOLF+ 13].

Overview and Detail

Um den Anwender bei der Positionsbestimmung innerhalb der Übersicht zu unterstützen, kann ein *Overview-and-Detail*-Ansatz (siehe Abschnitt 2.17.3) mit einem zusätzlichen Detail-Fenster verfolgt werden. Dieses Fenster ist im Idealfall verschiebbar und bei Bedarf auch ausblendbar, um die Objekte in der Gesamtübersicht nicht zu verdecken.

Die Überlagerung der Elemente hinter dem Fenster kann alternativ auch durch Transparenz verhindert werden. Welcher Ansatz der bessere ist, ist Geschmackssache. Am besten wird beides implementiert, damit der Anwender die Wahl hat. Abbildung 6.7 zeigt, wie eine auf Overview and Detail basierende Übersicht der Softwarestände im Prinzip aussehen könnte.

Räumliche Anordnung durch Gruppierung

Die Anordnung und Verteilung der einzelnen Server in der Gesamtübersicht könnte entweder zufällig oder nach einem bestimmten Schema erfolgen. Eine zufällige Anordnung hat den Nachteil, dass sich der Administrator bei jedem Aufruf dieser Ansicht erneut darin zurechtfinden muss. Für die Orientierungsphase geht wertvolle Zeit verloren. Es ist also sinnvoller, ein System in die Anordnung der Server zu bringen.

Der zweidimensionale Raum kann zum Beispiel genutzt werden, um Server logisch zu gruppieren. Es ist denkbar, Gruppen anhand der Priorität der Systeme oder anhand der Software zu bilden, die auf den Maschinen installiert ist. Das linke Teilbild von Abbildung 6.8 zeigt ein Beispiel für diese Art der Gruppierung. Auf den Servern in der Gruppe oben links wird *OpenSSL* als Kryptobibliothek verwendet, auf den anderen *GnuTLS*. Um eine Gruppierung zu verdeutlichen, kann auf Grundlagen der Gestalt-Theorie zurückgriffen werden (siehe Abschnitt 2.11.2). Räumliche Nähe, Verbindungen und Rahmen um die Piktogramme sind wirksame Methoden zur visuellen Gruppierung.

Um dem zuvor genannten Mantra gerecht zu werden, müssen die Inhalte der Gruppen nicht von Anfang an sichtbar sein. Wenn beispielsweise ein Server innerhalb der Gruppe *openssl* verwundbare Software enthält, wird zunächst die gesamte Gruppe optisch hervorgehoben, ohne dass weder der betroffene Server noch irgendein anderer in der Gesamtübersicht zu sehen ist. Nachdem die Gruppe die Aufmerksamkeit des Anwenders auf sich gezogen hat, kann er die Gruppe öffnen und den Zustand der darin enthaltenen Server abfragen. Indem unnötige und redundante Elemente von Anfang an nicht zu sehen sind, wird eine hohe Übersichtlichkeit gewährleistet.

Die Gruppierung von Servern muss nicht auf eine einzige hierarchische Ebene beschränkt sein: Mit Hilfe von Baumstrukturen in einer Datenbank lassen sich auch beliebig komplexe Enthaltenseinsbeziehungen abbilden. In einer dazugehörigen Visualisierung arbeitet sich der Anwender durch sukzessives Wechseln der Ansicht zu den Blattknoten vor, die die einzelnen Server darstellen. Dort angelangt sieht der Anwender dann, welche konkreten Pakete Aktualisierungen benötigen.

Mapping von Attributen auf Abszisse und Ordinate

Die zweidimensionale Fläche der Gesamtübersicht ermöglicht es, in die Position der einzelnen Serverpiktogramme zusätzliche Informationen zu kodieren. Der Anwender hat die Möglichkeit, zwischen mehreren Bedeutungen von Abszisse und Ordinate umschalten, damit die Server in der Übersicht nach unterschiedlichen Kriterien sortiert werden. Als eine Kategorie ist zum Beispiel die Priorität der Server bzw. die der dazugehörigen Sicherheitszone denkbar.

Im Kontext der Softwarestände ist es naheliegend, auch den Zeitpunkt der letzten Aktualisierung auf eine Achse abbilden zu können. So sieht der Anwender sofort, bei welchen Systemen dringender Handlungsbedarf besteht. Diese Entscheidung kann der Administrator anhand der Visualisierung sogar dann treffen, wenn keine Referenzdaten vorliegen, mit denen die Softwarestände der einzelnen Linux-Server verglichen werden könnten.

Abbildung 6.8 zeigt ein solches Beispiel im rechten Teilbild. Der Server ganz oben rechts stellt einen Outlier dar, der aufgrund seiner Position außerhalb der Häufung sofort ins Auge sticht. Es handelt sich in die-

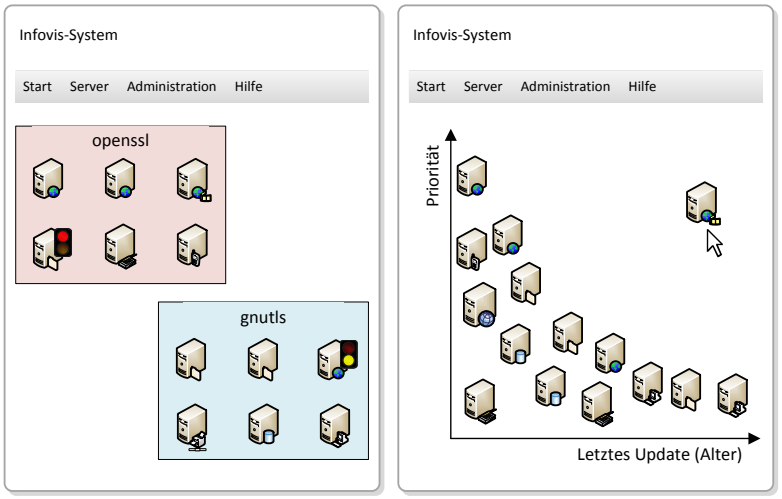


Abbildung 6.8.: Gruppierung von Servern anhand von Sicherheitszonen (links) und Anordnung anhand von letztem Update-Zeitpunkt und Priorität (rechts).

sem Fall offensichtlich um einen Streaming Server, der von sehr hoher Priorität ist, aber schon lange keine Update-Installation mehr erhalten hat. Selbst ohne vorliegende Referenzdaten über die zur Zeit aktuellen Softwarestände erkennt der Administrator sofort, dass dort Handlungsbedarf besteht.

Nicht jedes Software-Paket im Repository eines Linux-Distributors wird in regelmäßigen kurzen Abständen gepatcht. Für einen Server, der nur wenig Software einer Stable- oder Oldstable-Distribution enthält, stehen oft für relativ lange Zeit überhaupt keine Updates zur Verfügung. Aus diesem Grund darf nicht (nur) der Zeitpunkt der tatsächlich erfolgten Patch-Installation als Grundlage für eine Sortierung anhand der letzten Aktualisierung verwendet werden. Die Visualisierung würde sonst den falschen Eindruck vermitteln, dass die Software auf dem Server lange Zeit vernachlässigt wurde. Stattdessen muss visualisiert werden, wann das Update Tool zum letzten Mal gestartet und erfolgreich beendet wurde, auch wenn es nichts zu patchen gab.

Damit sich der Anwender besser in der räumlichen Anordnung der Server zurechtfinden kann, bietet es sich an, dass er per Direct Manipulation (siehe Abschnitt 2.14.2) die Position der Piktogramme nach seinen eigenen Bedürfnissen verändern kann. Ein derart personalisiertes Positionsschema kann abgespeichert und zu beliebigen Zeitpunkten wieder aufgerufen werden. Dies stellt eine Ergänzung zu den zuvor genannten Schemas dar.

Filterung

Als Alternative oder Ergänzung zu der Gruppierung der Server oder der auf Achsen basierten Positionierung bietet es sich an, die Systeme auch mit individuellen Metadaten in Form von Tags bzw. Stichwörtern zu versehen. Mit Hilfe von Dynamic Queries kann der Anwender anhand der gezielten Auswahl einzelner oder mehrerer Tags die Anzahl der auf der Gesamtübersicht angezeigten Linux-Server sukzessive verringern, bis nur noch die interessanten Systeme übrig bleiben.

Sinnvoll ist auch eine Filterfunktion, die lediglich diejenigen Systeme inklusive eines entsprechenden Ampel-Piktogramms anzeigt, bei denen Handlungsbedarf besteht. Wenn kritische Fehler auf Servern existieren, interessieren den Administrator die nicht betroffenen Systeme vorerst nicht. Sie stellen Visual Clutter dar und verringern die Übersichtlichkeit. Aus diesem Grund können sie ausgeblendet werden.

Sobald alle nicht betroffenen Systeme ausgeblendet sind, ist es denkbar, reduzierte Ampel-Piktogramme für die Server mit veralteter Software zu verwenden. Die Farbe Grün wird nach dem gerade erfolgten Filterschritt schließlich nicht mehr benötigt und kann weggelassen werden. Dies führt zu einem weiteren Abbau von Visual Clutter und erhöht die Übersichtlichkeit.

Abbildung 6.9 zeigt zwei solche Filterschritte mit reduzierten Piktogrammen. Im linken Bild sind noch alle

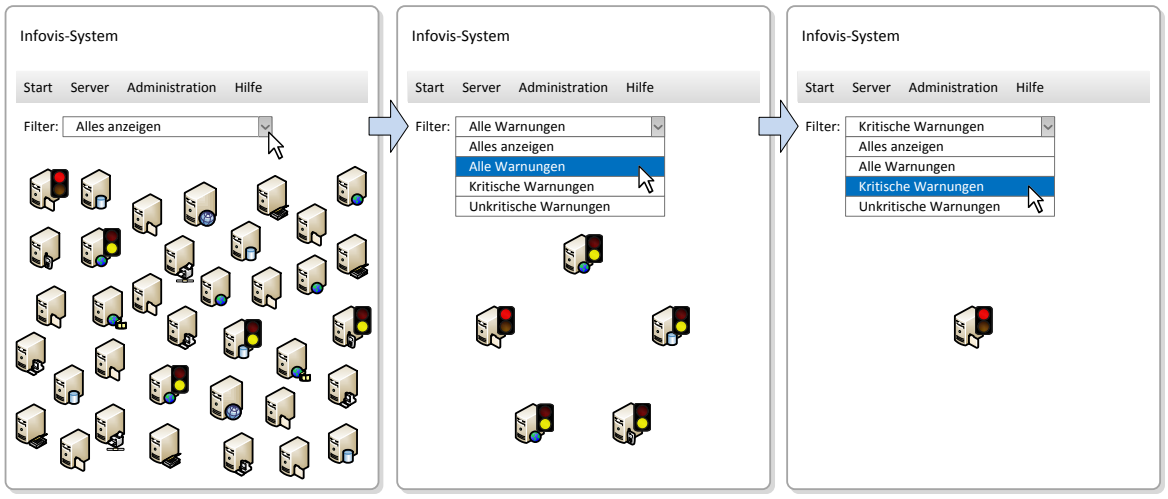


Abbildung 6.9.: Schrittweise Filterung von Linux-Servern anhand von Software-Update-Warnstufen.

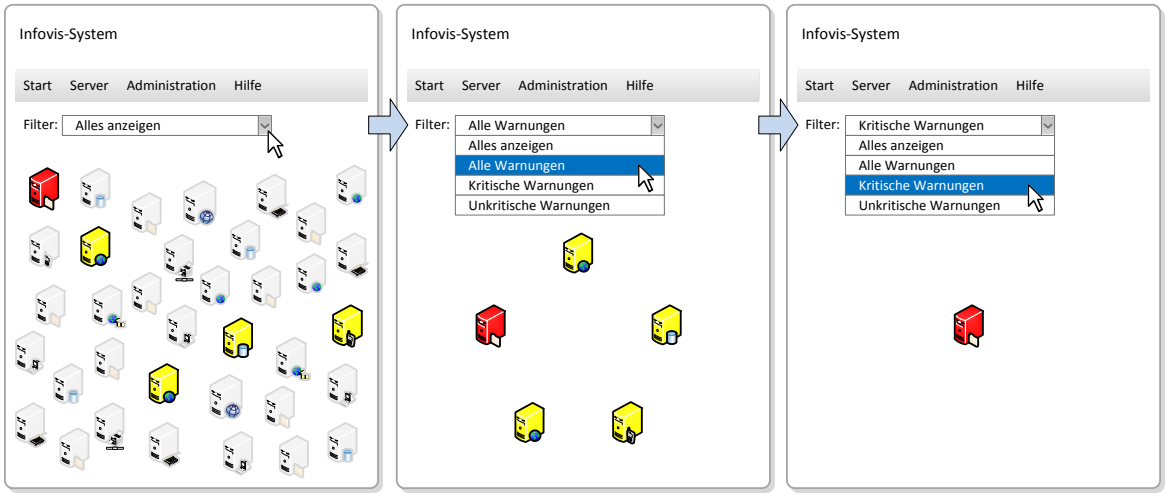


Abbildung 6.10.: Schrittweise Filterung von Linux-Servern anhand farbkodierter Software-Update-Warnstufen.

Server zu sehen. Das mittlere Bild zeigt nur noch diejenigen Server an, die veraltete Software-Pakete besitzen. Das rechte Bild beinhaltet nur noch die Server, auf denen Programme installiert sind, zu denen sicherheitskritische Exploits bekannt sind.

Benutzerdefinierbare Ansichten

Zuvor wurde erwähnt, dass es für den Anwender hilfreich ist, die Positionen der nach seinen Bedürfnissen angeordneten Server-Piktogramme zu speichern, um sie später wiederherstellen zu können. Diese Speicherfunktion sollte idealerweise auch die Filter berücksichtigen, die der Benutzer ausgewählt hat. Er kann sich so verschiedene Ansichten mit häufig benötigten Übersichten zusammenstellen, die per Mausklick sofort abrufbar sind. Dadurch spart er Zeit und kann sich besser auf seine Aufgaben konzentrieren.

Durch die Möglichkeit, mehrere benutzerdefinierte Ansichten abzuspeichern und wieder abzurufen wird das Informationssystem ein Stück mehr Ben Shneidermans Information Seeking Mantra gerecht. In der inhaltlich reduzierten Ansicht, in der die Piktogramme bereits nach einem bestimmten Schema angeordnet sind, kann der Anwender immer noch dieselben Schritte durchführen, die das Mantra propagiert, er kommt allerdings noch schneller zum Ziel, da er vor dem Laden der entsprechenden Ansicht schon Vorarbeit geleistet hat.

Verringerung von Visual Clutter

Um Visual Clutter weiter zu verringern, bietet es sich an, die Ampel-Piktogramme komplett wegzulassen und stattdessen die Warnstufen auf eine andere Art zu kodieren. Dazu eignet sich zum Beispiel der Farbton der Server-Piktogramme (siehe Abbildung 6.10). Server mit aktuellem Softwarestand erhalten eine unaufdringliche Farbe, während gefährdete Server rot oder gelb eingefärbt werden.

Bei diesem Vorgehen muss allerdings ein guter Kontrast sichergestellt sein, sodass sich alle Piktogramme einerseits gut vom Hintergrund abheben, andererseits die Warnfarben sofort ins Auge stechen. Hier ist es empfehlenswert, die zuvor genannten Ampelfarben zu verwenden.

Um den Anteil an Visual Clutter von Anfang an gering zu halten, bietet es sich an, die Anzahl der in der Übersicht eingeblendeten Elemente vom Zoomgrad abhängig zu machen. Je weiter der Anwender in die Übersicht hineinzoomt und je weniger Server-Piktogramme dadurch im Anzeigefenster übrig bleiben, desto mehr Zustandsinformationen können direkt in der Übersicht in der Nähe des Servers eingeblendet werden. Der Anwender müsste diese Informationen dann nicht für jeden Server einzeln per Klick darauf abrufen. Das zugrunde liegende Prinzip wurde im Abschnitt 2.12.7 als *konstante Informationsdichte* bezeichnet.

Detailansicht

Um dem letzten Teil von Shneidermans Mantras “details on-demand” gerecht zu werden, muss der Anwender nach dem Filterprozess oder nach der Alarmierung durch eine rote oder gelbe Ampel von einzelnen Systemen Detailinformationen abrufen können. Im Anwendungskontext der Softwarestände will der Anwender wissen, welche Software auf einem konkreten System installiert ist. Dies geschieht durch einen Klick auf eines der Systeme, wodurch in eine andere Detailansicht gewechselt wird.

Ein Linux-Server beinhaltet nicht nur die installierten Server-Anwendungen, wie zum Beispiel Datenbank- oder Webserver, sondern auch zahlreiche Abhängigkeiten. Einen Großteil macht die Standardsoftware aus, die unabhängig vom Anwendungszweck auf einem Linux-Server mit installiert wird. Dazu zählen zum Beispiel alle Kommandozeilentools, die ein Portable Operating System Interface (POSIX)-konformes System enthalten sollte. Die Liste der installierten Pakete fällt daher ziemlich lang aus und sollte nur nach einer weiteren Anforderung durch den Anwender komplett angezeigt werden. Sie sollte sich nach den Anforderungen des Anwenders sortieren lassen. Generell sollten standardmäßig nur diejenigen Pakete aufgelistet werden, die Updates benötigen oder sicherheitsbedenklich sind.

Eine Information, die in Zusammenhang mit diesem Anwendungsfall in der Detailansicht auf jeden Fall auftauchen muss, ist der Zeitpunkt des letzten Updates. Die Anzahl der installierten Pakete ist eine weitere nützliche Information. Sie dient als Indikator für die generelle Angriffsfläche des Servers, die sich mit der Menge an installierten Programmen vergrößert. Angaben über den Festplattenplatz und die derzeitige Belegung sind ebenfalls nützliche Informationen.

Die Software-Detailansicht eines Servers dient nicht nur für reaktive Zwecke, um veraltete Paketversionen zu sehen, sondern auch zur Vorabinformation. Um sich gezielt über die auf einem Server installierte Software informieren zu können, sollte die Detailübersicht die Möglichkeit bieten, Software anhand ihrer Kategorien zu unterscheiden. Der Einsatz von geläufigen Piktogrammen trägt zur Verbesserung der Übersichtlichkeit bei.

Als Alternative zum Umschalten in eine dedizierte Detailansicht bietet es sich an, nach dem Klick auf ein Server-Piktogramm Detailinformationen in einem nichtmodalen Fenster anzuzeigen. Dieses Fenster kann der Anwender frei auf dem Bildschirm positionieren, sodass es keine wichtigen Bereiche in der Visualisierung verdeckt. Falls der Ansatz mit dem Detailfenster verfolgt wird, gibt es zwei prinzipielle Möglichkeiten, wie die Informationen mehrerer nacheinander angeklickter Server dargestellt werden sollen: Entweder wird für jeden Server ein eigenes Detailfenster angezeigt, das solange sichtbar ist bis es geschlossen wird oder es ist immer nur ein Fenster sichtbar, das mit den Detailinformationen des Servers aktualisiert wird, der zuletzt angeklickt wurde.

Beide Varianten haben Vor- und Nachteile: Mehrere gleichzeitig geöffnete Fenster benötigen viel Platz auf dem Bildschirm und setzen einen Bildschirm mit hoher Auflösung voraus. Außerdem verdecken Sie Bereiche der zoombaren Bedienoberfläche. Bei einem einzigen Fenster bestehen diese Probleme nicht. Allerdings fallen

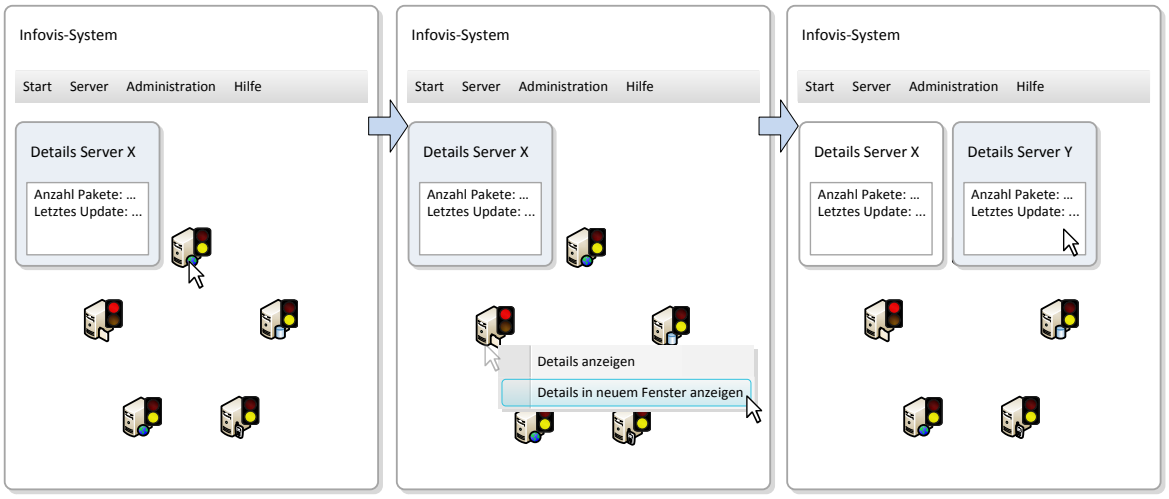


Abbildung 6.11.: Abruf von Server-Detailinformationen in separaten Fenstern.

dem Anwender dann Vergleiche zwischen den Zustandsinformationen zweier Server schwer, da er immer zwischen den beiden umschalten muss. Ein sinnvoller Kompromiss besteht darin, dass beim Klick auf einen Server nur dann ein weiteres Detailfenster geöffnet wird, wenn der Anwender gleichzeitig eine bestimmte Taste drückt oder im Kontextmenü des Server-Piktogramms diese Option explizit auswählt (siehe Abbildung 6.11).

6.1.3. Unterstützung proaktiver Tätigkeiten

Der Einsatzzweck des Informationssystems muss sich nicht darauf beschränken, dass der Anwender nur nach Systemen sucht, bei denen konkreter Handlungsbedarf herrscht. Er kann es auch nutzen, um sich persönlich einen besseren Überblick über die Konfiguration der vielen Linux-Server zu verschaffen und anhand dieser Informationen proaktiv tätig zu werden.

Suche nach spezifischen Paketversionen

Mit der Hilfe von Dynamic Queries kann der Anwender zum Beispiel in kurzer Zeit überprüfen, auf welchen Servern bestimmte Programme oder Programmversionen installiert sind. Diese Funktion kann dabei behilflich sein, um Server ausfindig zu machen, die Kandidaten für eine Konsolidierung sind. Überflüssige Software lässt sich so ebenfalls auf den Systemen aufspüren, um sie im Anschluss zu deinstallieren. Diese Maßnahmen sparen insgesamt nicht nur Speicherplatz, sondern verringern auch die Verwundbarkeit.

Abbildung 6.12 zeigt die Sequenz einer Anwendung von Dynamic Queries anhand der Suche aller Linux-Rechner, die den Webserver *nginx* installiert haben. Das Filterfeld verfügt über eine Autovervollständigungsfunktion, ähnlich wie es beim Suchformular der Google-Suchmaschine der Fall ist. Während der Anwender einen Filtertext Zeichen für Zeichen eingibt, verringert sich die Anzahl der übrigen Server automatisch. Die Anwendung blendet nach jedem Tastendruck Stichwörter als Suchvorschläge ein, die den Suchtext enthalten. Aus diesen kann der Anwender wählen, um nicht das komplette Wort eingeben zu müssen.

Das linke Bild zeigt eine Übersicht aller möglicher Linux-Server, im mittleren Bild sind nur noch Server mit Paketen zu sehen, die die Zeichenkette `ngi` enthalten, im rechten Bild bleiben schließlich nur noch nginx-Webserver übrig. Diese kann der Anwender dann anklicken, um weitere Details abzufragen. Durch die sukzessive Reduzierung der Suchergebnismenge findet der Anwender Schritt für Schritt zum Ziel.

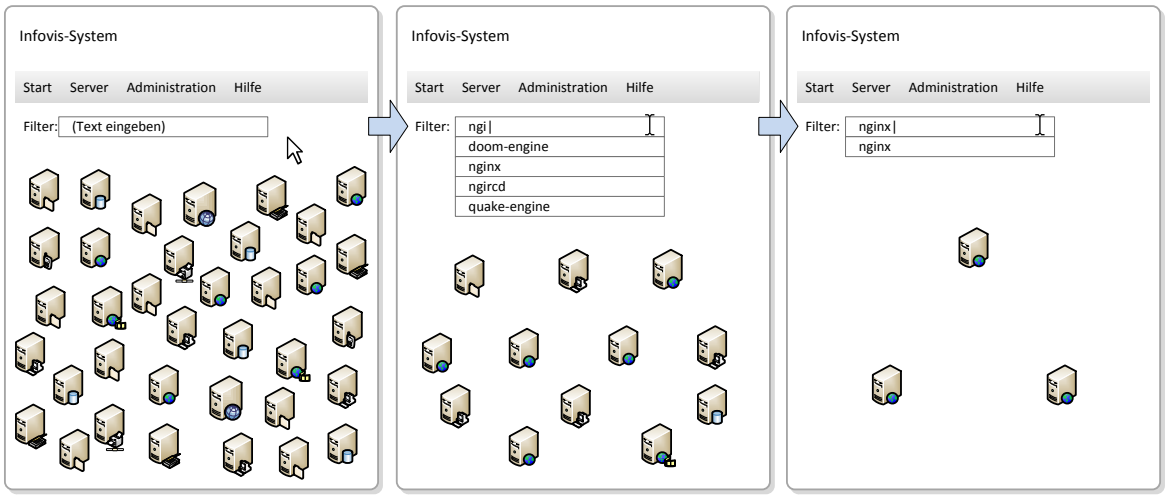


Abbildung 6.12.: Schrittweise Filterung von Linux-Servern anhand installierter Softwarepakete.

Wildwuchs bekämpfen

Nicht nur die Suche nach bestimmten Paketversionen kann zur Planung proaktiver Tätigkeiten nützlich sein. Um Systeme mit potenziellem Handlungsbedarf ausfindig zu machen, ist auch die Anzahl der darauf installierten Pakete ein nützlicher Indikator. Ein Server, auf dem besonders viele Pakete installiert sind, ist potenziell anfälliger für Angriffe als ein Minimalsystem, das nur einem konkreten Zweck dient.

Um die Anzahl der zu einem bestimmten Zeitpunkt auf einem Server installierten Pakete zu ermitteln, kann das Informationssystem auf die Paketliste zugreifen, die in seiner zentralen Datenbank gespeichert ist. Es genügt, die Anzahl der vorhandenen Datensätze zu zählen. Dies geschieht für jeden registrierten Server. Mit diesen gewonnenen Daten als Grundlage zeichnet das Informationssystem dynamisch generierte Diagramme, mit deren Hilfe sich der Administrator einen groben Überblick über die Verteilung der Anzahl installierter Pakete verschaffen kann.

Abbildung 6.13 zeigt beispielsweise ein Histogramm, das aus den Zahlen von 25 Debian-Servern aus einem Firmennetz generiert wurde. Diesem ist zu entnehmen, dass am häufigsten zwischen 500 und 550 Pakete installiert sind. In einem Informationssystem sind neben dem Histogramm noch statistische Kennzahlen wie Minimum, Maximum, Median, Mittelwert und Standardabweichung anzuzeigen, damit der Anwender die Art der Verteilung besser interpretieren kann.

Um die Anzahl installierter Pakete auch für die Server im Einzelnen anschaulich visualisieren zu können, wird die in Abschnitt 6.1.2 erwähnte Sortierfunktion anhand von Abszisse und Ordinate noch einmal aufgegriffen: Durch die Abbildung der Anzahl der installierten Pakete auf eine der Bildachsen sieht der Anwender auf einen Blick, auf welchen Servern Wildwuchs herrscht. Diese Systeme kann sich der Administrator dann genauer ansehen und nach Programmen suchen, die gar nicht oder nicht mehr benötigt werden, um sie dann deinstallieren zu können.

Je weniger Programme auf einem Server vorhanden sind, desto kleiner ist seine potenzielle Angriffsfläche. Nebenbei verringert sich auch der Aufwand, die Anwendungen zu patchen, sowie der benötigte Speicherplatz des Servers. Der zuletzt genannte Vorteil fällt allerdings nur marginal aus, falls es sich bei den Servern hauptsächlich um virtuelle Maschinen handelt. Durch die Speicherung ihrer virtuellen Festplatten auf einem dedizierten Storage-System sorgt die Datendeduplikation bereits für eine deutliche Einsparung von benötigtem Speicher, wenn identische Programme auf mehreren Servern vorhanden sind.[1]

Wenn davon ausgegangen werden kann, dass alle überwachten Server das gleiche Betriebssystem verwenden

[1] Datendeduplikation spart nicht nur in homogenen Serverlandschaften deutlich Speicher, sondern bietet auch Vorteile bei wissenschaftlichen Anwendungen. Als Beispiel sei an dieser Stelle die Genforschung genannt. Dort fallen extreme Datenmengen an, deren Speicherung durch Deduplikation günstiger und performanter wird [NET 08]. Neben dedizierten Storage-Lösungen gibt es auch Dateisysteme, die Deduplikation beherrschen. Dazu zählen beispielsweise B-tree File System (btrfs), Opendedup und Zettabyte File System (ZFS).

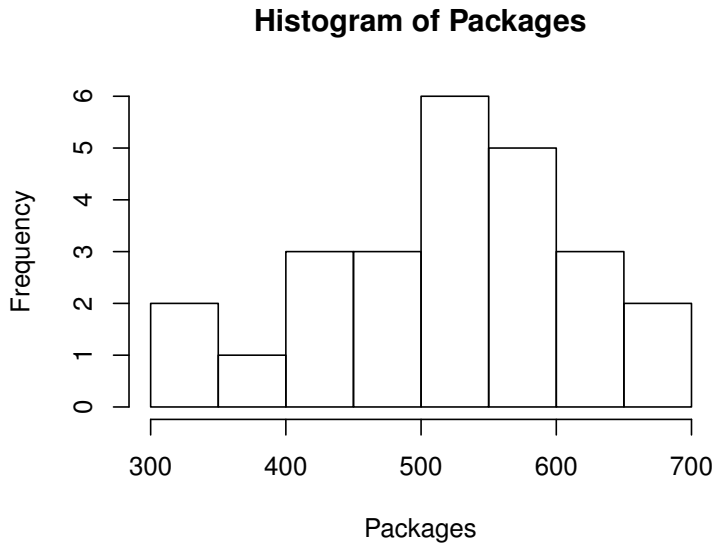


Abbildung 6.13.: Histogramm der Anzahl installierter Pakete von 25 Linux-Servern.

und dort als Ausgangslage dasselbe Minimalsystem installiert wurde, gibt es eine bestimmte Mindestanzahl an Paketen, die jeder Server besitzt. In einer Visualisierung der Paketanzahl, deren Ursprung bei der Null beginnt, führt dies zu einer Lücke, die wertvolle Bildschirmfläche wegnimmt. Indem die Skala erst beim tatsächlichen Minimalwert beginnt, wird die Bildfläche besser ausgenutzt. Abbildung 6.14 veranschaulicht die bessere Ausnutzung der zur Verfügung stehenden Bildfläche.

Dass diese Möglichkeit optional ist und auf Wunsch des Anwenders wahrgenommen werden kann ist wichtig, da hier nicht nur der Bildschirmplatz optimaler ausgenutzt wird, sondern die Daten dabei streng genommen auch verzerrt dargestellt werden. Dieses Dilemma wurde schon von Edward Tufte im Zusammenhang mit dem Thema *Graphical Integrity* angesprochen (siehe Abschnitt 2.7.3).

6.1.4. Verwendung historischer Daten

Bisher wurde beschrieben, wie sich ein Administrator anhand von Methoden der Datenvisualisierung einen Überblick über den Ist-Zustand der Softwarestände verschaffen kann, um eine Entscheidungsgrundlage zu erhalten, auf welchen Linux-Servern zeitnah Updates eingespielt werden müssen. Dabei handelt der Administrator immer nach dem selben Schema und arbeitet mit aktuellen Soll- und Istdaten. Durch die Sammlung und Visualisierung historischer Daten kann die Anwendung dem Administrator helfen, weitere Fragen zu beantworten und so das Sicherheitsniveau der Server wirksam auf einem hohen Level zu halten.

Nachvollziehbarkeit von Installationen

Die Soll-Ist-Vergleiche, die das Informationssystem durchführt, liefern nur dann zuverlässige Informationen, wenn die Referenzdaten fehlerfrei sind. Es kann vorkommen, dass alle Ampeln grün leuchten, obwohl eine kritische Lücke in einem Programm klafft, weil sie noch nicht bekannt ist. Um im Nachhinein nachvollziehen zu können, welche Paketversion seit wann, für wie lange und auf welchen Servern installiert war, ist die Speicherung historischer Daten erforderlich. Damit können Administratoren nach einem Sicherheitsvorfall gezielter reagieren und sich auf die relevanten Server konzentrieren.

Der zeitliche Verlauf der auf einem Server installierten Versionen eines bestimmten Pakets lässt sich grafisch in Form eines Stufendiagramms darstellen. Die Zeit ist dort an der Abszisse angetragen. Die einzelnen Paketversionen, die dem Informationssystem bekannt sind, werden in der Reihenfolge ihrer Veröffentlichung an der

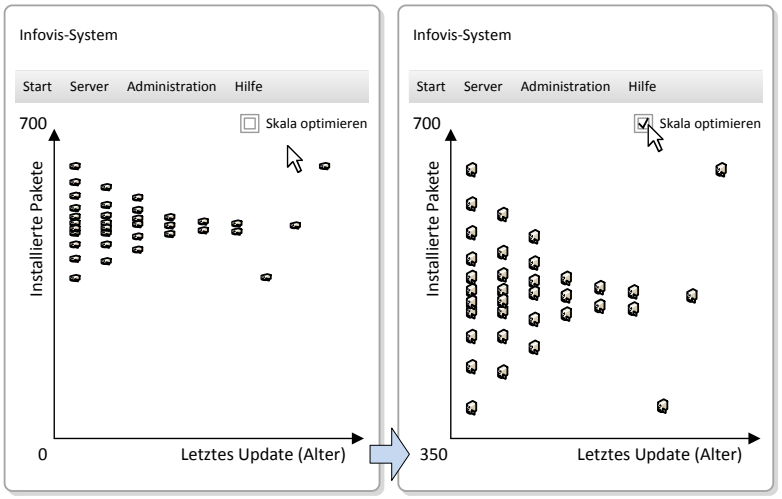


Abbildung 6.14.: Bessere Ausnutzung der Anzeigefläche durch Änderung des Ursprungs im Koordinatensystem.

Ordinate angeordnet. Das Resultat ist eine Visualisierung, die wie die Stufen einer nach rechts aufsteigenden Treppe aussieht. Je breiter ein Plateau ist, desto länger war eine bestimmte Version des betrachteten Pakets auf dem Server installiert.

Um die Aussagekraft des Diagramms zu erhöhen, muss es um Referenzdaten erweitert werden. Eine zusätzliche Treppenfunktion wird zu diesem Zweck über das vorhandene Bild gelegt, um die Zeitpunkte sichtbar zu machen, an denen eine neue Version des betrachteten Pakets im Repository des Distributors aufgetaucht ist.

Im Idealfall werden Paketupdates unmittelbar nach dem Erscheinen der neuen Version auf dem Server installiert, zum Beispiel durch ein getriggertes automatisches Update. Dies würde dazu führen, dass die beiden Treppenfunktionen identisch sind. In der Realität findet die Installation aber erst nach einer gewissen Verzögerung statt, zum Beispiel weil Updates erst getestet und danach manuell installiert werden. Da die Installation einer neuen Version nicht vor ihrem Erscheinungstermin möglich ist, hinkt die Treppenfunktion der installierten Versionen der Funktion mit den Referenzdaten immer hinterher, das heißt ein neues Plateau wird immer nach einer zeitlichen Verzögerung erreicht.

Da es für den Administrator von großem Interesse ist, wie lange eine veraltete Version auf dem Server installiert war, muss in der Visualisierung die Fläche zwischen den beiden Treppenfunktionen durch eine Signalfarbe wie beispielsweise Rot hervorgehoben werden. Je größer eine solche Fläche ist, desto länger war eine veraltete Version auf dem Server im Betrieb.

Durch die präattentive Wahrnehmung der Signalfarbe stechen die ungepatchten Zeiträume sofort ins Auge. Da lange Zeiträume zu großen Flächen führen, ziehen sie zusätzlich die Aufmerksamkeit des Betrachters auf sich. Ein Beispiel für eine derartige Visualisierung ist in Abbildung 6.15 zu sehen. Die blaue Treppe kennzeichnet die Referenzdaten, die gelbe Treppe die tatsächlich installierten Versionen und die rote Farbe die potenziell gefährlichen ungepatchten Zustände.

Das Softwareprojektplanungstool *Jira* des australischen Herstellers *Atlassian* verwendet eine ähnliche Form der Visualisierung, allerdings in einem anderen Kontext. Das Programm ermöglicht dem Projektleiter die Definition konkreter Aufgaben, die von den Softwareentwicklern umgesetzt werden müssen. Neben der Beschreibung der Aufgabe legt der Projektleiter auch die vorgesehene Bearbeitungszeit fest. Nachdem ein Softwareentwickler eine Aufgabe erledigt hat, gibt er die tatsächlich benötigte Zeit ins System ein. Aus diesen Informationen generiert Jira ein Übersichtsdiagramm. Wenn Aufgaben später erledigt wurden als ursprünglich geplant war, führt dies zu einer rot markierten Fläche zwischen den beiden Funktionen. Der Projektleiter sieht auf einen Blick, dass dort ein Problem besteht. Andererseits ist es aber auch möglich, dass eine Aufgabe bereits vor dem geplanten Ende erledigt wurde. In diesem Fall erreicht die Istdatenfunktion früher ein neues Plateau als die Referenzdatenfunktion. Jira kennzeichnet die Fläche zwischen den beiden Funktionen dort mit der Signalfarbe Grün. Ein Beispiel für ein solches Diagramm aus Jira ist in Abbildung 6.16 zu sehen.

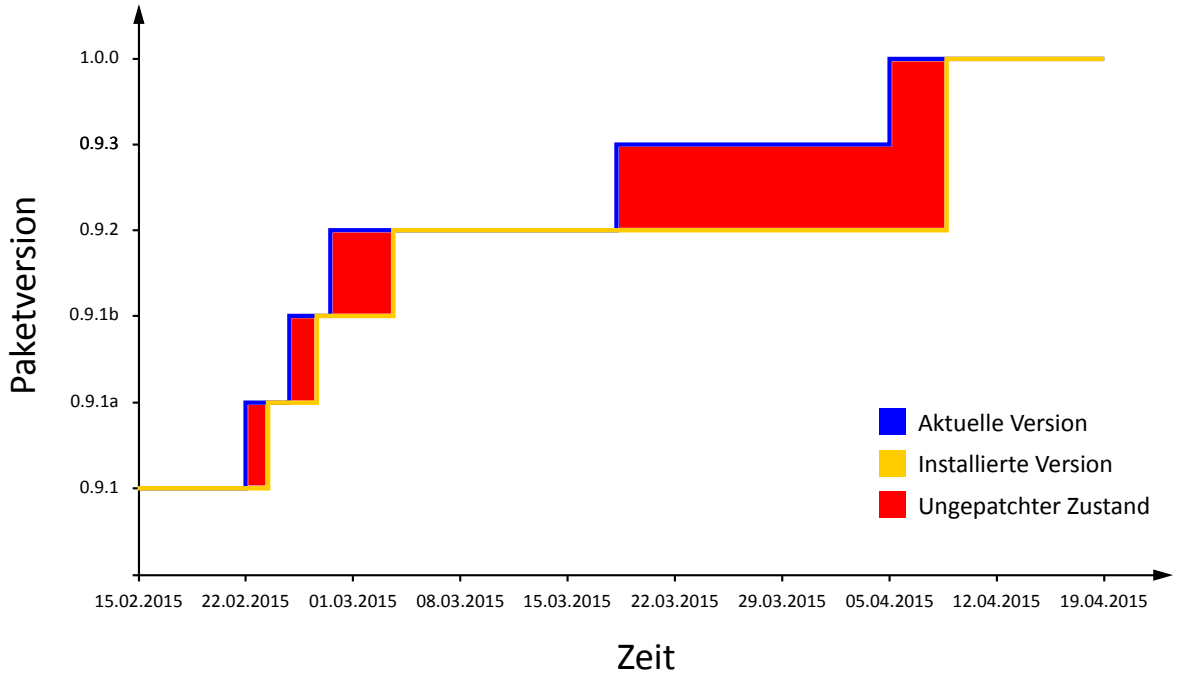


Abbildung 6.15.: Visualisierung der aktuellen und installierten Versionen eines Pakets im Lauf der Zeit.

Die Aussagekraft des Diagramms kann noch weiter ausgebaut werden, indem Informationen zu kritischen Softwarelücken mit aufgenommen werden. Dies kann durch eine zusätzliche Farbe geschehen, indem die Treppenstufen der betroffenen Versionen hervorgehoben werden. Ebenfalls denkbar ist eine waagerechte Linie, die das Diagramm in zwei Hälften aufteilt. In der unteren Hälfte befinden sich die Paketversionen, die von einer schwerwiegenden Softwarelücke betroffen waren, in der oberen Hälfte sind die fehlerbereinigten Versionen enthalten. Sofern der kritische Fehler nicht von Anfang an in dem Programm enthalten war, kann das Erscheinen der Lücke durch eine zusätzliche Trennlinie gekennzeichnet werden. Die Einfärbung der Treppenstufen zwischen diesen beiden Trennlinien in einer Signalfarbe ist auch möglich.

Unapplied Patch Latency

Ein wertvoller Indikator, der auf historischen Daten basiert, ist laut dem Sicherheitsexperten Andrew Jaquith die sog. *Unapplied Patch Latency* [JAQ 07]. Es handelt sich dabei um die durchschnittliche Zeitdifferenz (oder den Median) zwischen der Veröffentlichung neuer Programmupdates und der verifizierten Patch-Installation auf den Zielsystemen. Zur Berechnung dieser Kennzahl kann auf die regelmäßig abgefragten Paketlisten der einzelnen Server zurückgegriffen werden.

Im einfachsten Fall kann die Änderung der Unapplied Patch Latency über die Zeit anhand eines Liniendiagramms visualisiert werden. Aus dieser Visualisierung kann unmittelbar abgelesen werden, wie sich die Reaktionszeit der Administratoren durch den Einsatz des Informationssystems über einen längeren Zeitraum verändert. Im Idealfall sollte eine Verkleinerung dieser Kennzahl zu verzeichnen sein.

Eine weitere Möglichkeit ist die Gegenüberstellung dieser Kennzahlen oder Kennzahlveränderungen unterschiedlicher Teams. Somit kann herausgefunden werden, welche Gruppe von Administratoren Updates im Mittel schneller bereitstellt. Zur Visualisierung eignen sich nebeneinander abgebildete oder übereinandergelegte Liniendiagramme, wie es das Beispiel in der Abbildung 6.17 zeigt.

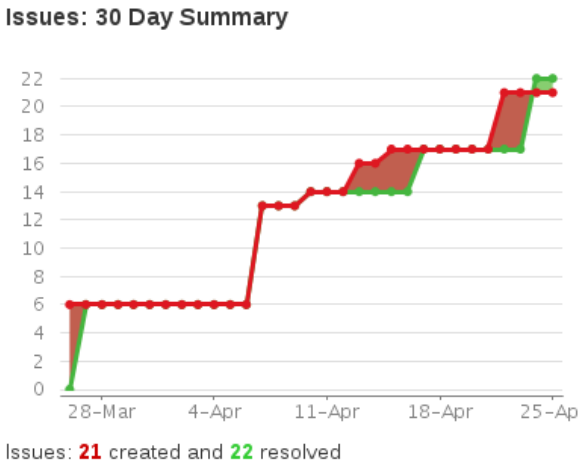


Abbildung 6.16.: Visualisierung des Fortschritts eines Softwareentwicklungsprojekts in Jira.

Installierte Patches pro Periode

Für den Rückblick auf die Vergangenheit kann es interessant sein, wie viele Patches in einem bestimmten Zeitraum installiert wurden. Die dafür erforderlichen Daten können aus den regelmäßig aktualisierten Paketlisten der einzelnen Server generiert werden. Die Wahl des Zeitraums nimmt der Anwender per Dynamic Query vor. Es ist möglich, diese Kennzahl für einzelne Server zu ermitteln und miteinander zu vergleichen oder die Summe über alle Server zu bilden. Als Visualisierung der zeitlichen Veränderung dieser Kennzahl eignen sich dynamisch generierte Liniendiagramme. Zum Vergleich der Veränderung mehrerer Kennzahlen können die Liniendiagramme auch übereinandergelegt werden, ähnlich wie es bereits bei der Unapplied Patch Latency der Fall ist.

6.1.5. Umgang mit Fremdsoftware

Paketverwaltungssysteme sind unverzichtbare Hilfsprogramme, um Software auf einem Server zu verwalten. Es gibt aber auch Fälle, wo die manuelle Installation eines Programms unumgänglich ist. Dafür gibt es folgende Gründe:

- Die Programmversion im Repository des Distributors ist veraltet, es wird aber eine neuere Version aufgrund des Funktionsumfangs oder weiterentwickelter Basistechnologien benötigt.
- Das Programm befindet sich nicht im Repository des Distributors.

Das erstgenannte Problem taucht häufig bei Stable- und Oldstable-Distributionszweigen auf, die lediglich Sicherheitsupdates für eine bestimmte Programmversion und keine Versionsupdates bereitstellen. Dies ist zum Beispiel bei Debian der Fall, wo der Fokus auf Stabilität und Sicherheit liegt. Der zweite Fall tritt ein, wenn der Distributor die Softwarelizenz eines Programms für unverträglich hält.

Der Java Application Server *GlassFish* ist beispielsweise nur in einer stark veralteten Version im Repository der zum Zeitpunkt der Verfassung dieses Textes aktuellen Debian-Stable-Version 7 zu finden, die lediglich Java Enterprise Edition (EE) 5 aus dem Jahr 2006 implementiert (aktuell ist Java EE 7). Diese Version stammt noch aus Zeiten bevor ihr Hersteller *Sun Microsystems* von *Oracle* aufgekauft wurde [WILK 09]. Im Repository von openSUSE ist GlassFish überhaupt nicht zu finden. Aus diesem Grund ist dieses Programm ein Kandidat für eine manuelle Installation.

Programme, die nicht über das Paketmanagementsystem bezogen werden, benötigen nicht nur eine manuelle Erstinstallation, sondern müssen von den Administratoren auch manuell gepatcht werden. Manche Programme bringen zwar eigene Update-Mechanismen mit, sie sorgen allerdings für zusätzlichen Aufwand. Da das zentrale Informationssystem, wie es bisher beschrieben wurde, nur den Softwarestand des Paketmanagement-

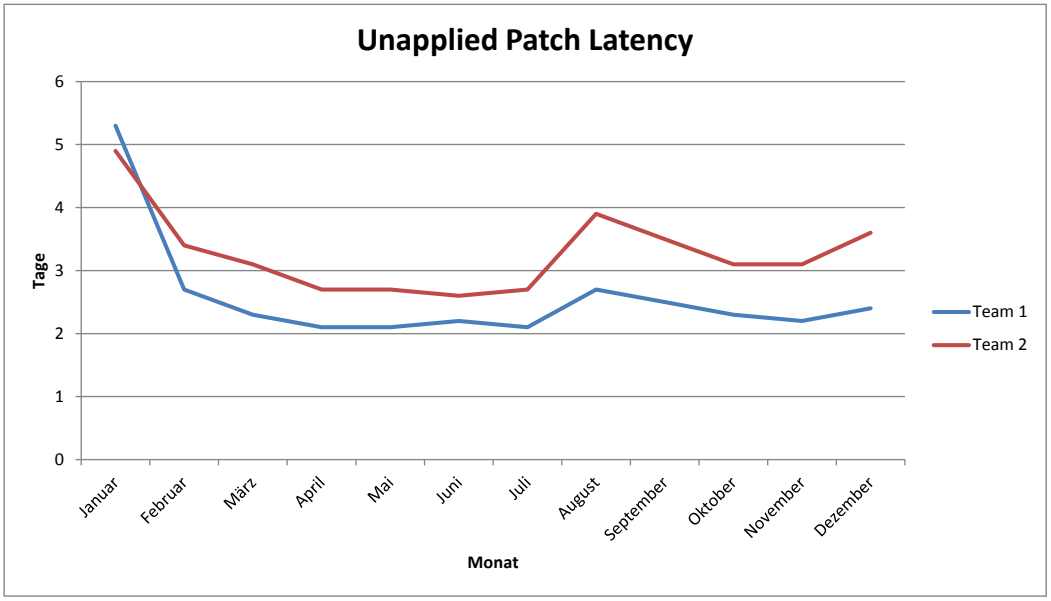


Abbildung 6.17.: Beispiel für die Visualisierung der Reaktionszeit zweier Teams bei neuen Updates über ein Kalenderjahr.

systems abfragt, liegen ihm keine Informationen über die Fremdsoftware vor. Um Fremdsoftware dennoch zu unterstützen, muss wie im folgenden Abschnitt beschrieben vorgegangen werden.

Fremdsoftware erkennen

Damit das Informationssystem den Administrator überhaupt auf veraltete Fremdsoftware hinweisen kann, muss sie zunächst erkannt werden. Es ist naheliegend, zu diesem Zweck die laufenden Prozesse des zu überwachenden Servers mit einem Tool wie `top` oder `htop` abzufragen und dort nach laufenden Programmen zu suchen, die nicht aus dem Paketsystem stammen.

Leider funktioniert dieser Ansatz nicht überall: Bei der Fremdsoftware könnte es sich um eine Webanwendung handeln, die über einen Web Server wie zum Beispiel Apache bereitgestellt wird. In diesem Fall findet man in der Prozesstabelle keinen Hinweis auf die Anwendung selbst, sondern sieht nur die Prozesse von Apache (siehe Abbildung 6.18). Dieser Ansatz führt also nicht immer zum Ziel.

Einen weiteren Anhaltspunkt für installierte Fremdsoftware kann auch ein Portscan liefern. Je nach Anwendung lässt ein Portscan aber nicht immer einen Rückschluss auf eine konkrete Anwendung oder Paketversion zu. Außerdem lauscht nicht jede Fremdsoftware zwingend an einem eigenen Port oder eine Personal Firewall blockiert die Pakete des Informationssystems an den Ports des Dienstes, sodass ein Portscan überhaupt kein Ergebnis liefern kann. Es ist zum Beispiel auch denkbar, dass es sich bei der Fremdsoftware um eine Enterprise-Java-Anwendung handelt, die von einem Application Server bereitgestellt wird. In diesem Fall sieht der Portscanner nur die Standardports des Application Servers. Portscans stellen allein betrachtet also auch keine Lösung dar, sondern eignen sich eher als Ergänzung.

Da es keinen allgemeingültigen Ansatz zum eindeutigen Identifizieren von Fremdsoftware gibt, muss ein neuer geschaffen werden. Dieser sollte aufgrund der Praktikabilität möglichst einfach gehalten sein und dem Zweck dienen, dem Informationssystem die notwendigen Informationen zu liefern.

Bei der manuellen Installation von Fremdsoftware müssen sich Administratoren von Anfang an auf ein gemeinsames Schema einigen. Um automatisch über ein Skript oder einen Hintergrundprozess den Zustand dieser Programme prüfen zu können, müssen sie ausnahmslos in einem auf allen Servern einheitlichen Verzeichnis installiert werden. Der FHS sieht hierfür das Verzeichnis `/opt/` vor.

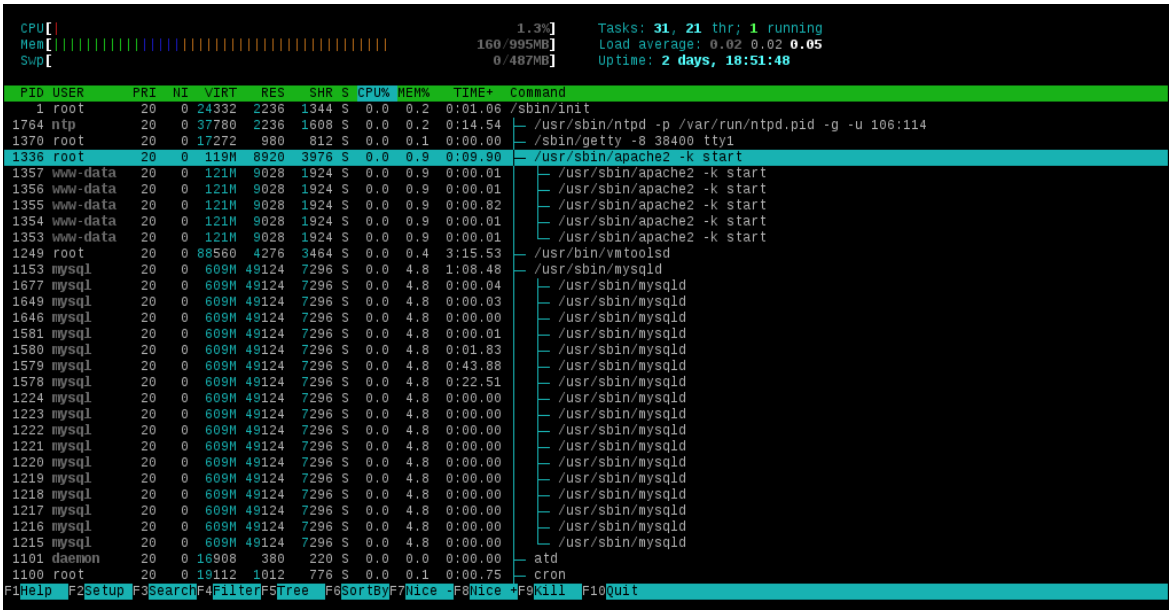


Abbildung 6.18.: Ausschnitt der Prozessliste eines virtuellen Servers, der eine Webanwendung durch Apache bereitstellt.

Bereits in anderen Verzeichnissen installierte Fremdsoftware muss nachträglich in den Ordner `/opt/` verschoben werden, damit auf allen Servern dasselbe Prüfskript oder derselbe Hintergrundprozess als Datenquelle für das Informationssystem verwendet werden kann. Alternativ können auch *symbolische Links* oder *Bind Mounts* verwendet werden, um sämtliche Fremdsoftware über das `/opt/`-Verzeichnis erreichbar zu machen.

Einzelne Anpassungen an abweichende Pfade in dem Prüfskript mögen zwar im Einzelfall schnell von der Hand gehen, dadurch steigt aber auch die Spezifität dieses Skripts. Es wird unübersichtlicher und schwieriger zu warten. Im Endeffekt bereiten Anpassungen an Sonderfälle den Administratoren mehr Arbeit als ein von Anfang an befolgtes einheitliches System.

Durch die Prüfung des Vorhandenseins von Inhalt im `/opt/`-Verzeichnis kann das Informationssystem den Administrator immerhin darauf aufmerksam machen, auf welchen Linux-Servern Fremdsoftware vorhanden ist. Dies lässt sich zum Beispiel durch ein zusätzliches visuelles Attribut in den Server-Glyphen der Gesamtübersicht bewerkstelligen. Durch eine zusätzliche Filteroption, die nur Server mit gefundener Fremdsoftware einblendet, sieht der Anwender sofort alle betroffenen Systeme.

Versionsprüfung von Fremdsoftware

Damit der Administrator nicht auf Verdacht die Server mit Fremdsoftware auf veraltete Versionen überprüfen muss, ist eine automatische Versionsprüfung wünschenswert. Da es keinen einheitlichen Standard für Softwarepakete von Drittherstellern gibt, muss selbst ein möglichst einfaches System eingeführt werden. Zwei Ansätze hierfür sind denkbar:

- Suche nach Versionsangaben, die programmspezifisch in unterschiedlichen Dateien und Formaten vorliegen können.
- Definition eines gemeinsamen und einheitlichen Formats zur Speicherung der Versionsangaben unterschiedlicher Programme.

Die erste Möglichkeit besteht darin, dass das Versionsprüfprogramm auf einem Server an bekannten Orten nach Versionsangaben sucht. Fremdsoftware liegt oft in Form einer Archivdatei vor, die auf dem Zielsystem entpackt werden muss. Das Hauptverzeichnis, das alle Dateien des Programms enthält, beinhaltet oft bereits die Versionsnummer in seinem eigenen Namen.

Beispiele hierfür sind `otrs-4.0.7` und `phpMyAdmin-4.4.3-all-languages`. Der komplette Name des Ordners kann mit einer Referenzdatenbank verglichen werden, um die Aktualität der Fremdsoftware zu überprüfen. Dies setzt allerdings voraus, dass sämtliche Fremdsoftware auf allen Servern nach diesem Schema installiert wird.

Falls der Name eines Programmverzeichnisses in einer Konfigurationsdatei angegeben werden muss, kann sich der Administrator bei einer Aktualisierung der Fremdsoftware die Arbeit sparen, die Konfigurationsdatei an den neuen Verzeichnisnamen anzupassen, indem ein symbolischer Link oder ein Bind Mount verwendet wird, der auf die aktuelle Version verweist und selbst keine Versionsbezeichnung im eigenen Namen enthält. Angenommen der Ordner `/opt/` beinhaltet die Programmverzeichnisse `otrs-3.0.7` und `otrs-4.0.7`. In diesem Fall muss der symbolische Link mit dem Namen `/opt/otrs` auf das Verzeichnis mit der jeweils aktuellen Version verweisen.

Bei vielen Programmen liegt die Versionsbezeichnung in Form einer Textdatei innerhalb des Installationsverzeichnisses vor, zum Beispiel in den *Release Notes*. Diese Versionsangaben dienen nicht nur zur Information des Administrators, sondern werden ggf. auch von Update-Routinen benutzt, um sicherzustellen, dass die Ausgangsversion kompatibel ist oder um versionsspezifische Änderungen an bestimmten Dateien oder der dazugehörigen Datenbank vorzunehmen.

Ein gutes Beispiel hierfür ist das populäre Trouble-Ticket-System *Open Ticket Request System (OTRS)*[2]. Die Anwendung legt ihre Versionsangaben in der Textdatei `RELEASE` im Programmverzeichnis ab. Es erweist sich als vorteilhaft, dass das Format dieser Datei bereits seit mehreren Jahren unverändert eingesetzt wird. Dadurch erübrigt sich der Aufwand, den Parser des Prüfprogramms an neue Versionen anzupassen.

Listing 6.1: Versionsangaben von OTRS 3.0.7

```
1 PRODUCT = OTRS
2 VERSION = 3.0.7
3 BUILDDATE = Fri Mar 25 06:39:58 CET
      2011
4 BUILDHOST = lusen.otrs.org
```

Listing 6.2: Versionsangaben von OTRS 4.0.7

```
1 PRODUCT = OTRS
2 VERSION = 4.0.7
3 BUILDDATE = Thu Mar 26 00:55:56 CET
      2015
4 BUILDHOST = otrsbuild.otrs.com
```

Die beiden Listings 6.1 und 6.2 verdeutlichen diesen Sachverhalt. Die erste Datei stammt aus der Version 3.0.7 vom Jahr 2011, die zweite aus der zur Zeit aktuellen Version 4.0.7. Das Dateiformat wurde im Lauf der Jahre beibehalten. Für den Parser des Prüfprogramms sind dort in erster Linie die Attribute `PRODUCT` und `VERSION` (Zeilen 1 – 2) von Interesse.

Einen ähnlichen Ansatz verfolgt der australische Softwarehersteller Atlassian bei seinem kommerziellen Wiki- und Kollaborationssystem *Confluence*[3]. Die Versionsangaben sind dort zwar so gut versteckt, dass man besser eine Suchmaschine benutzt, um ihren Ort zu bestimmen,[4] dafür ist aber auch dort das Dateiformat versionsübergreifend einheitlich gehalten, was die Implementierung eines Parsers erleichtert.

Listing 6.3: Versionsangaben von Confluence 3.5.11

```
1 #Generated by Maven
2 #Wed Aug 17 23:53:04 CDT 2011
3 version=3.5.11
4 groupId=com.atlassian.confluence
5 artifactId=confluence-webapp
```

Listing 6.4: Versionsangaben von Confluence 5.5.1

```
1 #Generated by Maven
2 #Tue May 20 05:00:57 UTC 2014
3 version=5.5.1
4 groupId=com.atlassian.confluence
5 artifactId=confluence-webapp
```

[2] OTRS-Website: `http://www.otrs.com/`

[3] Confluence-Website: `https://de.atlassian.com/software/confluence`

[4] Unter Verwendung des vom Hersteller empfohlenen Installationspfads befinden sich die Versionsangaben von Confluence in der Datei `/opt/confluence/confluence/META-INF/maven/com.atlassian.confluence/confluence-webapp/pom.properties`. Dieser umständliche Dateipfad ist der Tatsache geschuldet, dass es sich bei Confluence um eine Enterprise-Java-Anwendung handelt, die in der Regel vom mitgelieferten Application Server *Apache Tomcat* (`http://tomcat.apache.org/`) bereitgestellt wird. Zudem benutzt Atlassian offensichtlich das Build-Management-Werkzeug *Apache Maven* (`http://maven.apache.org/`), um die Entwicklung des Java-Programms zu standardisieren und die Code-Verwaltung zu vereinfachen.

In den beiden Listings 6.3 und 6.4 sind Dateien mit Versionsangaben von Confluence aus den Jahren 2011 und 2014 zu sehen. Auch hier ist das Format unter anderem dank Apache Maven über die Jahre unverändert geblieben.

Leider verfolgt Atlassian dieses einheitliche Schema der Versionierung nicht konsequent über seine eigene Produktpalette hinweg. Während etwa das Softwareprojektplanungstool *Jira*[5] dem Prinzip von Confluence folgt, weicht das komplementär dazu angebotene Quellcodeverwaltungsprogramm *FishEye*[6] davon ab. Eine mögliche Quelle für die Programmversion ist dort eine Readme-Datei, die im HTML-Format vorliegt. Die zuverlässige und versionsübergreifende Extraktion der Versionsangabe ist in diesem Fall ein Glücksspiel.

Die Implementierung einer Versionsprüfung ist zwar möglich, aber umständlich, wenn die Versionsangaben nur in Dateien vorliegen, die an Menschen adressierte Prosatexte und kein strukturiertes Datenformat beinhalten. Dort besteht immer das Risiko, dass die Versionsermittlung bei einer neuen Programmversion nicht mehr funktioniert. Beispiele für Open-Source-Anwendungen ohne strukturierte Versionsangaben sind die Webmail-Applikation *Roundcube*[7] und die Mediendatenbank *ResourceSpace*[8].

In einigen Fällen kann auch die Hauptanwendung mit einem speziellen Kommandozeilenparameter wie z. B. `-version` oder `--version` aufgerufen werden, um an die genaue Versionsbezeichnung zu gelangen. In beiden Fällen muss aber das Versionsprüfprogramm speziell für jede einzelne Anwendung angepasst werden, um alle Versionen korrekt zu erkennen. Nach einer Anpassung des Prüfprogramms an eine aktuelle Version muss es noch dazu auf allen Servern neu eingespielt werden.

Sofern die Menge an Fremdsoftware, die im Rechenzentrum installiert wurde, überschaubar ist, stellt die Implementierung der Versionserkennung im Prüfprogramm kein großes Problem dar. Hier gilt es abzuwägen, ob der Aufwand vertretbar ist oder ob man auf eine alternative Methode setzt, die im Folgenden Beschrieben wird.

Der zweite der zuvor genannten Lösungsansätze sieht keine speziellen Anpassungen des Versionprüfprogramms an jede in Frage kommende Anwendung vor. Versionsangaben werden in diesem Fall durch den Administrator nach der Installation der Fremdsoftware einer speziellen Datei im Anwendungsverzeichnis abgelegt. Diese Datei hat stets denselben Namen und strukturellen Aufbau, sodass das Prüfprogramm zu jedem Programmverzeichnis die Version ermitteln kann.

Beim Aufbau dieser Datei könnte man sich an der auf vielen zeitgemäßen Linux-Systemen vorhandenen Datei `/etc/os-release` orientieren (ein Beispiel hierfür ist in Listing 6.5 zu sehen). Diese Datei wurde zusammen mit dem Init-System *systemd* eingeführt und beinhaltet Versionsangaben zum laufenden Betriebssystem.

Listing 6.5: Die Konfigurationsdatei `/etc/os-release` einer openSUSE-Installation

```
1  NAME=openSUSE
2  VERSION="13.2 (Harlequin)"
3  VERSION_ID="13.2"
4  PRETTY_NAME="openSUSE 13.2 (Harlequin) (x86_64)"
5  ID=opensuse
6  ANSI_COLOR="0;32"
7  CPE_NAME="cpe:/o:opensuse:opensuse:13.2"
8  BUG_REPORT_URL="https://bugs.opensuse.org"
9  HOME_URL="https://opensuse.org/"
10 ID_LIKE="suse"
```

Die zentrale Speicherung der Versionen für mehrere Programme in einer einzigen Datei im `/opt/`-Verzeichnis ist ebenfalls denkbar. In diesem Fall ist ein Dateiformat erforderlich, das mehrere Datensätze für eine programmatische Weiterverarbeitung speichern kann. Hier kommen zum Beispiel JavaScript Object Notation (JSON), Extensible Markup Language (XML) oder YAML Ain't Markup Language (YAML) in Frage.

Da die zusätzliche Datei mit den Versionsangaben ggf. redundante Informationen enthält, besteht die Gefahr der Inkonsistenz. Wenn eine Anwendung manuell durch eine neue Version ersetzt wird, muss diese Datei

[5] Jira-Website: `https://www.atlassian.com/software/jira/`
[6] FishEye-Website: `https://www.atlassian.com/software/fisheye/`
[7] Roundcube-Website: `http://roundcube.net/`
[8] ResourceSpace-Website: `http://www.resourcespace.org/`

angepasst werden. Die Folgen des Vergessens dieser Anpassung sind allerdings nicht fatal. Das zentrale Informationssystem würde den Administrator darauf hinweisen, dass weiterhin eine veraltete Programmversion installiert ist. Dieser Hinweis genügt, um ihn zur nachträglichen Anpassung der Versionsdatei zu bewegen.

Damit das Informationssystem überhaupt einen Versionsvergleich durchführen kann, benötigt es auch hierfür eigene Referenzdaten mit den aktuellen Programmversionen. Diese Referenzdaten werden an zentraler Stelle in einer Datenbank gespeichert. Ähnlich wie die Anwendungen aus einem Repository können auch sie mit der Information versehen werden, ob eine sehr kritische Sicherheitslücke existiert. Die Visualisierung erfolgt auf dieselbe Art, wie bereits zuvor genannt wurde.

6.2. Firewall-Konfiguration

Auch wenn der Zugriff auf einzelne Dienste eines Servers durch eine zentrale Firewall Appliance wie zum Beispiel eine Cisco Adaptive Security Appliance (ASA) reglementiert wird, machen zusätzliche Personal Firewalls auf den Servern Sinn. Sie tragen zur Verbesserung der Sicherheit bei, indem eine zusätzliche unabhängige Sicherheitsebene eingeführt wird, die ein Angreifer überwinden muss. Dieses Sicherheitsprinzip wird auch als *Defense in Depth* bezeichnet [SCHN 06][NSA 01]. Im Fehlerfall dienen Personal Firewalls auch als Fallback-Lösung. Zum Fehlerfall kommt es zum Beispiel durch eine Fehlkonfiguration, die Ausnutzung von Schwachstellen oder einfach durch den Ausfall der Appliance.

Dieser Abschnitt befasst sich damit, wie ein Informationssystem die Firewall-Konfigurationen einzelner Server überwacht und dem Administrator die Zustände durch Methoden der Informationsvisualisierung augenblicklich mitteilt. Für Grundlagen zum Thema Firewalls wird an dieser Stelle auf Abschnitt 3.4.1 verwiesen.

6.2.1. Vorüberlegungen

Personal Firewalls auf virtuellen Servern stellen eine kostengünstige und praktikable Lösung dar, um den Zugriff auf die Server innerhalb eines Firmen- oder Institutsnetzes zu beschränken. Eine VM besitzt oft nur eine virtuelle Netzwerkschnittstelle, die sowohl zur Erbringung ihres Dienstes als auch zur Administration per SSH dient. Aus diesem Grund ist es sinnvoll, den SSH-Zugang durch eine Personal Firewall derart zu beschränken, dass nur Administratoren aus ihrem eigenen Netz oder von ausgewählten Management-PCs aus darauf zugreifen können.

Ein anderes Beispiel für eine sicherheitsrelevante Zugriffsbeschränkung betrifft die Software-Updates, die bereits im Abschnitt 6.1 angesprochen wurden. Hypertext Transfer Protocol (HTTP)-Verbindungen nach außen sollten nur zu einem Proxy Server erlaubt sein und dem Zweck dienen, auf das Repository der Distribution zurückzugreifen. Falls sehr viele Linux-Server vorhanden sind, macht auch der Betrieb eines eigenen Spiegelservers Sinn. Das verringert nicht nur die Downloads aus dem Internet deutlich, sondern erhöht auch die Sicherheit, indem die Firewall-Regeln der Clients nur ausgehende HTTP-Verbindungen zum eigenen Spiegelserver erlauben.

Welche zusätzlichen Firewall-Regeln auf einer einzelnen VM sinnvoll sind, hängt von ihrem konkreten Einsatzzweck ab. Unabhängig davon muss eine Überwachung der Firewall-Zustände über das zentrale Informationssystem möglich sein. Hier stellt sich zunächst die Frage, wie das System an diese Informationen gelangt.

Wie bereits in Abschnitt 4.2.2 erwähnt wurde, können die Daten, die die einzelnen Server regelmäßig an das zentrale Informationssystem übertragen, potenziell auch von Angreifern missbraucht werden können. Die Kenntnis über die einzelnen Firewall-Konfigurationen der Server ist für Angreifer von enormem Vorteil. Aus diesem Grund müssen Sicherheitsmaßnahmen ergriffen werden, um dies zu verhindern. Die wichtigsten Maßnahmen wurden bereits in Abschnitt 3.4 angesprochen.

6.2.2. Abgrenzung von vorhandenen Lösungen

Firewalls haben unmittelbar etwas mit Rechnernetzen zu tun, denn ohne Netze wären auch keine Firewalls notwendig. Es liegt deshalb nahe, auch die Topologien der Netze selbst zu visualisieren, in denen die über-

wachten Server eingebunden sind. Derartige Visualisierungen geben sofort Aufschluss über die möglichen Wechselwirkungen zwischen den Netzelementen.

Die Auslastung einzelner Systeme und der Traffic zwischen ihnen sind sowohl mess- als auch visualisierbar. Eine grafische Unterscheidung der einzelnen Sicherheitszonen ist ebenfalls denkbar. Einen Vorschlag zur Visualisierung von Netzwerkverkehr für Administratoren veröffentlichten *Robert Ball et al.* bereits im Jahr 2004. Bei ihrem Konzept hielten sie sich streng an Ben Shneidermans Information Seeking Mantra (siehe Abschnitt 6.1.2) [BALL+ 04].

Da es bereits diverse Produkte auf dem Markt gibt, die diesem Zweck dienen, wird auf die Visualisierung ganzer Netze an sich in dieser Arbeit nicht weiter eingegangen. Das im Rahmen dieser Arbeit angesprochene Informationssystem soll auch nicht dem Zweck dienen, wie viele bereits vorhandene Produkte Echtzeitdaten wie Netzverkehr oder Arbeitsspeicherauslastungen zu verarbeiten und anzuzeigen. Stattdessen soll aufgezeigt werden, wie die Konfigurationen einzelner Personal Firewalls überwacht und optimiert werden können.

6.2.3. Firewall-Zustand auf den Servern ermitteln

Die naheliegendste Möglichkeit, um an die Firewall-Konfigurationen zu gelangen, besteht in der Abfrage der Regeldateien, die beim Start der Server automatisch geladen werden. Das im vorherigen Abschnitt bereits erwähnte Prüfprogramm würde diese Dateien einlesen und zum zentralen Informationssystem senden. Dies setzt voraus, dass die Regeldateien aller Server in einem identischen Verzeichnis gespeichert sind. Darauf müssen Administratoren von Anfang an achten, da es auf lange Sicht nicht praktikabel ist, Sonderfälle in das Prüfprogramm einzubauen.

Um etwas mehr Flexibilität zu gewinnen, reicht es aus, sich nur auf ein bestimmtes Verzeichnis zu einigen, das beliebige Regeldateien enthält. Diese können dann unabhängig von ihrer Anzahl und ihrem Namen vom Prüfprogramm berücksichtigt werden. Besser ist es allerdings, auf distributionsspezifische Standardverzeichnisse und -dateien zurückzugreifen, sofern sie vorhanden sind.

Regeln auslesen bei openSUSE

Da bei openSUSE standardmäßig eine Personal Firewall mit installiert wird, ist dort bereits ein einheitlicher Ort für die Firewall-Konfiguration vorgegeben. Die Firewall-Konfiguration befindet sich in der Datei `/etc/sysconfig/SuSEfirewall2`. Die Dienstspezifischen Parameter sind bei openSUSE in separate Dateien innerhalb des Verzeichnisses `/etc/sysconfig/SuSEfirewall2.d/services/` ausgelagert. Diese beinhalten zum Beispiel die Transmission Control Protocol (TCP)- und User Datagram Protocol (UDP)-Ports, die für den jeweiligen Dienst geöffnet werden müssen.

Regeln auslesen bei Debian

Bei der auf Servern weit verbreiteten Linux-Distribution Debian sind nach einer erstmaligen Installation keine Firewall-Konfigurationsdateien vorhanden. Prinzipiell können sie an beliebigen Orten gespeichert werden, in der Regel werden sie aber in einem Verzeichnis unterhalb des Ordners `/etc/` aufbewahrt.

Da bei Debian nichts Firewallspezifisches vorinstalliert ist, gibt es auch verschiedene Möglichkeiten, Firewall-Konfigurationen beim Hochfahren des Systems zu laden. Es bietet sich beispielsweise an, ein Skript zur Initialisierung der Firewall-Regeln im Verzeichnis `/etc/network/if-up.d/` abzulegen. Die darin enthaltenen Skriptdateien werden automatisch ausgeführt, sobald das Netzwerkinterface aktiv ist.

Eine andere Möglichkeit bei Debian ist die Verwendung des Programms `iptables-persistent`. Es wird als Dienst ausgeführt und kümmert sich automatisch um das Laden der Firewall-Konfiguration beim Systemstart. Im laufenden Betrieb geänderte oder hinzugefügte Firewall-Regeln speichert das Programm automatisch an einer einheitlichen Stelle: für IPv4-Regeln in der Datei `/etc/iptables/rules.v4`, bei Internet Protocol Version 6 (IPv6) in der Datei `/etc/iptables/rules.v6`. Da `iptables-persistent` nicht nur das Laden, Entladen und Aktualisieren der Regeldateien übernimmt, sondern auch ein einheitliches Dateischema benutzt, ist die Verwendung dieses Tools auf Debian-Servern empfehlenswert.

Probleme beim Auslesen der Regeldateien

Bereits anhand der beiden Beispiele openSUSE und Debian wird deutlich, dass in einer heterogenen Linux-Serverlandschaft die Komplexität hoch ist. Das Prüfprogramm müsste sowohl Standardverzeichnisse als auch oft verwendete Verzeichnisse nach den Firewall-Konfigurationsdateien durchsuchen. Die Komplexität steigt potenziell mit jeder hinzukommenden Linux-Distribution.

Beim Auslesen der Regeldateien besteht allerdings das Problem, dass die Firewall-Konfiguration, die sich im Arbeitsspeicher befindet, von ihnen abweicht. Dies ist dann der Fall, wenn im laufenden Betrieb über die Kommandozeile Änderungen an den aktiven Regeln vorgenommen wurden. Im schlimmsten Fall ist die Firewall – obwohl strenge Regelsätze in Form von Dateien vorliegen – überhaupt nicht aktiv. Dieser Zustand ist zum Beispiel auf fehlerhafte Konfigurationsdateien zurückzuführen. Ein anderer Grund mag sein, dass ein Administrator die Personal Firewall absichtlich vorübergehend deaktiviert hat, weil eine Anpassung der Firewall-Regeln für einen bestimmten Eingriff zu lange gedauert hätte, und er danach vergessen hat, die Firewall wieder anzuschalten.

Listing 6.6: Beispiel für die Konfiguration der Personal Firewall eines Linux-Servers

```
1  Chain INPUT (policy DROP)
2  target     prot opt source               destination
3  ACCEPT     all  --  anywhere             anywhere
4  ACCEPT     tcp  --  10.123.10.0/24       anywhere             tcp dpt:ssh state
       NEW,ESTABLISHED
5  ACCEPT     udp  --  anywhere             anywhere             udp spt:domain
6  ACCEPT     udp  --  anywhere             anywhere             udp spt:ntp
7  ACCEPT     udp  --  anywhere             anywhere             udp dpt:ntp
8  ACCEPT     icmp --  anywhere             anywhere             icmp echo-reply
9  ACCEPT     icmp --  anywhere             anywhere             icmp echo-request
10 ACCEPT     tcp  --  proxy.mydomain.com   anywhere             tcp spt:3128 state
       ESTABLISHED
11 ACCEPT     tcp  --  mysql.mydomain.com   anywhere             tcp spt:mysql state
        ESTABLISHED
12 ACCEPT     tcp  --  ldap.mydomain.com    anywhere             multiport sports
       ldap,ldaps state ESTABLISHED
13 ACCEPT     tcp  --  10.123.20.0/24       anywhere             multiport dports
       http,https state NEW,ESTABLISHED
14
15
16 Chain FORWARD (policy DROP)
17 target     prot opt source               destination
18
19 Chain OUTPUT (policy DROP)
20 target     prot opt source               destination
21 ACCEPT     all  --  anywhere             anywhere
22 ACCEPT     tcp  --  anywhere             10.123.10.0/24       tcp spt:ssh state
       ESTABLISHED
23 ACCEPT     udp  --  anywhere             anywhere             udp dpt:domain
24 ACCEPT     udp  --  anywhere             anywhere             udp dpt:ntp
25 ACCEPT     udp  --  anywhere             anywhere             udp spt:ntp
26 ACCEPT     icmp --  anywhere             anywhere             icmp echo-request
27 ACCEPT     icmp --  anywhere             anywhere             icmp echo-reply
28 ACCEPT     tcp  --  anywhere             proxy.mydomain.com   tcp dpt:3128 state
       NEW,ESTABLISHED
29 ACCEPT     tcp  --  anywhere             mysql.mydomain.com   tcp dpt:mysql state
        NEW,ESTABLISHED
30 ACCEPT     tcp  --  anywhere             ldap.mydomain.com    multiport dports
       ldap,ldaps state NEW,ESTABLISHED
31 ACCEPT     tcp  --  anywhere             10.123.20.0/24       multiport sports
       http,https state ESTABLISHED
```

Listing 6.6 zeigt ein Beispiel für die Ausgabe des Befehls `iptables -L` auf einem Server für eine interne Webanwendung. Der Einfachheit halber wurde hier auf die standardmäßig vorhandenen Chains `INPUT`, `FORWARD` und `OUTPUT` zurückgegriffen.

Den Zeilen 1, 16 und 19 ist zu entnehmen, dass es sich um eine Whitelist handelt, da die Standard-Policy für alle Chains auf `DROP` eingestellt ist. Die Firewall erlaubt den SSH-Zugriff nur aus einem speziellen Management-Netz heraus (Zeilen 4 und 22) und den Zugang zur Webanwendung des Servers ausschließlich über ein anderes internes Netz (Zeilen 13 und 31). Darüber hinaus ist die Nutzung einiger Standard-Netzwerkdienste erlaubt (Zeilen 5 – 9 und 23 – 27). Die VM darf für Software-Updates auf einen Proxy Server (Zeilen 10 und 28) und zur Benutzerauthentifizierung auf einen LDAP Server (Zeilen 12 und 30) zugreifen. Die Datenbank der Webanwendung befindet sich auf einem dedizierten Datenbankserver (Zeilen 11 und 29).

Um den aktuellen Zustand der Firewall zu ermitteln, kann das Prüfprogramm auf die Ausgabe des Kommandos `iptables -L` zurückgreifen. Dieses Kommando gibt aber nur die Regeln für IPv4 aus. Der IPv6 Stack des Betriebssystems muss ebenfalls berücksichtigt werden, da er bei einem frisch installierten Debian oder openSUSE standardmäßig aktiviert ist. Falls er auf dem Server nicht nachträglich deaktiviert wurde, muss zusätzlich die Ausgabe des Befehls `ip6tables -L` berücksichtigt werden. Die Ausgaben der beiden Programme können vom Prüfprogramm dann an das zentrale Informationssystem übertragen werden, wo sie danach ausgewertet werden.

6.2.4. Firewall-Zustand durch Portscans ermitteln

Als Ergänzung zu den Firewall-Zustandsdaten, die die Server selbst bereitstellen, können Portscans von außen durchgeführt werden. Sie sind zwar aufwendiger durchzuführen und erzeugen während der Laufzeit eine Last im Netz und auf den zu prüfenden Systemen, dafür liefern sie die Informationen, die auch ein potenzieller Angreifer von außen sehen würde. Zudem bieten Portscans den Vorteil, dass sie auch für Server Daten liefern, bei denen das Prüfprogramm nicht korrekt konfiguriert ist, nicht läuft oder überhaupt nicht vorhanden ist.

Es ist denkbar, das zentrale Informationssystem mit Portscan-Daten verschiedener Quellen zu versorgen und durch die Aggregation der Daten schneller an sicherheitsrelevante Informationen zu gelangen. Durch die Automatisierung der Bereitstellung von Portscan-Ergebnissen verringert sich der manuelle Aufwand und Administratoren können schneller auf Sicherheitslücken reagieren. Es gibt Dienstleister auf dem Markt, die sich darauf spezialisiert haben, Sicherheitslücken in Unternehmensnetzen aufzuspüren, unter anderem auch durch Penetrationstests und Portscans. Die Ergebnisse solcher Dienstleister stellen eine mögliche Datenquelle für das zentrale Informationssystem dar.

6.2.5. Firewall-Zustand auswerten

Bei der Auswertung der Firewalls prüft das zentrale System zunächst, ob die Firewall aktiv ist und ein Whitelist-Ansatz verfolgt wird. Whitelist-Firewalls sind zwar anfangs umständlicher zu konfigurieren, aber der sicherere Ansatz. Die Überprüfung der Sicherheit ist dann auch für einen Algorithmus einfacher, da er die konkreten Regeln nur mit Referenzdaten vergleichen muss, um die Sicherheit der Firewall zu bewerten. Aus diesem Grund muss das zentrale Informationssystem eine Möglichkeit bieten, Referenzdaten zu verwalten.

Es existieren Tools auf dem Markt, die zwar die Erstellung von Firewall-Regeln erleichtern, zum Vergleichen von Regeln sind sie aber nur bedingt geeignet. Mit der algorithmischen Vergleichbarkeit von Regelsätzen haben sich vor einigen Jahren *Lu et al.* von der Wollongong-Universität in Australien beschäftigt und ein einen Lösungsvorschlag veröffentlicht [LU+ 07].

Je unterschiedlicher die Anwendungsfälle sind, die von den Servern abgedeckt werden, desto schwieriger ist es, Referenzdaten zu finden, die auf alle Server gleichermaßen anzuwenden sind. Die kleinste Schnittmenge an Firewall-Regeln, die auf allen Servern erlaubt ist, würde nur aus dem SSH-Zugriff für die Administratoren und einigen wenigen Standard-Netzprotokollen bestehen. Jede Abweichung davon würde als unsicher bewertet werden, was aber nicht unbedingt der Realität entspricht. Aus diesem Grund muss das Informationssystem

berücksichtigen, welche konkreten Dienste ein bestimmter Server anbietet und anhand dieser Information beurteilen, welche Verbindungen zusätzlich erlaubt sein dürfen.

Sofern das zentrale Informationssystem, wie bereits zuvor vorgeschlagen wurde, eine Kategorisierung der Server gestattet, können Referenzdaten verwendet werden, die spezifisch für jede einzelne Kategorie hinterlegt wurden. Dadurch lassen sich erlaubte Regeln typischer Netzdienste abdecken, zum Beispiel ein geöffneter Port 443 auf einem Webserver, der auf HTTPS-Verbindungen von außen wartet. Im einfachsten Fall würde lediglich geprüft werden, welche Ports geöffnet sind.

Falls zusätzlich der rechner- oder netzspezifische Zugriff auf bestimmte Ports geprüft werden soll, werden feingranularere Referenzdaten benötigt. Das erhöht zwar die Erkennungsleistung des zentralen Systems, sorgt aber gleichzeitig für einen deutlichen Verwaltungsaufwand. Je feingranularer die Referenzdaten sind, desto geringer ist die Anzahl der Server, mit denen sie verglichen werden können. Im schlimmsten Fall müsste auf dem zentralen System für jeden einzelnen Server eine eigene Regeldatei als Referenz abgelegt und gepflegt werden.

Die Sicherheit wäre in diesem Fall optimal, da jede kleinste Abweichung vom Sollzustand entdeckt wird. Dieser Ansatz ist aber nur dann praktikabel, wenn davon ausgegangen werden kann, dass sich die Regeln der einzelnen Firewalls nur selten ändern. Bei diesem Thema ist eine genaue Abwägung notwendig, um einen Kompromiss zwischen Erkennungsgenauigkeit und Handhabbarkeit zu finden.

6.2.6. Mögliche Visualisierung

Die Überprüfung der Firewall-Regeln erfolgt beim zentralen Informationssystem ebenfalls durch einen Soll-Ist-Vergleich. Entweder stimmen allen Regeln eines Servers mit den dazugehörigen Referenzdaten überein oder es gibt Abweichungen. Diese Abweichungen können unterschiedlich gewichtet werden. Eine Abweichung ist zum Beispiel als harmlos zu bewerten, wenn eine Whitelist Firewall einen Port öffnet, an dem überhaupt kein Prozess nach Verbindungsversuchen lauscht. Als schwerwiegend ist die komplette Funktionslosigkeit der Firewall einzustufen. Damit das Informationssystem den Sicherheitszustand korrekt visualisiert, ist ein genaues Bewertungsschema als Grundlage erforderlich.

Einsatz von Signalfarben

Soll-Ist-Vergleiche wurden bereits in ähnlicher Form im Abschnitt 6.1 erwähnt. Es liegt also nahe, für die Visualisierung der Firewall-Zustände ähnliche Methoden zu verwenden. Auf dem Startbildschirm der Anwendung kann zu diesem Zweck ein zusätzliches Ampelpiktogramm untergebracht werden, das mit den zuvor genannten Signalfarben arbeitet.

Um die Bedeutungen der Signalfarben von denen der Softwarestatus-Ampel unterscheiden zu können, muss dem Anwender eine Legende bereitgestellt werden. Um Visual Clutter zu vermeiden, sollte die Legende nur auf Anforderung durch den Anwender eingeblendet werden. Die Farben könnten die folgenden Bedeutungen haben:

Grün signalisiert den Optimalzustand, der aus einem abweichungslosen Soll-Ist-Vergleich resultiert. Die Farbe Rot kennzeichnet eine schwerwiegende Sicherheitslücke, z. B. den kompletten Ausfall einer Firewall. Übrig bleibt noch die Farbe Gelb für Ereignisse, deren Sicherheitsrelevanz zwischen den beiden zuvor genannten Fällen liegt. Sie kann für harmlosere Zustände verwendet werden wie offene Ports ohne lauschenden Dienst oder Chains, die nicht erreicht werden.

Server-Statistiken

Ähnlich wie im Bereich Software-Verwaltung sollte das Informationssystem dem Anwender Diagramme mit allgemeinen Kennzahlen zur Verfügung stellen. Eine denkbare Datenquelle sind zum Beispiel die durchschnittliche Anzahl geöffneter Ports pro Serverkategorie. Diese Daten lassen sich aufgrund der leichten Erfassbarkeit und Vergleichbarkeit als Säulendiagramme darstellen.

Anhand dieser Gegenüberstellung sieht der Anwender sofort, welche Serverkategorien potenzielle Einfallstore für Angriffe darstellen, denn je mehr Ports geöffnet sind, desto größer ist auch die Angriffsfläche. Wie bereits zuvor genannt wurde, hilft die Kategorisierung außerdem dabei, die für die betroffenen Server verantwortlichen Ansprechpartner zu ermitteln, um sie zu einer konstruktiven Lösung zu bewegen.

Die Anzahl der Server, die aufgrund ihrer Firewall-Konfiguration mit einer der zuvor genannten Ampelfarben gekennzeichnet sind, kann ebenfalls als Säulendiagramm abgebildet werden. Während der Startbildschirm nur eine einzige Ampelfarbe beinhaltet, die den kritischsten Firewall-Zustand aller betrachteten Server kennzeichnet, bietet dieses Säulendiagramm den Vorteil, dass die gesamte Sicherheitssituation besser bewertet werden kann. Wenn aus dem Diagramm ersichtlich wird, dass nur ein einziger Server in die Kategorie “kritisch” fällt, ist das bei weitem besser als eine zweistellige Anzahl.

Als optional einblendbare Alternative für das zuvor genannte Säulendiagramm bietet sich auch ein Kreisdiagramm an. Aus ihm lassen sich zwar keine absoluten Zahlen ablesen, dafür stechen die relativen Verhältnisse besser ins Auge. Beispiele für beide Diagramme sind in Abbildung 6.19 zu sehen.

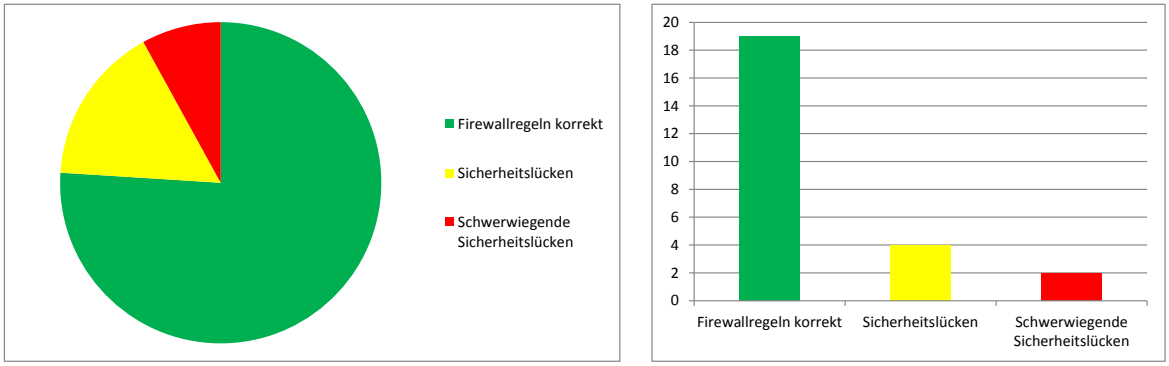


Abbildung 6.19.: Firewall-Status aller Server als Kreisdiagramm (links) und als Säulendiagramm (rechts).

Damit Administratoren schnell auf Bedrohungen reagieren können, müssen auch die bei diesem Anwendungsfall angezeigten Schaubilder Interaktivität bieten. Durch den Klick ins Diagramm muss die Anwendung direkt zur zoombaren Bedienoberfläche umschalten und ausschließlich die betroffenen Server anzeigen.

Zoombare Gesamtübersicht mit Filterung und Sortierung

Das Konzept der zoombaren Gesamtübersicht aller Server lässt sich ebenfalls auf Firewall-Zustände übertragen. Der Anwender hat die Möglichkeit, die Menge der angezeigten Server mit Hilfe von Filteroperationen einzuschränken und nach bestimmten Kriterien am Bildschirm anordnen zu lassen.

Als Kategorie für die Gruppierung bietet sich im Kontext der Firewall-Zustände zum Beispiel die Sicherheitszone an, in der sich ein Server befindet. Falls eine Institution mehrere DMZ-Netze betreibt, eignen sich diese Netze als Gruppierungskriterium. Die Serverkategorie ist eine weitere Gruppierungsmöglichkeit.

Wenn ein Server mehrere virtuelle Ethernet-Schnittstellen in unterschiedlichen Sicherheitszonen besitzt, gibt es zwei Möglichkeiten: Entweder wird er in einer speziellen Gruppe angezeigt oder er taucht mehrmals in der Gesamtübersicht auf, sofern mindestens zwei der betroffenen Gruppen eingeblendet werden. Um Verwirrungen zu vermeiden sollte auf jeden Fall die Möglichkeit bestehen, speziell nach Servern zu filtern, die mehrere Schnittstellen besitzen.

Um die Serverpiktogramme räumlich zu sortieren, bietet sich als Kategorie zum Beispiel die Anzahl geöffneter Ports an. Die Kategorie oder die Priorität der Server kann ebenfalls auf eine Achse des Koordinatensystems abgebildet werden. Weitere Möglichkeiten werden im Abschnitt 6.2.7 genannt.

Detailansicht

Um Ben Shneidermans Information Seeking Mantra gerecht zu werden, soll der Anwender nur so viele Details zu sehen bekommen, wie er tatsächlich sehen will. Das Problem von Firewall-Regelsätzen ist, dass sie sehr komplex werden können. Die Anzeige kompletter Regelsätze in der Detailansicht sollte nur eine Option sein, die dem Anwender der Vollständigkeit halber angeboten wird. Ebenso sollte die Option bestehen, durch *Syntax Highlighting* die Unterscheidbarkeit der einzelnen Elemente einer Regel zu erhöhen. Zum Bezug von Informationen über Firewall-Regeln sollte das Informationssystem aber primär Visualisierungen anbieten.

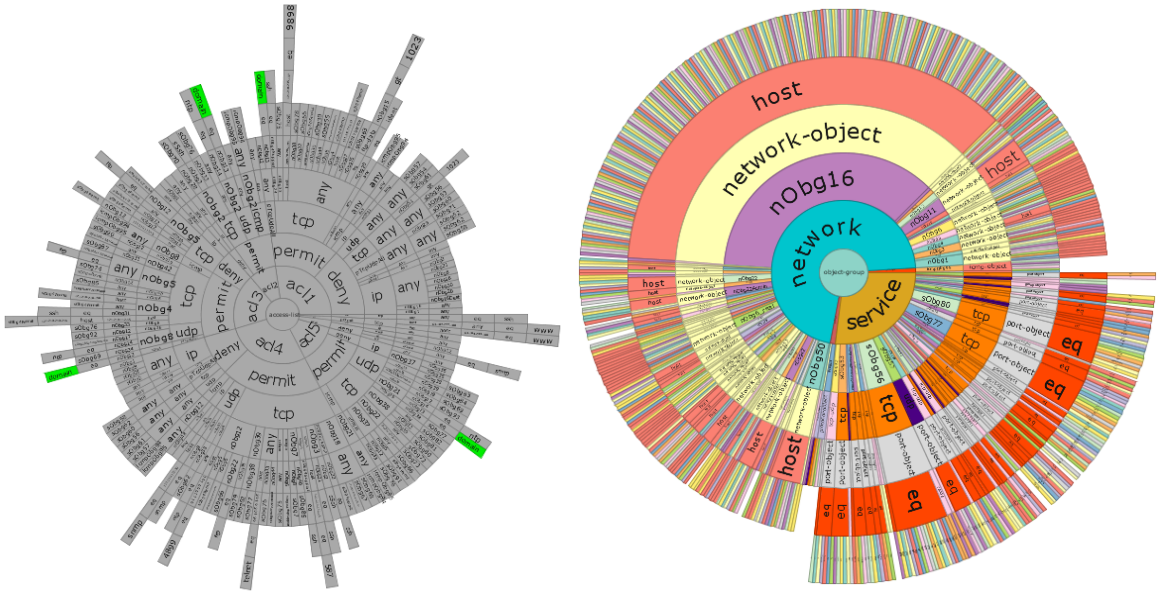


Abbildung 6.20.: Suche nach einem Schlüsselwort (links) und vollständig ausgeklappter Teilbaum des Regelwerks (rechts).

Bei der Generierung von Visualisierungen kann man sich die inhärente Eigenschaft von Firewall-Regeln zunutze machen, dass es sich dabei um strukturierte Texte handelt, die Gruppierungen von einzelnen Regeln und anderer struktureller Elemente wie etwa Ports ermöglichen. Ein konkretes Beispiel für die Visualisierung von Firewall-Regeln wurde bereits in Abbildung 6.1 gezeigt. Mansmann et al. verwenden ein sog. *Sonnenstrahldiagramm*, das einen Einblick in den hierarchischen Aufbau der Regeln gewährt [MAN+ 12].

Abbildung 6.20 zeigt weitere Screenshots von Mansmanns Visualisierungskonzept: Im linken Teilbild ist die Suche nach dem Schlüsselwort `domain` dargestellt. Die Treffer werden durch eine grüne Hintergrundfarbe hervorgehoben. Das rechte Teilbild zeigt einen vollständig aufgeklappten Teilbaum des Regelsatzes. Auf den ersten Blick wirkt dies sehr unübersichtlich, doch der Betrachter ist nicht gezwungen, alle verfügbaren Knoten zu expandieren. Durch die schrittweise Expansion von Objektgruppen vom Zentrum der Grafik nach außen wird die Visualisierung deutlich übersichtlicher und beinhaltet nur die Informationen, nach denen der Administrator tatsächlich sucht (siehe Abbildung 6.21).

6.2.7. Unterstützung proaktiver Tätigkeiten

Auch beim Thema Firewall muss der Administrator nicht nur auf sicherheitsrelevante Ereignisse reagieren. Das Informationssystem lässt sich auch zur proaktiven Planung von Tätigkeiten nutzen. Hier kommt die zoombare Gesamtübersicht mit ihrer Filter- und Sortierfunktion ins Spiel:

Es ist zum Beispiel denkbar, das Alter der letzten Änderung an der Firewall-Konfiguration als Sortierkriterium für die Serverpiktogramme zu verwenden. Diese Art der Sortierung bietet den Vorteil, dass sie auch dann funktioniert, wenn keine Referenzdaten auf dem Informationssystem vorliegen. Der Administrator sieht auf einen Blick, auf welchen Servern erst vor kurzen oder schon lange nicht mehr etwas an der Firewall-Konfiguration

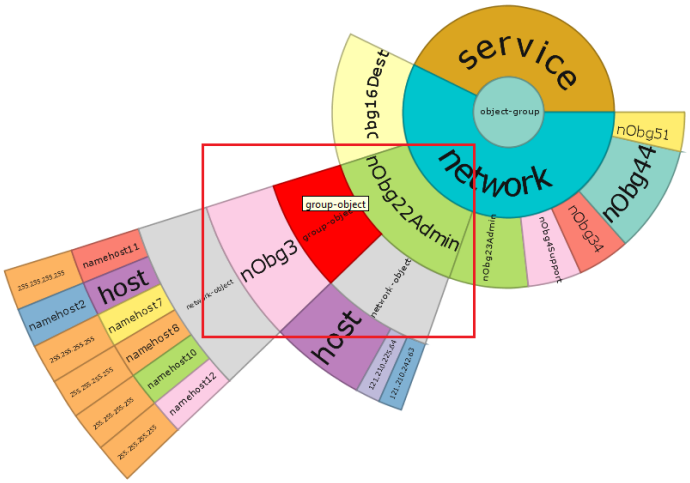


Abbildung 6.21.: Sukzessives Expandieren von Objektgruppen von innen nach außen.

geändert wurde. Bei älteren Versionen sollte sich auch ohne konkreten sicherheitsrelevanten Anlass ein Blick in die Firewall-Regeln lohnen. Änderungen, die erst kürzlich von Kollegen gemacht wurden, werden ebenfalls schneller sichtbar. Auch hier lohnt sich ein überprüfender Blick.

Für den Administrator von Interesse könnte auch die Größe der Firewall-Konfigurationsdateien sein. Je größer ein Regelsatz ist, desto komplizierter und fehleranfälliger ist er potenziell. Durch die Sortierung nach der Größe der Regelsätze sieht der Administrator sofort, welche Konfigurationsdateien eine Revision benötigen oder aufgeräumt werden sollten.

6.2.8. Unterstützung von Fehlersuche und Audits

Ein Informationssystem, das nicht nur Echtzeitdaten von Servern anzeigt, sondern auch historische Daten speichert und aufbereitet, bietet Vorteile für die Administration. Aus den aufgezeichneten Daten können Administratoren zum Beispiel nach einem Sicherheitsvorfall nachvollziehen, wie lange die Firewall-Konfigurationen der Server auf einem bestimmten Stand waren. Es könnte schließlich sein, dass lange Zeit alle Ampeln trotz einer Sicherheitslücke grün leuchteten, weil die Referenzdaten selbst nicht fehlerfrei waren.

Möglicherweise waren die Referenzdaten nicht von Anfang an fehlerhaft. Durch den Rückblick auf die historischen Daten können die fehlerhaften Referenzdaten besser überprüft werden. Sie können auch genutzt werden, um kontinuierlich die Qualität der Regeln zu optimieren. Zu diesem Zweck sind automatisierte Prüfmechanismen erforderlich. Ein Rückblick auf alte Firewall-Konfigurationen und frühere Referenzdateien ist auch für Sicherheitsaudits von Vorteil. Die Einhaltung von Sicherheitsstandards kann so rückwirkend beurteilt werden.

6.2.9. Erkennung von Trends

Auch aus allgemeinem Interesse können die historischen Daten von Nutzen sein. So ist es zum Beispiel denkbar, die durchschnittliche Anzahl geöffneter Ports einzelner Server oder ganzer Serverkategorien in Form eines Liniendiagramms zu visualisieren. Dort lassen sich Trends ablesen, die womöglich zum Überdenken des Sicherheitskonzepts führen können. Wenn im Laufe der Zeit immer mehr Ports geöffnet wurden, gibt es bestimmt Lösungen, um dem entgegenzuwirken und die Überschaubarkeit der Firewall-Regeln zu verbessern.

Durch die Speicherung von historischen Daten ist mit Hilfe des Informationssystems auch die Änderung der Größen der Regelsätze über die Zeit nachvollziehbar. In Form eines Liniendiagramms lassen sich so Wachstums- und Schrumpfungsphasen der Regelsätze einzelner Server visualisieren.

6.2.10. Gemeinsame Visualisierung der Anwendungsfälle

Da sich zur Visualisierung der Firewall-Zustände ähnliche Methoden wie beim Anwendungsfall Software-Verwaltung (siehe Abschnitt 6.1) anwenden lassen, ist es ein konsequenter Schritt, in einem Informationssystem diese beiden Anwendungsfälle miteinander zu kombinieren. Die ab Abschnitt 6.1.2 vorgestellte Methode mit Piktogrammen, die sich nach bestimmten Kriterien auf dem Bildschirm anordnen und beliebig heranzoomen lassen, kann auch durch Firewall-Informationen erweitert werden. Dies ist durch zusätzliche Visuelle Attribute möglich, die ab einem bestimmten Detailgrad sichtbar sind. Bei geringer Detailstufe werden die bekannten Signalfarben verwendet. Im Idealfall sollte der Anwender entscheiden können, welche Anwendungsfälle gemeinsam in der zoombaren Oberfläche visualisiert werden sollen. Indem er nur diejenigen Anwendungsfälle aktiviert, die ihn gerade interessieren, verringert er Visual Clutter und kann sich auf das Wesentliche konzentrieren.

Die gemeinsame Visualisierung verschiedener Anwendungsfälle bietet Vorteile gegenüber einer spezifischen Visualisierung für jeden Use Case: Indem der Anwender nicht ständig zwischen verschiedenen Ansichten hin- und herschalten muss, um sich Informationen zu den entsprechenden Use Cases einblenden zu lassen, spart er einerseits Zeit. Andererseits entgehen ihm keine sicherheitskritischen Meldungen eines Anwendungsfalls, wenn er ursprünglich einen anderen Anwendungsfall betrachten wollte. Schließlich kann es vorkommen, dass die Pakete auf allen Servern auf dem aktuellsten Stand sind, in einer Firewall-Konfiguration aber eine große Lücke klafft. In einer Gesamtübersicht, die beide Anwendungsfälle abdeckt, entgeht dem Administrator nichts.

Zuvor wurde vorgeschlagen, dass die gemeinsame Ansicht eine Auswahlmöglichkeit für die anzuzeigenden Anwendungsfälle bereitstellen soll. Dadurch entgehen dem Anwender ggf. wichtige Meldungen, die von einem ausgeblendeten Anwendungsfall stammen. Um dies zu verhindern, sollte ein Alarmierungsmechanismus vorhanden sein, der den Anwender trotz des deaktivierten Anwendungsfall über sicherheitskritische Zustandsänderungen von diesem Use Case informiert. Dies kann in Form eines Hinweisfensters erfolgen. Es sollte einen knappen Hinweis beinhalten und dem Administrator erlauben, mit nur einem Mausklick Informationen über den kritischen Anwendungsfall in die aktuelle Ansicht mit aufzunehmen. Über eine weitere Schaltfläche sollte das Hinweisfenster direkt zu einer Ansicht wechseln, das nur die kritischen Systeme des ausgeblendeten Anwendungsfalls beinhaltet. Dadurch gelangt der Administrator sofort ans Ziel.

6.3. Sicherheit privater X.509-Schlüssel

Die Geheimhaltung privater X.509-Schlüssel ist essentiell, um die Vertraulichkeit und Authentizität bei verschlüsselter Kommunikation zu gewährleisten. Dieser Abschnitt beschreibt, wie ein zentrales Informationssystem zur Sicherheit privater Schlüssel beitragen kann. Nähere Informationen zu privaten Schlüsseln und Schlüsselpaaren generell befinden sich in Abschnitt 3.4.1.

6.3.1. Vorüberlegungen

Server, die verschlüsselte Verbindungen von den Clients unterstützen sollen, greifen in der Regel auf SSL oder TLS zurück. Der Server stellt anfragenden Clients ein X.509-konformes Zertifikat bereit, das seinen öffentlichen Schlüssel, Informationen über den Server selbst und eine digitale Signatur vom Aussteller enthält. Den öffentlichen Schlüssel benötigen Clients, um verschlüsselte Nachrichten an den Server senden zu können und seine Identität sicherzustellen. Zusätzlich besitzt der Server einen privaten Schlüssel, den er zum Entschlüsseln an ihn gerichteter Nachrichten und zum Signieren eigener Nachrichten benötigt.

Sofern ein Server nicht auf ein Hardware Security Module (HSM)[9] zurückgreift, müssen die beiden Schlüssel persistent im Dateisystem abgelegt sein, damit sie auch nach einem Neustart noch vorhanden sind.

[9] Ein HSM ist ein Peripheriegerät, das kryptographische Operationen sicher und performant ausführt. HSMs liegen oft als Peripheral Component Interconnect Express (PCIe)-Erweiterungskarte oder als vernetzte Komponente vor. Viele kommerzielle Produkte ermöglichen die sichere Speicherung von privaten Schlüsseln. Private Keys können im laufenden Betrieb nicht aus einem HSM extrahiert werden. Um sie auch vor physikalischem Zugriff zu schützen, ist ein automatisches Löschen der Schlüssel, z. B. bei Öffnen des Gehäuses, möglich. Der Einsatz von HSMs wird unter anderem vom BSI zur Schlüsselspeicherung empfohlen [BSI 15].

Die Datei, die den privaten Schlüssel enthält, darf unter keinen Umständen an Unbefugte gelangen.[10] Aus diesem Grund muss eine Schlüsseldatei mit besonders restriktiven Dateizugriffsrechten ausgestattet sein.

In der Regel hat lediglich der Benutzer `root` Zugriff sowohl auf das Verzeichnis `/etc/ssl/private/` als auch auf die darin befindlichen Schlüsseldateien. Dies trifft auch für die Distribution openSUSE zu, die am LRZ hauptsächlich Verwendung findet. Bei Debian und seinen Derivaten gibt es zusätzlich noch die System-Benutzergruppe `ssl-cert`, die lediglich Lesezugriff für die privaten Schlüssel erhält.

6.3.2. Private Schlüssel aufspüren

Viele für den Serverbetrieb prädestinierten Linux-Distributionen besitzen Standard-Verzeichnisse zur Aufbewahrung der einzelnen Schlüssel. Öffentliche Schlüssel befinden sich bei Debian und diversen Derivaten im Ordner `/etc/ssl/certs/` und private Schlüssel in `/etc/ssl/private/`. openSUSE weicht etwas davon ab, da dort `/etc/ssl/certs/` kein echtes Verzeichnis, sondern ein symbolischer Link zu dem Ordner `/var/lib/ca-certificates/pem/` ist. Die Schlüssel liegen Privacy Enhanced Mail (PEM)-kodiert (*Base64*) vor.

Idealerweise ist auf einem Server nur ein Exemplar des privaten Schlüssels an einer zentralen Stelle gespeichert. Es kann aber auch sein, dass er an einem anderen Ort gespeichert ist oder gar mehrere Exemplare davon existieren. In beiden Fällen müssen alle Private Keys im Dateisystem aufgespürt werden, um ihre Dateizugriffsrechte zu prüfen.

Manche Administratoren neigen dazu, Sicherungskopien privater Schlüssel an einem bestimmten Ort, zum Beispiel innerhalb des `/root/`-Verzeichnisses anzulegen. Zu diesem Zweck legen sie zum Teil versteckte Ordner mit nichtssagenden Namen an, um darin die Backups zu speichern. Das ist aber nicht im Sinn eines private Keys, denn er sollte nur dort gespeichert sein, wo der Zugriff auf ihn erfolgt, und bei Außerdienststellung gelöscht werden.

Duplikate von privaten Schlüsseln können auch dann auftreten, wenn der Administrator die Datei zunächst per Secure Copy (SCP) oder SSH in ein temporäres Verzeichnis hochlädt und nach dem Kopieren der Datei an die richtige Stelle schlicht und einfach vergisst, die zuvor hochgeladene Datei wieder zu löschen. Besonders fatal ist dies, wenn sich die Datei in einem temporären Verzeichnis befindet, das keine besonderen Zugriffsbeschränkungen besitzt. Dies ist zum Beispiel bei Austauschordnern oder den `/tmp/`-Verzeichnissen der Fall, die sowohl weltlesbar als auch weltschreibbar sind. Dann ist das Abgreifen des Private Keys auch ohne Administratorrechte möglich.

Ein weiterer möglicher Grund für mehrere Private-Key-Dateien auf einem Server beruht auf der Tatsache, dass Zertifikate nur mit begrenzter Laufzeit ausgestattet werden und rechtzeitig durch neue ersetzt werden müssen. Wie bereits gesagt wurde, neigen manche Administratoren dazu, Sicherungskopien nicht mehr benötigter Schlüsselpaare auf dem Server anzulegen, zum Beispiel in einem versteckten Verzeichnis. Das ist nicht im Sinn der Informationssicherheit, sondern fahrlässig.

Ein Programm, das das komplette Dateisystem nach privaten Schlüsseln durchsucht, muss systembedingt auch mit Administratorrechten ausgeführt werden. Andernfalls kann es nicht alle Verzeichnisse durchsuchen. Es sollte nur Auskunft darüber geben, wo private Schlüssel gefunden wurden. Unter keinen Umständen darf es den Inhalt der Schlüssel auslesen und an das zentrale Informationssystem weiterleiten oder generell eine Schnittstelle nach außen bereitstellen, um den Inhalt abrufen zu können. Damit das zentrale System dennoch einzelne Schlüsselversionen voneinander unterscheiden kann, sollte stattdessen auf eine möglichst sichere Hash-Funktion wie zum Beispiel SHA-512 zurückgegriffen werden.

6.3.3. Sicherheit der Schlüssel prüfen

Sobald das Prüfprogramm den oder die Speicherorte festgestellt hat, muss es die Dateizugriffsrechte prüfen. Ein Ordner, der einen Private Key enthält, darf nur vom Systembenutzer `root` und ggf. einer speziellen

[10] Damit ein Angreifer an den privaten Schlüssel eines Servers gelangt, muss er nicht zwingend Zugriff auf sein Dateisystem besitzen. Der Private Key kann unter bestimmten Bedingungen unter Ausnutzung von Programm- oder Implementierungsfehlern auch über das Netz ausgelesen werden. So war es zum Beispiel bei der OpenSSL-Lücke Heartbleed der Fall [SCHM 14].

Gruppe geöffnet werden. Bei Debian ist gibt es zu diesem Zweck die Gruppe `ssl-cert`. Dies verhindert schon mal, dass andere Benutzer den Inhalt des Verzeichnisses sehen.

Die darin enthaltenen Schlüssel dürfen ebenfalls nur vom Benutzer `root` und ggf. dieser Gruppe gelesen werden. Um Manipulationen zu verhindern oder zumindest zu erschweren, darf kein Schreibzugriff erlaubt sein, auch nicht für `root`. Sollte er die Datei dennoch irgendwann löschen oder überschreiben müssen, nutzt er das Tool `chmod`, um den Schreibzugriff vorübergehend zu erlauben.

6.3.4. Mögliche Visualisierung

Im Anschluss werden mögliche Visualisierungen für den Anwendungsfall der Sicherheit privater X.509-Schlüssel angesprochen. Diese basieren ebenfalls auf den zuvor genannten Visualisierungen.

Einsatz von Signalfarben

Bei den zuvor genannten Anwendungsfällen basieren die Visualisierungen auf Soll-Ist-Vergleichen. Im Fall von Private Keys ist der Sollzustand gleichzeitig der Idealzustand, in dem jeweils nur ein aktuelles Exemplar eines privaten Schlüssels im Verzeichnis `/etc/ssl/private/` liegt. Zur Unterscheidung von Private Keys und ihrer Versionen nutzt das zentrale Informationssystem möglichst sichere Hash-Werte als Referenzdaten.

Zu den möglichen Abweichungen vom Sollzustand zählen die folgenden:

- Es existieren redundante Kopien eines Private Keys.
- Es sind mehrere Versionen eines Private Keys vorhanden.
- Die Zugriffsbeschränkungen sind nicht adäquat.

Da Soll-Ist-Vergleiche bereits in ähnlicher Form bei den vorherigen Use Cases verwendet wurden, liegt es nahe, sie auch für die Visualisierung der Sicherheit privater Schlüssel zu verwenden. Dies erfolgt durch ein zusätzliches Ampelpiktogramm auf der Startseite des Informationssystems.

Je mehr Ampelpiktogramme auf der Startseite vorkommen, desto wichtiger ist eine Legende mit den Bedeutungen der einzelnen Zustände. Die Einblendung der Legende sollte dem Anwender überlassen sein. Erstens werden dadurch die auf dem Bildschirm angezeigten Informationen auf das Notwendigste reduziert, zweitens kennt der Anwender die Bedeutung der einzelnen Farben jedes Anwendungsfalls aufgrund der ständigen Arbeit mit dem System irgendwann auswendig.

Grün signalisiert bei diesem Anwendungsfall den Optimalzustand, der aus einem abweichungslosen Soll-Ist-Vergleich resultiert. Die Farbe rot kennzeichnet eine schwerwiegende Sicherheitslücke. Dieser Fall tritt ein, sobald Zugriffsbeschränkungen unzureichend oder nicht vorhanden sind. Die Farbe Gelb wird für alle restlichen Zustände verwendet, die weder optimal noch schwerwiegend sind. Dies ist der Fall bei redundanten Kopien oder falls alte Schlüsselversionen gefunden wurden.

Server-Statistiken

Je nach Anwendungszweck ist nicht auf jedem Server ein eigenes Schlüsselpaar installiert. Dies ist zum Beispiel bei internen Webservern der Fall, die lediglich unkritische Informationen bereitstellen und keine benutzerspezifischen Werte wie etwa Zugangsdaten entgegennehmen. Andererseits können auf einem Server auch mehrere unterschiedliche private Schlüssel gespeichert sein. Ein Beispiel hierfür ist ein Apache-Webserver, der HTTPS-Verbindungen für verschiedene *Virtual Named Hosts* ermöglicht. Aus technischer Sicht spricht zwar nichts gegen die Verwendung vieler Private Keys pro Server. Sollte ein solcher Server aber gehackt werden, erhält der Angreifer im schlimmsten Fall Zugriff auf sämtliche Private Keys und kann damit – sofern kein PFS verwendet wurde – zuvor aufgezeichnete Kommunikationsinhalte trotz Transportverschlüsselung entschlüsseln.

Um Administratoren auf einem Blick zu vermitteln, wie hoch der Anteil der Server mit einer bestimmten Anzahl von Private Keys ist, bietet sich eine Art Histogramm an. Jede Säule des Histogramms steht für eine

mögliche Anzahl von Schlüsseln, beginnend mit der Zahl Null. Sollte es Server mit einer sehr großen Anzahl privater Schlüssel geben, kann die Anzahl der Histogrammklassen begrenzt werden, damit das Schaubild nicht zu viel Platz beansprucht. In diesem Fall würde die letzte Klasse keine obere Schranke besitzen.

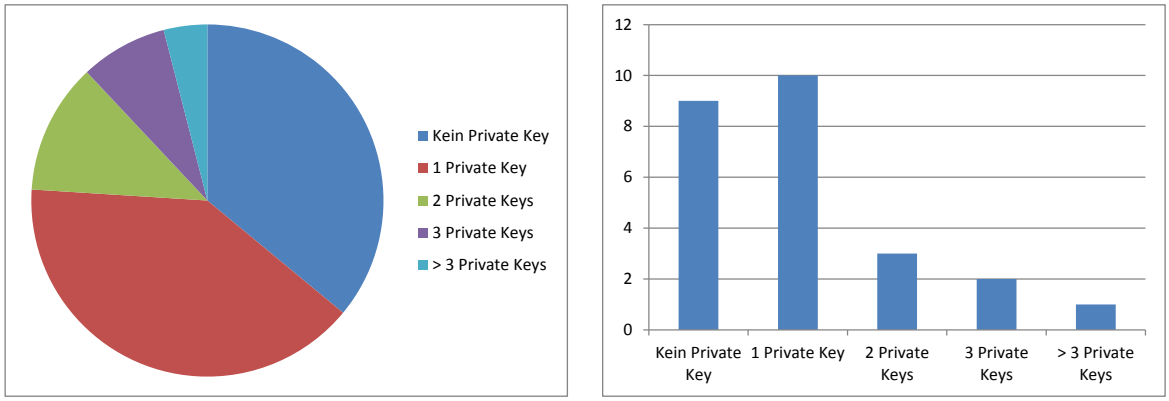


Abbildung 6.22.: Anteil der Server mit einer bestimmten Anzahl privater Schlüssel relativ (links) und absolut (rechts).

Zusätzlich zum Histogramm kann die Anwendung eine Visualisierung der relativen Anteile der Schlüsselanzahlen bereitstellen. Dies geschieht wie in den vorherigen Beispielen in Form eines Kreisdiagramms. Jedes Kreissegment steht dort für eine andere Anzahl von Schlüsseln pro Server. Ein Beispiel für beide Präsentationsformen ist in Abbildung 6.22 zu sehen.

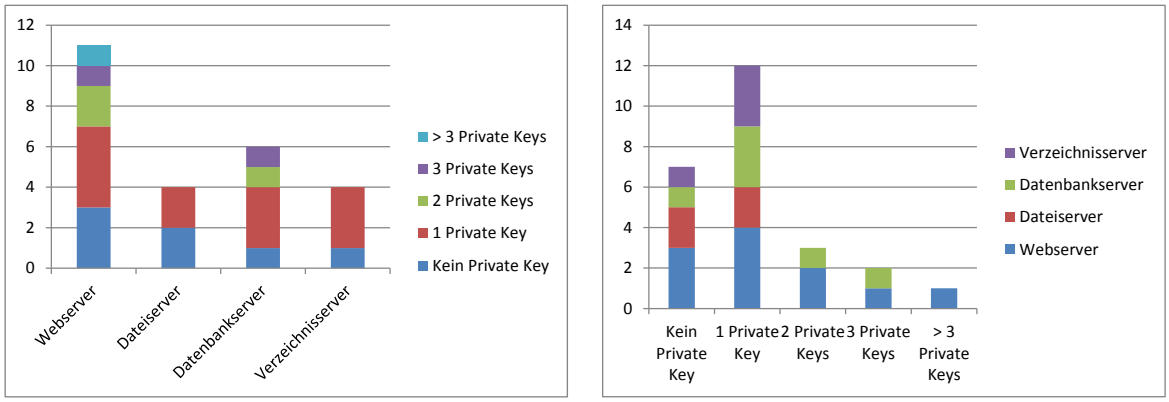


Abbildung 6.23.: Anteil der Server mit einer bestimmten Anzahl privater Schlüssel nach Serverkategorien (links) und nach Histogrammklassen sortiert (rechts).

Aus beiden Visualisierungen geht unmittelbar hervor, ob es Systeme mit einer hohen Anzahl privater Schlüssel gibt. Durch einen Klick auf die entsprechende Säule oder das Kreissegment wechselt das Informationssystem in eine Ansicht, die die konkreten Systeme anzeigt.

Sofern das Informationssystem in der Lage ist, Server nach Kategorien zu unterscheiden, kann diese Kategorisierung auch im Histogramm berücksichtigt werden. Ein Beispiel hierfür ist in Abbildung 6.23 zu sehen. Das rechte Teilbild entspricht dem vorherigen Histogramm, wurde aber um Serverkategorien erweitert. Die Anzahl der Server in einer Kategorie wird durch die Höhe des Stapels ausgedrückt.

Das linke Teilbild zeigt dieselben Daten, allerdings aufgeschlüsselt nach den Kategorien in der Abszisse. Während sich im rechten Teilbild die Gesamtanzahl der Server mit einer bestimmten Anzahl an Private Keys auf einen Blick ablesen lässt, sticht im linken Teilbild die gesamte Anzahl der Server einer bestimmten Klasse ins Auge.

Zoombare Gesamtübersicht mit Filterung und Sortierung

Um den Anwendungsfall *Sicherheit privater Schlüssel* in die zoombare Gesamtübersicht zu integrieren, ist zunächst zu ergründen, welche sinnvollen Filter- und Sortierkriterien es gibt. Für beide Fälle eignet sich zunächst die Anzahl der privaten Schlüssel, die auf einem Server gespeichert sind. Damit sind einerseits Server mit einer ungewöhnlich hohen Anzahl sofort erkennbar, andererseits findet man so schnell Systeme, die überhaupt keine Transportverschlüsselung unterstützen. Server ohne Transportverschlüsselung sollten nur zur Bereitstellung nicht-vertraulicher Daten eingesetzt werden bzw. zum Hosten unveränderlicher Daten. Dies ist zum Beispiel für Dokumentationen der Fall oder bei Archiven alter Intranetseiten.

Sobald personenbezogene Daten mit dem Server ausgetauscht werden, ist Verschlüsselung einzusetzen. Die mit Hilfe des Informationssystems herausgefilterten Server ohne Private Key könnten in einem weiteren Filterprozess nach Server-Kategorie, wie zum Beispiel nach Web Servern, gefiltert werden. In regelmäßigen Zeitabständen macht diese Kontrolle Sinn, um Server aufzuspüren, bei denen ggf. Verschlüsselung nachgerüstet werden muss.

Wer A sagt, muss auch B sagen. Im Zusammenhang mit diesem Use Case muss das zentrale Informationssystem deshalb auch Informationen über die öffentlichen Schlüssel und Zertifikate visualisieren, da Private Keys alleine nicht aussagekräftig genug sind. Eine für den Administrator in diesem Zusammenhang sinnvolle Filterfunktion wäre die Suche nach Servern, die ein Zertifikat von einem oder von mehreren bestimmten Ausstellern besitzen. Dies ist für den Fall nützlich, dass eine Zertifizierungsstelle den Dienst einstellt und zukünftige Zertifikate nur noch von einer neuen Stelle signiert werden sollen. Nützlich wäre diese Filterfunktion auch, um selbst signierte Zertifikate aufzuspüren, damit sie in einem Aufgabenpaket gebündelt durch vertrauenswürdigere Zertifikate ersetzt werden können, die von einer CA unterzeichnet wurden.

Ein Kriterium, das besonders für proaktive Zwecke interessant ist, ist die Gültigkeitsdauer der Serverzertifikate. Mit Hilfe des Informationssystems kann der Administrator dann alle Server abfragen, die beispielsweise in den nächsten drei Monaten ein neues Zertifikat benötigen. Es ist für Dienstanbieter essentiell, stets auf eine derartige Übersicht zurückgreifen zu können, um rechtzeitig die Zertifikate auszutauschen. Anwender können zwar auch im Fall eines abgelaufenen Zertifikats den Dienst nutzen, sie reagieren aber mehr oder weniger verunsichert, wenn sie beim Webseitenabruf eine TLS-Warnmeldung des Browsers zu sehen bekommen. Die häufige Konfrontation mit unverständlichen Warnmeldungen führt bei vielen Anwendern dazu, dass sie die Meldungen irgendwann nur noch ignorieren und ungelesen wegklicken. Dazu sollen Dienstanbieter nicht auch noch beitragen. Selbst für den Fall, dass sie die Warnmeldung verstehen, haben sie dennoch keine Sicherheit, ob sie mit dem richtigen Server verbunden sind. In jedem Fall ist es besser, jederzeit gültige Zertifikate auf den Servern bereitzustellen.

6.4. TLS-Konfiguration

Um eine sichere Kommunikation zwischen Clients und Servern zu ermöglichen, wird in den meisten Fällen auf das Transportverschlüsselungsverfahren TLS zurückgegriffen. Mögliche Anwendungsbereiche sind zum Beispiel Webserver oder Mailserver. Dieser Abschnitt beschreibt, wie ein zentrales Informationssystem mit Hilfe von Methoden der Informationsvisualisierung zur Optimierung der TLS-Konfiguration verwendet werden kann.

Da sich die TLS-Konfiguration von Dienst zu Dienst unterscheidet, konzentriert sich dieser Abschnitt auf einen möglichst weitverbreiteten Anwendungsfall von TLS. Die Wahl fällt hier auf die Absicherung des Webservers *Apache*, der laut einer aktuellen Erhebung von *Netcraft* der weitverbreitetste Webserver ist (siehe Abbildung 6.24) [NETC 15].

6.4.1. Vorüberlegungen

Nachdem im vorherigen Abschnitt die Sicherheit privater Schlüssel besprochen wurde, kommt an dieser Stelle die Anwendung der Private Keys zum Zug: Das Transportverschlüsselungsverfahren TLS greift auf den priva-

[11] Datenquelle: `http://news.netcraft.com/archives/2015/01/15/january-2015-web-server-survey.html`

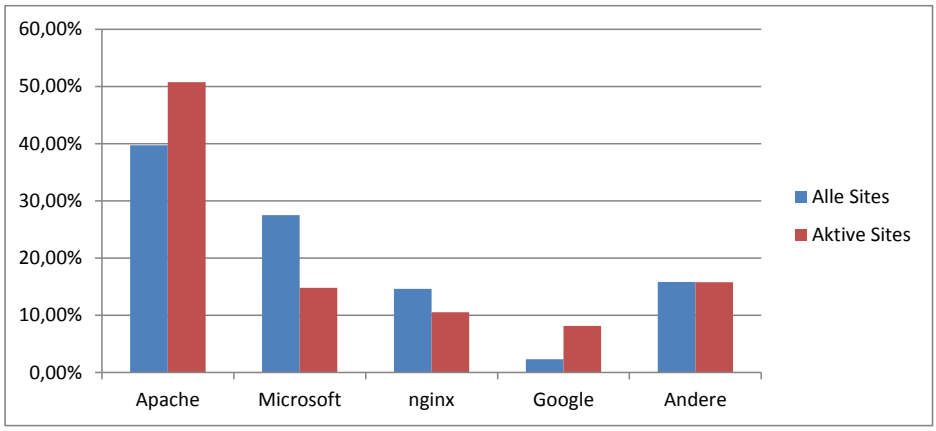


Abbildung 6.24.: Marktanteile der populärsten Webserver im Januar 2015.[11]

ten Schlüssel und das Serverzertifikat zu. Die Sicherheit der Schlüssel zu gewährleisten reicht nicht aus, um TLS-Verbindungen ebenfalls sicher zu machen. Das ist dadurch begründet, dass beim Server eine Vielzahl an Konfigurationsmöglichkeiten speziell für die Transportverschlüsselung existiert.

Zunächst steht der Administrator vor der Entscheidung, welche Versionen von TLS oder SSL der Server beim Verbindungsaufbau bereitstellen soll. Als nächstes besteht die Wahl zwischen unterschiedlichen *Cipher Suites*. Sie beschreiben Kombinationen kryptographischer Algorithmen für folgende Teilaufgaben von TLS:

- Authentifizierung
- Message Authentication Code (MAC)-Algorithmus
- Schlüsselaustauschverfahren
- Symmetrisches Verschlüsselungsverfahren

Je nach den verwendeten Teilalgorithmen und den Längen von Schlüsseln oder Hashwerten ist eine Cipher Suite als mehr oder weniger sicher einzustufen. In einer aktuellen Veröffentlichung beschreibt die Internet Engineering Task Force (IETF) den Stand der Technik, gibt Empfehlungen ab und warnt außerdem vor Algorithmen, die aus Sicherheitsgründen nicht mehr eingesetzt werden dürfen [IETF 15].

Es gibt bei Apache noch andere Konfigurationsparameter für TLS. Dazu zählen zum Beispiel die optionale Kompression, HTTP Strict Transport Security (HSTS) oder die Vorgabe der Reihenfolge der präferierten Cipher Suites durch den Server. In der Bewertung des Sicherheitszustands des Webservers muss das zentrale Informationssystem diese Parameter berücksichtigen.

6.4.2. TLS-Konfigurationen aufspüren

In einer heterogenen Linux-Serverlandschaft sind nicht alle Konfigurationsdateien eines Dienstes am selben Platz. Dies gilt auch für den Webserver Apache. Bei openSUSE sind die TLS-Einstellungen in der Regel bei den Konfigurationsdateien der Virtual Hosts zu finden. Diese befinden sich in der Regel im Verzeichnis `/etc/apache2/vhosts.d/` und besitzen die Dateiendung `.conf`. Bei dieser Distribution genügt es demnach, diese Dateien nach TLS-spezifischen Konfigurationsparametern zu durchsuchen.

Bei Debian und vielen seiner Derivate wird das Konzept verfolgt, dass die Virtual-Host-Konfigurationen in sog. *Sites* erfolgen, die mit den Befehlen `a2ensite` und `a2dissite` individuell aktiviert und deaktiviert werden können. Aus diesem Grund stellt Apache unter Debian zwei spezielle Ordner bereit:

/etc/apache2/sites-available/ Dieser Ordner beinhaltet alle zur Verfügung stehenden Sites.

/etc/apache2/sites-enabled/ Falls hier ein Symlink zu einer Konfigurationsdatei des zuvor genannten Verzeichnisses existiert, ist die *Site* aktiv.

Prinzipiell würde es genügen, bei Debian im Verzeichnis `sites-enabled` nach Konfigurationsparametern für TLS zu suchen. Es ist aber nicht ausgeschlossen, dass im Verzeichnis `sites-available` auch relevante Konfigurationsparameter vorhanden sind und die betroffenen Sites später von irgendjemandem aktiviert werden. Aus diesem Grund sollte auch dieses Verzeichnis bei der Suche in Betracht gezogen werden. Eine spätere Unterscheidung von aktivierten und deaktivierten Sites im zentralen Informationssystem sollte dennoch stattfinden, um die aktuelle Sicherheitslage von einem potenziellen Zustand unterscheiden zu können. Dies ist bei der Priorisierung von Aufgaben dienlich.

Die für die Transportverschlüsselung verantwortlichen Konfigurationsparameter können daran erkannt werden, dass sie alle mit dem Präfix `SSL` beginnen. Das Prüfprogramm muss nach diesen Schlüsseln suchen und sie zusammen mit den Werten an das zentrale Informationssystem übertragen. In Listing 6.7 ist der Ausschnitt einer Virtual-Host-Konfigurationsdatei zu sehen, die den für TLS relevanten Teil beinhaltet.

Listing 6.7: TLS-Konfiguration eines Virtual Hosts bei Apache

```
1  <VirtualHost *:443>
2      SSLEngine on
3      SSLProtocol -SSLv2 -SSLv3 -TLSv1 -TLSv1.1 +TLSv1.2
4      SSLCipherSuite ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:
           ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-
           GCM-SHA384:DHE-RSA-AES128-GCM-SHA256
5      SSLHonorCipherOrder on
6      SSLCompression off
7      SSLCertificateFile /etc/ssl/certs/myserver.crt
8      SSLCertificateKeyFile /etc/ssl/private/myserver.key
9      Header always set Strict-Transport-Security "max-age=31536000;
           includeSubDomains"
10     # ...
11 </VirtualHost>
```

6.4.3. Sicherheit der TLS-Konfigurationen prüfen

Die TLS-Konfiguration beim Web Server Apache umfasst die Protokollversion, die vom Server unterstützten Cipher Suites und diverse andere Konfigurationsparameter. Das Prüfprogramm muss diese Parameter auswerten, um die Sicherheit der TLS-Konfiguration zu bestimmen.

Damit das zentrale Informationssystem die Sicherheit der TLS-Konfigurationen der einzelnen Server beurteilen kann, ist zunächst ein Bewertungsmaßstab erforderlich. Das System muss Informationen darüber besitzen, welche kryptographische Algorithmen als sicher gelten, welche bedenklich sind und welche nicht mehr verwendet werden dürfen. Als Richtlinie können Administratoren hierfür auf aktuelle technische Richtlinien wie etwa vom BSI [BSI 15] oder von der IETF [IETF 15] zurückgreifen.

In der zuletzt genannten Richtlinie der IETF sind die bekannten Kryptographiealgorithmen in unterschiedliche Kategorien eingestuft, die bei der Aufstellung des Bewertungsmaßstabs behilflich sind. Algorithmen der Kategorie *MUST NOT* gelten als unsicher und dürfen unter keinen Umständen mehr verwendet werden. Überholte, aber immer noch ausreichend sichere Algorithmen werden als *SHOULD NOT* eingestuft. Darüber hinaus werden Empfehlungen mit *SHOULD* und alternativlose Verfahren mit *MUST* gekennzeichnet.

Die Empfehlungen für bestimmte Kryptographiesysteme, die in dieser Masterarbeit genannt werden, spiegeln den Stand der ersten Jahreshälfte von 2015 wider. Unabhängig von den in diesem Kapitel vorgeschlagenen Methoden der Informationsvisualisierung sollte bei der Implementierung dieses Anwendungsfalls in einem möglichen Produktivsystem darauf geachtet werden, auf Kryptographiealgorithmen zurückzugreifen, die zum jeweiligen Zeitpunkt als sicher bewertet werden.[12]

Bei der Auswahl der Kryptographiealgorithmen müssen Administratoren ebenfalls den Nutzerkreis der Dienste im Blick behalten. In einem Unternehmensnetz hat der Administrator im Normalfall die Kontrolle über

[12] Für nahezu jeden Kryptoalgorithmus existiert eine spezielle Angriffsmethode. Aus diesem Grund muss immer abgewogen werden, wie realistisch im Wirkbetrieb der Angriff auf einen bestimmten Kryptoalgorithmus unter Berücksichtigung der aktuellen technischen Möglichkeiten ist und wie realistisch er in den folgenden Jahren vermutlich sein wird.

die auf den Arbeitsplatzrechnern installierten Programme. Dort kann er sicherstellen, dass die Anwendungen auf diesen Rechnern die sicheren Kryptographiealgorithmen der eigenen Server auch unterstützen und der Nutzung der Dienste nichts im Weg steht.

Anders sieht es bei Betreibern öffentlich zugänglicher Webserver aus: Je sicherer ein Server konfiguriert ist, desto weniger Kryptographiealgorithmen akzeptiert er. Dies führt dazu, dass potenzielle Dienstnutzer mit älterer Client Software ausgesperrt werden. Besonders Betreiber von Suchmaschinen, Online-Händler und Banken sind von dieser Problematik betroffen und nehmen eine geringe Sicherheit in Kauf.

Da die Sicherheitsbewertungen ihrer Webserver in öffentlich zugänglichen Portalen wie *SSL Server Test* (siehe Abschnitt 5.1.2) abrufbar sind, veranlassen einige Betreiber mit schlechten Ergebnissen den Ausschluss ihrer Server von diesen Bewertungsportalen. Der Software-Architekt und Microsoft Most Valuable Professional (MVP) *Troy Hunt* hatte erst im Mai 2015 von einem derartigen Fall in seinem Blog berichtet [HUNT 15]. Es ist nur im Interesse der Kunden, dass derartige Vertuschungsaktionen einen Streisand-Effekt[13] nach sich ziehen.

TLS-Version

Beide der zuvor genannten technischen Richtlinien rufen dazu auf, nach Möglichkeit nur noch TLS Version 1.2 zuzulassen. Alle anderen Protokollversionen müssen in diesem Fall deaktiviert werden. Um diesen Umstand zu verifizieren, muss das Prüfprogramm in der Virtual-Host-Konfiguration nach dem Eintrag `SSLProtocol` suchen und die angehängten Parameter auswerten.

Mögliche Parameter sind `all`, `SSLv2`, `SSLv3`, `TLSv1`, `TLSv1.1` und `TLSv1.2`. Jedem Parameter kann ein Minuszeichen vorangestellt werden, um das entsprechende Protokoll auszuschließen. Ein Pluszeichen bewirkt das Gegenteil davon und ist optional. Ein Beispiel hierfür ist in Listing 6.7 in Zeile 3 zu sehen. Durch die Auswertung dieser Parameter kann das Prüfprogramm die vom Server erlaubten Protokollversionen unterstützen.

Es gibt Sonderfälle, bei denen die Parameter von `SSLProtocol` nicht eindeutig auf die vom Server angebotenen TLS-Protokollversionen schließen lassen. Ein Beispiel ist das Debian-Derivat Ubuntu in der Version 12.04 Long Term Support (LTS). Aufgrund des Langzeitsupports bis Mitte 2017 ist diese Version von Ubuntu noch auf zahlreichen Servern installiert. Die im Repository enthaltene Version 2.2.22 des Web Servers Apache unterstützt offiziell noch kein TLS 1.2. Der Support für TLS 1.2 wurde allerdings bei Ubuntu durch einen Backport nachgereicht. Als problematisch erweist sich die Konfiguration dieser speziellen Apache-Version: Einzelne TLS-Versionen lassen sich dort nicht selektiv aktivieren oder deaktivieren. Es gibt nur den `SSLProtocol`-Parameter `TLSv1`. Dieser aktiviert gleichzeitig alle Versionen bis einschließlich 1.2.[14] Ältere Versionen können deshalb nicht abgeschaltet werden. Angreifer profitieren davon, indem sie einen Verbindungsaufbau mit einer veralteten Protokollversion erzwingen und bekannte Schwachstellen ausnutzen.

Cipher Suites

Um die Sicherheit weiter zu erhöhen, dürfen nur besonders sichere Cipher Suites beim Verbindungsaufbau zugelassen werden. In der Konfiguration von Apache erfolgt dies durch den Konfigurationsparameter von `SSLCipherSuite`. Dort werden die bevorzugten Kombinationen eingetragen (siehe Zeile 4 in Listing 6.7). Die fünfte Zeile dient dazu, dass der Server beim Verbindungsaufbau seine eigene bevorzugte Reihenfolge der verfügbaren Cipher Suites forciert. Andernfalls würde er sich nach dem Client richten. Das erhöht die Sicherheit zusätzlich.

Die in der Beispielkonfiguration erlaubten Cipher Suites schreiben als Schlüsselaustauschverfahren Elliptic curve Diffie-Hellman Ephemeral (ECDHE) vor, ansonsten das weniger performante Diffie-Hellman Ephemeral (DHE) als Fallback-Lösung. In beiden Fällen unterstützt der Server PFS: Selbst nach Ausspähen des privaten Server-Schlüssels können Angreifer zuvor aufgezeichneten Datenverkehr nicht mehr entschlüsseln.

[13] "Als Streisand-Effekt wird ein Phänomen bezeichnet, wonach der Versuch, eine unliebsame Information zu unterdrücken oder entfernen zu lassen, öffentliche Aufmerksamkeit nach sich zieht und dadurch das Gegenteil erreicht wird, nämlich dass die Information einem noch größeren Personenkreis bekannt wird." (Quelle: `http://de.wikipedia.org/wiki/Streisand-Effekt`)

[14] Quelle: `https://bugs.launchpad.net/ubuntu/+source/apache2/+bug/1400473`

Die Beispielkonfiguration folgt dem Vorschlag der IETF, Advanced Encryption Standard (AES) mit Galois/-Counter Mode (GCM) einzusetzen. Es muss allerdings berücksichtigt werden, dass diese Kombination bisher nur von TLS 1.2 unterstützt wird. Die SHA-Hashfunktionen setzen in der Beispielkonfiguration Hashlängen von mindestens 256 Bit voraus.

Sonstige TLS-Konfigurationsparameter

Es gibt Seitenkanalangriffe wie zum Beispiel Compression Ratio Info-leak Made Easy (CRIME), die Eigenschaften von Datenverkehr ausnutzen, der durch TLS komprimiert wurde. Mit der Hilfe von CRIME kann es Angreifern gelingen, trotz aktiver Transportverschlüsselung Benutzersitzungen zu übernehmen und danach weitere Angriffe durchzuführen [FIS 12]. Da eine Datenkompression auch auf der Anwendungsschicht durch HTTP durchgeführt werden kann, ist es ratsam, die TLS-Kompression zu deaktivieren. Dies geschieht durch Zeile 6 in Listing 6.7.

Ein möglicher *Downgrade-Angriff* besteht darin, dass Kriminelle unverschlüsselte HTTP-Verbindungen erzwingen, obwohl der Anwender ursprünglich eine verschlüsselte Verbindung zum Webserver aufgebaut hat. Mit Hilfe von HSTS kann der Server den Browser dazu zwingen, sämtliche Verbindungen zu ihm nur per HTTPS herzustellen. In der Beispielkonfiguration wird HSTS in Zeile 9 aktiviert. Der Browser soll sich die Regel ein Jahr lang merken und auch auf Subdomänen anwenden.[15]

Ein weiterer erst Ende Mai 2015 bekannt gewordener Downgrade-Angriff ist *Logjam*. Er nutzt Implementierungsschwächen des Diffie-Hellman (DH)-Schlüsselaustauschverfahrens aus. Details zu diesem möglichen Angriff haben *David Adrian* et al. in einem Paper vorgestellt [ADR+ 15].

Signatur-Hashalgorithmus

Obwohl die Hashfunktion SHA-1 seit dem Jahr 2005 als anfällig für Kollisionsangriffe gilt, wurde sie bis heute noch als Hashalgorithmus zur Signatur von Serverzertifikaten eingesetzt [SCHE 14C]. Um die Sicherheit der Server zu verbessern, muss das Prüfprogramm in der Lage sein, den bei Zertifikaten verwendeten Signaturhashalgorithmus zu bestimmen.

Zu diesem Zweck ermittelt das Programm den Pfad zur Zertifikatdatei anhand der Virtual-Host-Konfiguration (in folgenden Beispiel vereinfacht `server.crt`). Danach prüft es den Hashalgorithmus mit dem Programm `openssl`:

```
1 openssl x509 -in server.crt -text | grep "Signature Algorithm"
```

Wenn die Bildschirmausgabe folgendermaßen lautet, handelt es sich noch um SHA-1 und es herrscht Handlungsbedarf:

```
1 Signature Algorithm: sha1WithRSAEncryption
```

Von der weiteren Verwendung von SHA-1 ist dringend abzuraten. Aktuelle Webbrowser warnen Anwender bereits vor HTTPS-Webseiten, die SHA-1 noch einsetzen. Bei Mozilla Firefox [MOZ 14b] und Google Chrome [PAL 14] sind stufenweise Verschärfungen der Warnungen geplant. Bei Firefox 36 erscheint der Warnhinweis momentan noch gut versteckt in der Web Console bei den Entwicklungswerkzeugen. Normale Anwender dürften damit nie in Kontakt kommen. Abbildung 6.25 zeigt den Screenshot einer solchen Warnmeldung.

[15] "Jede Medaille hat zwei Seiten" – Dieses Sprichwort gilt auch für HSTS. Die eigentlich sinnvolle Sicherheitsfunktion kann auch für Tracking missbraucht werden. Serverbetreiber können sog. *Supercookies* erstellen, die Browsern sogar im privaten Modus eindeutig zugeordnet werden können [EIK 15C].

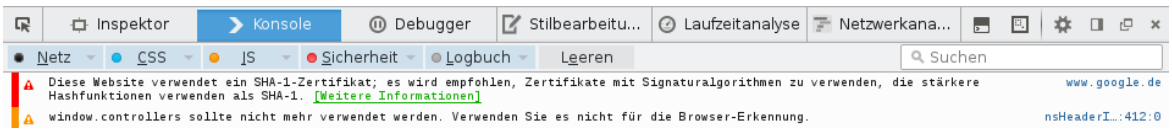


Abbildung 6.25.: Mozilla Firefox warnt in der Web Console vor Zertifikaten mit SHA-1 als Signaturhashalgorithmus.

6.4.4. Alternative Prüfmöglichkeiten

Bisher wurde das Sicherheitsniveau der TLS-Konfiguration nur durch das Betrachten der Konfigurationsdateien ermittelt. Dieser Ansatz liefert nur dann zuverlässige Ergebnisse, wenn der Webserver-Prozess auch tatsächlich die Einstellungen aus diesen Dateien ausgelesen hat. Um dies sicherzustellen, müsste das Prüfprogramm vor dem Auslesen dieser Dateien Apache dazu auffordern, diese Dateien erneut einzulesen. Zusätzlich muss es sicherstellen, dass das Einlesen auch funktioniert hat, indem es eventuell auftretende Fehlermeldungen von Apache berücksichtigt.

Die Ermittlung der TLS-Konfiguration kann aber auch erfolgen, indem das Prüfprogramm einen Webclient simuliert und so verschiedene Einstellungen überprüft. Dies ist zwar aufwendiger und lastet den Server während des Prüfprozesses etwas aus, die ermittelten Ergebnisse entsprechen aber dem realen Verhalten des Webservers. Es ist schließlich denkbar, dass der Webserver aufgrund eines Softwarefehlers ein anderes Verhalten an den Tag legt, als es seine Konfigurationsdateien vermuten lassen.

Als ein mögliches Werkzeug zum Ermitteln des tatsächlichen TLS-Verhaltens eignet sich das Kommandozeilentool `openssl`. Es stellt TLS-Verbindungen zum Webserver unter der Berücksichtigung der vom Benutzer angegebenen Parameter her und bereitet die Antworten des Servers in einer Form auf, die sowohl für Menschen lesbar als auch für Programme verwertbar ist.

Listing 6.8: Fehlgeschlagener Verbindungsversuch bei einem Test mit openssl.

```
root@server # openssl s_client -connect localhost:443 -ssl3
CONNECTED(00000003)
140658876614288:error:1409E0E5:SSL routines:SSL3_WRITE_BYTES:ssl handshake
    failure:s3_pkt.c:618:
---
no peer certificate available
---
No client certificate CA names sent
---
SSL handshake has read 0 bytes and written 0 bytes
---
New, (NONE), Cipher is (NONE)
Secure Renegotiation IS NOT supported
Compression: NONE
Expansion: NONE
SSL-Session:
    Protocol  : SSLv3
    Cipher    : 0000
    Session-ID:
    Session-ID-ctx:
    Master-Key:
    Key-Arg   : None
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    Start Time: 1431166849
    Timeout   : 7200 (sec)
    Verify return code: 0 (ok)
---
```

In Listing 6.8 ist ein Beispiel für die Verwendung des Programms `openssl` zu sehen. Das Kommando in der ersten Zeile löst einen Verbindungsaufbau zum Webserver auf demselben Host aus und gibt die nicht mehr als sicher zu betrachtende SSL-Version 3 als Voraussetzung für den Handshake vor. Da der Server diese Protokollversion nicht mehr unterstützt, sendet er eine entsprechende Fehlermeldung zum Client. Dadurch scheitert der Verbindungsaufbau (Zeile 3).

Mit etwas Programmieraufwand lässt sich `openssl` beispielsweise auch verwenden, um die Unterstützung einzelner Cipher Suites eines Servers zu testen. Einfacher geht es mit dem Portscanner `nmap`, wie in Listing 6.9 demonstriert wird.

Der Kommandozeilenparameter `--script ssl-enum-ciphers` weist `nmap` an, die verfügbaren Cipher Suites auf demselben Host zu überprüfen. Die Prüfung erfolgt für jede Protokollversion von TLS separat. Aus der Bildschirmausgabe des Programms geht hervor, dass der getestete Host nur Verbindungen via TLS 1.1 und 1.2 zulässt (Zeile 15 und 28).

Neben den Cipher Suites jeder Protokollversion (Zeile 17 – 25 bzw. 30 – 50) gibt der Portscanner eine Bewertung ab (`strong`). Eine Gesamtbewertung der Sicherheit der Cipher Suites erfolgt in Zeile 53.

Listing 6.9: Ermittlung der unterstützten Cipher Suites eines Servers mit dem Portscanner nmap.

```
1  root@server # nmap --script ssl-enum-ciphers localhost
2
3  Starting Nmap 6.47 ( http://nmap.org ) at 2015-05-09 12:25 CEST
4  Nmap scan report for localhost (127.0.0.1)
5  Host is up (0.000013s latency).
6  Other addresses for localhost (not scanned): 127.0.0.1
7  Not shown: 995 closed ports
8  PORT    STATE SERVICE
9  22/tcp  open  ssh
10 80/tcp  open  http
11 111/tcp open  rpcbind
12 443/tcp open  https
13 | ssl-enum-ciphers:
14 |   TLSv1.1:
15 |     ciphers:
16 |       TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA - strong
17 |       TLS_DHE_RSA_WITH_AES_128_CBC_SHA - strong
18 |       TLS_DHE_RSA_WITH_AES_256_CBC_SHA - strong
19 |       TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA - strong
20 |       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA - strong
21 |       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA - strong
22 |       TLS_RSA_WITH_3DES_EDE_CBC_SHA - strong
23 |       TLS_RSA_WITH_AES_128_CBC_SHA - strong
24 |       TLS_RSA_WITH_AES_256_CBC_SHA - strong
25 |     compressors:
26 |       NULL
27 |   TLSv1.2:
28 |     ciphers:
29 |       TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA - strong
30 |       TLS_DHE_RSA_WITH_AES_128_CBC_SHA - strong
31 |       TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 - strong
32 |       TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 - strong
33 |       TLS_DHE_RSA_WITH_AES_256_CBC_SHA - strong
34 |       TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 - strong
35 |       TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 - strong
36 |       TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA - strong
37 |       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA - strong
38 |       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 - strong
39 |       TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 - strong
40 |       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA - strong
41 |       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 - strong
42 |       TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 - strong
```

```
43 |       TLS_RSA_WITH_3DES_EDE_CBC_SHA - strong
44 |       TLS_RSA_WITH_AES_128_CBC_SHA - strong
45 |       TLS_RSA_WITH_AES_128_CBC_SHA256 - strong
46 |       TLS_RSA_WITH_AES_128_GCM_SHA256 - strong
47 |       TLS_RSA_WITH_AES_256_CBC_SHA - strong
48 |       TLS_RSA_WITH_AES_256_CBC_SHA256 - strong
49 |       TLS_RSA_WITH_AES_256_GCM_SHA384 - strong
50 |     compressors:
51 |       NULL
52 |_  least strength: strong
53
54 Nmap done: 1 IP address (1 host up) scanned in 3.05 seconds
```

Ein weiteres Bewertungskriterium für die Sicherheit der Transportverschlüsselung auf einem Server stellt auch die Aktualität der Softwaremodule dar, die für TLS benötigt werden. Auf Linux-Servern handelt es sich dabei hauptsächlich um *OpenSSL* und *GnuTLS*. Veraltete Versionen können Sicherheitslücken beinhalten, die die Informationssicherheit stark gefährden. Eine schwerwiegende Lücke in OpenSSL wurde beispielsweise im Jahr 2014 bekannt [SCHM 14]. Seitdem wurden auch zahlreiche kleinere Lücken geschlossen.

Die Aktualität dieser Programme kann prinzipiell durch den Mechanismus erfolgen, der bereits im Abschnitt 6.1 vorgestellt wurde. Der Vorteil dieses Ansatzes ist, dass er bereits vorhanden ist und zur Bewertung der TLS-Sicherheit herangezogen werden kann. Am aussagekräftigsten ist dieser Ansatz aber nur dann, wenn die Aktualität der Paketversionen und die Sicherheit der TLS-Konfigurationsdateien zum gleichen Zeitpunkt erfolgt. In der Praxis ist dies nicht der Fall, da die Prüfmechanismen für alle Anwendungsfälle Last auf den Servern erzeugen und diese möglichst gleichmäßig verteilt werden soll.

Eine zusätzliche Versionsprüfung, die zusammen mit der Prüfung der TLS-Konfiguration stattfindet, ergibt also Sinn: Sie liefert den aktuellen Wert und erzeugt im Gegensatz zu einer Prüfung sämtlicher Paketversionen deutlich weniger Overhead. Die zusätzliche Versionsprüfung bietet auch den Vorteil, dass sie unabhängig vom Prüfmodul für den Anwendungsfall aus Abschnitt 6.1 implementiert ist. Sollte diese Komponente aufgrund eines Fehlers oder Ausfalls versagen, ist eine Prüfung der Programmversionen von OpenSSL und GnuTLS durch die funktionale Entkoppelung weiterhin möglich.

6.4.5. Mögliche Visualisierung

Im Anschluss werden Visualisierungen für den Anwendungsfall der Sicherheit von TLS-Konfigurationen angesprochen. Diese basieren ebenfalls auf den zuvor genannten Visualisierungen.

Einsatz von Signalfarben

Beim Anwendungsfall *TLS-Konfiguration* kann im zentralen Informationssystem auf dieselben Methoden der Informationsvisualisierung zurückgegriffen werden wie es bei den zuvor betrachteten Use Cases der Fall ist. Zunächst ist eine Integration der Gesamtbewertung der TLS-Sicherheit aller Server in die Ampelübersicht durch ein zusätzliches Piktogramm denkbar. Als Grundlage für ein zusätzliches Ampelsymbol dienen die folgenden möglichen Zustände:

- Alle TLS-Konfigurationen sind sicher.
- Es gibt mindestens einen Server mit einer sicherheitsbedenklichen TLS-Konfiguration.
- Es existiert mindestens ein Server mit einer Sicherheitskritischen SSL-Konfiguration.

Der erste Punkt stellt den Idealfall dar und wird wie in den vorherigen Anwendungsfällen auf die Farbe Grün gemappt. Eine TLS-Konfiguration gilt als *bedenklich*, wenn überholte kryptographische Algorithmen vorgefunden wurden. Dies gilt zum Beispiel für das symmetrische Verschlüsselungsverfahren 3Data Encryption Standard (DES). Die IETF rät von dessen Verwendung ab [IETF 15]. Die veralteten SSL-Protokollversionen bis einschließlich 3 dürfen nicht mehr verwendet werden und fallen unter die Farbkategorie Rot. Sollte ein

Server anfällig für Downgrade-Angriffe sein, also dem Client die Wahl eines unsicheren Algorithmus überlassen, ist die Sicherheit der TLS-Konfiguration ebenfalls als kritisch einzustufen. Eine zusätzliche Legende für diesen Anwendungsfall erklärt dem Anwender die Bedeutung der Farben.

Server-Statistiken

Wie bei den zuvor genannten Anwendungsfällen ist es für den Administrator zunächst interessant, wie viele Server jeweils in eine der drei zuvor genannten Sicherheitskategorien fallen. Analog dazu eignen sich wieder Säulendiagramme zur Repräsentation absoluter Zahlen und Kreisdiagramme zur Repräsentation von Verhältnismäßigkeiten. Es werden dort jeweils die Farben Grün, Gelb und Rot verwendet (siehe Abbildung 6.26). Nach dem Klick auf eine der Farben bekommt der Anwender die konkreten Systeme in der zoombaren Oberfläche angezeigt, um sie nach seinen Bedürfnissen weiter zu filtern und anzuordnen. Dieses Verfahren ist bereits aus den vorherigen Anwendungsfällen bekannt.

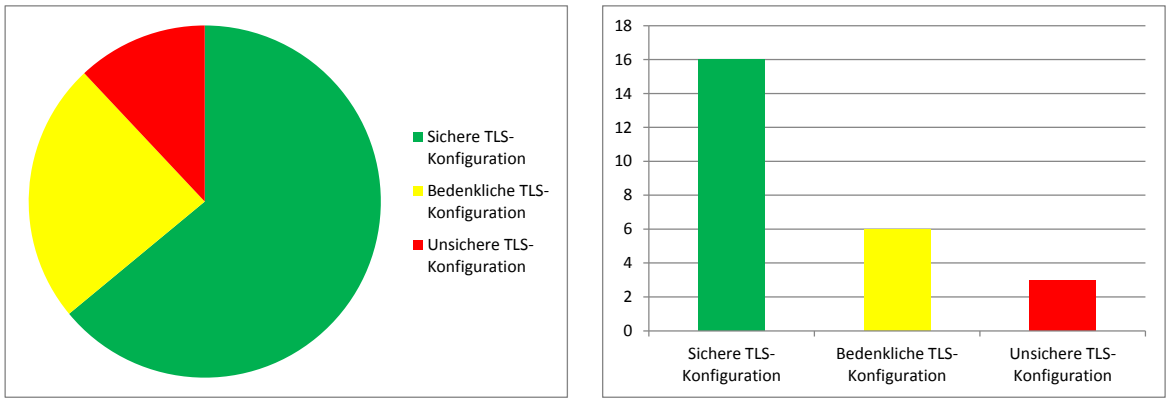


Abbildung 6.26.: Sicherheitszustand der TLS-Konfigurationen nach drei Kategorien geordnet.

Wenn die TLS-Konfiguration eines Servers als bedenklich oder gar kritisch bewertet wird, kann es mehrere Ursachen geben. Dazu zählen zum Beispiel unsichere Cipher Suites oder veraltete Protokollversionen. Um zunächst die Anzahl der betroffenen Systeme zu visualisieren, eignen sich Diagramme, die nach Cipher Suites oder Protokollversionen kategorisiert sind.

Da von einem Server in der Regel mehrere Cipher Suites oder Protokollversionen gleichzeitig angeboten werden, sind Kreisdiagramme oder andere Diagramme, die relative Werte ausdrücken zur Visualisierung ungeeignet. Abbildung 6.27 zeigt ein Beispiel hierfür.

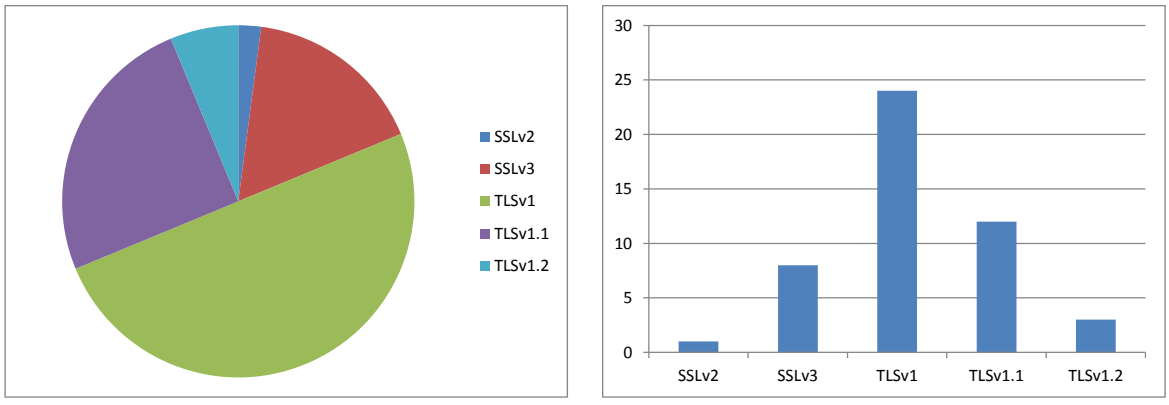


Abbildung 6.27.: Von Servern bereitgestellte TLS-Versionen relativ (links) und absolut (rechts). Das linke Teilbild vermittelt unterschwellig einen falschen Eindruck von der Anzahl der Systeme, da Überschneidungen möglich sind.

Da es in der Natur von Kreisdiagrammen liegt, relative Anteile darzustellen, vermittelt das linke Teilbild dem Betrachter, dass die Hälfte der Server TLS 1.0 verwendet und der Rest der Server die anderen Protokolle. In der Regel bietet ein Server aber mehr als ein Protokoll an. Wie aus der höchsten Säule im rechten Teilbild hervorgeht, gibt es im Extremfall nur 24 Server, von denen nicht die Hälfte, sondern jeder TLS 1.0 unterstützt, da Kombinationen unterschiedlicher Versionen nicht ausgeschlossen sind. Der falsche Eindruck wird im rechten Teilbild durch das Säulendiagramm nicht vermittelt.

Zoombare Gesamtübersicht mit Filterung und Sortierung

Auch bei diesem Anwendungsfall wird eine Integration in die zoombare Übersicht der Server angestrebt, die in Abschnitt 6.1 vorgestellt wurde. Durch die Integration aller Anwendungsfälle in diese Ansicht wird ein konsistentes Bedienkonzept geschaffen. Da das Bedienkonzept dem Anwender bereits bekannt ist, kommt er auch mit Anwendungsfällen zurecht, die im Wirkbetrieb zum zentralen Informationssystem hinzugefügt wurden und noch gar nicht Bestandteil dieser Ausarbeitung sind.

Die zoombare Übersicht der Server dient dem Anwender auch bei diesem Use Case als eine Art virtueller Sortiertisch. Er kann die Menge der angezeigten Server durch Filteroperationen reduzieren und die Server nach bestimmten Kriterien am Bildschirm gruppieren oder anordnen lassen. Ein mögliches Filterkriterium stammt zwar aus dem Anwendungsfall der Softwarepakete (siehe Abschnitt 6.1), ergibt im Kontext der TLS-Sicherheit Sinn: Immer wieder tauchen in Kryptobibliotheken wie OpenSSL Sicherheitslücken auf. Durch die Reduzierung der angezeigten Server auf eine bestimmte Paketversion können Probleme schneller eingegrenzt werden.

Nicht immer muss eine Sicherheitslücke ausschlaggebend für die Filterung nach bestimmten Paketversionen sein: Im Abschnitt 6.1 wurde eine spezielle Apache-Version als Beispiel genannt, die keine selektive Auswahl von TLS-Versionen zulässt und daher potenziell Downgrade-Angriffe zulässt. Der Anwender kann die Apache-Version als Filterkriterium benutzen, um auf einen Blick zu erkennen, welche Server ein Distributions-Upgrade benötigen.

Als sinnvolles Gruppierungskriterium eignet sich die Sicherheitszone, zu der ein Linux-Server gehört. Nachdem der Anwender zum Beispiel alle Systeme angefordert hat, die sicherheitskritische Lücken in den TLS-Einstellungen aufweisen, sieht er sofort, wo dringender Handlungsbedarf herrscht: Hängt der Server etwa am öffentlich zugänglichen Internet, müssen die Fehler dringender beseitigt werden als bei einem internen Webserver, der nur von einer kleinen und bekannten Personengruppe verwendet wird. Ein Beispiel für den zuletzt genannten Fall ist eine Instanz eines Softwareprojektplanungsprogramms, das für ein kleines Entwicklerteam angeschafft wurde.

Falls ein Server aufgrund mehrerer LAN-Schnittstellen mehreren Sicherheitszonen angehört, taucht er in der Gesamtübersicht mehrmals auf. Um Verwirrungen zu vermeiden, ist eine gesonderte Kennzeichnung für solche Systeme erforderlich. Eine Möglichkeit besteht darin, dass das Informationssystem ihn automatisch einer besonderen Gruppe zuweist, die für Dual Homed Hosts gedacht ist.

Ein weiteres sinnvolles Gruppierungskriterium ist auch bei diesem Anwendungsfall die Serverkategorie. Bisher wurde zwar nur die Bewertung der Sicherheit von TLS-Konfigurationen am Beispiel des Web Servers Apache behandelt, TLS findet aber auch bei anderen Diensten Anwendung. Dazu zählen zum Beispiel Datenbankserver oder Verzeichnisdienste. Im Prinzip muss das eingesetzte Prüfprogramm die gleichen Konfigurationsparameter betrachten, lediglich das Format der Konfigurationsdatei unterscheidet sich von Apache.

Neben der Gruppierung von Systemen ist auch eine Anordnung im Koordinatensystem in Abhängigkeit von bestimmten Attributen möglich. Ein naheliegendes Sortierkriterium speziell bei Webservern ist die durchschnittliche Anzahl von Webseitenbesuchern. Stark frequentierte Server können so schnell von weniger beanspruchten Systemen unterschieden werden. Eine Sortierung nach der letzten Änderung der TLS-Konfiguration ist ebenfalls denkbar.

6.5. Zusammenfassung

In diesem Kapitel wurden vier Anwendungsfälle aufgegriffen, die in der Linux-Server-Administration von großer Bedeutung sind und einen starken Bezug zur Informationssicherheit haben. Ausgehend vom Anwendungsfall Software-Verwaltung (siehe Abschnitt 6.1) wurden Visualisierungsmethoden vorgeschlagen, die eine sukzessive Vergrößerung des Detailgrads ermöglichen, um Ben Shneidermans Information Seeking Mantra (siehe Abschnitt 6.1.2) zur Bewältigung unüberschaubarer Datenmengen auf diesen Anwendungsfall zu übertragen. Es wurden auch Lösungsmöglichkeiten erarbeitet, die ein zentrales grafisches Informationssystem benötigt, um die für die Visualisierung erforderlichen Daten von den Servern zu erhalten.

Zunächst wurde für den Anwendungsfall mit der Abbildung der Gesamtsituation aller Server auf ein Ampelpiktogramm eine möglichst simple Visualisierung vorgestellt, die einen minimalen Detailgrad bereitstellt und vom Administrator lediglich einen kurzen Blick abverlangt, um die Serversicherheit bewerten zu können. Die verschiedenen Diagramme mit den statistischen Werten besitzen einen höheren Informationsgehalt, werden vom Administrator aber nur aufgerufen, falls konkretes Interesse besteht.

Den größten Informationsgehalt liefert schließlich eine zoombare Bedienoberfläche (siehe Abschnitt 2.17.2), die Dynamic Queries (siehe Abschnitt 2.14.3), Overview and Detail (siehe Abschnitt 2.17.3) und Filterfunktionen implementiert, um gezielt nach Detailinformationen zu bestimmten Servern zu suchen. Im Raum angeordnete Glyphen (siehe Abschnitt 2.12.4) repräsentieren die einzelnen Server. Der Administrator kann sie nach von ihm ausgewählten Kriterien auf dem Bildschirm anordnen lassen, sukzessive Filterungen vornehmen und durch das Zoomen in die Server-Übersicht durch die Implementierung von konstanter Informationsdichte (siehe Abschnitt 2.12.7) und abrufbarer Detailübersichten immer mehr sicherheitsrelevante Informationen von interessanten Systemen anfordern. Durch ästhetische Animationen wird ein Immersionseffekt erzielt, der Einstiegsbarrieren abbaut und die Bedienfreundlichkeit optimiert. Das ist eine wichtige Voraussetzung für den effizienten Einsatz dieser Art der Visualisierung in einer tatsächlichen Implementierung.

Der große Vorteil dieser Arten der Visualisierung ist die Kompatibilität zu den restlichen Anwendungsfällen. Sie lassen sich nicht nur auf ähnliche Art und Weise visualisieren, sondern auch in gemeinsame Visualisierungen integrieren. Die Erweiterbarkeit um Anwendungsfälle, die in dieser Ausarbeitung nicht angesprochen wurden, ist ebenfalls denkbar. Durch die gemeinsamen Visualisierungsmethoden für mehrere Anwendungsfälle sinkt der Implementierungsaufwand. Sie bieten auch den Vorteil, dass Administratoren nicht ständig zwischen unterschiedlichen Ansichten hin- und her wechseln müssen und dabei wichtige Informationen übersehen. Es spricht auch nichts dagegen, bereits existierende anwendungsfallspezifische Visualisierungen zusätzlich zu integrieren und bei Bedarf für einzelne Systeme abrufbar zu machen. Sie erfordern zwar Einarbeitung, können bestimmte Sachverhalte aber wirksam ausdrücken. Die zentrale Anlaufstelle bleibt allerdings das hier vorgeschlagene grafische Informationssystem.

Um die Implementierbarkeit und Wirksamkeit ausgewählter Visualisierungen zu demonstrieren, wird im Rahmen dieser Arbeit eine prototypische Anwendung entwickelt. Diesem Zweck widmet sich das folgende Kapitel.

7. Prototypische Implementierung

Dieses Kapitel widmet sich der Vorstellung der prototypischen Anwendung, die die Realisierbarkeit der in Kapitel 6 vorgeschlagenen Visualisierungen anhand eines ausgewählten Anwendungsfalls (siehe Abschnitt 7.1) demonstrieren soll. In Abschnitt 7.2 werden Grundlagen zur Implementierung angesprochen. Dazu zählen verwendete Technologien und Plattformen (siehe Abschnitt 7.2.1), Details zur Server- (siehe Abschnitt 7.2.2) und Client-Seite (siehe Abschnitt 7.2.3) sowie das Datenmodell in Abschnitt 7.2.4.

7.1. Auswahl eines Anwendungsfalls

Im Kapitel 6 wurden insgesamt vier Anwendungsfälle vorgestellt, deren Sicherheit sich durch Methoden der Informationsvisualisierung verbessern lässt. Bei der Auswahl eines Anwendungsfalls für die Implementierung im Demonstrator ist es ausschlaggebend, wie einfach die Daten bereitgestellt werden können, auf denen die Visualisierungen basieren, da die Realisierbarkeit der zuvor vorgeschlagenen Visualisierungen den Schwerpunkt dieser Anwendung darstellt.

Der Anwendungsfall *Software-Verwaltung* aus Abschnitt 6.1 eignet sich hervorragend für eine Implementierung im Demonstrator, da die Beispieldaten einfach zu erzeugen und in der Datenbank abzulegen sind. Es handelt sich um einfache aggregierte Werte, die direkt in eine Visualisierung einfließen können. Zu den beispielhaft betrachteten Werten für diesen Anwendungsfall zählen folgende:

- Anzahl der auf einem Server installierten Pakete
- Letzter Aktualisierungszeitpunkt der Pakete
- Allgemeiner Zustand der Sicherheit der Pakete auf einem Server (aktuell, nicht aktuell, kritische Sicherheitslücken)

Zusätzlich zu den für diesen Use Case spezifischen Werten können auch noch allgemeine Informationen in die Visualisierungen einfließen. Der Demonstrator greift zu diesem Zweck auf die Priorität der Server zurück, die vom Administrator festgelegt werden, und auf benutzerspezifische Tags.

Die restlichen Anwendungsfälle aus Kapitel 6 werden vom Demonstrator aber nicht vollkommen ignoriert. Die Statusübersicht mit den allgemeinen Sicherheitszuständen der einzelnen Use Cases, die in Form von Ampelpiktogrammen dargestellt werden, fand ihren Weg in die Implementierung.

7.2. Grundlagen zur Implementierung

In diesem Abschnitt wird auf grundlegende Herangehensweisen bei der Implementierung des Prototyps und auf verwendete Technologien eingegangen. Einerseits soll dadurch ein Überblick für potenzielle Entwickler bereitgestellt werden, um die Wiederverwendbarkeit des Programms für zukünftige Zwecke zu gewährleisten. Andererseits wird auf wichtige Implementierungsdetails eingegangen, die zur Gewährleistung der Informationssicherheit beitragen. Da es sich beim Prototyp um eine webbasierte Rich-Internet-Anwendung mit hohem JavaScript-Anteil handelt, werden serverseitige Komponenten und clientseitige JavaScript-Komponenten später separat vorgestellt.

7.2.1. Verwendete Technologien und Plattformen

Bei einer webbasierten Anwendung wie dem Prototyp kommen viele verschiedene Technologien auf dem Server und dem Client zum Einsatz. Dazu zählen Skriptsprachen, Auszeichnungssprachen, Frameworks, Web- und Datenbankserver. Dieser Abschnitt stellt die eingesetzten Technologien und Plattformen vor und begründet ihre Auswahl für den Prototyp.

Python

Für die Entwicklung der serverseitigen Komponente des Prototyps wurde die Skriptsprache Python[1] gewählt. Dafür spricht zunächst die hohe Verbreitung dieser Sprache im Webumfeld: Neben Java und PHP Hypertext Preprocessor (PHP) zählt Python laut einer aktuellen Statistik des Magazins *IEEE Spectrum* zu den populärsten Programmiersprachen für webbasierte Anwendungen [CASS 14] und erzielte dort Platz 2. Die leichte Erlernbarkeit und weniger Code Overhead im Vergleich zu Java Enterprise Edition sprechen für den Einsatz von Python beim Bau eines Demonstrators. Ein weiterer Vorteil ist, dass diese Sprache bereits für verschiedene Aufgaben am LRZ eingesetzt wurde und dort bereits Erfahrung damit besteht. Prinzipiell würde auch PHP für den Prototyp in Frage kommen, doch Python bietet im Gegensatz dazu noch weitere Vorteile:

Beim Entwurf der Sprache legte ihr Entwickler, *Guido van Rossum*[2], großen Wert auf die Lesbarkeit und Übersichtlichkeit von Python-Skripts. Die Sprache verfügt über eine überschaubare Menge von Schlüsselwörtern und verzichtet auf spezielle Zeichen zur Bildung von Code-Blöcken. Die Blockbildung wird stattdessen durch die Einrückung des Codes bestimmt. Dies hat zur Folge, dass Quelltexte in Python im Vergleich zu Sprachen wie C oder Java platzsparender ausfallen. Aufgrund der Blockbildung durch Einrückung zwingt Python den Entwickler regelrecht dazu, für die Allgemeinheit übersichtlichen Code zu schreiben, denn im Gegensatz zu C oder Java muss sich der Leser eines fremdes Skripts nicht erst an den speziellen Einrückungsstil[3] des Programmierers gewöhnen.

Als Multiparadigmensprache zwingt Python den Entwickler nicht zu einem bestimmten Programmierparadigma. Stattdessen entscheidet der Entwickler, welcher Programmierstil ihm beim Erledigen einer bestimmten Aufgabe schneller zum Ziel führt. Dies hat eine hohe Flexibilität zur Folge und macht Python für viele Einsatzbereiche tauglich. Python unterstützt unter anderem funktionale, objektorientierte und strukturierte Programmierung. Eine umfangreiche Programmbibliothek, besonders in Bezug auf Webanwendungen, spricht für Python.

Abbildung 7.1.: Bild- und Wortmarke der Programmiersprache Python.[4]

Web Framework Django

Wenn es um Webentwicklung geht, haben Entwickler die Auswahl unter einer Vielzahl von Frameworks. Das gilt nicht nur für Python, sondern auch für andere im Web relevanten Sprachen wie zum Beispiel PHP Hypertext Preprocessor, ursprünglich Personal Home Page Tools (PHP). Viele dieser Frameworks kommen und gehen bzw. werden von der Community nicht mehr weiterentwickelt. Beim Einsatz in einer Umgebung, die Wert auf Informationssicherheit legt, sollten die verwendeten Technologien aber noch aktiv gepflegt werden, um Sicherheitslücken zu beseitigen.

[1] Python-Website: `https://www.python.org`
[2] Guido van Rossums Homepage: `https://www.python.org/~guido/`
[3] Einrückungsstile: `http://en.wikipedia.org/wiki/Indent_style`
[4] Bildquelle: `http://upload.wikimedia.org/wikipedia/commons/f/f8/Python_logo_and_wordmark.svg`

Das auf Python basierte Web Framework *Django*[5] hat den Status einer Modeerscheinung schon längst hinter sich gelassen und blickt bereits auf ein zehnjähriges Bestehen zurück. Für den Einsatz von Django beim zu entwickelnden Prototyp spricht außerdem, dass das Framework bereits von LRZ-Mitarbeitern eingesetzt wurde.

Abbildung 7.2.: Das offizielle Logo des Django Frameworks.[6]

Ein zentrales Ziel von Django ist die Einhaltung des *Don't repeat yourself (DRY)*-Prinzips. Das Framework bietet einen objektrelationalen Mapper und unterstützt out of the box die Datenbanksysteme MySQL, PostgreSQL und Oracle. Die mögliche Anbindung einer SQLite-Datenbank verkürzt den zeitlichen Aufwand, eine Entwicklungsumgebung einzurichten und eine erste lauffähige Programmversion zu erhalten. Das Mapping zwischen URLs und Programmfunktionen erfolgt bei Django auf flexible Art und Weise über reguläre Ausdrücke in einer zentralen Datei. Für den Aufbau von Webseiten bietet das Framework eine funktionsreiche Template-Sprache, die auch Vererbung unterstützt. Django beherrscht Internationalisierung und Lokalisierung und lässt sich je nach Bedarf um weitere Module erweitern.

Um die Sicherheit von Webanwendungen zu gewährleisten, stellt Django unterschiedliche Funktionen zur Verfügung. Es besitzt einen Token-basierten Schutz gegen CSRF. Durch automatisches Escapen bestimmter Zeichen schützt das Template-System vor einer Vielzahl möglicher XSS-Angriffe. Der objektrelationale Mapper von Django generiert sichere SQL-Abfragen und beugt somit SQL Injections vor. Durch spezielle Anweisungen im HTTP Header werden Anwender vor Clickjacking geschützt. Weitere Schutzmechanismen sind auf der Website des Frameworks nachzulesen.[7]

HTML5

Da der Prototyp eine webbasierte Anwendung werden soll, ist die Wahl von HTML5 als Auszeichnungssprache für Webseiten nahezu alternativlos. Im Vergleich zur Vorgängerversion wurde HTML nicht einfach um zusätzliche Elemente erweitert, sondern es wurden auch neuartige Konzepte zur clientseitigen Speicherung von Nutzdaten und dem temporären Offline-Betrieb von Webanwendungen eingeführt. Erwähnenswert sind außerdem die durch das `<canvas>`-Element hinzugekommenen Grafikfunktionen von HTML5.

Abbildung 7.3.: Bild- und Wortmarke von HTML5, bereitgestellt vom W3C.[8]

[5] Django-Website: `https://www.djangoproject.com/`
[6] Bildquelle: `http://upload.wikimedia.org/wikipedia/de/0/0e/Django-logo.svg`
[7] Sicherheit bei Django: `https://docs.djangoproject.com/en/1.7/topics/security/`
[8] Bildquelle: `http://upload.wikimedia.org/wikipedia/commons/6/61/HTML5_logo_and_wordmark.svg`

jQuery

Als Basis für die clientseitige Programmierung greift der Demonstrator auf die JavaScript-Bibliothek jQuery[9] zurück. jQuery erweist sich im Gegensatz zu reinem JavaScript als äußerst hilfreich, wenn es um eine einfache Document Object Model (DOM)-Manipulation und -Navigation geht. Dies erweist sich auch im zu entwickelnden Prototyp als vorteilhaft.

Abbildung 7.4.: Das offizielle Logo von jQuery.[10]

Die Bibliothek wurde erstmals im Jahr 2006 veröffentlicht. Im Jahr 2013 erschien mit der Version 2.0 der zweite Major Release. Durch die Entfernung der Unterstützung für veraltete Webbrowser konnte der Code von jQuery umfassend aufgeräumt werden. Es mag zwar Vertreter der Ansicht geben, dass durch den Wegfall der Unterstützung älterer Browser die Bedienbarkeit eingeschränkt wird, veraltete Browser, die nicht mehr gepatcht werden, stellen aber ein Sicherheitsproblem dar. Aus Sicht der Informationssicherheit ist besonders in Hinblick auf das LRZ diese Entscheidung also zu verschmerzen.

jQuery ermöglicht mit Hilfe der sog. *Sizzle Selector Engine* die Selektion von DOM-Elementen in einer ähnlich komfortablen Art und Weise wie bei Cascading Style Sheets (CSS)3. jQuery stellt *Asynchronous JavaScript and XML (AJAX)*-Funktionalitäten bereit und verfügt über ein erweitertes System zur Ereignisbehandlung. Zudem stellt jQuery spezielle Funktionen für Grafikeffekte und Animationen bereit. Mit der `each()`-Hilfsfunktion kann im Gegensatz zu reinem JavaScript auf einfache Art und Weise über Datenstrukturen iteriert werden.

Durch die große Popularität von jQuery existieren auch zahlreiche andere JavaScript-Bibliotheken, die auf jQuery aufbauen. Dies gestaltet sich als Vorteil, da so weitestgehend auf die Einbindung unterschiedlicher Basisbibliotheken verzichtet werden kann. Durch jQuery als gemeinsame Basisbibliothek werden nicht nur Konflikte vermieden, sondern auch der Overhead verringert.

Um konsequent zu bleiben und Konflikte zu vermeiden, handelt es sich bei allen im im Demonstrator eingebundenen JavaScript-Bibliotheken entweder um Plug-ins von jQuery oder um jQuery selbst. Durch die Wahl von jQuery als einzige Basis-Bibliothek wird der Demonstrator auch einfacher wartbar und weiterentwickelbar, da sich die Entwickler nicht mit unterschiedlichen Technologien auseinandersetzen müssen.[11]

jQuery UI

Eine Anforderung an den Prototyp ist eine Bedienoberfläche zu schaffen, die sich durch Bedienfreundlichkeit und Plattformunabhängigkeit auszeichnet. Die Darstellung bestimmter Bedienelemente unterscheidet sich von Browser zu Browser. Manche Oberflächenelemente wie Schieberegler oder Fortschrittsleisten, die man von nativen Client-Anwendungen kennt, stehen in Browsern standardmäßig nicht zur Verfügung. Mit HTML5 wurden zwar unter anderem neue Formularelemente eingeführt, sie werden aber noch nicht von allen gängigen Browsern unterstützt.

Abbildung 7.5.: Das offizielle Logo von jQuery UI.[12]

[9] jQuery-Website: `http://jquery.com/`
[10] Bildquelle: `http://upload.wikimedia.org/wikipedia/de/f/fd/JQuery-Logo.svg`
[11] Neben jQuery existieren noch eine Reihe anderer JavaScript-Frameworks, die zur Zeit in Mode sind. Beispiele hierfür sind *Sencha Ext JS* (`https://www.sencha.com/products/extjs/`) und *Googles AngularJS* (`https://angularjs.org/`). jQuery bietet im Gegensatz zu anderen Frameworks den Vorteil, dass es sich auch hervorragend in bestehende Projekte einbinden lässt und die Einarbeitung relativ einfach ist.

Abhilfe schafft in diesen Fällen die JavaScript-Bibliothek jQuery UI[13]. Es handelt sich dabei um eine Erweiterung der zuvor genannten Bibliothek jQuery und ist auf eine plattformübergreifende Browser-Unterstützung ausgelegt.

Die Bibliothek jQuery UI bietet dem Entwickler Unterstützung beim Programmieren bestimmter Interaktionsformen wie zum Beispiel *Drag and Drop*. Sie stellt verschiedene vorprogrammierte Widgets zur Verfügung und erweitert das Angebot an Effekten von jQuery. Um Overhead zu minimieren, ist jQuery UI komplett modular aufgebaut. Der Entwickler kann auf der Projekt-Website die Komponenten, die er tatsächlich benötigt, selbst auswählen und so eine maßgeschneiderte Version der Bibliothek zusammenstellen. Ergänzend dazu bietet die Website die Möglichkeit, eigene Skins für Bedienelemente interaktiv zusammenzustellen und herunterzuladen.

jQuery Panzoom

Der Prototyp verwendet das jQuery Plug-in *Panzoom*[14] bei der zoombaren Übersicht der Server. Panzoom ermöglicht es, beliebige DOM-Elemente stufenlos vergrößerbar zu machen und innerhalb der vergrößerten Ansicht zu navigieren. Um die Performance zu erhöhen, unterstützt Panzoom hardwarebeschleunigte Berechnungen.

Die Server-Gesamtübersicht wird mit Hilfe dynamisch generierter Scalable Vector Graphics (SVG) realisiert. Zu diesem Zweck kommt eine weitere Bibliothek mit dem Namen *jQuery SVG* zum Einsatz, die in später noch vorgestellt wird.

jQuery Mouse Wheel

Die zuvor vorgestellte JavaScript-Bibliothek jQuery Panzoom ist so konzipiert, dass die Steuerung des Zooms und der angezeigten Bildposition mit Hilfe von eigenem JavaScript Code gesteuert werden kann. Ein häufiger Verwendungszweck dieser Bibliothek ist die geskriptete Animation von Fotos. Damit der Anwender auch die Anzeige selbst beeinflussen kann, ist die Anbindung von Buttons und Slidern denkbar. Am intuitivsten und einfachsten gestaltet sich die Interaktion mit der Grafik durch das Zoomen mit dem Mausrad. Hier kommt das Plug-in *jQuery Mouse Wheel*[15] ins Spiel.

Durch die Einbindung von jQuery Mouse Wheel in jQuery Panzoom kann der Anwender einen beliebigen Ort innerhalb der Anzeigefläche mit Hilfe des Mausrads vergrößern und verkleinern. Sobald der Mauszeiger auf einem Objekt positioniert ist, das der Anwender gerne näher betrachten will, verwendet jQuery Panzoom diese Koordinate als fokalen Punkt. Beim Hereinzoomen in die Grafik bleibt das anvisierte Objekt so stets unter dem Mauszeiger. Durch dieses Bedienkonzept kommt der Anwender sehr schnell ans Ziel und an die von ihm benötigten Detailinformationen und muss nicht ständig den Sichtbereich verschieben.

jQuery SVG

Die Bibliothek *jQuery SVG*[16] ermöglicht die Interaktion mit SVG-Dateien per JavaScript. Der Demonstrator nutzt die Funktionen dieser Bibliothek, um für seine zoombare Bedienoberfläche den Bildinhalt dynamisch zu generieren. SVG-Grafiken eignen sich für diesen Anwendungsfall hervorragend, da es sich um ein vektorbasiertes Format handelt. Auch bei starker Vergrößerung bleiben die Bildinhalte auf dem Bildschirm scharf.

Als Alternative zu SVG käme prinzipiell für clientseitig dynamisch generierte Grafiken auch das Bitmapbasierte *Canvas*-Element in Frage, das zusammen mit HTML5 eingeführt wurde. Im Gegensatz zu SVG ist damit aber einerseits keine Vergrößerung ohne Qualitätsverlust möglich, andererseits können auch keine Event Handler ohne Weiteres mit Teilen der Grafik verbunden werden. Damit scheidet *Canvas* für den Demonstrator

[12] Bildquelle: `http://upload.wikimedia.org/wikipedia/de/5/51/JQuery_UI-Logo.svg`
[13] jQuery-UI-Website: `http://jqueryui.com/`
[14] jQuery-Panzoom-Website: `https://github.com/timmywil/jquery.panzoom`
[15] jQuery-Mouse-Wheel-Website: `https://github.com/jquery/jquery-mousewheel`
[16] jQuery-SVG-Website: `http://keith-wood.name/svg.html`

aus. Eine Diskussion der Eignung von Canvas und SVG unter Berücksichtigung der benötigten Rechenleistung können Interessierte in einem Microsoft Developer Network (MSDN)-Artikel nachlesen [MSDN 11].

Im Paket von jQuery SVG ist unter anderem das optional verwendbare Plug-in *SVG DOM* enthalten. Mit diesem Plug-In ist es möglich, bestimmte jQuery-Funktionen, die ursprünglich für das HTML DOM entwickelt wurden, auch auf Elemente innerhalb einer SVG-Grafik anzuwenden. Im Prototyp findet diese Bibliothek Verwendung, um einzelne Objekte in den dynamisch generierten Visualisierungen mit Event Handlern zu versehen. Dadurch reagieren sie zum Beispiel auf Mouse-Over-Ereignisse oder das Klicken mit einer Maustaste, um dem Anwender bei Bedarf Zusatzinformationen bereitzustellen.

jQuery Cookie

Die Anwendung nutzt Cookies, um Zustände auf der Clientseite zu speichern und sie später wiederherzustellen. Dies ist der Fall bei Reitern im Visualisierungsbereich des Demonstrators. Der zuletzt vom Benutzer geöffnete Reiter wird in einem Cookie gespeichert, damit beim erneuten Aufruf der Webseite der gleiche Reiter automatisch wieder geöffnet wird. Dadurch gelangt der Anwender schneller zu häufig aufgerufenen Informationen. Zur einfachen Handhabung von Cookies greift der Demonstrator auf die Bibliothek *jQuery Cookie*[17] zurück.

Flot

Flot[18] ist eine auf jQuery basierende Bibliothek zur dynamischen Generierung unterschiedlicher Diagramme mit dem Schwerpunkt auf einfacher Bedienung und interaktiven Funktionen. Sie beherrscht bereits viele grundlegende Diagrammtypen, die jeweils in separate JavaScript-Dateien ausgelagert sind. Der Entwickler muss so nur diejenigen Dateien in eine Webseite einbinden, die dort auch tatsächlich benötigt werden. Dadurch wird unnötiger Overhead vermieden.

Abbildung 7.6.: Das offizielle Logo von Flot.[19]

Tablesorter

Neben Webseiten mit dynamisch generierten Visualisierungen stellt der Demonstrator vor allem im Administrationsbereich zahlreiche Informationen in tabellarischer Form dar. Die Übersichtlichkeit der Tabellen verbessert der Demonstrator indem er Interaktionen mit den Tabellen ermöglicht. Zu diesem Zweck greift die Anwendung auf einen Fork der auf jQuery basierenden Bibliothek *Tablesorter* zurück.[20] Der Fork verfügt über eine bessere Kompatibilität zu aktuellen jQuery-Versionen, was sich während der Implementierungsphase auch bewahrheitet hat.

Wie der Name bereits vermuten lässt, ermöglicht Tablesorter dem Anwender die lexikografische Sortierung von Tabelleninhalten anhand ausgewählter Spalten. Neben einer umschaltbaren auf- und absteigenden Sortierung unterstützt Tablesorter auch die Sortierung der Tabelle anhand mehrerer nacheinander ausgewählter Spalten.

[17] jQuery-Cookie-Website: `https://github.com/carhartl/jquery-cookie`
[18] Flot-Website: `http://www.flotcharts.org/`
[19] Bildquelle: `http://www.flotcharts.org/images/blog/2013-01-28-flots-new-logo.png`
[20] Tablesorter-Website (unofficial fork): `http://mottie.github.io/tablesorter/docs/`

Die Bibliothek ist durch weitere Plug-ins beliebig an die Anforderungen der eigenen Anwendung anpassbar. Der Demonstrator nutzt beispielsweise die Fähigkeit, die Kopfzeile von Tabellen beim Herunterscrollen des Bildschirminhalts dauerhaft einzublenden, um die Übersichtlichkeit der Tabelle zu erhalten. Mit Hilfe eines Pagers können Anwender durch Inhalte umfangreicher Tabellen blättern, während die zuvor ausgewählte Sortierung beibehalten wird. Eine benutzerdefinierte Anpassung der Spaltenbreite ist ebenfalls möglich.

Flat jQuery Responsive Menu

Das Hauptnavigationsmenü des Prototyps *Flat jQuery Responsive Menu*[21] stammt von der Website *CSS MenuMaker*[22]. Das Skript nutzt verschachtelte HTML-Listen als Datengrundlage zur Generierung dynamischer Navigationsmenüs. Der Quellcode wurde geringfügig angepasst, um eine harmonische Integration in die Oberfläche des Demonstrators zu gewährleisten.

spin.js

Nachdem der Webbrowser einen Request an den Server gesendet hat, kann es je nach angeforderter Information eine Weile dauern, bis sie auf dem Client angezeigt wird. Während dieser Zeit könnte der Anwender verunsichert reagieren und voreilig zu einer anderen Webseite wechseln. Um ihm deutlich zu signalisieren, dass Daten vom Server erwartet werden und das ggf. versehentliche Klicken währenddessen auf andere Bedienelemente zu verhindern, verwendet der Demonstrator eine kleine Animation, die den Bildschirminhalt im Hintergrund abdunkelt und sämtliche Mausereignisse von selbigem abfängt. Die Animation wird mit Hilfe des auf jQuery basierenden Skripts *spin.js*[23] realisiert.

AJAX

Wie bereits erwähnt wurde, warten klassische Webanwendungen darauf, dass ein Client einen Request an sie sendet, werten daraufhin diesen Request aus und senden dann eine Antwort an den Client zurück. Währenddessen muss der Anwender warten, bis der Browser die Antwort vom Server verarbeitet hat und mit der Arbeit fortfahren kann. Sofern der Server keine aufwendigen Berechnungen durchführen muss, ist das auch kein Problem, da die Antwortzeiten sehr kurz sind. In manchen Fällen macht es aber Sinn, mit Hilfe von JavaScript und AJAX nur bestimmte Informationen vom Server abzufragen, ohne die angezeigte Webseite komplett neu aufbauen zu müssen.

Durch den Einsatz von Django auf der Serverseite und dem JavaScript Framework jQuery auf der Clientseite ist der Prototyp für eine einfache Integration von AJAX gut gerüstet. Als Datenaustauschformat kommt JSON zum Einsatz. Es bietet den Vorteil, dass der JavaScript Client die übertragenen Datenstrukturen unmittelbar verarbeiten kann und verursacht bei der Übertragung weniger Overhead als zum Beispiel XML. Die Verwendung von AJAX macht beim Prototyp besonders dort Sinn, wo Visualisierungen automatisch neu berechnet werden sollen, um Aktualität zu gewährleisten. Der Benutzer muss so nicht ständig selbst die Webseite neu laden und ihm können so auch keine aktuellen Informationen entgehen.

Apache HTTP Server

Das Web Framework Django bringt zwar einen eigenen Webserver mit, dieser wurde aber explizit für die Entwicklungsphase von Django-basierten Projekten entwickelt. Er ist zwar schnell startbereit, lässt aber kaum Spielraum für seine Konfiguration. Für die Produktivsetzung des Prototyps wird deshalb ein leistungsfähiger HTTP Server benötigt. Hier fiel die Wahl auf den quelloffenen *Apache HTTP Server* der Apache Software Foundation.

Apache hat sich im Laufe der Zeit zum meistgenutzten Webserver der Welt entwickelt. Auch wenn sein Open-Source-Konkurrent *nginx* Microsofts kommerziellen IIS den Rang abgelaufen hat, verweilt Apache immer

[21] Flat-jQuery-Responsive-Menu-Website: `http://cssmenumaker.com/menu/flat-jquery-responsive-menu`
[22] CSS-MenuMaker-Website: `http://cssmenumaker.com/`
[23] spin.js-Website: `http://fgnass.github.io/spin.js/`

noch auf Platz eins [NETC 15]. Ein Diagramm mit den Marktanteilen der populärsten Browser ist in Abbildung 6.24 zu sehen. Die Entscheidung für Apache wird durch seine weite Verbreitung und der Tatsache begründet, dass er auch am LRZ in großer Stückzahl eingesetzt wird.

Abbildung 7.7.: Logo der Apache Software Foundation.[24]

Eine Grundinstallation von Apache kann bei Bedarf um zusätzliche Module erweitert werden, damit der Server neue Funktionen bereitstellt. Dies ist zum Beispiel erforderlich, um mit Hilfe von Skriptsprachen dynamische Webseiten zu generieren oder um sichere Verbindungen über TLS herzustellen. Von dieser Möglichkeit macht auch der Prototyp Gebrauch. Damit er Python als Skriptsprache für Webanwendungen unterstützt, muss beispielsweise das Modul `mod_wsgi` geladen werden. Für die Verschlüsselung wird `mod_ssl` benötigt.

MySQL

Beim Anlegen eines neuen Django-Projekts kann der Entwickler unmittelbar auf eine vorkonfigurierte SQLite-Datenbank zurückgreifen. So war es auch beim Prototyp der Fall. Die komplette SQLite-Datenbank befindet in einer einzigen Datei innerhalb des Projektverzeichnisses (`db.sqlite3`). Auf dem Entwicklungssystem ist das eine praktische Sache, da sich der Entwickler auf die eigentliche Anwendung konzentrieren kann. Von Vorteil ist dieser Ansatz auch, wenn die Anwendung abwechselnd auf unterschiedlichen Systemen weiterentwickelt wird oder Sicherungskopien des Quellcodes angefertigt werden sollen. Der aktuelle Datenbankinhalt ist dann ebenfalls in der Kopie enthalten.

Für die spätere Produktivsetzung des Demonstrators ist SQLite aber nicht optimal. Gründe hierfür sind die schlechtere Performance und die überhaupt nicht vorhandene Rechteverwaltung. Der Zugriff zur kompletten Datenbank der Anwendung wird dort lediglich durch die aktuellen Dateisystemrechte beschränkt.

Abbildung 7.8.: Wort- und Bildmarke von MySQL.[25]

Bei der Suche nach einem quelloffenen relationalen DBMS bleiben meist die beiden Konkurrenten *MySQL* und *PostgreSQL* in der engeren Auswahl übrig. Es existieren unter den Anwendern beider Systeme diverse stereotypische Auffassungen über den jeweiligen Konkurrenten, die sich über Jahre gehalten haben. Oft berücksichtigen sie dabei die aktuellen Entwicklungsstände überhaupt nicht. Sowohl PostgreSQL als auch MySQL sind mittlerweile sehr ausgereift und stellen eine ernstzunehmende Konkurrenz für kommerzielle Produkte dar. Eine relativ aktuelle Diskussion über dieses Thema ist bei `wikivs.com`[26] zu finden, einer Plattform, die Vergleiche zwischen verschiedenen Computertechnologien veröffentlicht.

Sofern keine sehr speziellen Anforderungen an das DBMS gestellt werden, ist die Entscheidung zwischen den beiden Programmen Geschmackssache. Idealerweise unterstützen Anwendungsprogramme die Anbindung unterschiedlicher DBMS. Dies ist ein weiterer Grund, weshalb das Web Framework Django für den Demonstrator gewählt wurde. Sein objektrelationaler Mapper erlaubt die Anbindung beliebiger DBMSs. Da am LRZ bereits auf vielen Systemen MySQL zum Einsatz kommt, wird dieses DBMS auch vom Demonstrator standardmäßig verwendet.

[24] Bildquelle: `http://upload.wikimedia.org/wikipedia/commons/f/ff/Apache_Software_Foundation_Logo.svg`
[25] Bildquelle: `http://upload.wikimedia.org/wikipedia/de/1/1f/Logo_MySQL.svg`
[26] MySQL vs. PostgreSQL: `http://www.wikivs.com/wiki/MySQL_vs_PostgreSQL`

Entwicklungsumgebung

Das Einsatzgebiet des Prototyps am LRZ ist eine Server-Landschaft, auf denen die quelloffenen Betriebssysteme *SLES* und Debian GNU's Not Unix (GNU)/Linux dominieren. Dies wurde bei der Wahl der Entwicklungsumgebung berücksichtigt. Aus Gründen der Aktualität der Pakete – insbesondere beim Framework Django – fiel die Wahl auf *openSUSE 13.2*[27] mit K Desktop Environment (KDE) als Bedienoberfläche.

Modernere Programmversionen lassen sich zwar manuell am Paketsystem vorbei installieren, das Patchen der Software bleibt einem in diesem Fall aber selbst überlassen. Aus diesem Grund wird eine Distribution mit einem aktuelleren Repository bevorzugt. Die Stable-Version von Debian 8 steht zwar angeblich vor der Tür, einen konkreten Veröffentlichungstermin gab es zum Zeitpunkt der Entscheidung aber noch nicht[28].

Um die Implementierung des Prototyps zu vereinfachen, wird ein Integrated Development Environment (IDE) benötigt, das im Idealfall auch Autovervollständigung für das Django-Framework unterstützt. Die Wahl fiel dabei auf das kostenlose und vielseitig einsetzbare Programm *Eclipse*[29]. Es ist zwar in erster Linie auf Java-Entwicklung ausgelegt, durch die Extension *PyDev*[30] wird Eclipse aber zu einem geeigneten Werkzeug für die Programmierung mit Python und Django. In Abbildung 7.9 ist ein Bildschirmfoto dieser Entwicklungsumgebung zu sehen.

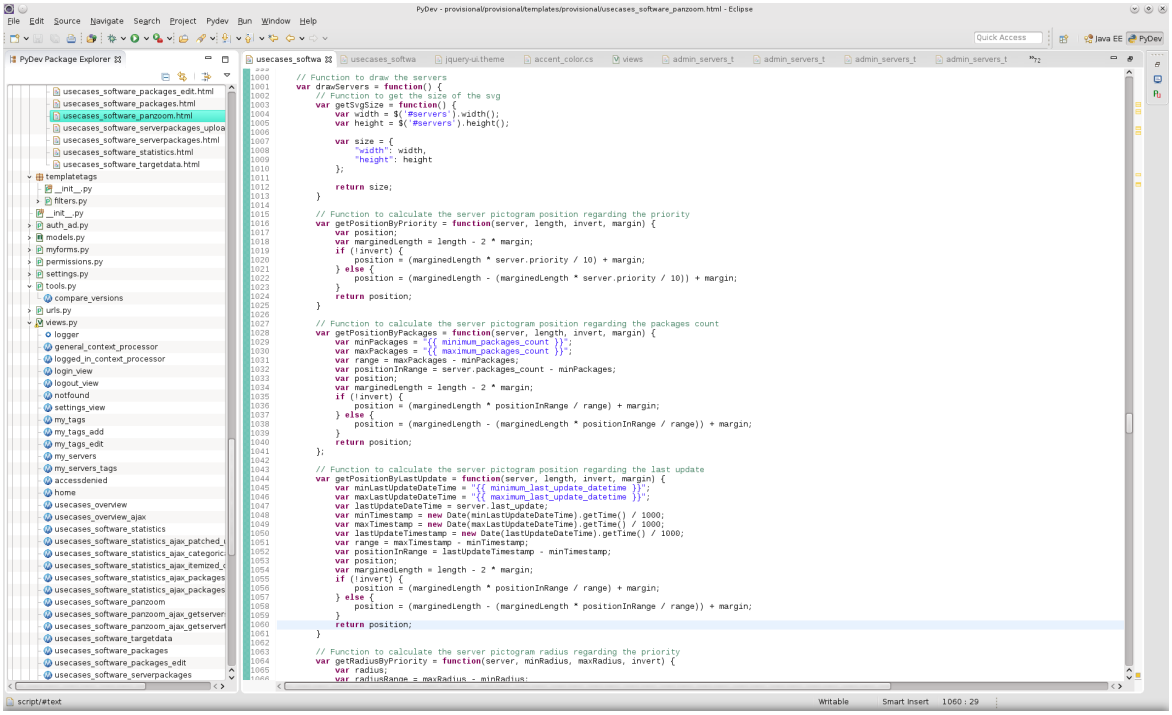


Abbildung 7.9.: Die Entwicklungsumgebung Eclipse mit PyDev.

7.2.2. Server-Seite

Dieser Abschnitt widmet sich den Implementierungsdetails des Demonstrators auf der Seite des Servers. Dazu zählen die Verzeichnisstruktur des Programmverzeichnisses, der Python-Programmcode und das Web Framework Django.

[27] openSUSE-Website: `https://www.opensuse.org/de/`
[28] Quelle: `https://www.debian.org/releases/jessie/`
[29] Eclipse-Website: `https://eclipse.org/`
[30] PyDev-Website: `http://pydev.org/`

Verzeichnisstruktur

Um die Übersicht zu verbessern und potenzielle Weiterentwicklungen zu erleichtern, hält sich der Prototyp an eine Verzeichnisstruktur, die von den Django-Entwicklern empfohlen wird. Im Folgenden wird kurz beschrieben, welche Komponenten das Projektverzeichnis beinhaltet und welche Rolle sie spielen (siehe Abbildung 7.10).

```
provisional/
├── changelog.txt
├── db.sqlite3
├── manage.py
├── provisional.log
└── provisional.sql
    ├── __pycache__/
    ├── fixtures/
    ├── locale/
    │   └── de/
    │       └── LC_MESSAGES/
    ├── static/
    │   └── provisional/
    │       ├── css/
    │       ├── img/
    │       └── js/
    │           ├── cssmenu/
    │           ├── fgnass.github.io/
    │           ├── flot/
    │           ├── jquery-2.1.3/
    │           ├── jquery-cookie-master/
    │           ├── jquery-mousewheel-master/
    │           ├── jquery-ui-1.11.3.custom/
    │           ├── jquery.panzoom-master/
    │           ├── jquery.svg.package-1.5.0/
    │           ├── tablesorter-master/
    │           ├── spinner.js
    │           └── tools.js
    ├── templates/
    │   └── provisional/
    ├── templatetags/
    ├── auth_ad.py
    ├── models.py
    ├── myforms.py
    ├── permissions.py
    ├── settings.py
    ├── tools.py
    ├── urls.py
    ├── views.py
    └── wsgi.py
```

Abbildung 7.10.: Verzeichnisstruktur des Prototyps

provisional/ Das Django-Projektverzeichnis des Prototyps.

provisional/changelog.txt Textdatei mit der Änderungshistorie des Demonstrators. Der Inhalt dieser Datei kann im Demonstrator über das Hilfe-Menü abgerufen werden.

provisional/db.sqlite3 SQLite-Datenbank für Entwicklungszwecke. Sie wird nur benötigt, solange kein richtiges DBMS angebunden ist.

provisional/manage.py Von Django erzeugtes Management-Skript. Es stellt einen einfachen HTTP Server bereit und bietet verschiedene Funktionen an, die für Entwicklungszwecke nützlich sind.

provisional/provisional.log Protokolldatei für Systemereignisse, die Anwender mit dem notwendigen Recht im Demonstrator aufrufen können.

provisional/provisional.sql Dump der MySQL-Datenbank des Demonstrators mit einer Reihe von Beispieldatensätzen, um nach einer Neuinstallation die dynamisch generierten Visualisierungen sofort demonstrieren zu können.

provisional/provisional/ Beinhaltet die eigentliche Applikation. Django unterstützt mehrere "Apps" pro Projekt. In diesem Fall handelt es sich nur um eine Haupt-Anwendung. Dadurch erklärt sich der doppelt vorkommende Name im Pfad.

provisional/provisional/__pycache__ Beinhaltet Bytecode, der vom Python-Interpreter aus den Skriptdateien erzeugt wurde. Er dient zur Erhöhung der Performance. Dieses Verzeichnis wird von Python ab Version 3 automatisch angelegt. Frühere Versionen speicherten den Bytecode direkt in dem Verzeichnis, wo auch die Skripte lagen.

provisional/provisional/fixtures/ Beinhaltet Initialwerte für die Datenbank in Form von JSON-Dateien. Zumindest die darin befindliche Datei `AdminAccount.json` muss in die Datenbank geladen werden, um sich mit dem Initialpasswort `admin123` als Administrator `admin` einloggen zu können.

provisional/provisional/locale/ Dieses Verzeichnis beinhaltet die GNU-gettext-Übersetzungsdateien in entsprechenden Unterverzeichnissen. Der Prototyp wird neben der englischen Standardsprache mit einer deutschen Sprachdatei ausgeliefert.

provisional/provisional/static/provisional/ Dieser Ordner ist für die Ablage statischer Dateien vorgesehen, die dem Client vom Webserver bereitgestellt werden.

provisional/provisional/static/provisional/css/ Hier werden die eigenen CSS-Dateien gespeichert.

provisional/provisional/static/provisional/img/ Dies ist der Ablageort für Bilddateien.

provisional/provisional/static/provisional/js/ Dieses Verzeichnis beinhaltet JavaScript-Dateien. In den einzelnen Unterordnern sind verschiedene quelloffene JavaScript-Bibliotheken enthalten.

provisional/provisional/templates/provisional/ In diesem Ordner sind die HTML-Template-Dateien des Prototyps zu finden. Hier werden auch die zuvor erwähnten statischen Dateien über dynamische Pfadangaben eingebunden.

provisional/provisional/models.py Beinhaltet sämtliche Model-Klassen der Anwendung. Sie dienen dem objektrelationalen Mapper zur Abbildung von Python-Objekten auf die Tabellen einer relationalen Datenbank.

provisional/provisional/templatetags/ Beinhaltet selbst definierte Filter für die Template Engine.

provisional/provisional/auth_ad.py Hilfsklasse für Anbindung an ein Active Directory zur Benutzerauthentifizierung.

provisional/provisional/myforms.py Beinhaltet alle Formular-Klassen des Prototyps.

provisional/provisional/permissions.py Stellt Hilfsfunktionen für das selbstgebaute Zugriffsrechtesystem bereit.

provisional/provisional/settings.py Dieses Skript wurde von Django automatisch angelegt. Dort werden zentrale Einstellungen für die App festgelegt. Dazu zählen zum Beispiel Verbindungsparameter für die Datenbank, die eingebundenen Django-Module, Sprache, Zeitzone etc.

provisional/provisional/tools.py Beinhaltet ausgelagerte Hilfsfunktionen, die im Controller wiederverwendet werden können.

provisional/provisional/urls.py Hier findet das zentrale Mapping von URLs auf View-Funktionen statt.

provisional/provisional/views.py Beinhaltet die Views, die vom URL Mapping referenziert werden. Sie nehmen HTTP Requests entgegen, verarbeiten diese, generieren mit Hilfe der Template Engine HTML Code und antworten schließlich dem anfragenden Client.

provisional/provisional/wsgi.py Django nutzt als Deployment-Plattform für den Produktiveinsatz vorzugsweise Web Server Gateway Interface (WSGI). Damit kann die Anwendung später über einen richtigen Webserver wie zum Beispiel Apache bereitgestellt werden. In dieser Datei findet die WSGI-Konfiguration statt.

Model

Der konzeptionelle Aufbau des Prototyps folgt dem Model View Controller (MVC)-Prinzip. Von Anfang an wird eine saubere Trennung von Datenmodell, Präsentation und Programmsteuerung eingehalten. Dadurch steigert sich die Übersichtlichkeit des gesamten Quellcodes. Ein hoher Grad an Übersichtlichkeit ist eine Voraussetzung, um Sicherheit und Flexibilität zu gewährleisten. Darauf wird Wert gelegt, um den Grundstein für potenzielle Weiterentwicklungen des Demonstrators in einem Umfeld mit hohem Anspruch an Informationssicherheit zu legen.

Die MVC-Komponente *Model* wird durch Djangos objektrelationalen Mapper und sog. *Model*-Klassen realisiert. Der objektrelationale Mapper ermöglicht den Zugriff auf Datenbankinhalte über Python-Objekte. Er entkoppelt die Anwendung von der relationalen Datenbank, indem er mit Hilfe von automatisch generiertem SQL-Code mit der Datenbank kommuniziert. Dem Entwickler wird nicht nur die Arbeit abgenommen, selbst syntaktisch korrektes SQL zu schreiben, der Mapper trägt auch zur Sicherheit der Anwendung bei: Durch das automatische Escapen von Benutzereingaben besitzt er beispielsweise ein wirksames Mittel gegen Angriffe wie SQL Injections.

Die Model-Klassen beschreiben die Struktur der Daten, mit denen der Prototyp arbeitet. Jede Model-Klasse wird vom Framework auf eine Relation in einer Datenbank abgebildet. Diese Klassen haben gemeinsam, dass sie von der Klasse `django.db.models.Model` abgeleitet sind. Das ist erforderlich, damit sie über den objektrelationalen Mapper zugreifbar sind.

In Listing 7.1 ist zur Veranschaulichung ein Ausschnitt der Datei `models.py` zu sehen. Diese Datei beinhaltet die einzelnen Model-Klassen. Der Ausschnitt zeigt die Klasse `UserSetting`, die zum Erfassen benutzerspezifischer Programmeinstellungen dient.

Listing 7.1: Ausschnitt der Modell-Definitionen

```
1 from django.contrib.auth.models import User
2 from django.db import models
3
4 # User specific settings
5 class UserSetting(models.Model):
6     user = models.ForeignKey(User)
7     key = models.CharField(max_length=64)
8     value = models.CharField(max_length=255)
```

Jede einzelne Benutzereinstellung setzt sich aus dem Tripel $(user,key,value)$ zusammen. Das Attribut `user` ist ein Fremdschlüssel, der sich auf ein Benutzerkonto der Django-eigenen Relation `User` bezieht. Gekennzeichnet wird dies durch die Methode `ForeignKey` in Zeile 6 und dem entsprechenden Attribut `User` im Argument. Die dazu erforderliche Model- Klasse `django.contrib.auth.models.User` wird in der ersten Zeile eingebunden. Schlüssel-Wert-Paare mit dem Namen der Einstellungsoption und dem dazugehörigen Wert werden in den Attributen `key` und `value` abgelegt (Zeile 7 und 8). Bei den Datentypen handelt es sich um Strings, deren maximale Länge durch das Attribut `max_length` vorgegeben wird. Der objektrelationale Mapper bildet diese Datentypen auf *Varchars* mit der entsprechenden Länge ab.

Nach der Definition der Model-Klassen muss das Schema in die Datenbank übertragen werden. Das Django-Skript `manage.py` legt die einzelnen Tabellen und die dazugehörigen deklarativen Constraints zu diesem Zweck automatisch an. Auf Wunsch des Entwicklers gibt das Skript auch die SQL-Anweisungen aus, die es an die Datenbank zur Bereitstellung des Schemas senden würde, ohne jedoch die Datenbank zu verändern.

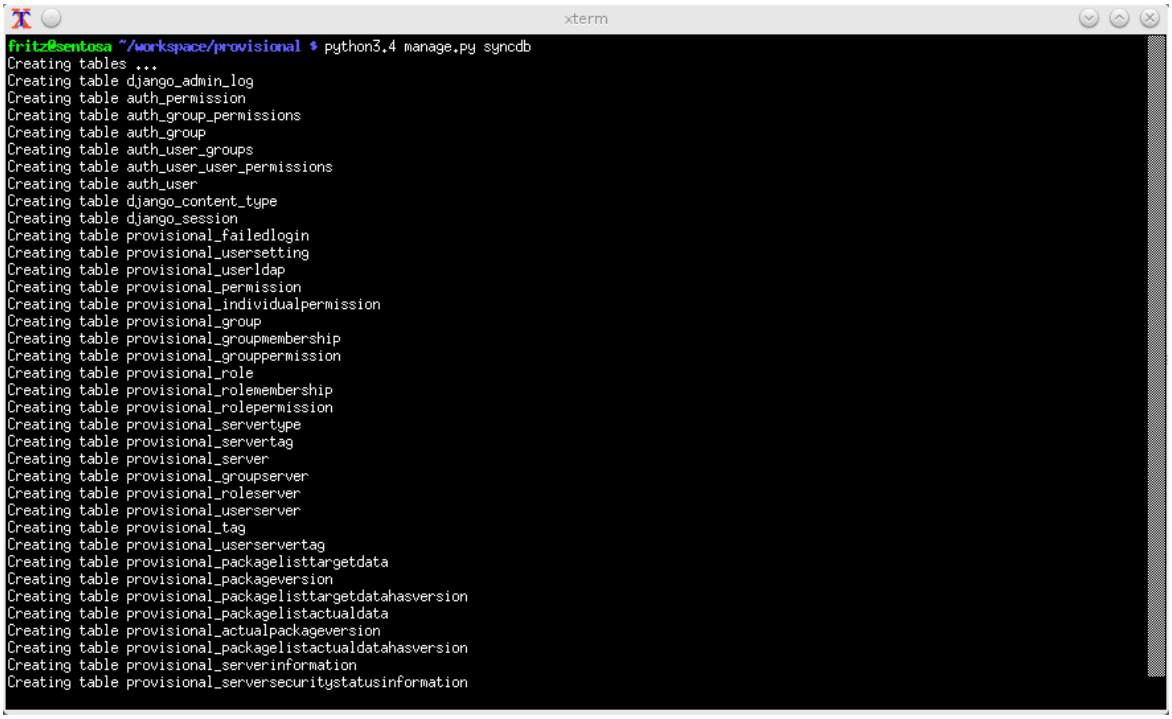


Abbildung 7.11.: Automatische Erzeugung der Relationen in der Datenbank durch Django.

Das automatische Anlegen der Tabellen geschieht durch den Befehl `python3.4 manage.py syncdb`. Abbildung 7.11 zeigt einen Screenshot von diesem Vorgang.

In Listing 7.2 ist beispielsweise der automatisch generierte SQL-Code zu sehen, der zur Erzeugung der Relation für die Benutzereinstellungen dient. Der Tabellenname in Zeile 3 wird mit einem einheitlichen Präfix versehen, das den Namen der Anwendung widerspiegelt. Das hat den Vorteil, dass Kollisionen zwischen mehreren Anwendungen vermieden werden, falls nur eine einzige Datenbank zur Verfügung steht. Die Attribute in den Zeilen 4 – 6 entsprechen den Attributen in den Zeilen 6 – 8 der Model-Klasse aus Listing 7.1. Für Transaktionssicherheit beim Einspielen des Schemas in die Datenbank sorgen die Zeilen 2 und 10.

Listing 7.2: Generierter SQL-Code für das Model der Benutzereinstellungen

```
1  fritz@linux-jjha:~/workspace/provisional> python3.4 manage.py sql provisional
2  BEGIN;
3  CREATE TABLE "provisional_usersetting" (
4      "user_id" integer NOT NULL PRIMARY KEY REFERENCES "auth_user" ("id"),
5      "key" varchar(64) NOT NULL PRIMARY KEY,
6      "value" varchar(255) NOT NULL
7  )
8  ;
9
10 COMMIT;
```

Standardmäßig erzeugt Django für jede Model-Klasse automatisch einen künstlichen Primärschlüssel in Form eines numerischen `id`-Attributs. Sobald in einer Model-Klasse ein eigenes Attribut als Primärschlüssel ausgezeichnet wird, tritt dieser Automatismus nicht in Kraft.

View

Antworten des Prototyps an den Client beinhalten HTML Code, der die Benutzeroberfläche beschreibt und vom Browser interpretiert wird. Um eine saubere Trennung der MVC-Komponenten zu gewährleisten, nutzt

der Prototyp die *Template Engine* des Frameworks. Zusammen mit HTML-Template-Dateien realisiert sie die Aufgabe der *View*-Komponente.

Zum dynamischen Füllen der Templates mit Inhalt stellt Django eine leistungsstarke Template-Sprache bereit. Sie unterstützt Variablen und spezielle Tags. Mit Hilfe der Tags lassen sich Templates mit einem gewissen Grad an Logik ausstatten. Damit sind zum Beispiel bedingte Ausgaben und Iterationen über Ergebnislisten von Datenbankabfragen möglich. Inhalte können über bereits vorhandene und selbst definierte Filtermethoden noch verändert werden, bevor die Template Engine den finalen HTML-Code erzeugt. Um Cross-Site-Scripting-Angriffe durch manipulierte Benutzereingaben zu verhindern, werden bestimmte Metazeichen automatisch durch entsprechende *HTML Entities* ersetzt.

Auch beim Prototyp erweist sich die Vererbbarkeit von Templates als besonders praktisch. Dadurch ist es möglich, eine Art Basis-Template bereitzustellen, das anwendungsweit genutzten HTML-Code beinhaltet und sog. Blöcke zu definieren, die von Kind-Templates mit dem eigenen Inhalt gefüllt werden. Da das System auch Template-Hierarchien unterstützt, sind flexible Verschachtelungen möglich. Dadurch wird redundanter Code, der zum Beispiel häufig benötigte Elemente der Bedienoberfläche beschreibt, in den einzelnen Template-Dateien vermieden. Das steigert nicht nur die Übersichtlichkeit des Codes, sondern wirkt sich auch vorteilhaft auf die Wiederverwendbarkeit einzelner Templates aus.

In Listing A.1 ist zur Veranschaulichung ein Auszug der Basis-Template-Datei des Prototyps zu sehen. Der Tag `{% load i18n %}` in Zeile 1 ist für die Unterstützung mehrerer Sprachen notwendig. Darauf wird später noch eingegangen. Nach dem *Doctype* für HTML5 (Zeile 3) und dem einleitenden `<html>`-Tag (Zeile 4) folgt der HTML-Header (Zeile 5 – 14). Dort werden statische *CSS*- und JavaScript-Dateien eingebunden, die in der kompletten Anwendung benötigt werden. Zur Steigerung der Flexibilität werden bei den `src`-Attributen keine absoluten Pfade verwendet. Stattdessen generiert die Template Engine die richtigen Dateipfade mit Hilfe des Tags `{% static %}` automatisch. Um diesen Tag in einem Template verwenden zu können, ist die Anweisung `{% load staticfiles %}` in Zeile 2 erforderlich. An der Stelle `{% block base_content %}{% endblock %}` können Inhalte anderer Templates eingefügt werden werden.

Auf unterster Ebene fallen die Templates relativ schlank aus. Listing A.2 zeigt einen Ausschnitt des Templates für die Benutzerkontenübersicht. Die Anweisung in der ersten Zeile bewirkt, dass der HTML-Code, der sich im Block `content` (ab Zeile 4) dieser Datei befindet, in den gleichnamigen Platzhalter der übergeordneten HTML-Datei `frontend.html` hineinkopiert wird. Diese Datei überschreibt wiederum Platzhalter in hierarchisch höher gelegenen Templates, bis schließlich das Basis-Template erreicht und mit sämtlichen Inhalten gefüllt wird.

Die beiden Template-Beispiele stammen jeweils aus der höchsten (Listing A.1) und aus der niedrigsten (Listing A.2) Template-Hierarchieebene. Was hierbei auffällt ist, dass noch gar kein HTML-Code vorhanden ist, der das Layout der vollständigen Bedienoberfläche vorgibt. Hier zeigt sich der große Vorteil der Template-Vererbung: Sie begünstigt eine Trennung von Layout und der Darstellung von Nutzinhalten der Anwendung und ermöglicht so eine hohe Flexibilität bei der Gestaltung der Bedienoberfläche. Von Anfang an wird dadurch beim Prototyp Redundanz in den Templates vermieden und die Übersichtlichkeit der Quelldateien gesteigert.

Die Benutzerkontenübersicht besteht aus einer Tabelle (Zeile 5 – 26), die die Template Engine mit Nutzinhalten füllt. In den Zeilen 16 – 24 wird pro Benutzerkonto eine eigene Tabellenzeile erzeugt. Attribute wie Benutzername, Vorname, Nachname etc. werden in entsprechende Platzhalter eingefügt, die im Gegensatz zu Tags durch zwei geschweifte Klammern gekennzeichnet sind. Als Datenquelle für diese Iteration dient die Datenstruktur `users` (Zeile 16). Pro Iterationsschritt sind die Attribute des jeweiligen Benutzerkontos über die lokale Variable `user` zugreifbar. Die Datenquelle wird vom Controller bereitgestellt, der im anschließenden Abschnitt behandelt wird.

Damit der Anwender die Tabelle nach seinen eigenen Bedürfnissen sortieren kann, wird auf den zuvor vorgestellten Tablesorter zurückgegriffen. Die Initialisierung des Tablesorters erfolgt in den Zeilen 30 – 32. In den Zeilen 33 – 35 befindet sich ein Event Handler für eine Schaltfläche zum Hinzufügen eines neuen Benutzerkontos. Er bewirkt eine Weiterleitung zur entsprechenden Funktion der Anwendung.

Das Framework unterstützt *GNU gettext*[31] und bietet so umfassende Möglichkeiten für die Internationalisierung und Lokalisierung von Applikationen. Der Prototyp nutzt dieses Angebot, um Anwendern eine mehr-

[31] GNU-gettext-Website: `http://www.gnu.org/software/gettext/gettext.html`

sprachige Bedienoberfläche zur Verfügung zu stellen (aktuell stehen die Sprachen Deutsch und Englisch zur Auswahl). Im einfachsten Fall passt sich der Server an die bevorzugte Sprache automatisch an, die der aufrufende Client im HTTP-Header mitsendet, ansonsten ist die Standardsprache Englisch.

Um ein Template mehrsprachig zu machen, muss die Anweisung {`% load i18n %`} eingebunden werden (Zeile 2). Übersetzbare Strings der Standardsprache Englisch werden in {`% trans %`}-Tags eingesetzt. Nach dem Fertigstellen eines Templates ruft der Entwickler ein bestimmtes Kommandozeilentool auf, das alle in Frage kommenden Dateien nach diesen Tags durchsucht und die Strings in eine zentrale Sprachdatei übernimmt. Die deutsche Sprachdatei `django.po` befindet sich z. B. im Unterordner `locale/de/LC_MESSAGES/` des Applikations-Verzeichnisses.

Listing A.3 zeigt einen Ausschnitt der deutschen Sprachdatei. Sie setzt sich im Prinzip aus Blöcken mit den folgenden Bestandteilen zusammen:

- Kommentarzeile mit Positionsangabe eines Oberflächentextes (`#:...`, Zeilen 1, 2, 6, 10, 14, 18, 22)
- Oberflächentext in Originalsprache (`msgid`, Zeilen 3, 7, 11, 15, 19, 23)
- Platzhalter für Übersetzung (`msgstr`, Zeilen 4, 8, 12, 16, 20, 24)

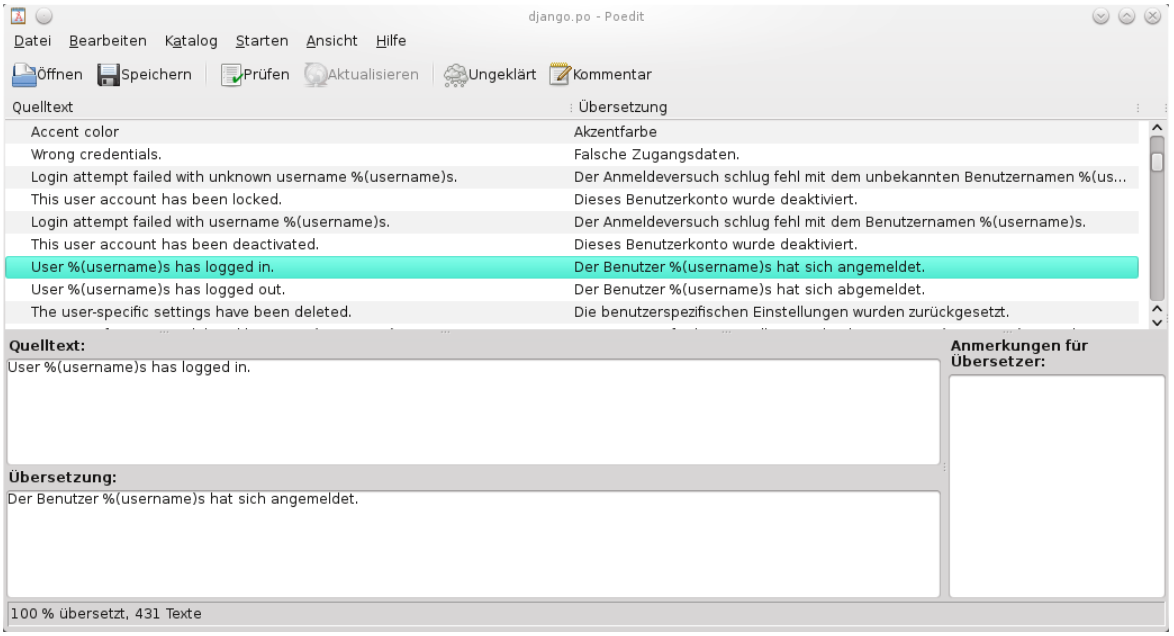


Abbildung 7.12.: Das Programm Poedit erleichtert die Bearbeitung von Sprachdateien im gettext-Format.

Anfangs sind die Platzhalter noch leere Strings. Mit einem Texteditor oder einem speziellen Programm wie z. B. *Poedit*[32] werden die übersetzten Texte dort eingefügt (siehe Abbildung 7.12). Im Anschluss wird diese Klartext-Datei von einem Kommandozeilentool in eine binäre Datei mit dem Namen `django.mo` umgewandelt. Der Prototyp greift schließlich auf diese binäre Sprachdatei zurück, um daraus die Oberflächentexte zu beziehen. Ein Neustart des Entwicklungsservers ist hierzu notwendig. Die Umwandlung in ein Binärformat bietet Performance-Vorteile gegenüber dem Klartextformat.

Die Template Engine von Django dient nicht nur zum dynamischen Füllen von Webseiten mit Inhalten, die für den Anwender interessant sind, sondern kommt auch zum Einsatz, um in den Webseiten enthaltene JavaScript-Programmcodes dynamisch zu komplettieren. Diese Technik findet zum Beispiel bei häufig wiederkehrenden Bildschirmelementen wie Fortschrittsbalken Verwendung. Durch die serverseitige Generierung von JavaScript Code verringert sich der Implementierungsaufwand auf der Seite des Clients.

Ein weiterer Anwendungsfall für die Template Engine ist die dynamische Einbindung von JavaScript-Dateien: Der Demonstrator greift auf eine Reihe etablierter JavaScript-Bibliotheken zurück, die im Abschnitt 7.2.1 be-

[32] Poedit-Website: `http://poedit.net/`

reits vorgestellt wurden. Diese Bibliotheken sind in separate Dateien ausgelagert, die im `<head>`-Element des Basis-Templates eingebunden werden. Um die Performance der Anwendung zu erhöhen und sowohl Netzwerkverkehr als auch Serverlast zu verringern, fordert der Client nur diejenigen JavaScript-Bibliotheken vom Server an, die tatsächlich von der momentan angezeigten Webseite benötigt werden. Die bedingte Einbindung der Bibliotheken geschieht durch die Prüfung spezieller Flags, die von der jeweiligen View in die `context`-Datenstruktur eingefügt wurden.

Listing 7.3: Bedingte Einbindung der der JavaScript-Bibliothek Flot und einiger Plug-ins.

```
1 {% if load_flot %}
2 <script type="text/javascript" src="{% static 'provisional/js/flot/jquery.flot.js
      ' %}"></script>
3 <script type="text/javascript" src="{% static 'provisional/js/flot/jquery.flot.
      pie.js' %}"></script>
4 <script type="text/javascript" src="{% static 'provisional/js/flot/jquery.flot.
      resize.js' %}"></script>
5 <script type="text/javascript" src="{% static 'provisional/js/flot/jquery.flot.
      stack.js' %}"></script>
6 <script type="text/javascript" src="{% static 'provisional/js/flot/flot-
      axislabels-master/jquery.flot.axislabels.js' %}"></script>
7 {% endif %}
```

In Listing 7.3 ist ein Beispiel für die bedingte Einbindung von JavaScript-Dateien zu sehen. Der Demonstrator verwendet auf einigen Webseiten die JavaScript-Bibliothek Flot zusammen mit einigen dazugehörigen Plug-ins zur dynamischen Generierung von Diagrammen. Die insgesamt fünf dafür benötigten Quellcode-Dateien werden durch die Prüfung des Flags mit der Bezeichnung `load_flot` nur auf den Webseiten eingebunden, wo sie tatsächlich benötigt werden.

Controller

Bei auf Django basierenden Projekten wird der *Controller* durch als *Views* bezeichnete Funktionen in der Datei `views.py` realisiert – auch wenn der Dateiname etwas anderes vermuten lässt. Eine View entspricht einer funktionalen Komponente der Anwendung bzw. im ursprünglichen Sinn einer konkreten Webseite. Dazu zählen zum Beispiel die Benutzerübersicht oder der Benutzereinstellungsdialog.

Listing 7.4: View für die Benutzerkontenübersicht im Administrationsbereich.

```
1  # Administration: Users overview
2  @login_required
3  @cache_control(no_cache=True, no_store=True)
4  def admin_users(request):
5      context = {}
6
7      # Get users from database
8      users = User.objects.all()
9      context.update({'users': users})
10
11     return render_to_response('provisional/admin_users.html', context,
           context_instance=RequestContext(request, processors=[
           general_context_processor, logged_in_context_processor]))
```

Die Hauptaufgabe einer View ist es, Informationen vom aufrufenden Client entgegenzunehmen, zu verarbeiten, Datenbankoperationen durchzuführen und dem Client eine Antwort in Form einer sog. `HttpResponse` zurückzugeben. In der Regel handelt es sich dabei um `HTML`-Code, aber auch Inhalte mit abweichendem Multipurpose Internet Mail Extensions (MIME)-Type sind möglich. Zur Generierung von HTML-Code wird auf die Template Engine zurückgegriffen. Um dynamisch generierte Webseiten zu erhalten, sind eine Template-Datei und eine `context`-Datenstruktur notwendig. Letztere besteht aus einem Dictionary, das die Nutzinhalte für das Template bereitstellt.

In Listing 7.4 ist anhand der Benutzerkontenübersicht ein einfaches Beispiel für den grundlegenden Aufbau einer View zu sehen. Die Benutzerkontenübersicht besteht aus einer Tabelle, die alle dem System bekannten Benutzerkonten und einige Zusatzinformationen anzeigt.

Bei den Zeilen 2 und 3 handelt es sich um Annotationen der View-Funktion. `@login_required` wird vom Framework bereitgestellt und stellt sicher, dass die View nur dann aufgerufen werden kann, wenn der Benutzer eingeloggt ist (Benutzerrechte sind in diesem Beispiel noch nicht implementiert). Das Verhalten dieser Annotation ist in der Datei `settings.py` so konfiguriert, dass ggf. eine Umleitung zum Anmeldebildschirm erfolgt.

Die Annotation `@cache_control` in Zeile 3 beeinflusst das Caching des aufrufenden Browsers. Hier wird der Browser über entsprechende Einträge im HTTP-Header angewiesen, die Webseiten des Prototyps immer komplett neu anzufordern und nicht auf den eigenen Cache zurückzugreifen. Ein wichtiger Grund dafür liegt in der höheren Sicherheit.[33]

In Zeile 8 wird der objektrelationale Mapper benutzt, um alle Benutzerkonten der Datenbank abzufragen. Das Abfrageergebnis wird in der Datenstruktur `users` zwischengespeichert. Anschließend werden die Benutzer in die `context`-Datenstruktur aufgenommen (Zeile 9), die als Datenquelle für die Template Engine dient. Die `context`-Datenstruktur wurde zuvor in Zeile 5 initialisiert.

In Zeile 11 wird das `context`-Dictionary der Funktion `render_to_response()` übergeben. Unter der Angabe der entsprechenden Template-Datei generiert diese Funktion ein `HttpResponse`-Objekt, das den HTML-Code der Benutzerkontenübersicht enthält, und sendet eine Antwort an den Client.

`render_to_response` erlaubt die Verwendung mehrerer sog. *Context Processors*. Das sind separate Funktionen, die unabhängig von der Datenquelle der View zusätzliche Datenquellen bereitstellen. Der Prototyp nutzt diese Möglichkeit, um Inhalte zur Verfügung zu stellen, die auf mehreren oder allen Webseiten der Anwendung benötigt werden. Dabei handelt es sich zum Beispiel um den Namen des Programms, der immer in der Titelleiste des Browsers angezeigt werden soll.

Durch die Auslagerung von häufig genutzten Inhalten in Context Processors wird von Anfang an Redundanz im Controller vermieden und die Übersichtlichkeit erhöht. Änderungen an diesen Inhalten sind an zentraler Stelle möglich und wirken sich auf alle Views aus, die den jeweiligen Context Processor verwenden.

Die Tatsache, dass die Datei `views.py` die komplette Controller-Komponente der Anwendung bereitstellt, führt zwangsweise zu einer Datei mit mehreren tausend Zeilen. Es ist auch denkbar, von dem der Django-Entwickler propagierten Konzept abzuweichen und separate Dateien für die einzelnen Controller-Bestandteile zu verwenden. Bei der Implementierung des Prototyps wurde auf die klassische Django-Variante zurückgegriffen, um die Einarbeitungszeit für potenzielle Erweiterungen durch andere Personen zu verringern.

URL Mapping

Bisher wurden mit den Views, Templates und Models essentielle Bestandteile des Prototyps vorgestellt. Da es sich um eine webbasierte Anwendung handelt, müssen ihre Ressourcen aber auch über HTTP-Methoden wie `GET` und `POST` erreichbar sein. Aus diesem Grund ist ein Mapping der einzelnen Views auf eindeutige URLs notwendig. Dies geschieht an zentraler Stelle in der Datei `urls.py`. Listing A.4 zeigt einen Ausschnitt dieser Datei.

Kernbestandteil dieser Datei sind URL Pattern. In dieser Liste werden einzelne Views der Datei *views.py* durch reguläre Ausdrücke mit URLs assoziiert (Zeilen 6 – 25). Beim Aufruf einer Seite prüft der Server diese Liste der Reihe nach von oben nach unten, bis ein passender Eintrag gefunden wird und ruft die entsprechende View mit einem `HttpRequest`-Objekt als Parameter auf. Diese Datenstruktur stellt der View unter anderem Informationen aus dem HTTP Header des Absenders bereit.

[33] Manche Webanwendungen mit Benutzerauthentifizierung bieten Angreifern die Möglichkeit, über die Zurück-Schaltfläche an Inhalte zu gelangen, die ein Anwender zuvor aufgerufen hat – selbst nach einem erfolgten Log-out. Indem der Browser dazu angewiesen wird, den Verlaufspeicher zu umgehen, kann das nicht passieren. Der Prototyp führt beim Versuch, vorherige Seiten aufzurufen, immer eine Umleitung zur Anmeldemaske durch. Aufgrund unterschiedlicher Implementierungen zeigen leider nicht alle Browser dieses gewünschte Verhalten. *Mozilla Firefox* verhält sich hier zumindest vorbildlich. Es gibt aber noch weitere Methoden, die zum Teil auf JavaScript zurückgreifen, um einen ähnlichen Effekt herbeizuführen. Durch die Deaktivierung von JavaScript können diese aber leicht umgangen werden.

Da die Pattern auf regulären Ausdrücken basieren, können auch Platzhalter in die URLs eingebaut werden. Platzhalter können beispielsweise für Identifiers (IDs) von Datensätzen vorgesehen werden, damit ein bestimmter Datenbankinhalt direkt über die Adresse referenziert werden kann. Ein anderer denkbarer Einsatzbereich ist die Unterbringung einer Jahreszahl in der Adresse, um auf Ressourcen eines bestimmten Jahres zuzugreifen. Beim Aufruf einer URL, deren Platzhalter durch Werte vom richtigen Datentyp ersetzt wurden, übergibt das Framework diese Werte in der Reihenfolge ihres Auftretens als zusätzliche Funktionsparameter an die entsprechende View. Diese Art von Platzhaltern stellt eine Alternative zu herkömmlichen URL-Parametern dar, die mit einem Fragezeichen an die Adresse angehängt werden. Die Extraktion von URL-Parametern und die Typprüfung bliebe sonst Aufgabe der jeweiligen View.

Übersichtlicher ist es allerdings, sog. *Named Regular Expression Groups* zu verwenden. Sie besitzen die Syntax `(?P<name>pattern)`, wobei `name` ein vom Entwickler wählbarer Bezeichner für die Gruppe ist und `pattern` den regulären Ausdruck für die Variable enthält. Wird eine entsprechende URL aufgerufen, übergibt das Framework die Zusatzinformationen in Form von Schlüssel-Wert-Paaren als zusätzliche Parameter an die entsprechende View-Funktion. Dieses Vorgehen hat den Vorteil, dass sowohl im URL Mapping als auch in der View übereinstimmende Bezeichner für die Variablen verwendet werden und trägt zur Übersichtlichkeit des Programmcodes bei.

Listing A.4 besitzt in Zeile 18 eine Named Group. Am Ende der URL `admin/users/edit/` wird eine obligatorische Ganzzahl erwartet, die hier für die ID des zu bearbeitenden Benutzerkontos steht. Wenn der Client die Ressource beispielsweise mit einer an der Adresse angehängten 2 anfordert, ruft das Framework die Funktion `admin_users_edit(request, user_id=2)` auf.

Praktisch ist, dass das Framework beim URL Mapping bereits eine Typprüfung durchführt. Es wird nur dann eine View mit den Variablen aus der URL aufgerufen, wenn ein passendes Pattern existiert. Diese Typprüfung muss dann nicht mehr in den Views durchgeführt werden, wodurch der Programmcode kompakter und übersichtlicher ausfällt. Eine View muss auch nicht selbst überprüfen, ob ein obligatorisches Attribut tatsächlich vorhanden ist. Sollte das nicht der Fall sein, wird sie gar nicht erst aufgerufen.

Beim URL Mapping wird von Anfang an Wert auf ein einheitliches und nachvollziehbares Schema gelegt. Gleichzeitig wird bereits zu Beginn des Anwendungsentwurfs der Grundstein dafür gelegt, potenzielle Änderungen oder Erweiterungen des Schemas zu erleichtern. Als besonders praktisch erweist sich zu diesem Zweck die Möglichkeit, den einzelnen URL Pattern Namen zu geben. Der Grund ist folgender: In den HTML-Templates des Prototyps kommen zahlreiche URLs vor – entweder als herkömmliche Hyperlinks via `<a>`-Tag oder als Umleitungen innerhalb von JavaScript-Code. Wenn Änderungen am URL Mapping durchgeführt werden, müssten normalerweise alle Templates nach dieser URL durchsucht und alle Treffer an das neue Mapping angepasst werden. Dies kann zum Beispiel dann notwendig sein, wenn eine Programmfunktion an eine andere Stelle verschoben wird. Je nachdem, wie viel sich am URL Mapping ändert, kann das sehr viel Zeit in Anspruch nehmen.

Um dies zu vermeiden, verzichtet der Prototyp weitestgehend auf hart kodierte Adressen. Durch die Verwendung von `{% url %}`-Tags müssen die Templates nach einer Änderung des Mappings gar nicht mehr angetastet werden. Innerhalb dieser Tags wird lediglich auf die Namen der URL Pattern verwiesen, die in der Datei `urls.py` zentral festgelegt wurden. Das Anhängen von Werten an URLs oder das Ausfüllen dort enthaltener Variablen ist innerhalb von Templates weiterhin möglich.

Im zuvor erwähnten Listing A.2 wird beispielsweise vom `{% url %}`-Tag Gebrauch gemacht. In Zeile 34 erfolgt mit dessen Hilfe ein Redirect via JavaScript zu der Adresse, die beim URL Mapping den Namen `admin_users_add` erhalten hat. In diesem Fall handelt es sich um ein Formular zum Erfassen der Daten für einen neu anzulegenden Benutzer-Account. In Zeile 18 ersetzt der Tag eine hart kodierte URL in einem Hyperlink. Dort wird die URL noch dazu um ein Attribut ergänzt, damit das Framework ein passendes Mapping findet. Diese URL ruft ein Formular auf, das zur Bearbeitung des Benutzerkontos mit der spezifizierten ID dient.

Formulare

Der Prototyp stellt nicht nur Informationen in eine Richtung dar, sondern ermöglicht auch Benutzereingaben über Formularfelder. Dies ist zum Beispiel im administrativen Bereich der Fall, wenn ein neues Benutzerkonto

angelegt werden soll. Zu diesem Zweck sind Angaben wie Benutzername, Vorname, Nachname, Passwort und die E-Mail-Adresse erforderlich. Dort wo eine webbasierte Anwendung Benutzereingaben erlaubt, ist besondere Vorsicht geboten, denn sie ermöglicht Angreifern die Ausnutzung von Programmschwächen. Um das Rad nicht neu zu erfinden, setzt der Prototyp bei Eingabeformularen auf die bewährten Sicherheitsfunktionen, die das Django-Framework bereitstellt.

Das fängt bereits bei der Erstellung der Formulare an. Anstatt wie bei klassischen Anwendungen die Formularfelder als HTML-Code hart in den Templates zu kodieren, verwendet der Prototyp einen ähnlich deklarativen Ansatz wie auch bei der Definition des Datenmodells: Jedes Eingabeformular wird durch eine eigene Python-Klasse beschrieben, die sich in der Datei `myforms.py` befindet. Diese Klassen sind abgeleitet von der Django-eigenen Klasse `django.forms.Form`. Sie bestimmt, wie Formulare funktionieren und auf dem Bildschirm erscheinen.

Eine Formular-Klasse besteht im einfachsten Fall aus den einzelnen Formularelementen eines bestimmten Typs. Das Framework unterstützt die klassischen Formularelemente, die HTML mit sich bringt. Dazu sind zum Beispiel Textfelder, Drop-Down-Menüs und Checkboxen. Herkömmliche HTML-Formularelemente können mit Hilfe des Frameworks um spezielle Typprüfungen erweitert werden. Das umfasst zum Beispiel Textfelder mit einer Typprüfung für E-Mail-, IP- oder MAC-Adressen. Darüber hinaus können Entwickler auch eigene Feldtypen oder Typprüfungen für Eingabefelder mit Hilfe von regulären Ausdrücken programmieren. Der Prototyp nutzt die Formularklassen außerdem für die mehrsprachige Beschriftung der Formularfelder und für die Beeinflussung ihrer Darstellung. Diese saubere Trennung von Formularen und HTML-Templates wirkt sich vorteilhaft auf die Übersichtlichkeit des Quellcodes aus und ermöglicht außerdem die Wiederverwendung von Formularen.

Listing A.5 zeigt einen Ausschnitt der Formularklassen des Prototyps. Zu sehen sind dort die Formulare für das Anmeldefenster und die Erfassung der Daten für ein neu anzulegendes Benutzerkonto. Das Erste Formular besteht aus zwei Textfeldern, deren maximale Eingabekapazitäten über Attribute auf sinnvolle Werte beschränkt sind (`max_length`). Das Attribut `required` gibt an, ob ein Formularfeld zwingend ausgefüllt werden muss. Über `label` wird die Beschriftung des Felds festgelegt. Da der Prototyp, wie bereits erwähnt wurde, auf eine konsequente Unterstützung von Internationalisierung und Lokalisierung ausgelegt ist, werden die Beschriftungen über die Methode `django.utils.translation.ugettext_lazy()` bereitgestellt. Sie ersetzt die übergebenen Strings durch die Zeichenketten aus der passenden Sprachdatei.

Nachdem die Formulare definiert wurden, können sie im Controller verwendet werden. Im Folgenden wird als Anschauungsbeispiel das Anmeldeformular `LoginForm` betrachtet. Die Verwendung von Formularen lässt sich an diesem einfachen Beispiel gut erklären, da das Anwendungsprinzip aller Formulare ähnlich ist. Listing A.6 zeigt die dazugehörige View.

In Zeile 8 prüft die Anwendung, ob der Anwender das Formular abgesendet hat. Falls nicht, wird in Zeile 37 ein leeres Anmeldeformular erzeugt und ab Zeile 39 zum Seiteninhalt hinzugefügt. Falls es schon abgesendet wurde, werden zunächst die Inhalte wiederhergestellt, die der Anwender in die Felder eingegeben hat (Zeile 9). Dies geschieht durch die Übergebe von `request.POST` an den Konstruktor des Formulars. In Zeile 10 werden dann die Benutzereingaben geprüft. Felder, die in der Formularklasse als `required` gekennzeichnet wurden, dürfen nicht leer sein, sonst ist die Eingabe nicht valide. Ebenso müssen die Eingaben den festgelegten Datentypen entsprechen.

Wenn alle Eingaben syntaktisch korrekt sind, prüft der Prototyp, ob die Zugangsdaten korrekt sind (Zeile 11). `django.contrib.auth.authenticate()` wird dazu aufgerufen. Der Benutzername und das Passwort werden nicht direkt vom Benutzer übernommen, sondern aus einem assoziativen Array namens `form.cleaned_data`. Dieses Array wird automatisch erzeugt, wenn die Validitätsprüfung bestanden wurde. Die darin enthaltenen Formularwerte wurden bereits zu einem konsistenten Format normalisiert und sind besser zu handhaben als die Werte, die direkt aus dem assoziativen Array `request.POST[]` stammen.

Wenn die Zugangsdaten falsch sind (Zeile 14) oder das Benutzerkonto von einem Administrator deaktiviert wurde (Zeile 21), wird die Anmeldung abgebrochen und das Anmeldefenster erneut aufgerufen. In einem Popup-Fenster, das mit jQuery UI realisiert wird, bekommt der Anwender den Grund des fehlgeschlagenen Logins angezeigt.

Wenn alles in Ordnung ist, wird der Benutzer eingeloggt (Zeile 29). Das bewirkt, dass eine Benutzersitzung gestartet wird, deren Informationen in der Datenbank gespeichert werden. Zur Identifizierung des Browsers

wird ein Session-Cookie verwendet. Nach der Anmeldung erfolgt eine Umleitung zur Startseite des Prototyps (Zeile 35). Falls eine spezielle Funktion der Anwendung über ein Lesezeichen aufgerufen wurde oder nach langer Inaktivität die Benutzersitzung zum Zeitpunkt des Aufrufs einer Funktion abgelaufen war, erfolgt die Umleitung direkt zu der entsprechenden URL (Zeile 33).

Auch in den Views verwendet der Prototyp keine hart kodierten URLs. Die Funktion `reverse()` aus dem Package `django.core.urlresolvers` nimmt in Zeile 35 den Namen der Adresse aus dem URL Mapping entgegen und ersetzt sie automatisch durch die tatsächliche Adresse. Dies erhöht die Wartungsfreundlichkeit der Anwendung.

Schutzmechanismen

In den vorherigen Abschnitten wurde bereits erwähnt, dass der Prototyp ein bestimmtes Maß an Sicherheit durch das Escapen von Benutzereingaben und eine konsequente Typprüfung bietet. Auch gegen CSRF-Angriffe besitzt der Prototyp einen Schutzmechanismus. Da soeben der Einsatz von Formularen besprochen wurde, bietet sich die Gelegenheit an, den CSRFSchutzmechanismus zu erklären.

Listing A.7 zeigt die Template-Datei des Login-Formulars. Das Formular-Element mit der `id login_form` wird von der Template-Engine automatisch mit den entsprechenden Formularfeldern ausgefüllt. In Zeile 7 befindet sich ein Template Tag mit der Bezeichnung {`% csrf_token %`}. Er ist Bestandteil des CSRF-Schutzmechanismus des Frameworks. Vor dem Ausliefern des Formulars fügt die Template Engine an dieser Stelle ein verstecktes Formularelement ein, dass einen zufälligen Wert enthält. Diesen Wert enthält das Formular im Browser des Anwenders nur, wenn es ordnungsgemäß aufgerufen wurde. Ein Angreifer, der diesen Wert nicht kennt, kann zwar manipulierte Formulardaten mit Hilfe eines HTTP Clients wie *cURL*[34] an den Server senden, dieser ignoriert sie aber aufgrund des fehlenden CSRF Tokens.

Veraltete Browser-Versionen stellen ein grundsätzliches Problem dar, nicht nur am LRZ. Wenn der Hersteller den Browser nicht mehr mit Sicherheits-Updates versorgt, können Lücken im Programm von präparierten Webseiten ausgenutzt werden, um an persönliche Daten des Anwenders zu gelangen. Ein anderer Grund, um veraltete Browser auszusperren, besteht in der fehlenden Unterstützung sicherheitsrelevanter Technologien wie zum Beispiel TLS 1.2. Microsofts Internet Explorer unterstützt diese Protokollversion beispielsweise erst ab Version 11.[35]. Sowohl die IETF als auch das BSI empfehlen seit kurzem, nach Möglichkeit nur noch TLS 1.2 auf den Servern bereitzustellen [IETF 15] [BSI 15].

Fehlende oder schlecht unterstützte Basistechnologien der Webentwicklung wie HTML5 oder CSS3 sind weitere Gründe, um veraltete Browser auszusperren. Es ist zwar möglich, durch Workarounds und die Nutzung spezieller Web Frameworks einige Defizite alter Browser zu kompensieren. Dadurch erhöht sich aber auch der Implementierungs- und Wartungsaufwand. Betreiber öffentlich zugänglicher Websites nehmen diesen Aufwand in Kauf, da viele Anwender mit einem veralteten Browser surfen. Laut einer Studie von *Kaspersky* aus dem Jahr 2012 greifen rund ein Viertel der Anwender auf veraltete Browser-Versionen zurück [KAS 12].

Bei einem System wie dem Demonstrator, das nur von einem begrenzten internen Anwenderkreis am LRZ verwendet wird, lohnt sich der Zusatzaufwand nicht, veraltete Browser-Versionen zu unterstützen. Es ist davon auszugehen, dass die Zielgruppe der Anwendung aktuelle und gepatchte Browser verwendet. Darüber hinaus kann die fehlende Unterstützung zeitgemäßer Kryptographiealgorithmen auch durch Workarounds nicht kompensiert werden.

Webanwendungen können anhand des User Agent (UA) Strings erkennen, welchen Browser der jeweilige Anwender verwendet (oder zumindest, welcher er vorgibt, zu sein) und darauf reagieren. Die Verwendung des Prototyps ist Browsern untersagt, die nach Ermessen des Betreibers als unsicher einzustufen sind. Er nutzt dazu eine eingebaute Funktion des Frameworks, um bestimmte UAs auszuschließen. Dies geschieht über die Liste `DISALLOWED_USER_AGENTS` in der Datei `settings.py`, die mit regulären Ausdrücken versehen wird. Setzt ein Anwender einen Browser ein, dessen UA String mit einem der regulären Ausdrücke zusammenpasst, wird er automatisch auf eine Webseite mit dem Hinweis *FORBIDDEN* umgeleitet. Das gilt unabhängig davon, welche konkrete URL aufgerufen wird. Listing 7.5 zeigt einen Ausschnitt dieser Liste.

[34] cURL-Website: `http://curl.haxx.se/`
[35] Quelle: `http://en.wikipedia.org/wiki/Internet_Explorer_11`

Bei Django-basierten Anwendungen ist standardmäßig eine Funktion aktiv, die Clickjacking-Angriffen entgegenwirkt. Durch die Anweisung `X-Frame-Options SAMEORIGIN` im Antwort-HTTP-Header wird der Browser angewiesen, nur dann eine Ressource in einem Frame oder Inline Frame (Iframe) zu laden, wenn auch der verursachende Request von derselben Website kam.

Der im Rahmen dieser Masterarbeit zu entwickelnde Demonstrator geht noch einen Schritt weiter, indem er komplett auf potenziell unsichere Frames und Iframes verzichtet. Das flexible Template-System von Django und die Gestaltungsmöglichkeiten von CSS machen dies möglich. Sofern es sich nicht um Iframes handelt, verbietet das W3C den Einsatz von Frames ab HTML5 sogar [W3C 14].

Listing 7.5: Einzelne Browser können von der Anwendung ausgeschlossen werden.

```
1  import re
2
3  # Disallow some unsafe Browsers
4  DISALLOWED_USER_AGENTS = (
5      re.compile(r'Chrome'),
6      re.compile(r'Google'),
7      re.compile(r'MSIE 9.0'),
8      re.compile(r'MSIE 8.0'),
9      re.compile(r'MSIE 7.0'),
10     re.compile(r'MSIE 6.0'),
11 )
```

7.2.3. Client-Seite

Dieser Abschnitt widmet sich Implementierungsdetails, die für die dynamische Erzeugung von Visualisierungen auf der Client-Seite bedeutsam sind.

AJAX

Bei traditionellen Webanwendungen dient der Browser nur als Anzeigeinstrument für Webseiten und zur Erfassung von Benutzereingaben. Im Kontext von MVC übernimmt der Browser hier in erster Linie die Aufgabe der View. Viele moderne Webanwendungen beauftragen den Browser mittlerweile auch mit Aufgaben aus den Bereichen Model und Controller. Sie setzen komplexe JavaScript Frameworks ein, die die Gestaltung von Webseiten und Bedienoberflächen komplett im Browser ablaufen lassen. Der Datenaustausch zwischen dem Client und dem Server beschränkt sich dort nur noch auf Datenstrukturen mit den konkreten Anwendungsdaten.

Um den Implementierungsaufwand für einen Demonstrator gering zu halten, wird in dieser Arbeit ein vernünftiger Mittelweg eingeschlagen: Der Prototyp entspricht einer klassischen Webanwendung, die an sinnvollen Stellen um JavaScript-Komponenten erweitert wurde. Zur Abfrage aktueller Nutzdaten vom Server kommen dennoch effiziente AJAX-Aufrufe zum Einsatz, um Ladezeiten und die Netzlast zu verringern. Durch den gezielten Einsatz von AJAX-Aufrufen fühlt sich die Bedienung für den Anwender flüssiger an, da nicht ständig komplette Webseiten vom Server generiert und neu geladen werden müssen.

Das eingesetzte JavaScript Framework jQuery stellt Funktionen bereit, um AJAX-Operationen in Effizienter Weise auf Webseiten zu integrieren. In Listing 7.6 ist ein Beispiel für einen AJAX-Aufruf zu sehen. In Zeile 1 wird eine jQuery bereitgestellte Funktion für allgemeine AJAX-Operationen aufgerufen, die standardmäßig einen HTTP GET Request durchführt.

Die aufzurufende Adresse auf dem Webserver wird durch eine Datenstruktur in Zeile 2 übergeben. In den Zeilen 3 bis 7 wird eine anonyme Callback-Funktion angegeben, die jQuery aufruft, sobald der Webserver erfolgreich geantwortet hat. Sie überprüft, ob der Server tatsächlich Nutzdaten liefern konnte (Zeile 4) und leitet die Nutzdaten im Erfolgsfall an eine Funktion weiter, die sich um die Aktualisierung eines Diagramms mit den neuen Kennzahlen kümmert.

Bei den Daten, die per AJAX vom Server abgefragt werden, handelt es sich um JSON-Datenstrukturen. Sie zeichnen sich im Gegensatz zu XML durch einen geringeren Overhead aus und können vom JavaScript-Programmcode direkt verarbeitet werden, ohne sie zuvor in ein kompatibles Format konvertieren zu müssen.

Listing A.8 beinhaltet den Auszug einer solchen Antwort. Dort ist zu erkennen, dass ausschließlich strukturierte Kennzahlen als Grundlage für eine dynamisch generierte Visualisierung übertragen werden. Es ist kein HTML oder CSS Code vorhanden.

Listing 7.6: Ein AJAX-Aufruf zur Abfrage von aktuellen Kennzahlen für die Visualisierung von Softwarepaket-Statistiken.

```
1 $.ajax({
2     url: "{% url 'usecases_software_statistics_ajax_itemized_categories' %}",
3     success: function(result) {
4         if (result['success']) {
5             updateItemizedCategoriesData(result['data']);
6         }
7     }
8 });
```

Der JSON Code in diesem Ausschnitt beinhaltet die Anzahlen von Servern, die über einen aktuellen Softwarestand verfügen (Zeile 5 – 14), aufgeschlüsselt nach den Kategorien, zu denen die Server gehören (Zeile 18 – 21). Die in Zeile 16 herausgekürzten Abschnitte für Server mit veralteten Paketen und Paketen mit kritischen Lücken besitzen den gleichen Aufbau wie in den Zeilen 4 – 16.

Dynamisch generierte Visualisierungen

Die Client-Komponente des Prototyps nutzt die auf jQuery basierende JavaScript-Bibliothek Flot zum Zeichnen dynamischer Visualisierungen. Das bedeutet, dass der Server keine vorgefertigten Bilder ausliefert, sondern lediglich die Daten, auf denen ein Diagramm beruht. Diese Daten ruft der Client mit Hilfe der zuvor vorgestellten AJAX-Technologie asynchron vom Server ab.

Das Zeichnen eines Diagramms mit Flot erfolgt durch die Funktion `$.plot()`. Als Funktionsparameter nimmt sie ein `<div>`-Element als Platzhalter für die Visualisierung, die Nutzdaten im JSON-Format und eine Datenstruktur mit Formatierungsoptionen entgegen. Der `<div>` Container ist Bestandteil der HTML-Template-Datei der Webseite.

Listing A.9 zeigt die Datenstruktur mit den Formatierungsoptionen und den Code, der das Diagramm zeichnet. Die Datenstruktur beinhaltet Formatierungsanweisungen für die Abszisse (Zeile 2 – 8) und die Ordinate (Zeile 9 – 13). Der Code in den Zeilen 14 – 23 weist Flot an, ein gestapeltes Säulendiagramm zu zeichnen und gibt an, wie die Säulen zu zeichnen sind. Die Hilfslinien des Diagramms werden in den Zeilen 24 – 29 formatiert.

Das eigentliche Zeichnen des Diagramms mit allen erforderlichen Optionen findet in Zeile 37 statt. Die dem Diagramm zugrundeliegenden Daten werden der Funktion über die Variable `data` bereitgestellt. Sie enthält JSON Code, der zuvor per AJAX Request vom Server angefordert wurde.

7.2.4. Datenmodell

In diesem Abschnitt wird auf das Datenmodell des Demonstrators eingegangen. Es wurde im Lauf dieser Arbeit in mehreren Schritten erweitert, um die einzelnen Funktionen des Prototyps realisieren zu können.

Während der Implementierungsphase stellte sich heraus, dass der objektrelationale Mapper von Django in der aktuell verwendeten Version keine zusammengesetzten Primärschlüssel unterstützt. Diese Tatsache wirkt sich auch auf das Datenmodell aus: Fremdschlüssel, die Primärschlüsselkandidaten sind, bleiben herkömmliche Fremdschlüssel. Bei Datenbankabfragen ist es aus diesem Grund die Aufgabe der Anwendung, Prüfungen durchzuführen, die mit zusammengesetzten Primärschlüsseln überflüssig wären. Das Ergebnis ist ein etwas größerer Overhead, der aber noch zu verschmerzen ist.

Es stellte sich während der Implementierungsphase ebenfalls heraus, dass der objektrelationale Mapper von Django voraussetzt, dass jede Tabelle genau einen Primärschlüssel besitzen muss.[36] Aus diesem Grund enthält jede Tabelle den künstlichen Primärschlüssel `id`, der mit jedem neu angelegten Datensatz hochgezählt wird. Die Verwendung von `id`-Primärschlüsseln ist bei Django gängige Praxis, da diese automatisch angelegt werden, sofern kein alternativer Primärschlüssel vorgeschlagen wird.

Da der Schwerpunkt dieser Arbeit auf der Visualisierung sicherheitsrelevanter Serverzustände liegt, greift der Demonstrator nicht auf Echtzeitdaten zurück, die von den Servern kommen. Stattdessen greift er auf aggregierte Daten zurück, die den Zustand einer Server-Landschaft zu einem bestimmten Zeitpunkt widerspiegeln. Das Datenmodell ist aber flexibel genug ausgelegt, um in einer möglichen Folgearbeit echte Daten dort ebenfalls ohne große Umstrukturierungsmaßnahmen speichern zu können. Im Großen und Ganzen setzt sich das Datenmodell des Demonstrators aus drei essentiellen Bereichen zusammen:

Benutzer- und Rechteverwaltung Dieser Bereich ist für Benutzerkonten und benutzerspezifische Einstellungen zuständig. Da ein Schwerpunkt des Prototyps auch auf Informationssicherheit liegt, umfasst dieser Bereich auch ein auf Gruppen und Rollen basierendes Rechtemodell.

Server und Zustandsinformationen Dieser Bereich speichert Informationen über die der Anwendung bekannten Server und aggregierte Zustandsdaten als Grundlage für die Visualisierung.

Paketversionen und Referenzlisten Vorbereitung des Datenmodells für reale Serverdaten des Anwendungsfalls Software-Verwaltung.

Die einzelnen Bereiche des Modells werden in den folgenden drei Abschnitten genau beschrieben. Eine Schnellübersicht gibt es in dem darauf folgenden Abschnitt 7.2.4.

Modell der Benutzer- und Rechteverwaltung

Der Prototyp unterstützt eine Rechteverwaltung, die auf individuell zuweisbaren Rechten und Gruppen oder als Alternative dazu auf Rollen basiert. Die Benutzer- und Rechteverwaltung stellt den ersten essentiellen Bestandteil des Datenmodells dar, dessen Bestandteile in diesem Abschnitt betrachtet werden.

Anwendungen, die mit dem Django-Framework erstellt werden, können zwar auf ein bereits vorhandenes Rechtesystem zurückgreifen, das ebenfalls auf Gruppen basiert, es ist allerdings auf die Zugriffsbeschränkung von Datenbankinhalten ausgelegt. Zur Regelung des Zugriffs auf View-Ebene ist es nicht geeignet. Um einzelne Programmfunktionen des Prototyps dennoch mit einer feingranularen Zugriffsbeschränkung auszustatten, wurde deshalb ein eigenes System implementiert. Der dazugehörige Ausschnitt des Datenmodells ist in Abbildung 7.13 dargestellt.

Die Tabelle `User` stammt von Djangos eigener Benutzerverwaltung und wird vom Prototyp unverändert benutzt. Zu den benutzerbezogenen Daten, die sie speichert, zählen Benutzername, Vor- und Nachname und die E-Mail-Adresse. Im Attribut `password` befindet sich das ein Hash-Wert des vom Benutzer vergebenen Passworts, mit dem er sich bei der Anwendung einloggt. Das Attribut `is_active` kann benutzt werden, um einzelne Benutzerkonten zu deaktivieren. Einen Anmeldeversuch verweigert der Prototyp in diesem Fall mit einer entsprechenden Fehlermeldung.

Benutzerkonten können nicht nur manuell von einem Administrator deaktiviert werden. Um die Sicherheit zu erhöhen, sperrt der Prototyp Benutzerkonten automatisch, sobald mehrere Anmeldeversuche unmittelbar hintereinander fehlschlagen. Informationen über fehlgeschlagene Anmeldeversuche speichert der Prototyp in der Tabelle `FailedLogin`. Sie verfügt über eine Referenz zur Tabelle `User`, um den Bezug zu dem Benutzerkonto herzustellen. Das Attribut `when` speichert den Zeitpunkt des fehlgeschlagenen Logins.

Wenn das Attribut `is_superuser` in der Tabelle `User` auf `true` gesetzt wird, unterliegt das jeweilige Benutzerkonto keinerlei Einschränkungen. Die tatsächlich zugewiesenen Rechte spielen dann keine Rolle mehr. Da in diesem Fall die Rechteverwaltung komplett umgangen wird, ist es empfehlenswert, Superuser nur für Testzwecke zu verwenden und nach dem Test das Attribut wieder auf `false` zu setzen.

[36] Quelle: `https://docs.djangoproject.com/en/1.6/topics/db/models/#automatic-primary-key-fields`

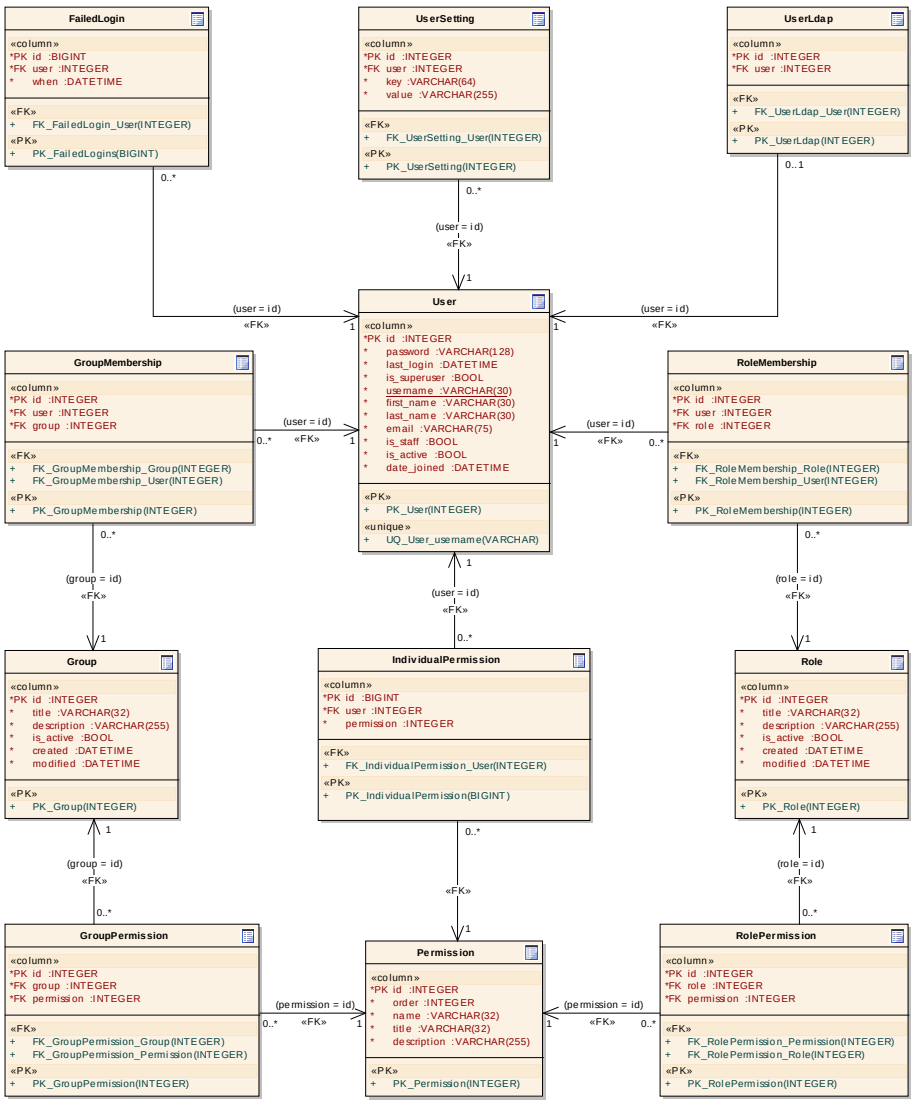


Abbildung 7.13.: Datenmodell der Benutzer- und Rechteverwaltung.

Die auf Benutzername und Passwort basierende Authentifizierung kommt standardmäßig mit den Attributen aus, die sich in der von Django bereitgestellten Tabelle `User` befinden. Sobald ein Anwender eine Sitzung gestartet hat, speichert Django Sitzungsinformationen in einer eigenen Tabelle. Diese und weitere Standard-Tabellen von Django werden in diesem Abschnitt nicht näher betrachtet, da sie keine Besonderheit des Prototyps darstellen.

Der Prototyp unterstützt eine Benutzerauthentifizierung über einen externen Active Directory Server. Zur Kennzeichnung eines Benutzerkontos für die auf LDAP basierende Authentifizierungsmethode genügt ein zusätzlicher boolescher Wert. Um die Struktur der vom Framework vorgegebenen `User`-Tabelle nicht zu verändern, wurde eine weitere Tabelle mit der Bezeichnung `UserLdap` eingeführt. Bei jedem Benutzerkonto, das über einen Fremdschlüssel in dieser Tabelle referenziert wird, erfolgt die Authentifizierung über das Active Directory.

Neben den Attributen, die der Administrator den einzelnen Benutzerkonten zuweist, existieren noch weitere Attribute, die die Anwender selbst und unabhängig voneinander speichern können. Dazu zählen zum Beispiel die persönlichen Anzeigeeinstellungen des Programms. Sie werden in Form von Schlüssel-Wert-Paaren in der Tabelle `UserSetting` abgelegt. Benutzerspezifisch werden diese Datensätze durch einen Fremdschlüssel, der die Tabelle `User` referenziert.

Die Zugriffsrechte, die der Prototyp kennt, sind in der Tabelle `Permission` gespeichert. Diese Tabelle ist von Anfang an mit den notwendigen Datensätzen gefüllt. Eine Bearbeitung der Inhalte aus der Anwendung heraus ist nicht vorgesehen. Das Attribut `title` beinhaltet den Namen des Zugriffsrechts, so wie er in der Administrationsoberfläche des Prototyps angezeigt wird. Im Feld `name` wird der Name des Rechts gespeichert, der programmintern für die Rechteprüfung verwendet wird. Das Attribut `description` ist für einen kurzen Text vorgesehen, der das jeweilige Zugriffsrecht für den Administrator der Anwendung beschreibt. Durch das Feld `order` können die Rechte der Übersichtlichkeit halber sortiert werden.

Einzelnen Benutzern können individuelle Rechte zugewiesen werden. Die dazu notwendigen Many-to-Many-Beziehungen zwischen den beiden Relationen `User` und `Permission` werden durch die Assoziationstabelle `IndividualPermission` hergestellt. Benutzergruppen werden in der Tabelle `Group` abgebildet. Diesen können ebenfalls Rechte zugewiesen bekommen, in diesem Fall über die Tabelle `GroupPermission`. Das gilt auch für Rollen der Tabelle `Role`. Sie erhalten ihre Rechte über die Relation `RolePermission`.

Durch die Zuweisung eines Benutzers zu einer oder mehreren Gruppen über die Tabelle `GroupMembership` erhält ein Benutzerkonto alle Rechte, die diesen Gruppen zugewiesen wurden. Diese Rechte addieren sich zu eventuell vorhandenen individuellen Benutzerrechten. Die Zuweisung eines Benutzerkontos zu einer Rolle erfolgt über die Relation `RoleMembership`. Auch in diesem Fall erhält der Benutzer alle Rechte der zugewiesenen Rolle, eventuell vorhandene individuelle Rechte und Rechte aus Gruppenzugehörigkeiten werden aber dann ignoriert.

Server und Zustandsinformationen

Der Bereich *Server und Zustandsinformationen* ist der zweite wesentliche Bestandteil des Datenmodells. Die dazugehörigen Tabellen befinden sich in Abbildung 7.14 Während sich beim vorherigen Bereich alles um die Relation `User` drehte, ist es hier die Tabelle `Server`. Damit der Demonstrator Zustandsdaten verschiedener Server unterscheiden zu kann, muss er auch Informationen über die einzelnen Server besitzen.

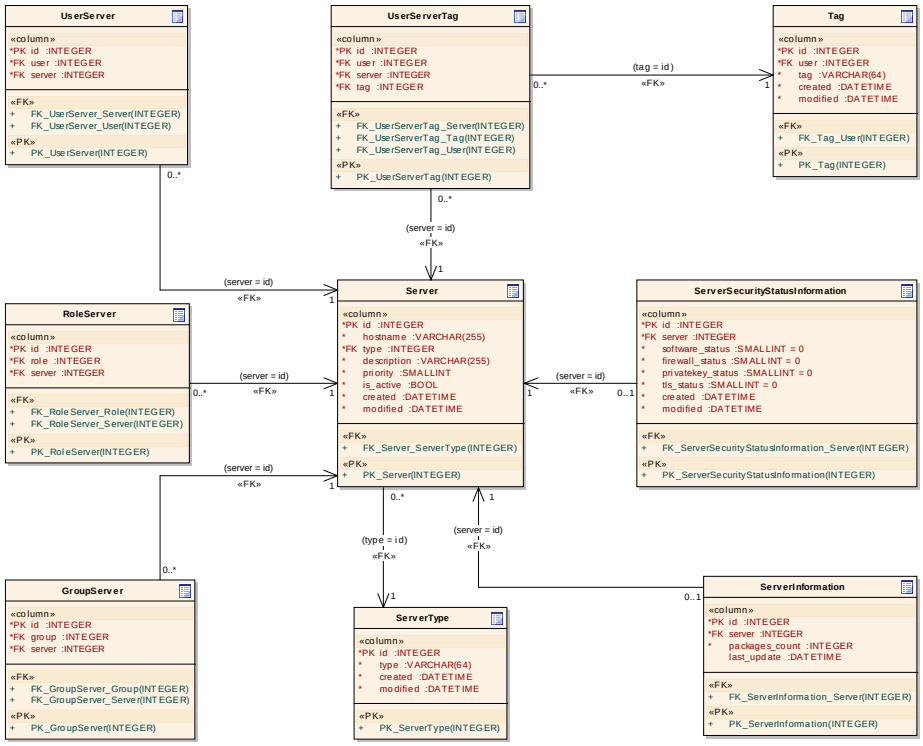


Abbildung 7.14.: Datenmodell der Server und Zustandsinformationen.

Jeder Server verfügt über einen eindeutigen Hostnamen als Identifikationsmerkmal für die Anwender des

Demonstrators. Beim Anlegen eines Servers in der Datenbank durch einen Administrator weist er dem Server eine Kategorie zu. Kategorien fassen Server zusammen, die einem ähnlichen Zweck dienen, z. B. ob es sich um einen Web Server oder einen Datenbankserver handelt. Kategorien können, wie später noch gezeigt wird, vom Administrator frei definiert werden. Im Auslieferungszustand kennt der Demonstrator bereits ca. 30 Stück. Gespeichert werden sie in der Tabelle `ServerType`. Die Referenzierung einer Kategorie erfolgt beim Server über das Attribut `type`.

Beim Erfassen eines neuen Servers in der Datenbank kann der Administrator einen frei wählbaren Beschreibungstext eingeben. Dieser wird in der Spalte `description` eines Server-Datensatzes gespeichert. Das Attribut `priority` speichert einen ganzzahligen Wert, der zur Unterscheidung der Wichtigkeit unterschiedlicher Server dient.

Der Wert des Attributs `is_active` kennzeichnet ähnlich wie in der Relation `User`, ob der Datensatz noch gültig ist oder nicht. Nicht mehr benötigte Server können vom Administrator als inaktiv gekennzeichnet werden. Sie tauchen dann in keiner der Visualisierungen mehr auf. In einer möglichen Erweiterung des Demonstrators würde das zentrale Informationssystem dann auch keine weiteren Daten vom Server mehr abfragen und aufzeichnen.

Als Datengrundlage für die Visualisierung der Serverzustände greift der Demonstrator auf aggregierte Werte zurück, die in zwei zusätzlichen Tabellen gespeichert sind. In einer möglichen Erweiterung des Demonstrators würden die Werte dieser beiden Tabellen regelmäßig neu berechnet und aktualisiert werden. Als Grundlage für diese Berechnung dienen die Daten, die von den in Kapitel 6 angesprochenen Prüfprogrammen auf den Servern bereitgestellt werden.

Die Tabelle `ServerInformation` besitzt für jeden Server einen Eintrag, der Auskunft über die derzeitige Anzahl an installierten Paketen und über den Zeitpunkt des letzten Updates gibt. Informationen über die allgemeinen Zustände, also die Ampelfarben für jeden Anwendungsfall, werden in der Tabelle mit dem Namen `ServerSecurityStatusInformation` gespeichert.

Das Administrationsinterface des Demonstrators ermöglicht es, Benutzerkonten bestimmte Server individuell und über ihre Gruppenzugehörigkeit zuzuweisen. Ähnlich wie bei den zuvor vorgestellten Benutzerrechten können Server einem Anwender auch über die Rollenmitgliedschaft zugewiesen werden. Um dies zu ermöglichen, kommen die drei Assoziationstabellen `UserServer`, `GroupServer` und `RoleServer` zum Einsatz. Aus Gründen der Übersichtlichkeit sind die mit den Assoziationstabellen verknüpften Relationen `User`, `Group` und `Role` und die damit zusammenhängenden Tabellen aus der Benutzer- und Rechteverwaltung in Abbildung 7.14 nicht mehr eingezeichnet.

Anwender des Prototyps können Server, die ihnen zugewiesen wurden, nach Belieben mit Zusatzinformationen in Form frei definierbarer Tags versehen, um zum Beispiel in den Visualisierungen gezielt nach Servern mit bestimmten Tags suchen zu können. Die benutzerspezifischen Tags werden in der Tabelle `Tag` gespeichert. Zur Verknüpfung von Tags und Servern dient die Assoziationstabelle UserServerTag. Auch hier wurde aus Gründen der Übersichtlichkeit auf das Einbinden der referenzierten Relation `User` in die Abbildung verzichtet.

Modell der Paketversionen und Referenzlisten

Obwohl die Visualisierungen des Prototyps zu Demonstrationszwecken dienen und deshalb auf aggregierten Daten eines Bestimmten Zeitpunkts basieren, wurde bereits Vorarbeit geleistet, um die Überführung der Anwendung in den Wirkbetrieb zu erleichtern. Das Datenmodell enthält aus diesem Grund bereits Tabellen zur Aufnahme von Informationen über Paketversionen und Referenzlisten von echten Servern.

Eine Referenzliste umfasst Informationen über alle in einem Software Repository verfügbaren Pakete, die zu einem bestimmten Zeitpunkt aktuell waren. Sie dienen dem Soll-Ist-Vergleich mit den tatsächlich auf einem Server installierten Paketversionen. Die einzelnen Paketversionen, die sich in einer Referenzliste befinden können, werden in der Tabelle `PackageVersion` gespeichert, für die Paketlisten selbst ist die Tabelle `PackageListTargetData` vorgesehen. Die Zuweisung von Paketversionen zu Referenzlisten erfolgt über die Assoziationstabelle `PackageListTargetDataHasVersion`.

Der Datensatz einer Paketversion setzt sich aus folgenden Bestandteilen zusammen:

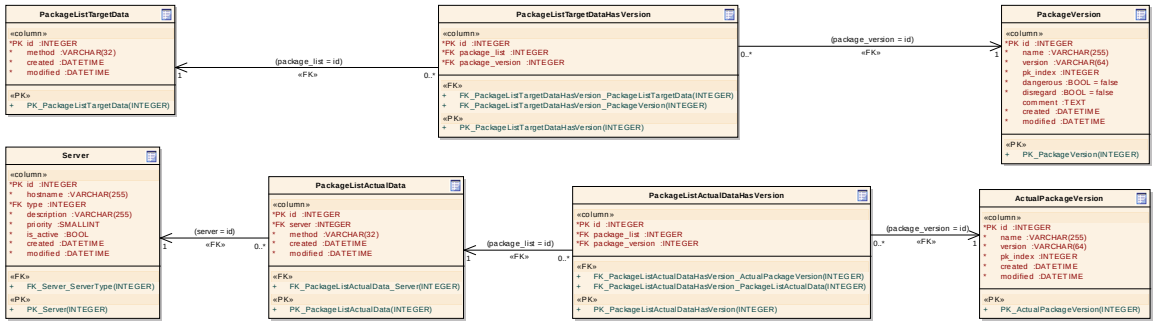


Abbildung 7.15.: Datenmodell der Paketversionen und Referenzlisten.

id Vom objektrelationalen Mapper vorgegebener künstlicher Primärschlüssel.

name Name des Pakets.

version Versions-Zeichenkette mit nicht näher spezifizierter Struktur.

pk_index Interner Zähler zur Sortierung der Paketversionen.

dangerous Kennzeichnet die Existenz sicherheitskritischer Lücken in einer Paketversion.

disregard Kennzeichnet, ob eine Paketversion bei Vergleichsoperationen ignoriert werden soll.

comment Optionaler Beschreibungstext.

created Erstellung des Datensatzes.

modified Letzte Änderung des Datensatzes.

In der unteren Bildhälfte von Abbildung 7.15 sind die Tabellen zu sehen, die einem Server konkrete Paketversionen zuweisen. Der Vollständigkeit halber ist die `Server`-Relation dort ebenfalls, allerdings ohne weitere Abhängigkeiten, eingezeichnet. Die Tabelle `PackageListActualData` dient ähnlich wie die zuvor erwähnte Tabelle `PackageListTargetData` zum Unterscheiden von Paketlisten. Allerdings handelt es sich hier nicht um Referenzlisten, sondern um Listen von Paketen, die tatsächlich auf einem Server zu einem bestimmten Zeitpunkt installiert waren. Der Bezug zum Server wird über den gleichnamigen Fremdschlüssel hergestellt.

Die Paketversionen, die in den Serverpaketlisten referenziert werden können, sind in der Tabelle mit dem Namen `ActualPackageVersion` gespeichert. Die Verbindung von Paketversionen zu einer Serverpaketliste erfolgt durch die Assoziationstabelle `PackageListActualDataHasVersion`.

Beim Betrachten des Modells der Paketversionen und Referenzlisten fällt auf, dass die Tabelle mit der Bezeichnung `ActualPackageVersion` scheinbar redundant ist. Ihre Attribute sind bereits in der Tabelle `PackageVersion` vollständig enthalten. Es liegt zunächst nahe, auf die redundante Tabelle zu verzichten und stattdessen Assoziationen zwischen den Serverpaketlisten und den Paketversionen aus der Tabelle mit dem Namen `PackageVersion` direkt herzustellen.

Obwohl die Tabelle mit den auf den Servern installierten Paketversionen zusätzlichen Implementationsaufwand nach sich zieht, macht sie dennoch Sinn: Es kann durchaus möglich sein, dass bestimmte Paketversionen auf einem Server ermittelt werden, die in keiner der bekannten Referenzlisten enthalten sind. Dies ist zum Beispiel der Fall bei Software, die aus einem zusätzlich eingebundenen fremden Repository heruntergeladen und installiert wurde. Eine andere Möglichkeit besteht darin, dass die Aktualität der Referenzliste den Servern hinterherhinkt. In beiden Fällen ist ein vollständiger Überblick der installierten Paketversionen dennoch wichtig. Ohne die redundante Tabelle wären die Angaben über die auf einem Server installierten Pakete unvollständig. Potenziell sicherheitskritische Informationen wären dann nicht in der Datenbank vorhanden. Durch die zusätzliche Tabelle ist eine lückenlose Erfassung der Istzustände möglich.

Gesamtes Modell

Dieser Abschnitt verschafft noch einmal einen Überblick über alle Relationen und ihre Beziehungen untereinander, die in der prototypischen Anwendung anzutreffen sind. Diese Zusammenfassung dient als Schnellübersicht, da dort Informationen aus den vorherigen detaillierteren Abschnitten wiederholt werden. Eine Grafik des vollständigen Modells ist in Abbildung 7.16 dargestellt.

Die Relationen der Benutzer- und Rechteverwaltung sind darin im Bereich links unten enthalten. Die Relation `FailedLogin` erfasst fehlgeschlagene Login-Versuche, um bei einer Überschreitung eines Schwellwerts das Benutzerkonto zu sperren. In `UserSettings` werden benutzerspezifische Programmeinstellungen wie zum Beispiel die Akzentfarbe gespeichert. Benutzerkonten, die über eine LDAP-Authentifizierung verfügen, besitzen einen Eintrag in der Tabelle `UserLdap`.

Neben den Benutzern stellen die Linux-Server des LRZ einen weiteren Schwerpunkt im Datenmodell dar. Die dazugehörige Relation `Server` speichert den Namen eines Servers, einen Fremdschlüssel zu einer Kategorie, in der sich der Server einordnen lässt (siehe Tabelle `ServerType`), einen Beschreibungstext, einen numerischen Wert, der die Priorität eines Servers im Vergleich zu anderen ausdrückt und die Information darüber, ob der Server noch in Betrieb ist oder ausgemustert wurde.

Als Grundlage für dynamisch generierte Visualisierungen greift der Demonstrator der Einfachheit halber auf aggregierte Werte zurück, die sich in der Datenbank befinden. Im Auslieferungszustand handelt es sich dabei um Beispielwerte und Zufallszahlen. Dazu zählen die Werte aus den Tabellen `ServerInformation` und `ServerSecurityStatusInformation`. Die erste Relation beinhaltet die Anzahl der aktuell auf einem Server installierten Softwarepakete und den Zeitpunkt des letzten Software Updates. Die zweite Relation bestimmt, welche Signalfarbe anhand des zugrundeliegenden Patch-Status im Demonstrator angezeigt wird.

Falls der Prototyp in Zukunft zu einem Produktivsystem weiterentwickelt werden sollte, können die Relationen, die die Beispielwerte enthalten, beibehalten werden. Die darin enthaltenen Werte müssten lediglich von einem externen Mechanismus bereitgestellt werden, der echte Daten von den zu überwachenden Servern liefert.

Auch aus Sicht der Performance und Skalierbarkeit machen die aggregierten Werte in den Tabellen Sinn: Der Aufruf einer Visualisierung durch einen Anwender bewirkt dann lediglich, dass der Server ein paar Leseoperationen in der Datenbank durchführen muss. Echtzeitdaten für eine Visualisierung von hunderten Servern gleichzeitig abzufragen würde zu spürbaren Verzögerungen führen und unnötige Last im Netz und auf den Servern nach sich ziehen. In welchen Abständen die Datenbank mit aktuellen Werten gefüllt wird und in welcher Reihenfolge die Daten von den Servern abgerufen werden sollen ist dann eine Frage der Optimierung an die Bedürfnisse des LRZ.

Alle Server, die in der gleichnamigen Relation gespeichert sind, sind in der Administrationsoberfläche des Prototyps einsehbar und editierbar. Da nicht jeder Anwender mit eingeschränkten Rechten alle Server sehen soll oder darf, ist eine Einschränkung der Sichtbarkeit der Systeme möglich. Die Tabellen `GroupServer` und `RoleServer` assoziieren ausgewählte Server mit Benutzergruppen oder Rollen. Ein Benutzer, der Mitglied einer oder mehrerer Gruppen ist, kann im Prototyp auf Informationen aller Server zurückgreifen, die dieser oder diesen Gruppen vom Administrator zugewiesen wurden. Eine individuelle Zuordnung von einzelnen Servern zu Benutzern geschieht über die Assoziationstabelle `UserServer`.

Anwender können die ihnen zugewiesenen Server durch selbst definierbare Tags nach eigenen Bedürfnissen mit Zusatzinformationen versehen. Dazu dienen die beiden Relationen `Tag` und `UserServerTag`.

Wie bereits zuvor erwähnt wurde, basieren die Visualisierungen des Prototyps auf Beispielwerten. Sie dienen als Grundlage, um die prinzipielle Funktion der Visualisierungen zu demonstrieren. Das Datenmodell verfügt aber bereits über Ansätze für einen Ausbau der Anwendung im Bereich des Anwendungsfalls der Software-Patchstände: Das Modell berücksichtigt bereits den Import von Referenzpaketlisten, die Informationen über die jeweils aktuellsten Paketversionen im Repository der Server bereitstellen. Zu diesem Zweck dienen die drei Relationen in der rechten oberen Ecke von Abbildung 7.16. Die übrigen drei Relationen in der rechten unteren Ecke stellen eine Beziehung zwischen Servern und den tatsächlich darauf installierten Paketen her.

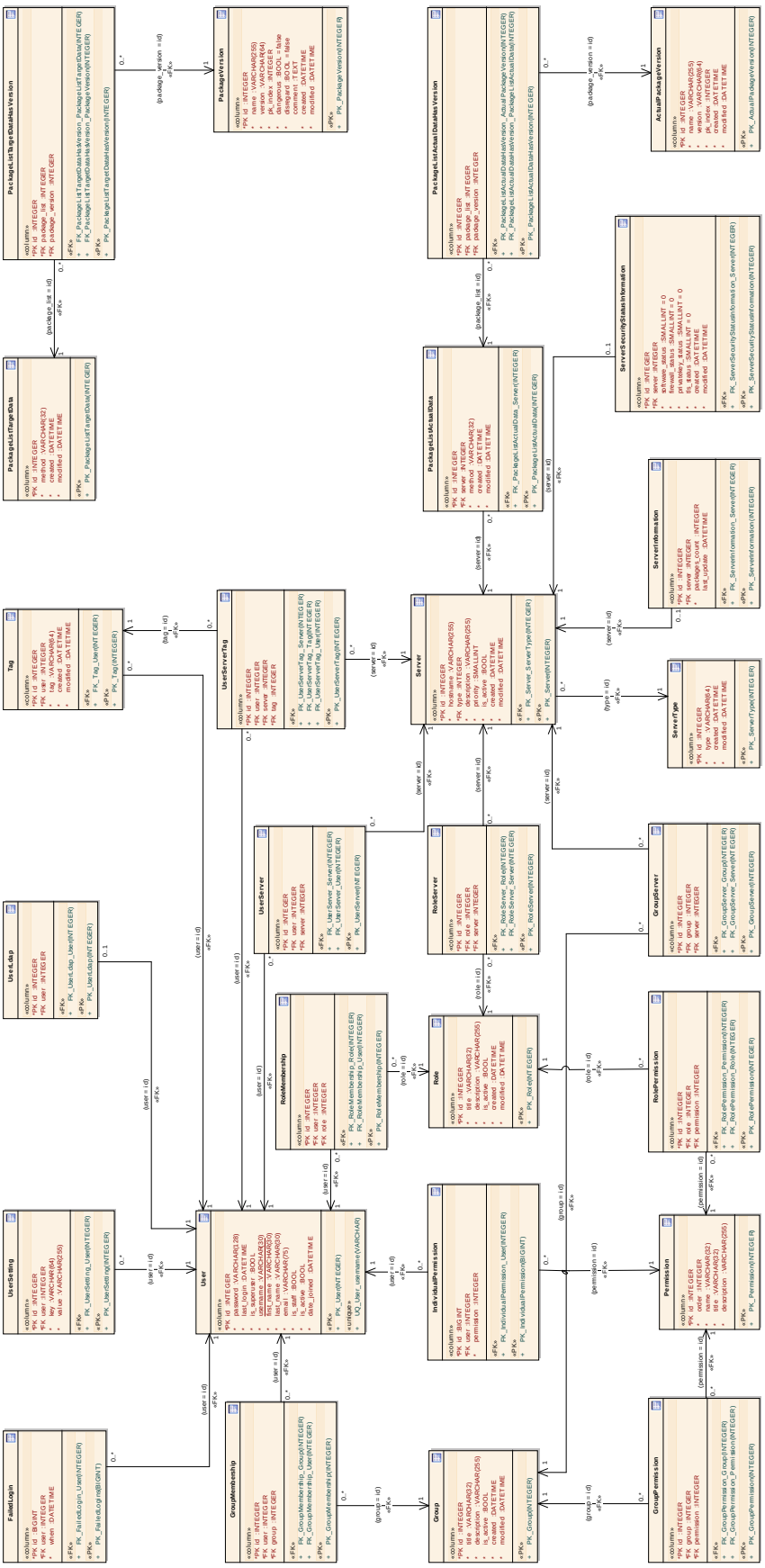


Abbildung 7.16.: Komplettes Datenmodell des Demonstrators.

Mit Hilfe dieser Informationen kann in einer Weiterentwicklung des Prototyps ein tatsächlicher Versionsvergleich durchgeführt werden, um die Sicherheit des Patchzustands eines Servers bewerten zu können. Außerdem ist damit kontinuierlich nachvollziehbar, bis zu welchem Zeitpunkt ein Paket mit einer bestimmten Version auf einem Server installiert war, bis es upgedatet oder deinstalliert wurde. Diese Informationen unter anderem für Auditierungszwecke wertvoll.

7.3. Bedienoberfläche des Prototyps

Dieser Abschnitt widmet sich der Bedienoberfläche der prototypischen Anwendung. Zunächst wird auf die Design-Prinzipien eingegangen, die beim Entwurf der UI berücksichtigt wurden. Danach werden die funktionalen Einheiten der Bedienoberfläche vorgestellt.

7.3.1. Designprinzipien

Beim Design der Bedienoberfläche des Prototyps wurde Wert auf Designrichtlinien gelegt, die bereits in den Kapiteln 2 erwähnt wurden, insbesondere in Abschnitt 2.13.3. Ebenso wurde darauf Wert gelegt, keine Fehler zu wiederholen, die in den bereits existierenden Lösungen aus Kapitel 5 vorkommen.

Aus dem zuletzt genannten Grund setzt der Prototyp Farben nur sehr sparsam ein. Die Elemente der grafischen Bedienoberfläche sind hauptsächlich in einer Handvoll Graustufen eingefärbt, um sie gut voneinander unterscheidbar zu machen. Damit folgt der Prototyp Ben Shneidermans Rat, die Bedienoberfläche in einem ersten Schritt zunächst ohne Farben zu entwerfen. Gleichzeitig wurde dabei darauf geachtet, dass Texte durch ihren Kontrast auch auf dunklem Hintergrund noch gut lesbar sind.

Farben wurden bewusst für die Visualisierungen im Prototyp reserviert. Lediglich ein grüner Farbton wird in der Oberfläche als farbige Designkomponente verwendet, um sie optisch ansprechender zu gestalten und dem Anwender einen vertrauenserweckenden Eindruck zu vermitteln. Um Ben Shneidermans Designrichtlinien für Bedienoberflächen gerecht zu werden, ist diese Akzentfarbe benutzerspezifisch änderbar.

Um der Bedienoberfläche ein konsistentes Aussehen zu verleihen, wurde für die eingebundene JavaScript-Bibliothek jQuery UI eine benutzerdefinierte Stylesheet-Datei erstellt. Damit fügen sich Bedienelemente wie Schaltflächen und Dialogfelder besser in die Anwendung ein. Dasselbe gilt für das Tablesorter-Plug-in, das Tabelleninhalte auf der Client-Seite nach den Anforderungen des Anwenders sortiert.

7.3.2. Menüführung

Die einzelnen Funktionen des Prototyps sind über ein per JavaScript generiertes Navigationsmenü aufrufbar, das auf Mouse-Over-Ereignisse reagiert und die gewünschten Untermenüs automatisch aufklappt. Abbildung 7.17 zeigt einen Bildschirmausschnitt mit dem Menü. Das Hauptmenü umfasst auf oberster Ebene die folgenden Menüpunkte:

Startseite Wird unmittelbar nach dem Login bei der Anwendung angezeigt, sofern keine spezifischere URL angegeben wurde.

Anwendungsfälle Beinhaltet Anwendungsfälle aus dem Bereich der Linux-Server-Sicherheit.

Administration Administrationsbereich für privilegierte Benutzer.

Hilfe Vorgesehener Platz für Hilfe und Programminformationen.

Persönliches Menü Beinhaltet benutzerspezifische Einstellungen, Tags und Server sowie den Abmelde-Button.

Wie auch der Rest der Oberfläche nutzt das Navigationsmenü lediglich die global einheitliche Akzentfarbe zum Hervorheben von Informationen. Der Menüpunkt, der sich aktuell unter dem Mauszeiger befindet, wird damit eingefärbt, um dem Benutzer ein optisches Feedback zu geben. Darüber hinaus befindet sich über dem aktuell

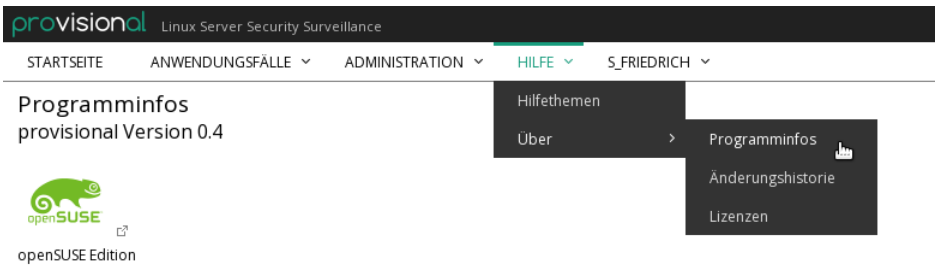


Abbildung 7.17.: Hauptmenü zum Zugriff auf die Programmfunktionen des Prototyps.

gewählten Menüpunkt ein schmaler farbiger Balken, der ebenfalls der Hervorhebung dient. Nachdem in einem der Menüs eine Programmfunktion aufgerufen wurde, bleibt der farbige Balken über dem dazugehörigen Hauptmenüpunkt eingeblendet. Dadurch kann der Anwender auf einen Blick einordnen, in welchem Bereich der Anwendung er sich gerade befindet.

Die zur Darstellung des Menüs verwendete JavaScript-Bibliothek unterstützt auch kleinere Bildschirme. Sobald das Programmfenster nicht mehr groß genug ist, um alle Menüpunkte ohne Scrollen anzeigen zu können, schaltet die Bibliothek das Menü in einen platzsparenderen Modus um. Alle Menüpunkte sind zunächst eingeklappt und werden auf Anforderung durch den Anwender untereinander angezeigt. Dieses Bedienkonzept spart Platz und dürfte vor allem Besitzern von Smartphones bekannt vorkommen.

7.3.3. Administration

Dieser Abschnitt befasst sich mit den administrativen Funktionen des Prototyps. Die einzelnen Punkte im Menü *Administration* erfüllen folgende Zwecke:

Server Verwaltung der Linux-Server, die als Datenquelle für Visualisierungen dienen.

Gruppen Verwaltung von Benutzergruppen.

Rollen Verwaltung von Benutzerrollen als Alternative zu Gruppen.

Benutzer & Rechte Verwaltung von Benutzerkonten und Zuweisung von Zugriffsrechten.

Server Übersicht der vom Demonstrator verwalteten Server.

Sitzungen Übersicht der aktuell eingeloggten Anwender.

Protokoll Systemprotokoll des Prototyps.

In den folgenden Abschnitten wird auf die einzelnen Funktionen des Administrationsbereichs detailliert eingegangen.

Serververwaltung

Der Anwendungszweck des Prototyps ist es, sicherheitsrelevante Zustandsinformationen von Linux-Servern zu visualisieren, um Methoden aufzuzeigen, die für die Administration gewinnbringend eingesetzt werden können. Auch wenn es sich beim Demonstrator nicht um ein Produktivsystem handelt, setzt dies dennoch voraus, dass er auf eine Datenbank zurückgreifen kann, die Informationen über Server beinhaltet. Zu diesem Zweck dient der Menüpunkt *Server* im Administrationsbereich.

Ursprünglich war es angedacht, den Prototyp mit Zustandsinformationen realer Server des LRZ zu befüllen, um als Grundlage für Visualisierungen zu dienen. Da der Fokus dieser Arbeit auf der gewinnbringenden Visualisierung dieser Informationen liegt, wurde die Datenbank in einem späteren Entwicklungsstadium mit frei erfundenen Daten befüllt. Es wurden manuell 100 Server eingetragen, die eine typische Serverlandschaft eines wissenschaftlichen Instituts widerspiegeln. Bei den Hostnamen wurde der Einfachheit halber auf die Namen von Inseln zurückgegriffen.

Obwohl die Serverinformationen frei erfunden sind, wirkt sich diese Tatsache nicht auf das Datenmodell der Anwendung aus. Die zu visualisierenden Informationen würden in einer weiterentwickelten Version für den Produktiveinsatz lediglich durch reale Werte ersetzt werden, die automatisch in die Datenbank geschrieben werden. Eine Änderung am Datenbankschema ist nur erforderlich, falls zusätzliche aggregierte Informationen über die Serverzustände gespeichert werden sollen.

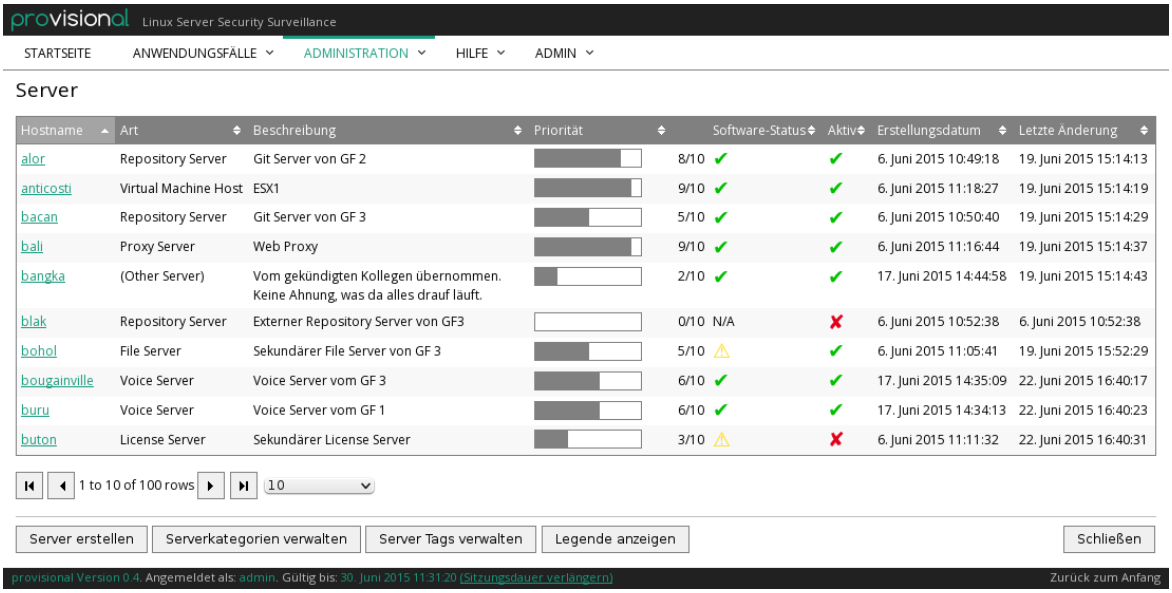


Abbildung 7.18.: Serververwaltung des Prototyps.

In Abbildung 7.18 ist ein Bildschirmfoto der Serverübersicht zu sehen. Das Programm präsentiert die Server in einer Tabelle. Bei der Tabelle handelt es sich nicht um eine statische HTML-Tabelle. Um den Umgang mit großen und unübersichtlichen Datenmengen für den Anwender zu vereinfachen, wurden die Tabellen des Prototyps mit Hilfe einer JavaScript-Bibliothek um nützliche Interaktionsmöglichkeiten erweitert. Da die Server-Tabelle die erste ist, die in diesem Abschnitt vorgestellt wird, werden bei dieser Gelegenheit die Vorzüge der dynamisch generierten Tabellen vorgestellt.

Die meisten Tabellen nutzen die volle Fensterbreite aus, um Zeilenumbrüche innerhalb der einzelnen Zellen zu minimieren. Dadurch bleibt die Höhe der angezeigten Zeilen konstant und der optische Gesamteindruck der Bedienoberfläche wirkt einheitlicher. Die Breite der Spalten passt sich automatisch an die Inhalte an. Sollte der Anwender dennoch den Wunsch verspüren, Einfluss auf die einzelnen Spaltenbreiten zu nehmen, kann er die Breite durch Klickziehen mit der Maus an seine eigenen Bedürfnisse anpassen.

Der Anwender hat die Möglichkeit, Tabellen nach seinen eigenen Anforderungen zu sortieren. Die Sortierung erfolgt spaltenspezifisch und wird per Mausklick auf einen Spaltentitel durchgeführt. Ein zusätzlicher Mausklick kehrt die Sortierreihenfolge um.

Kleine Piktogramme in der Kopfzeile der Tabelle erleichtern die Arbeit mit der eingebauten Sortierfunktion. Das Doppelpfeilpiktogramm, das in Abbildung 7.18 neben allen Spaltenbeschriftungen außer der ersten zu sehen ist, teilt dem Anwender mit, dass die Tabelle durch einen Mausklick auf den Titel nach den Inhalten dieser Spalte sortierbar ist, aber noch nicht sortiert wurde. Ein einfacher Pfeil, der nach oben oder nach unten zeigt, deutet an, dass die Spalte bereits nach diesem Attribut entweder in absteigender oder in aufsteigender Richtung sortiert wurde.

Nicht immer macht es Sinn, Tabellen anhand eines bestimmten Attributs zu sortieren. Das ist zum Beispiel der Fall, wenn jede Zeile den gleichen Inhalt wie einen Link mit identischer Beschriftung enthält. Bei solchen Attributen wurde die Sortierfunktion gezielt deaktiviert, da sie keinen Zweck erfüllt. Nicht sortierbare Spalten beinhalten keines der zuvor genannten Piktogramme neben dem Titel. Dies ist auch ein geringfügiger Beitrag, um unnötigen Bildschirminhalt zu reduzieren und somit die Übersichtlichkeit zu erhöhen.

Um die Übersichtlichkeit weiter zu erhöhen, werden Attribute in der Kopfzeile, nach denen eine Sortierung

stattfindet, durch eine etwas hellere Hintergrundfarbe hervorgehoben. Dadurch sieht der Anwender schneller, nach welchen Attributen die Inhalte der Tabelle sortiert wurden.

Die Sortierfunktion von Tabellen ist nicht auf ein einziges Attribut beschränkt. Wenn der Anwender die Inhalte zunächst nach einem bestimmten Attribut und danach zusätzlich nach einem weiteren Attribut sortieren möchte, geschieht dies durch Gedrückthalten der `Shift`-Taste während dem Klicken auf das zusätzliche Attribut. Auf der Serverübersichtsseite ist diese Funktion nützlich, um zum Beispiel die Server zunächst nach ihrer Priorität und dann nach dem Software-Status zu sortieren. Dies erleichtert die Suche nach wichtigen Servern mit kritischem Patch-Zustand enorm.

Unter der Tabelle befindet sich ein Pager zum Durchblättern der segmentierten Datenseiten. Über die Schaltflächen des Pagers kann der Anwender wahlweise Seite für Seite durch die Inhalte der Tabelle blättern oder direkt zur ersten bzw. zur letzten Bildschirmseite wechseln. Auf den Schaltflächen, die zu diesem Zweck dienen, sind keine Texte, sondern Piktogramme mit hohem Wiedererkennungswert untergebracht. Zusätzlich verfügen diese Schaltflächen über Tooltip-Texte, die ihre Funktion beschreiben.

Die Anzahl der angezeigten Zeilen in der Tabelle kann der Benutzer frei wählen. Dies geschieht über ein Dropdownfeld rechts neben den Schaltflächen des Pagers. Dieses Dropdownfeld schlägt eine Auswahl vordefinierter Zeilenanzahlen vor. Zusätzlich dazu beinhaltet das Feld einen Wert, den der Benutzer in den persönlichen Einstellungen selbst festlegen kann. Dieser Wert wird standardmäßig auch bei anderen Tabellen innerhalb der Anwendung vorausgewählt. Diese komfortable Art der Arbeit mit Tabellen wird im kompletten Demonstrator konsequent unterstützt.

Zurück zu den Servern: Jeder Server, den der Prototyp verwaltet, ist einer Kategorie wie zum Beispiel *Web Server* oder *File Server* zugeordnet. Obwohl der Prototyp im Auslieferungszustand ca. 30 unterschiedliche Serverkategorien kennt, ist der Administrator nicht auf diese Menge beschränkt, da sie nicht hart im Programmcode kodiert ist, sondern in der Datenbank liegt.

Die Kategorien können vom Administrator individuell an den Einsatzort der Anwendung angepasst werden. Über die Schaltfläche mit der Bezeichnung *Serverkategorien verwalten* kann sich der Administrator die verfügbaren Kategorien auflisten lassen und Änderungen daran durchführen. In Abbildung 7.19 ist die Webseite mit den Serverkategorien zu sehen.

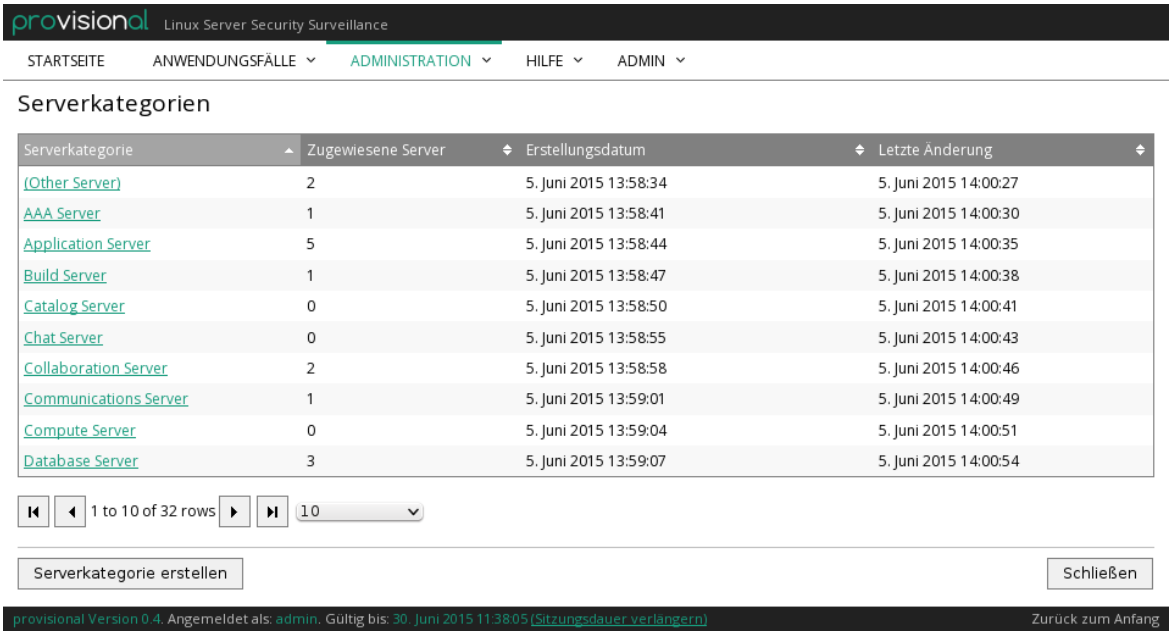


Abbildung 7.19.: Auflistung der vom Administrator festgelegten Serverkategorien.

Die einzelnen Serverkategorien werden in einer Tabelle aufgelistet. In der ersten Spalte befindet sich der Name der Kategorie. Die Zahlen in der zweiten Spalte geben Auskunft darüber, wie viele Server bereits mit

der entsprechenden Kategorie assoziiert wurden. Hier ist die zuvor vorgestellte Sortierfunktion nützlich, da der Administrator auf einen Klick sehen kann, bei welchen Kategorien es sich um die am häufigsten verwendeten handelt oder welche Kategorien nicht verwendet werden und Kandidaten zum Löschen sind. Die restlichen beiden Spalten geben Auskunft darüber, wann eine Kategorie angelegt bzw. wieder umbenannt wurde.

Das Anlegen eigener Serverkategorien geschieht durch einen Mausklick auf die Schaltfläche mit der Beschriftung *Serverkategorie erstellen*. Eine bestehende Kategorie wird durch einen Klick auf ihren Namen zur Bearbeitung geöffnet. In beiden Fällen erscheint ein Eingabeformular, das lediglich den Namen der Kategorie abfragt. Vor dem Speichern achtet der Prototyp darauf, dass der eingegebene Name nicht schon vergeben wurde, denn Duplikate sind nicht erlaubt.

Neben einem frei definierbaren Beschreibungstext besitzt jeder der Server eine Priorität, die vom Administrator festgelegt wird. Dabei handelt es sich der Einfachheit halber um einen ganzzahligen Wert zwischen 0 und 10, wobei 10 die höchste Priorität widerspiegelt. Bei diesem Attribut erweist sich die Sortierfunktion der Tabelle als praktisch, da der Anwender die verfügbaren Server per Mausklick nach ab- und aufsteigender Priorität sortieren lassen kann. Besonders wichtige (oder unwichtige) Systeme sind so auf einen Blick sichtbar.

Da die Methoden der Informationsvisualisierung im Demonstrator sich primär mit dem Anwendungsfall *Software-Verwaltung* (siehe Abschnitt 6.1) auseinandersetzen, gibt es ein weiteres Serverattribut, das den Patch-Zustand kategorisiert. Die zur Verfügung stehenden Kategorien entsprechen denen aus dem zuvor genannten Abschnitt. Das Attribut *Aktiv* in der Tabelle gibt Auskunft darüber, ob ein Server noch für den produktiven Einsatz verwendet wird oder ausgemustert wurde.

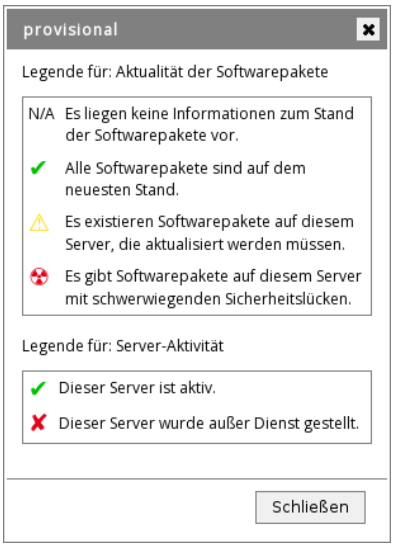


Abbildung 7.20.: Optional einblendbare Legende zu den Piktogrammen der Serververwaltung.

Um auf sich wiederholende Texte zu verzichten, wird in den beiden zuletzt genannten Spalten auf farbige Piktogramme zur Kennzeichnung des jeweiligen Zustands zurückgegriffen. Die Bedeutung der Piktogramme kann sich der Anwender bei Bedarf durch einen Klick auf die Schaltfläche mit der Bezeichnung *Legende anzeigen* einblenden lassen. Abbildung 7.20 zeigt einen Screenshot dieser Legende. Es handelt sich um ein Fenster, das der Anwender beliebig auf dem Bildschirm verschieben und wieder schließen kann.

Nach einem Klick auf *Server* erstellen kann der Administrator neue Server ins System aufnehmen, unabhängig davon, wie die Zustandsinformationen in die Datenbank gespielt werden. Zu Demonstrationszwecken werden dort ebenfalls erfundene Werte eingetragen. Die Eigenschaften bereits bestehender Server können nach einem Klick auf den jeweiligen Hostnamen bearbeitet werden. Dieses Bedienprinzip zieht sich durch den kompletten Demonstrator. In Abbildung 7.21 wurde ein Server zur Bearbeitung geöffnet.

Die Formulare zur Bearbeitung bestehender und zum Anlegen neuer Server setzen sich aus den folgenden Bestandteilen zusammen: Ein Textfeld ist für den Hostnamen des Servers vorgesehen. In dem Dropdownfeld mit der Bezeichnung *Art* kann der Administrator dem Server eine der zuvor vorgestellten Kategorien zuweisen.

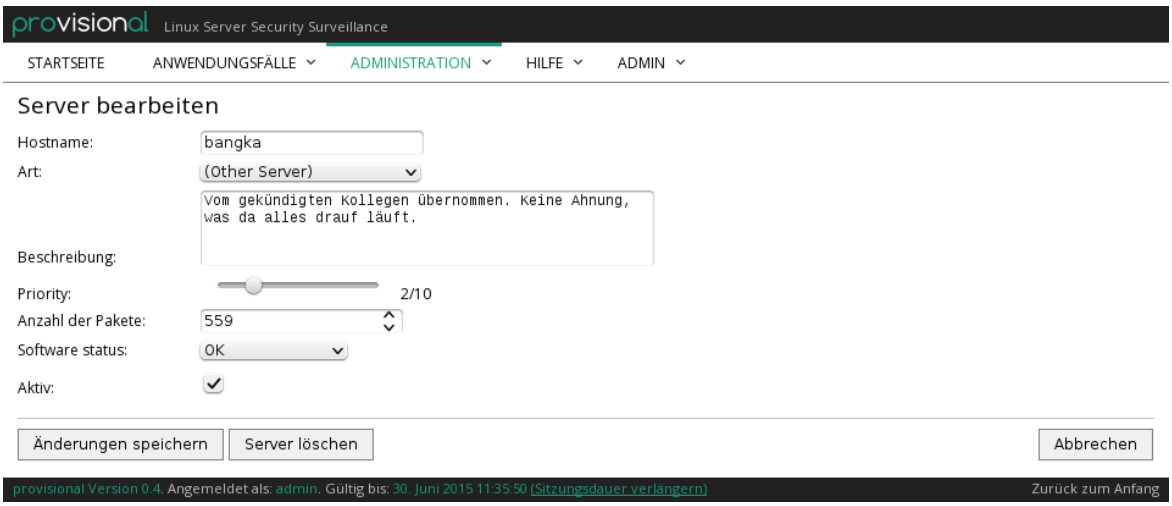


Abbildung 7.21.: Bearbeitung der Eigenschaften eines Servers.

Das große Textfeld darunter ist für einen optionalen Beschreibungstext vorgesehen.

Die Priorität des Servers stellt der Administrator mit Hilfe eines auf HTML5 basierenden Sliders ein. Das ist eine intuitivere und schnellere Art und Weise, als eine Zahl in ein Textfeld einzugeben. Zudem ist die syntaktische Korrektheit der Eingabe gewährleistet. Da es sich um eine Ganzzahl zwischen 0 und 10 handelt, liegen die einzelnen Ticks des Sliders ausreichend weit auseinander, um mit dem Mauszeiger nicht absolut exakt zielen zu müssen.

Bei der Anzahl der Pakete handelt es sich um einen aggregierten Wert. Da der Demonstrator aus den bereits genannten Gründen keine tatsächlichen Daten von Servern visualisiert, wurde in dem Eingabeformular ein Feld geschaffen, um die Anzahl installierter Pakete in einem Rutsch dort zu hinterlegen. Dadurch gestaltet sich die Eingabe von Beispieldaten einfacher. In einem Produktivsystem, das echte Serverdaten visualisiert, ist dieses Eingabefeld überflüssig. Dasselbe gilt für das Dropdownfeld mit der Bezeichnung *Sicherheitsstatus*. In einem Produktivsystem würde es sich hierbei ebenfalls um einen aggregierten Wert handeln, der aus der Bewertung der Aktualität der installierten Softwarepakete eines Servers resultiert.

Das letzte Eingabefeld in dem Formular ist eine Checkbox mit der Bezeichnung *Aktiv*. Nicht mehr benötigte Server können so entsprechend gekennzeichnet werden. Inaktive Server werden in den Visualisierungen, die der Prototyp generiert, nicht berücksichtigt.

Da es sich bei diesem Formular bisher um das komplexeste handelt, wird bei der Gelegenheit noch auf generelle Eigenschaften von Eingabeformularen in der gesamten Anwendung eingegangen: Durch einen Klick auf die Schaltfläche *Änderungen speichern* übernimmt das System die eingegebenen Serverinformationen in die Datenbank. Vor dem Schreiben prüft der Server die Benutzereingaben auf syntaktische Korrektheit. manche Eingabefelder wie der Hostname sind obligatorisch und dürfen nicht leer sein. Andernfalls wird ein entsprechender Hinweis ausgegeben und der Speichervorgang wird abgebrochen. Der Server achtet auch auf die Sinnhaftigkeit der Daten, indem er zum Beispiel Duplikate beim Hostnamen nicht akzeptiert.

Gruppenverwaltung

Um nicht jedem Benutzer der Anwendung individuelle Zugriffsrechte und Server zuweisen zu müssen, unterstützt der Prototyp das aus vielen anderen Anwendungen bekannte Konzept von Benutzergruppen. Sie fassen Zugriffsrechte für Programmfunktionen und Mengen von Servern unter einem aussagekräftigen Namen zusammen. Mitglieder einer Gruppe erhalten automatisch die Rechte und Server zugeteilt, die der Gruppe zugewiesen wurden. Gehört ein Anwender mehreren Gruppen an, summieren sich die Rechte bzw. Server, die ihm zur Verfügung stehen. Neben dem geringeren Konfigurationsaufwand erlauben Gruppen auch schnelle Änderungen an den Rechten großer Anwenderzahlen.

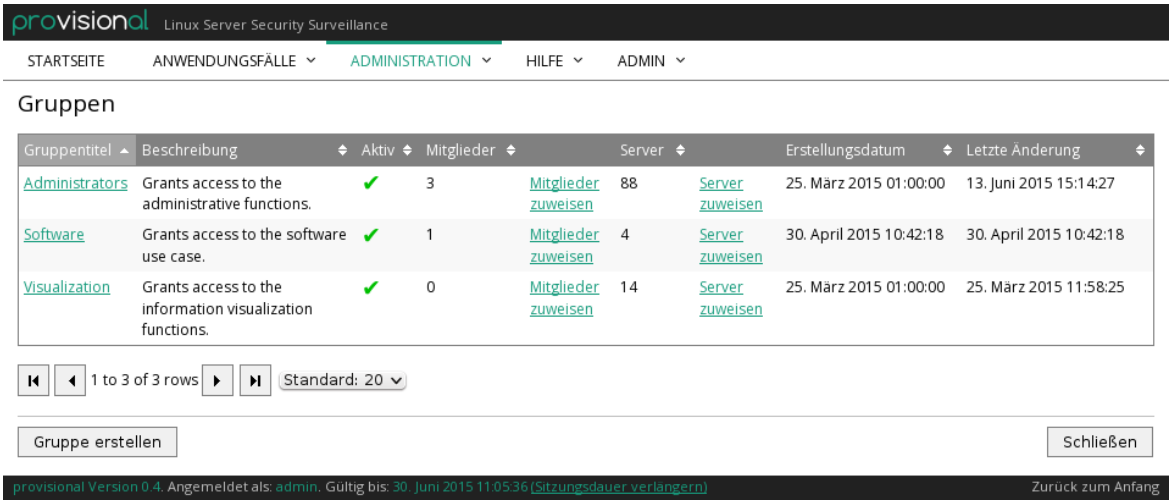


Abbildung 7.22.: Gruppenverwaltung des Prototyps.

Abbildung 7.22 zeigt die Verwaltung der Benutzergruppen. Sie beinhaltet eine Übersichtstabelle mit den wichtigsten Attributen einer Gruppe. Die Tabelle beinhaltet die folgenden Informationen: Die erste Spalte zeigt den Namen der Gruppe an. Die Namen, die dort aufgelistet sind, spielen später bei der Definition von Gruppenmitgliedschaften eine Rolle. Die zweite spalte ist für einen kurzen Beschreibungstext vorgesehen.

Gruppen können ebenso wie Server als inaktiv gekennzeichnet werden. Gruppen, die nicht aktiv sind, vererben keine Rechte und keine Server an ihre Gruppenmitglieder. Durch das Deaktivieren einer Gruppe können einer großen Anzahl von Anwendern besonders schnell Rechte und Server entzogen werden. Über den Zustand der Aktivität gibt ein Piktogramm in der Spalte *Aktiv* Auskunft.

Die Zahl in der Spalte *Mitglieder* gibt Auskunft darüber, wie viele Benutzerkonten in die jeweilige Gruppe aufgenommen wurden. Durch die Sortierfunktion können Administratoren mit nur einem Mausklick feststellen, welche Gruppen die meisten Mitglieder haben.

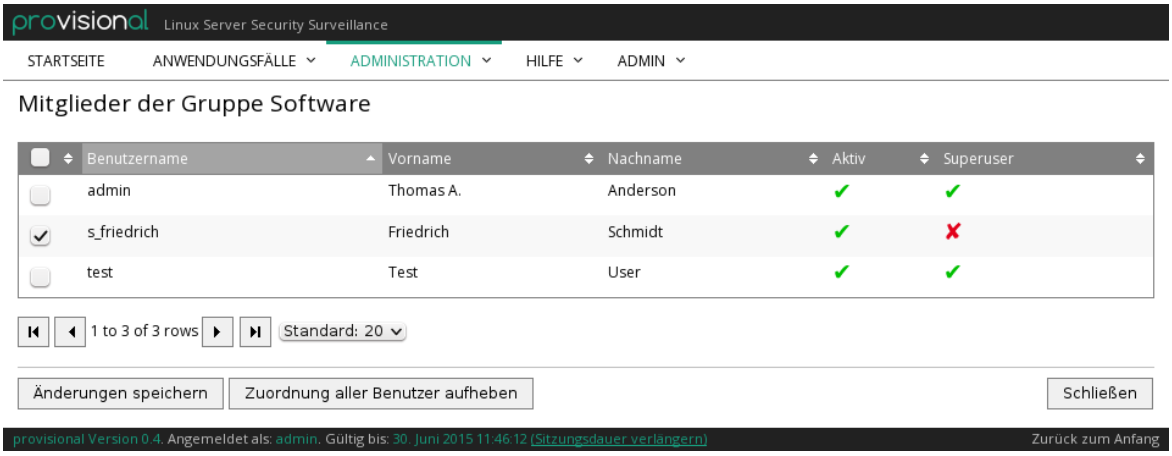


Abbildung 7.23.: Zuweisung von Benutzern zu einer Gruppe.

Es ist auch möglich, die Mitglieder der Gruppe unmittelbar abzufragen oder Änderungen an den Gruppenmitgliedschaften vorzunehmen. Dies geschieht durch einen Klick auf den Link *Mitglieder zuweisen*. Es öffnet sich eine Tabelle aller vorhandener Benutzer, die die Gruppenmitgliedschaft durch den Zustand der Checkbox in der ersten Spalte kennzeichnet (siehe Abbildung 7.23). Auch wenn die Spalte keinen erkennbaren Titel besitzt, ist sie sortierbar. Auf diese Weise können die Gruppenmitglieder mit einem Klick ermittelt werden.

Um den Mitgliedern einer Gruppe ihre Rechte zu entziehen, ist es nicht immer wünschenswert, die Gruppe zu

deaktivieren, da sie möglicherweise weiterverwendet werden soll. Für diesen Fall befindet sich unter der Mitgliedschaftstabelle eine Schaltfläche mit der Bezeichnung *Zuordnung aller Benutzer aufheben*. Dieser Button bewirkt, dass alle Mitgliedschaften der geöffneten Gruppe sofort gelöscht werden. Die Gruppe selbst bleibt weiterhin aktiv.

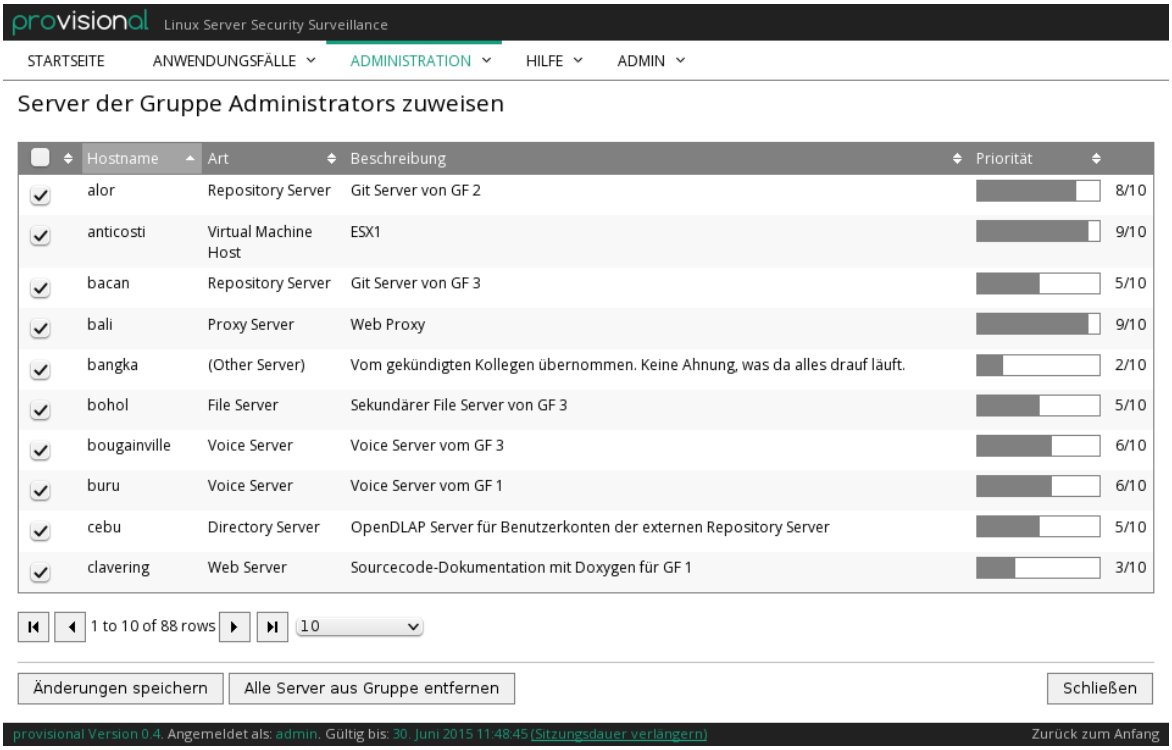


Abbildung 7.24.: Zuweisung von Servern zu einer Gruppe.

Analog zu den Mitgliedern besitzt die Gruppentabelle eine Spalte *Server*, die die Anzahl der einer Gruppe zugeteilten Server enthält. Daneben befindet sich ebenfalls ein Link, mit dem bereits zugewiesene Server abgefragt und Änderungen an der Zuordnung der Server zu dieser Gruppe durchgeführt werden können. Ein Klick auf den Link mit der Bezeichnung *Server zuweisen* führt zu einer Tabelle aller als aktiv gekennzeichneten Server, deren Enthaltensein in der Gruppe durch eine aktivierte Checkbox in der ersten Spalte gekennzeichnet wird (siehe Abbildung 7.24). Analog zur zuvor genannten Mitgliederverwaltung können auch bei dieser Webseite alle Server aus der Gruppe entfernt werden, ohne die Gruppe deaktivieren zu müssen.

Die letzten beiden Spalten in der Gruppentabelle geben Auskunft über den Zeitpunkt ihrer Erstellung und der letzten Änderung des zugrundeliegenden Datensatzes.

Neue Gruppen können in der Gruppenübersicht über den Button *Gruppe erstellen* angelegt werden. Um die Attribute und Rechte einer bestehenden Gruppe zu bearbeiten, genügt ein Klick auf den entsprechenden Link in der Spalte *Gruppentitel*. In Abbildung 7.25 wurde eine Gruppe zur Bearbeitung geöffnet. Die Eingabeformulare sind in beiden Fällen identisch:

Eine Gruppe besitzt in ihrer Minimalkonfiguration lediglich einen Namen und den Zustand *aktiv* oder *inaktiv*. Der Beschreibungstext ist optional, das Eingabeformular sollte aber genutzt werden, um Hinweise zum Zweck der Gruppe zu hinterlegen. Zur Ausstattung der Gruppe mit Zugriffsrechten dienen die Checkboxen in der unteren Hälfte des Eingabeformulars. Eine aktivierte Checkbox kennzeichnet das Enthaltensein des entsprechenden Zugriffsrechts.

Neben dem Button zum Speichern der Änderungen an der Gruppe gibt es weitere Schaltfläche, die die zuvor genannte Mitgliederverwaltung aufruft. Möglicherweise durchgeführte Änderungen an den Formularelementen werden bei einem Klick auf diese Schaltfläche nicht gespeichert. Ein weiterer Button ermöglicht die Zuweisung von Servern zu einer Gruppe. Er führt direkt zu der zuvor genannten Serverzuweisung, ebenfalls ohne

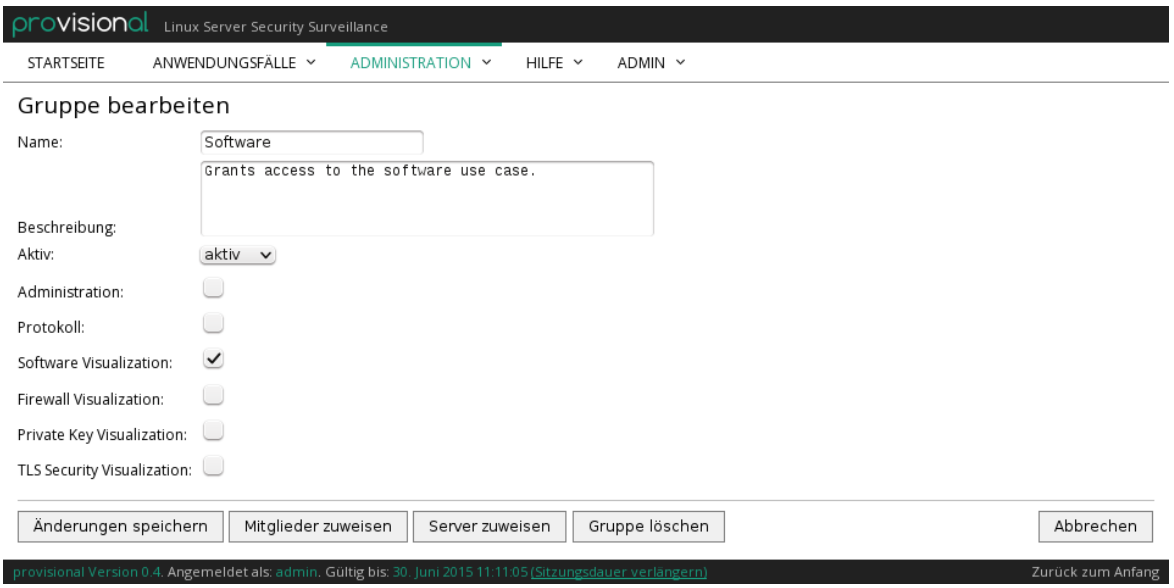


Abbildung 7.25.: Bearbeitung einer Benutzergruppe im Prototyp.

etwaige Änderungen an der Gruppe selbst abzuspeichern.

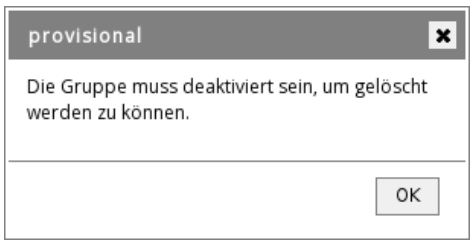


Abbildung 7.26.: Meldung beim Versuch, eine aktivierte Gruppe zu Löschen.

Das Löschen einer Gruppe erfolgt ebenfalls über das Bearbeiten-Formular. Als zusätzliche Sicherheitsfunktion muss eine Gruppe vor dem Löschen zunächst vom Administrator in dem Formular deaktiviert und dieser Zustand gespeichert werden. Dadurch werden Gruppen vor versehentlichem Löschen geschützt. Andernfalls erscheint eine Fehlermeldung in einem modalen Fenster (siehe Abbildung 7.26).

Rollenverwaltung

Alternativ zu den Gruppen bietet der Prototyp Rollen an. Rollen bieten genauso wie Gruppen die Möglichkeit, Zugriffsrechte und eine Auswahl von Servern unter einem aussagekräftigen Namen zusammenzufassen. Im Gegensatz zu Gruppen kann ein Benutzerkonto aber maximal nur einer Rolle zugewiesen werden.

Die Rollenverwaltung inkl. ihrer Bearbeitungsfunktion ist nahezu vergleichbar mit der Gruppenverwaltung, die im vorherigen Abschnitt vorgestellt wurde. Aus diesem Grund wird nicht weiter auf ihre Bedienoberfläche eingegangen.

Benutzer- und Rechteverwaltung

Abbildung 7.27 zeigt die Benutzerverwaltung des Prototyps. Die Tabelle listet alle bekannten Benutzerkonten und einige wichtige Informationen dazu auf. Ein neues Benutzerkonto kann durch den Klick auf die Schaltfläche *Benutzerkonto erstellen* hinzugefügt werden.

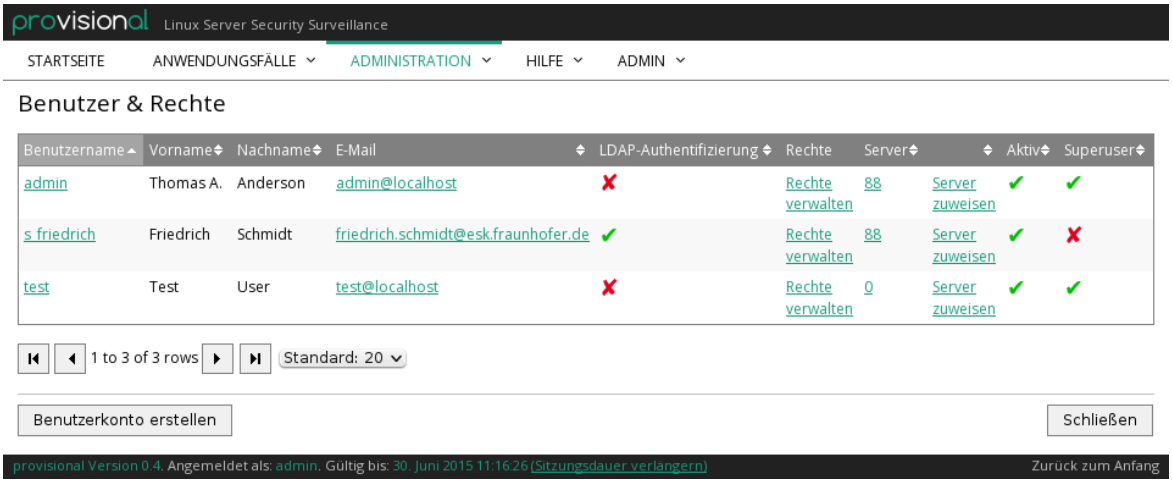


Abbildung 7.27.: Benutzerverwaltung des Prototyps.

Die Tabelle *Benutzer & Rechte* zeigt neben dem Loginnamen den realen Namen und die im System hinterlegte E-Mail-Adresse des Anwenders an. Das Piktogramm in der Spalte *LDAP-Authentifizierung* gibt Auskunft darüber, ob bei der Anmeldung das eingegebene Passwort mit dem in der lokalen Datenbank gespeicherten Passwort-Hash verglichen werden soll oder ob zu diesem Zweck auf ein Active Directory mit zentral gespeicherten Zugangsdaten zurückgegriffen werden soll.

Über den Link *Rechte verwalten* kann der Administrator das Benutzerkonto in Gruppen aufnehmen oder ihm eine Rolle zuweisen. Die Zuweisung individueller Zugriffsrechte ist dort auch möglich. Abbildung 7.28 zeigt ein Beispiel für das entsprechende Eingabeformular.

In der Ansicht *Rechte verwalten* befindet sich eine Tabelle, die sämtliche Rechte zusammenfasst, die dem Benutzer über die folgenden Arten zugewiesen wurden:

- Durch die Mitgliedschaft in einer oder mehrerer Gruppen.
- Durch individuelle Zuweisung konkreter Rechte.
- Alternativ durch die Mitgliedschaft in einer Rolle.

Für jeden dieser genannten Punkte existiert in der Rechteverwaltung ein eigener Tab unterhalb der Rechteauflistungstabelle. Die Änderungen in jedem Tab werden unabhängig voneinander gespeichert. Sobald dort eine Rolle ausgewählt wurde, erhält der Benutzer nur die Rechte dieser Rolle. Alle anderen Rechtezuweisungen spielen dann keine Rolle mehr.

Eine Ausnahme stellen Benutzerkonten mit aktiviertem Attribut *Superuser* dar. Sie unterliegen keinerlei Rechteeinschränkungen. Einzelne Benutzerkonten sollten nur für vorübergehende Testzwecke zu Superusern gemacht werden.

Der in der Spalte *Server* der Benutzertabelle eingeblendete Wert entspricht der Anzahl der Server, die einem Anwender insgesamt zugewiesen wurden. Es handelt sich dabei um die Summe der Server durch Gruppenmitgliedschaften und individuell zugeteilten Servern oder um die Anzahl der Server, die mit der Rolle des Benutzers assoziiert wurden. Ein Klick auf diese Zahl führt zu einer Auflistung aller Server des entsprechenden Benutzers. Diese Übersicht beinhaltet eine Tabelle mit den Hostnamen, den Serverkategorien, dem Beschreibungstext und der Serverpriorität. Es handelt sich dabei um die gleichen Informationen, die in der Serververwaltung des Administrationsbereichs definiert wurden.

Die hinzugekommene Spalte mit dem Titel *Ursprung* ist besonders nützlich: Sie beinhaltet für jeden Server in der Tabelle eine Auskunft über den Grund der Zuweisung des Servers zum jeweiligen Benutzer. Wie bereits zuvor erwähnt wurde, erfolgt die Zuweisung durch Gruppen, Rollen oder individuell. Wenn der Anwender einen Server zum Beispiel über die Gruppe *Administrators* geerbt hat, wird diese Information in der Spalte Ursprung angezeigt. Auch Kombinationen von mehreren Ursprüngen sind möglich. Mit Hilfe dieser Informa-

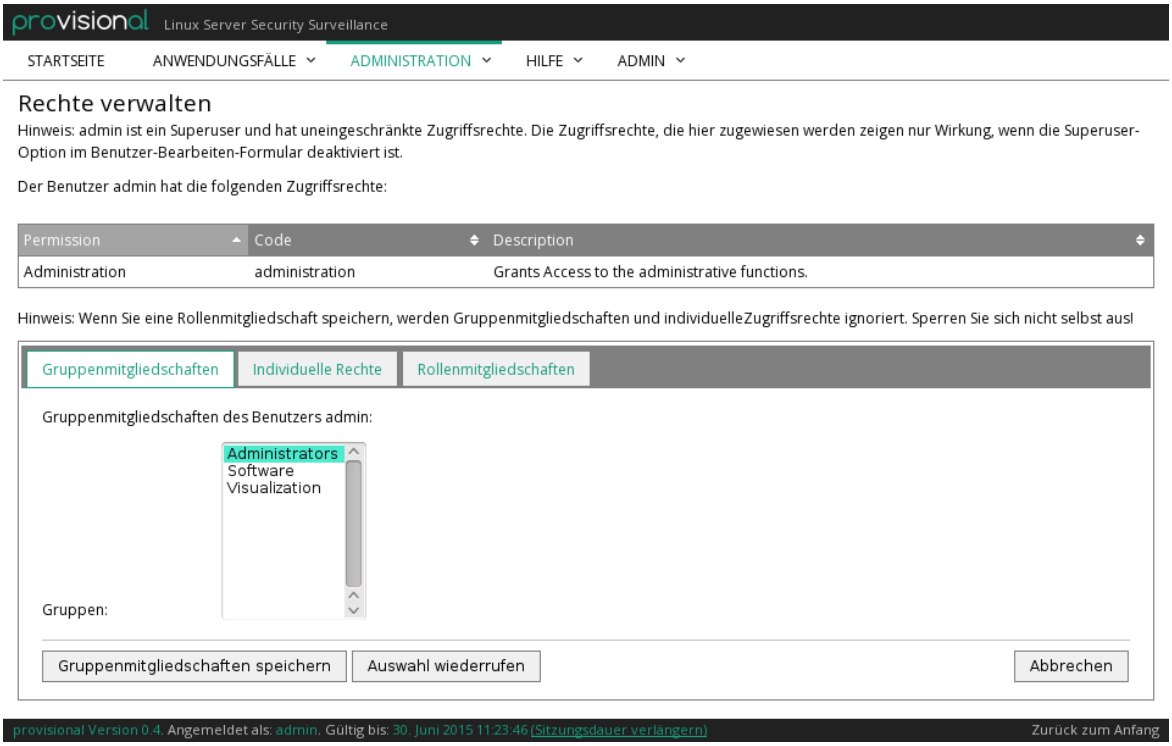


Abbildung 7.28.: Festlegen der Zugriffsrechte eines Benutzerkontos im Prototyp.

Abbildung 7.29.: Ausschnitt der einem Anwender zugewiesenen Server mit Information über die Herkunft der Zuweisung.

tion kann der Administrator dem Anwender schneller Server wieder entziehen, ohne lange nach der Stelle der Zuweisung suchen zu müssen. Abbildung 7.29 zeigt einen Ausschnitt dieser Übersicht.

Durch einen Klick auf *Server zuweisen* in der Benutzerübersicht kann der Administrator dem jeweiligen Benutzerkonto bei Bedarf individuelle Server zur Verfügung stellen (siehe Abbildung 7.30).

Benutzerkonten können entweder im Bearbeiten-Dialog vom Administrator selbst oder automatisch deaktiviert werden. Ein deaktiviertes Benutzerkonto ist nicht mehr zum Login bei der Anwendung berechtigt. Ein Benutzerkonto wird aus Sicherheitsgründen automatisch gesperrt, sobald eine bestimmte Anzahl von Anmeldeversuchen nacheinander fehlgeschlagen sind. Dadurch werden Angriffsversuche erschwert.

Ein bestehendes Benutzerkonto wird durch einen Klick auf den entsprechenden Link in der Spalte *Benutzername* zur Bearbeitung geöffnet. In Abbildung 7.31 ist ein Beispiel hierfür zu sehen. In dieser Ansicht können Konten auch wieder gelöscht werden. Wie auch bei den zuvor genannten Gruppen und Rollen gilt auch hier, dass das Benutzerkonto hierzu zuerst deaktiviert werden muss. Dadurch wird ein versehentliches Löschen verhindert.

Das Formular *Benutzerkonto Bearbeiten* besitzt auch eine Schaltfläche mit der Bezeichnung *Rechte verwalten*. Durch Betätigen dieser Schaltfläche öffnet sich ebenfalls die Rechteverwaltung für das gewählte Konto (siehe

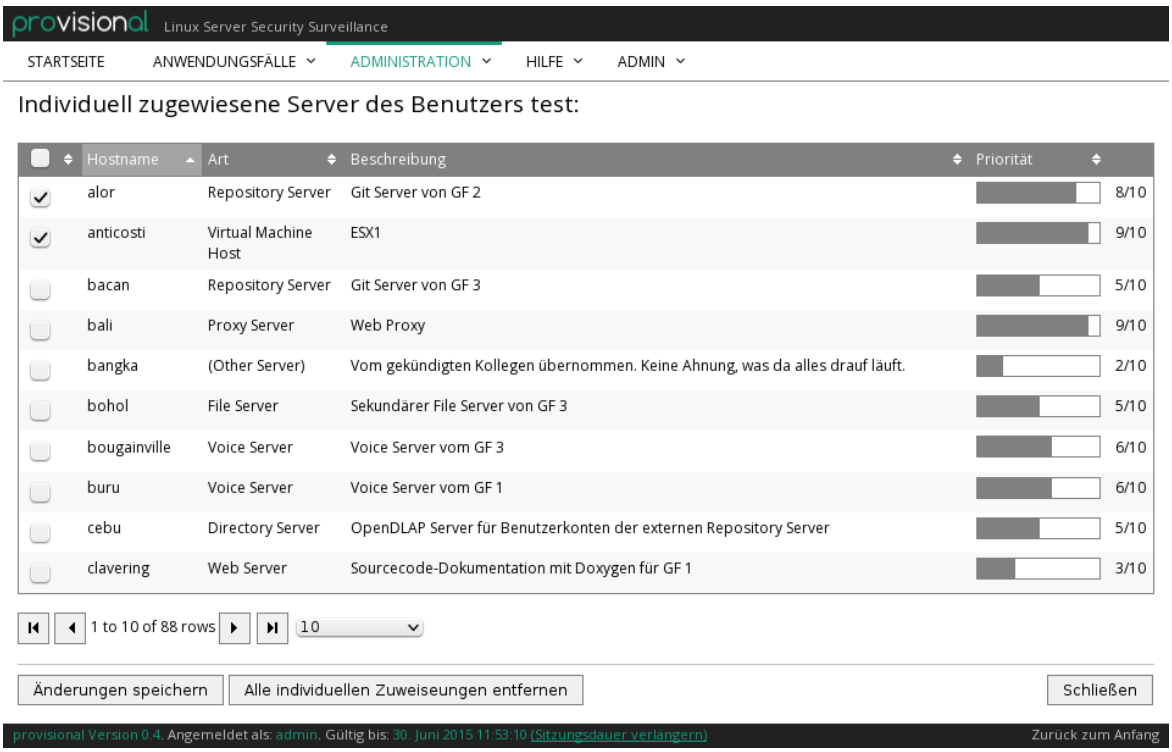


Abbildung 7.30.: Individuelle Zuweisung von Servern zu einem Benutzerkonto.

Abbildung 7.31).

Sitzungsverwaltung

Über den Punkt *Sitzungen* im Administrationsmenü erhält der Administrator Informationen über die Benutzer, die im Moment bei der prototypischen Anwendung eingeloggt sind. Die Tabelle zeigt unter anderem an, wie lange die Sitzung noch gültig ist, bis der Anwender nach einer Inaktivitätsphase automatisch wieder abgemeldet wird. Abbildung 7.32 zeigt ein Beispiel für eine solche Tabelle. Dort sind im Moment zwei Benutzer gleichzeitig eingeloggt.

Über die Sitzungsverwaltung können Sitzungen anderer eingeloggter Anwender auf der Stelle von einem Administrator beendet werden. Zu diesem Zweck dient der Link mit der Bezeichnung *Diesen Benutzer jetzt ausloggen*. Nach einem Klick auf einen solchen Link erscheint ein modales Dialogfenster, das noch einmal nachfragt, ob die Sitzung wirklich beendet werden soll. Dabei unterscheidet das Programm zwischen Benutzerkonten anderer Personen und dem eigenen Benutzerkonto und zeigt für beide Fälle einen unterschiedlichen Hinweistext an. Einem versehentlichen Beenden der eigenen Sitzung wird so entgegengewirkt.

Das Beenden einer fremden Sitzung hat keinen Einfluss auf den Bildschirminhalt, den der jeweilige Anwender momentan zu sehen bekommt. Sobald er aber auf irgendeine Schaltfläche oder Link klickt oder zuvor besuchte Webseiten der Anwendung aus dem Verlaufspeicher abrufen will, wird er unmittelbar zum Logout-Dialog umgeleitet. In Kombination mit der Möglichkeit, Benutzerkonten zu deaktivieren, können einzelne Personen bei Bedarf komplett aus der Anwendung ausgesperrt werden.

Systemprotokoll

Wichtige Systemereignisse zeichnet der Prototyp in einer Protokolldatei auf, die sich im Projektverzeichnis der Anwendung befindet. Der Name der Datei lautet `provisional.log` und ist in der globalen Einstellungs-Datei `settings.py` festgelegt. In Zeile 2 von Listig 7.7 ist dieser Eintrag in der Konfigurationsdatei zu

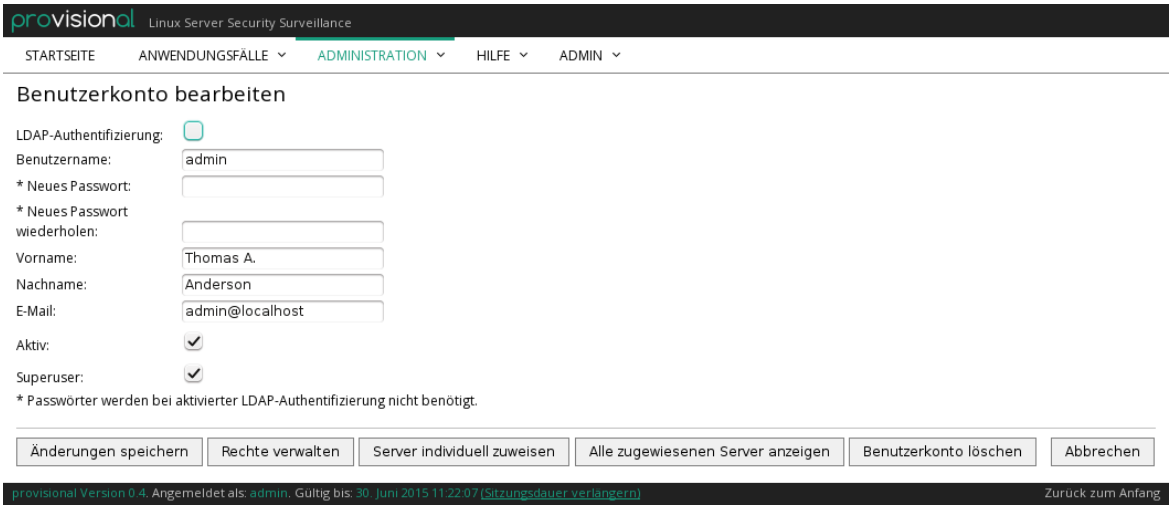


Abbildung 7.31.: Bearbeiten eines Benutzerkontos im Prototyp.

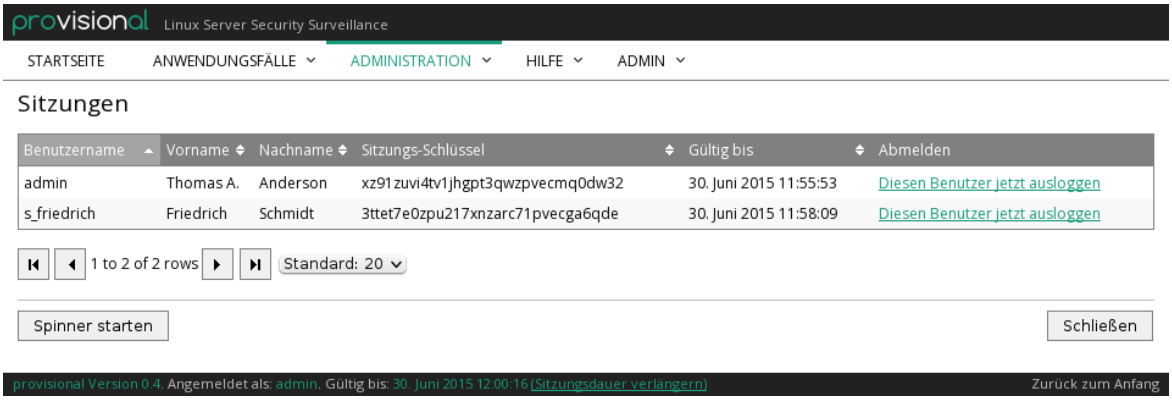


Abbildung 7.32.: Sitzungsverwaltung mit Übersicht der eingeloggten Benutzer im Prototyp.

sehen. Er wird aus dem Namen der Anwendung, die in der Variable `APPLICATION_NAME` gespeichert ist, und der Dateiendung `.log` zusammengesetzt.

Innerhalb der Logdatei sind mehrere Loglevel möglich. Sie dienen der Unterscheidung verschiedener Wichtigkeitsgrade von Logeinträgen. Der Prototyp verwendet die folgenden Loglevel:

INFO Allgemeine Informationen mit Hinweisen für den Entwickler, z. B. Login-Ereignisse.

WARNING Warnmeldungen aufgrund von Verhalten der Anwender, z. B. fehlgeschlagene Anmeldeversuche.

ERROR Zur Laufzeit aufgetretene Programmfehler.

CRITICAL Schwerwiegende zur Laufzeit aufgetretene Programmfehler.

Listing 7.7: Ausschnitt der globalen Einstellungsdatei mit Pfadangaben

```
1 # Logging
2 LOGGING_FILENAME = "%s.log" % APPLICATION_NAME
3
4 # Changelog filename
5 CHANGELOG_FILENAME = "changelog.txt"
```

Die Logdatei kann von Anwendern mit der entsprechenden Berechtigung auch direkt über die Weboberfläche des Prototyps aufgerufen werden, und zwar über den Punkt *Protokoll* im Administrationsmenü. Abbildung 7.33 zeigt ein beispielhaftes Bildschirmfoto von dieser Programmfunktion.

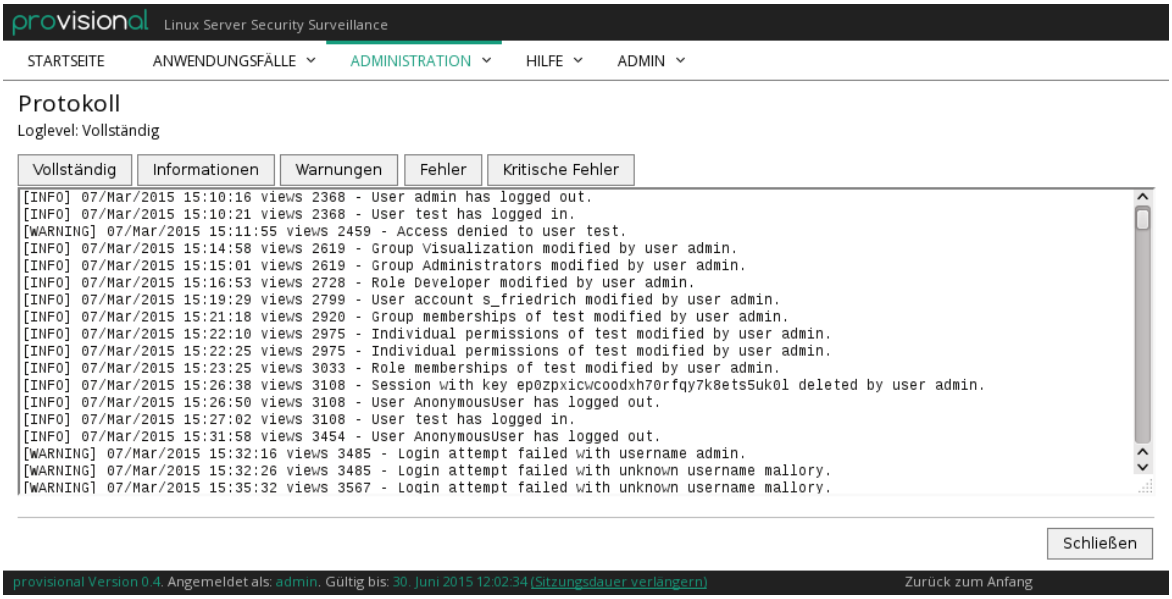


Abbildung 7.33.: Protokoll mit Systemereignissen der Kategorie “Warnungen” im Prototyp.

Standardmäßig wird beim Aufruf dieser Seite der vollständige Inhalt der Protokolldatei in einem scrollbaren Textfenster angezeigt. Der Anwender hat aber auch die Möglichkeit, gezielt Einträge mit einem bestimmten Loglevel abzurufen. Zu diesem Zweck dienen die Schaltflächen über dem Textfenster. Durch einen Klick auf einen der Buttons werden nur noch Ereignisse mit dem entsprechenden Wichtigkeitsgrad eingeblendet. Die Schaltfläche mit der Beschriftung *Vollständig* bewirkt die Deaktivierung der Filterfunktion.

7.3.4. Persönliches Menü

Neben den fixen Einträgen im Navigationsmenü der Anwendung gibt es noch einen Menüpunkt, der mit dem Namen des angemeldeten Benutzers beschriftet ist. In den Screenshots, die bisher zu sehen waren, handelt es sich dabei größtenteils um das Standard-Administratorkonto mit der Bezeichnung *admin*. In Abbildung 7.17 ist zu erkennen, dass ein anderer Benutzer eingeloggt war.

Dieser Menüeintrag dient nicht nur zur Information, welcher Anwender gerade an einem Rechner eingeloggt ist, sondern beherbergt auch Programmfunktionen zur Verwaltung benutzerspezifischer Informationen. Dazu zählen zunächst die persönlichen Einstellungen.

Persönliche Einstellungen

Die Seite *Einstellungen* beinhaltet mehrere Tabs mit benutzerdefinierbaren Einstellmöglichkeiten. Abbildung 7.34 zeigt der Einfachheit halber den Inhalt aller Tabs nebeneinander.

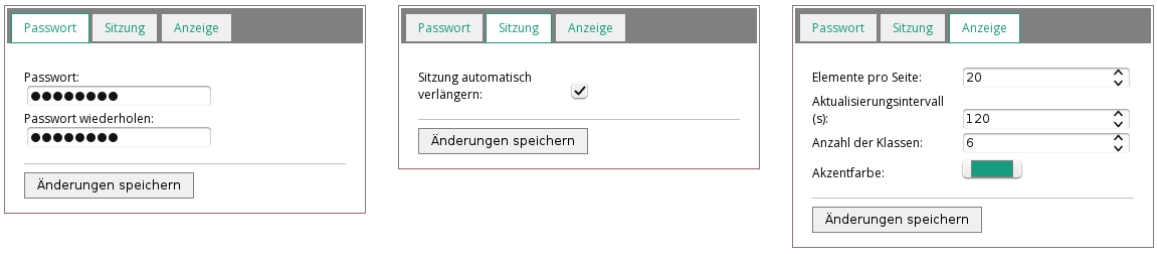


Abbildung 7.34.: Benutzerspezifische Einstellmöglichkeiten des Prototyps.

Der Tab mit der Beschriftung *Passwort* dient zur Änderung des Kennworts, mit dem sich der Anwender beim Prototyp einloggt. Beim Anlegen eines neuen Benutzerkontos durch den Administrator erhalten Anwender zunächst ein Initialpasswort, dass sie über diese Programmfunktion zur Erhöhung der Sicherheit ändern können. Der Passwort-Tab ist nur bei Benutzerkonten sichtbar, die über eine lokale Benutzerauthentifizierung verfügen. Bei Konten mit LDAP-basierter Authentifizierung kann das Passwort nicht über die Bedienoberfläche des Prototyps geändert werden. Aus diesem Grund macht es auch keinen Sinn, den Tab in diesem Fall einzublenden.

Der Reiter mit dem Titel *Sitzung* beinhaltet die Option, die Benutzersitzung trotz Inaktivität des Anwenders automatisch zu verlängern. In einem Abstand von wenigen Minuten führt die Anwendung dann im Hintergrund einen für den Benutzer nicht bemerkbaren AJAX-Aufruf durch, der beim Server eine Verlängerung der Sitzung um die in der Datei `settings.py` eingestellte Höchstdauer bewirkt. Gleichzeitig fragt der Client ab, wie lange die Sitzung nach der Erneuerung gültig ist und blendet den neuen Wert in der Fußzeile des Prototyps ein. Diese Funktion wurde in erster Linie implementiert, um die Entwicklungsphase des Prototyps angenehmer zu gestalten. Der wiederholte Login während des Programmierens war dann nicht mehr notwendig.

In einem Produktivsystem muss eine derartige Funktion jedoch deaktiviert oder auskommentiert werden: Sobald ein Anwender vergisst, sich auszuloggen, und den Arbeitsplatz verlässt ohne den Bildschirm zu sperren, könnte ein Angreifer mit den Rechten des eingeloggten Anwenders auf das Programm und damit auf sicherheitskritische Serverinformationen zugreifen. Selbst wenn der Angreifer keinen direkten Zugang zum Arbeitsplatz des eingeloggten Anwenders hat, stellen nicht endende Sitzungen ein Sicherheitsproblem dar: Da sich der Sitzungsschlüssel des Clients nie ändert, ist es leichter für einen Angreifer, die Sitzung zu übernehmen und von einem anderen System aus die Zugriffsrechte des eingeloggten Anwenders zu erhalten. Häufig wechselnde Sitzungsschlüssel wirken dieser Gefahr entgegen.

Der dritte Tab mit der Beschriftung *Anzeige* speichert Standardeinstellungen, die sich auf die Darstellung von Diagrammen und Tabellen innerhalb der Anwendung auswirken. Der Wert im Eingabefeld *Elemente pro Seite* wirkt sich auf die maximale Anzahl von Zeilen aus, die standardmäßig in Tabellen sichtbar sind. Der Wert im Feld *Aktivierungsintervall (s)* beeinflusst die Dauer bis zum automatischen Neuzeichnen dynamisch generierter Visualisierungen im Prototyp. Das dritte Feld *Anzahl der Klassen* speichert die Anzahl der Buckets, die standardmäßig in Histogrammen angezeigt werden sollen.

Das letzte Eingabefeld in dem Tab erlaubt dem Anwender, die Akzentfarbe der Anwendung nach den eigenen Bedürfnissen anzupassen. Sie wirkt sich unter anderem auf die Farbe von Hyperlinks, Schaltflächentexten und den Hervorhebungen im Navigationsmenü aus. Bei dem Eingabefeld handelt es sich um einen Element, das mit HTML5 eingeführt wurde. Ein Klick auf das Feld öffnet ein Dialogfenster zur Farbwahl, dessen Aufbau von der individuellen Browser-Implementierung abhängig ist.

Die Eingabefelder des Anzeige-Tabs sind standardmäßig leer. Werte, die nicht vorhanden sind, ersetzt die Anwendung automatisch durch Standardwerte, die in der globalen Konfigurationsdatei `settings.py` eingetragen sind. Listing 7.8 zeigt den entsprechenden Ausschnitt aus dieser Datei.

Listing 7.8: Ausschnitt der globalen Einstellungsdatei mit Standardwerten für Anzeigeeinstellungen

```
1  # Default paginator size (if no user-specific setting available)
2  DEFAULT_PAGINATOR_SIZE = 25
3  # Minimum and Maximum paginator size
4  MINIMUM_PAGINATOR_SIZE = 1
5  MAXIMUM_PAGINATOR_SIZE = 1000
6
7  # Default refresh interval for certain graphics (in seconds)
8  DEFAULT_REFRESH_INTERVAL = 60
9
10 # Default number of classes in a histogram
11 DEFAULT_HISTOGRAM_CLASSES = 6
12
13 # Default accent color
14 DEFAULT_ACCENT_COLOR = "#179c7d"
```

Der Anwender kann seine benutzerspezifischen Einstellungen bei Bedarf auch wieder löschen, indem er auf die Schaltfläche mit der Beschriftung *Werkseinstellungen wiederherstellen* klickt. In diesem Fall treten dann unmittelbar wieder die Standardeinstellungen aus der zentralen Konfigurationsdatei `settings.py` in Kraft.

Persönliche Tags

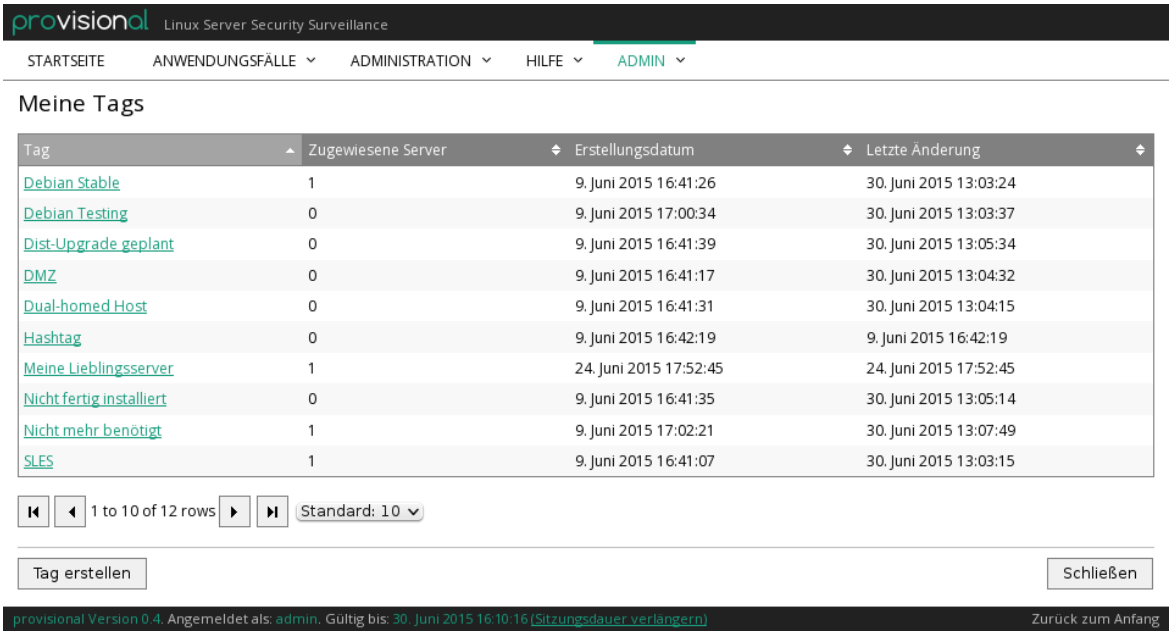


Abbildung 7.35.: Verwaltung benutzerdefinierbarer Tags zur Kennzeichnung der Server.

Anwender des Prototyps bekommen vom Administrator Server aus einer Gesamtauswahl zugewiesen, die als Datengrundlage für Visualisierungen dienen. Diese Server können Anwender nach ihren eigenen Bedürfnissen mit zusätzlichen Metadaten in Form von Tags versehen. Über den Menüpunkt *Meine Tags* öffnen Anwender eine Übersicht der benutzerdefinierten Tags in tabellarischer Form. Diese Tabelle beinhaltet in der Spalte *Zugewiesene Server* auch die Information darüber, wie vielen Servern ein bestimmter Tag bereits zugeordnet wurde (siehe Abbildung 7.35).

Persönliche Server

Die eigentliche Zuordnung von Tags zu den eigenen Servern geschieht über den Menüpunkt *Meine Server*. Ein Screenshot der dazugehörigen Webseite ist in Abbildung 7.36 dargestellt. Der Punkt führt zu einer tabellarischen Übersicht der dem Anwender zugeordneten Server. Die dort angezeigten Informationen entsprechen größtenteils denen, die auch in der Serververwaltung des Administrationsbereichs vorhanden sind.

Die Übersicht *Meine Server* gestattet dem Anwender aber nicht die Änderungen an Serverinformationen, die im Administrationsbereich möglich sind. Lediglich über den Link *Tags zuweisen* kann er seinen Servern individuell selbst erstellte Tags zuordnen.

7.3.5. Hilfemenü

Das Hilfemenü wurde zur Unterbringung von Zusatzinformationen vorgesehen, die nicht direkt zur Funktion der Visualisierungen beitragen. Die Zusatzinformationen befinden sich im darin enthaltenen Untermenü mit der Bezeichnung *Über* und werden im Folgenden der Vollständigkeit halber kurz vorgestellt.

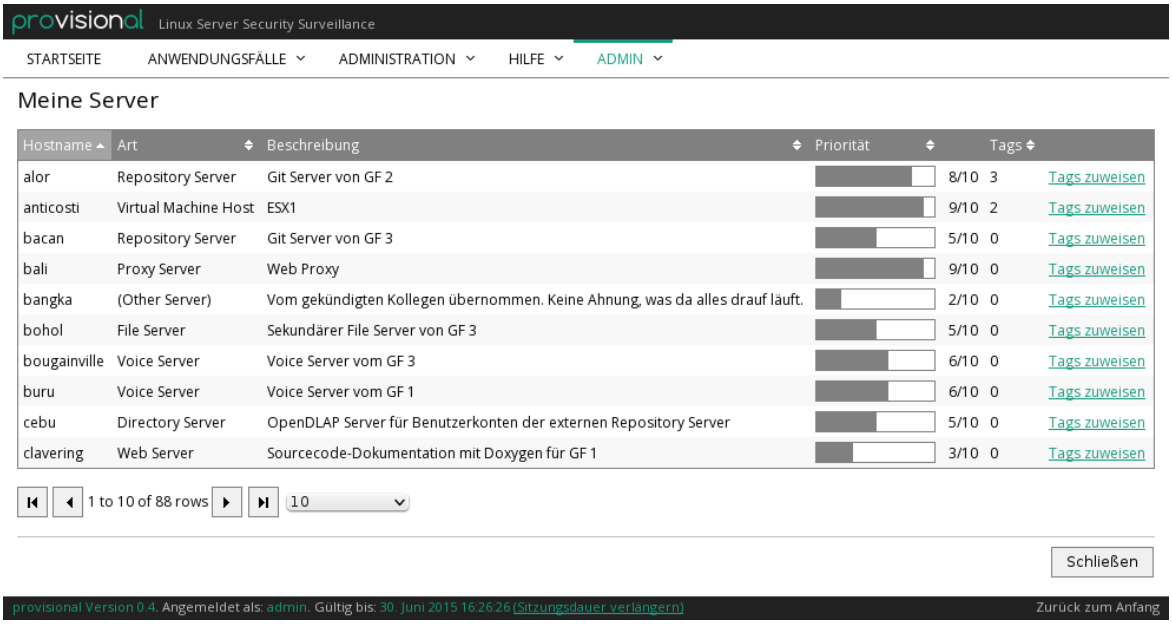


Abbildung 7.36.: Übersicht der eigenen Server.

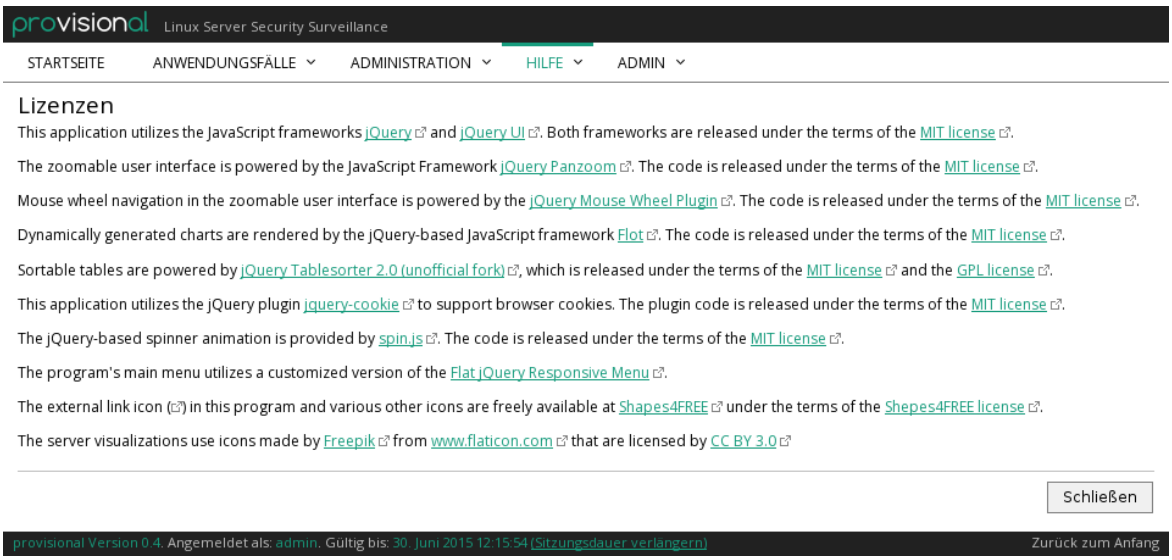


Abbildung 7.37.: Auflistung der eingebundenen Fremdsoftware und dazugehörige Lizenzen.

Programminfos

Hinter dem Punkt *Programminfos* des Über-Menüs verbergen sich Versions- und Copyright-Angaben des Prototyps.

Lizenzen

Um das Rad nicht neu zu erfinden, wurde bei der Entwicklung der Client-Komponente des Prototyps auf quelloffene JavaScript-Bibliotheken zurückgegriffen. Diese Bibliotheken wurden bereits in Abschnitt 7.2.1 vorgestellt. Über den Punkt *Lizenzen* des Über-Menüs erhalten Anwender eine Übersicht der verwendeten

Bibliotheken inklusive der Angabe der Urheber und der Lizenzbedingungen. Abbildung 7.37 zeigt ein Bildschirmfoto dieser Webseite.

Änderungshistorie

Die schrittweise Implementierung der einzelnen funktionalen Einheiten des Prototyps wurde in einem Changelog festgehalten, das sich hinter dem Punkt *Änderungshistorie* des Über-Menüs verbirgt. Ein Bildschirmfoto hiervon ist in Abbildung 7.38 zu sehen.

Als Datenquelle für das Changelog dient die Textdatei mit dem Namen `changelog.txt`. Sie befindet sich im Projektverzeichnis des Demonstrators. Damit die Anwendung auf der Webseite der Änderungshistorie den Inhalt dieser Datei einbindet, ist ein Verweis darauf in der globalen Einstellungsdatei `settings.py` vorhanden (siehe Zeile 5 von Listing 7.7).

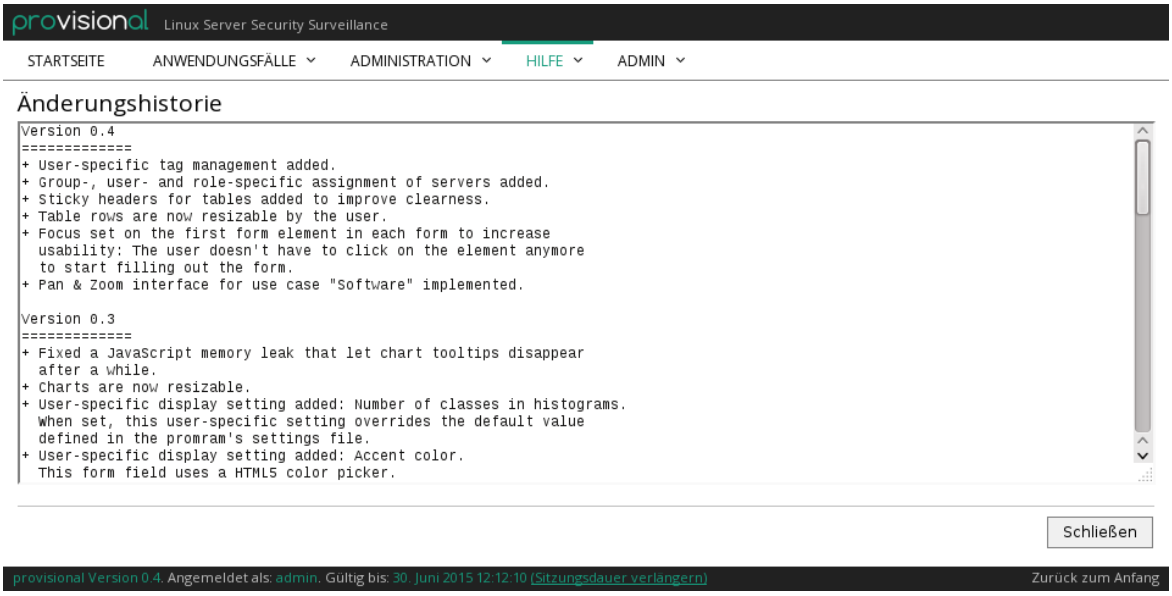


Abbildung 7.38.: Die Änderungshistorie der prototypischen Anwendung.

7.4. Visualisierung von Serverzuständen

Bisher wurden hauptsächlich die administrativen Komponenten der Bedienoberfläche des Prototyps vorgestellt. Dieser Abschnitt widmet sich den dynamisch generierten Visualisierungen, die im Demonstrator implementiert wurden. Sie basieren auf Vorschlägen, die im Kapitel 6 erarbeitet wurden und stellen eine repräsentative Auswahl davon dar.

Anhand der implementierten Visualisierungen wird gezeigt, dass sicherheitsrelevante Serverinformationen unter Berücksichtigung von Grundlagen der Informationsvisualisierung in einem webbasierten Administrationstool dargestellt werden können. Dabei handelt es sich einerseits um traditionelle Visualisierungsmethoden wie Diagramme, andererseits um eine im Anwendungsbereich der Linux-Serveradministration neue Visualisierungsmethode mit einer zoom- und filterbaren Bedienoberfläche, die mit Dynamit Quereis (siehe Abschnitt 2.14.3) arbeitet.

7.4.1. Gesamtübersicht von Serverzuständen

In Abschnitt 6.1.2 wurde der Vorschlag gemacht, Ampelpiktogramme zu verwenden, um dem Anwender unterschiedliche sicherheitsrelevante Informationen der Server auf einen Blick zu vermitteln. Die Implementierung

einer solchen Übersicht befindet sich im Demonstrator im Menü *Anwendungsfälle* unter dem Punkt mit dem Namen *Gesamtübersicht*.

Da der Demonstrator im Gegensatz zu einem Produktivsystem keine Live-Informationen von hunderten Servern erhält, greift er auf einen unveränderten Datenbestand in seiner Datenbank zurück, der einen bestimmten Zeitpunkt widerspiegelt. Da mit dieser unveränderlichen Datenquelle die Ampeln der Gesamtübersicht nie umschalten würden, liefert der Server beim Demonstrator zufällige Statusinformationen. Bei jedem erneuten Laden der Visualisierung oder der gesamten Webseite führen neue Zufallsdaten zu unterschiedlichen Ampelfarben.

Die Tatsache, dass in der Gesamtübersicht zufällige Daten visualisiert werden, hat keine nachteilige Auswirkung auf das Software Design. In einem Produktivsystem müsste lediglich die Serverkomponente angepasst werden, um reale Daten zu liefern. Der Programmcode des Clients und die AJAX-Schnittstelle bleiben unverändert.

Abbildung 7.39 zeigt einen Screenshot der angesprochenen Gesamtübersicht. Die Zustandsinformationen entsprechen exakt den vier Anwendungsfällen, die im Kapitel chap:loesungsmoeglichkeiten vorgestellt wurden. Jede Ampel repräsentiert einen anderen Use Case.

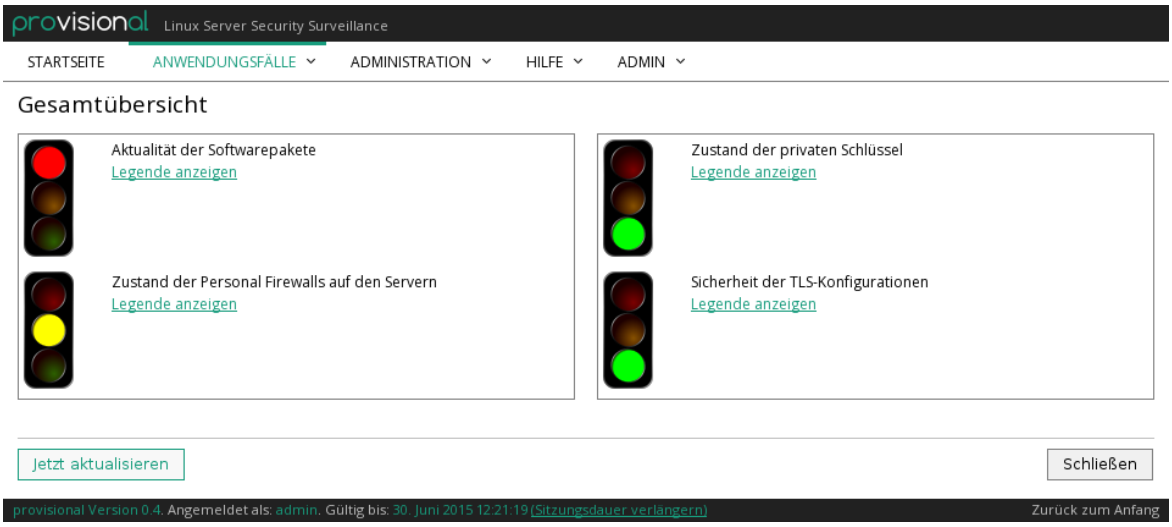


Abbildung 7.39.: Gesamtübersicht der Zustände der Server nach Anwendungsfällen aufgeschlüsselt.

Um die Bedienfreundlichkeit zu erhöhen und gleichzeitig den Bildschirm möglichst frei von potenziell ablenkenden Inhalten zu lassen, verfügt jede der Ampeln über eine optional einblendbare Legende. Sie ist über den Link mit der Bezeichnung *Legende anzeigen* abrufbar. Da es sich bei den Legenden um nichtmodale Fenster handelt, können sie auch alle gleichzeitig sichtbar sein und beliebig innerhalb des Browserfensters angeordnet werden. Abbildung 7.40 zeigt ein solches Fenster am Beispiel des Anwendungsfalls *Aktualität der Softwarepakete*.

7.4.2. Server-Statistiken

Im Abschnitt 6.1.2 wurden verschiedene Diagrammtypen mit Statistiken zu den Patch-Zuständen der Server vorgeschlagen. Eine Implementierung dieser Diagramme befindet sich im Demonstrator als Unterpunkt *Statistiken* im Untermenü *Software* des Anwendungsfälle-Menüs. Die dort gezeigten dynamisch generierten Visualisierungen basieren aus dem zuvor genannten Grund ebenfalls auf Zufallsdaten. Auch hier gilt die Aussage, dass lediglich die Serverkomponente angepasst werden muss, um tatsächliche Daten zu liefern. Die Client-Seite und auch das Format der übertragenen Daten bleiben unangetastet.

Die Webseite besteht aus mehreren Tabs, die jeweils einen Diagrammtyp beinhalten. Zunächst wird der Tab *Allgemeiner Patch-Zustand* betrachtet. Ein Screenshot hiervon ist in Abbildung 7.41 zu sehen. Bildschirmfotos von den restlichen Diagrammen sind in den Abbildungen 7.42, 7.43, 7.44, 7.45 und 7.46 zu dargestellt.

Abbildung 7.40.: Gesamtübersicht der Zustände der Server nach Anwendungsfällen aufgeschlüsselt.

Das Diagramm in Abbildung 7.41 visualisiert die Anzahl der Server, deren Softwareaktualität in eine der Kategorien *gepatcht*, *nicht gepatcht* und *kritischer Paketzustand* fällt. Diese Zustände sind in Form eines Stapeldiagramms dargestellt, das einen unmittelbaren Vergleich der Verhältnisse und ein Ablesen der absoluten Werte zulässt. Die Signalfarben Grün, Gelb und Rot erlauben dem Betrachter intuitiv, die Bedeutung der Säulen zu erfassen, ohne die Achsbeschreibungen lesen zu müssen. Das Diagramm verfügt über Tooltip-Texte, die dem Anwender den exakten Wert eines Stapels liefern.

Visual Clutter wurde auf ein Minimum reduziert, um die Übersichtlichkeit zu optimieren. Dazu zählen zum Beispiel unnötige Hilfslinien, die die JavaScript-Bibliothek, die zur Zeichnung des Diagramms verwendet wird, standardmäßig einblenden würde. Um die Aufmerksamkeit des Betrachters auf die Zustände der Softwarestände zu lenken, kommen Farben in dem Diagramm nur bei den drei Säulen zur Verwendung. Da auf eine flächendeckende Hintergrundfüllung verzichtet wurde, ist die dynamisch generierte Visualisierung nicht nur übersichtlicher, sondern auch noch druckerfreundlich.

Die Größe des Diagramms ist vom Benutzer an seine eigenen Bedürfnisse oder an die Größe des Ausgabemediums anpassbar. Die Größenänderung erfolgt über einen Anfasspunkt unten rechts im Bild.

Das in Abbildung 7.42 dargestellte Diagramm greift auf dieselbe Datenquelle des Servers zurück wie das zuvor genannte Säulendiagramm. Das hier zu sehende Kreisdiagramm visualisiert ebenfalls die Anteile der Server der zuvor genannten Software-Zustände und nutzt auch die gleichen Signalfarben. Da es sich hier um ein Kreisdiagramm handelt, liegt der Fokus auf die relativen Verhältnisse. Um absolute Werte zu erhalten, ist das Diagramm in Abbildung 7.41 besser geeignet.

Das Kreisdiagramm ist nicht bis zum Mittelpunkt mit Farbe gefüllt, sondern stellt eher einen Ring dar, der in Segmente unterteilt ist. Diese Form der Visualisierung wurde gewählt, da es dieselben Informationen ausdrückt wie ein herkömmliches Kreisdiagramm, jedoch ein besseres Data-Ink-Verhältnis besitzt (siehe Abschnitt 2.7.1). Zudem wird bei der Ausgabe auf Papier Druckertinte bzw. Toner eingespart.[37]

Wie bereits in Abschnitt 6.1.2 vorgeschlagen wurde, kann die Aussagekraft des Diagramms aus Abbildung 7.41 erhöht werden, indem Serverkategorien als zusätzliche Information aufgenommen werden. Dies geschieht dadurch, indem das Säulendiagramm in Stapel unterteilt wird. Jeder der Stapel entspricht einer Serverkategorie, die farblich eindeutig kodiert ist.

Da der Prototyp bereits im Auslieferungszustand ca. 30 Serverkategorien unterscheidet und Administratoren auch eigene Kategorien hinzufügen können, muss bei dieser Darstellungsform die Anzahl der angezeigten Kategorien eingeschränkt werden, da ihre Farben sonst kaum noch voneinander unterscheidbar sind. Auf diese

[37] Wenn es nach Edward Tufte ginge, würde sich das Diagramm vermutlich aus einem Kreisumfang mit einer Breite von lediglich einem Pixel zusammensetzen, um das Data-Ink-Verhältnis zu optimieren. Da die einzelnen Segmente dann aber kaum noch zu unterscheiden wären, ist es naheliegend, in begründeten Fällen gegen dieses Dogma zu verstoßen.

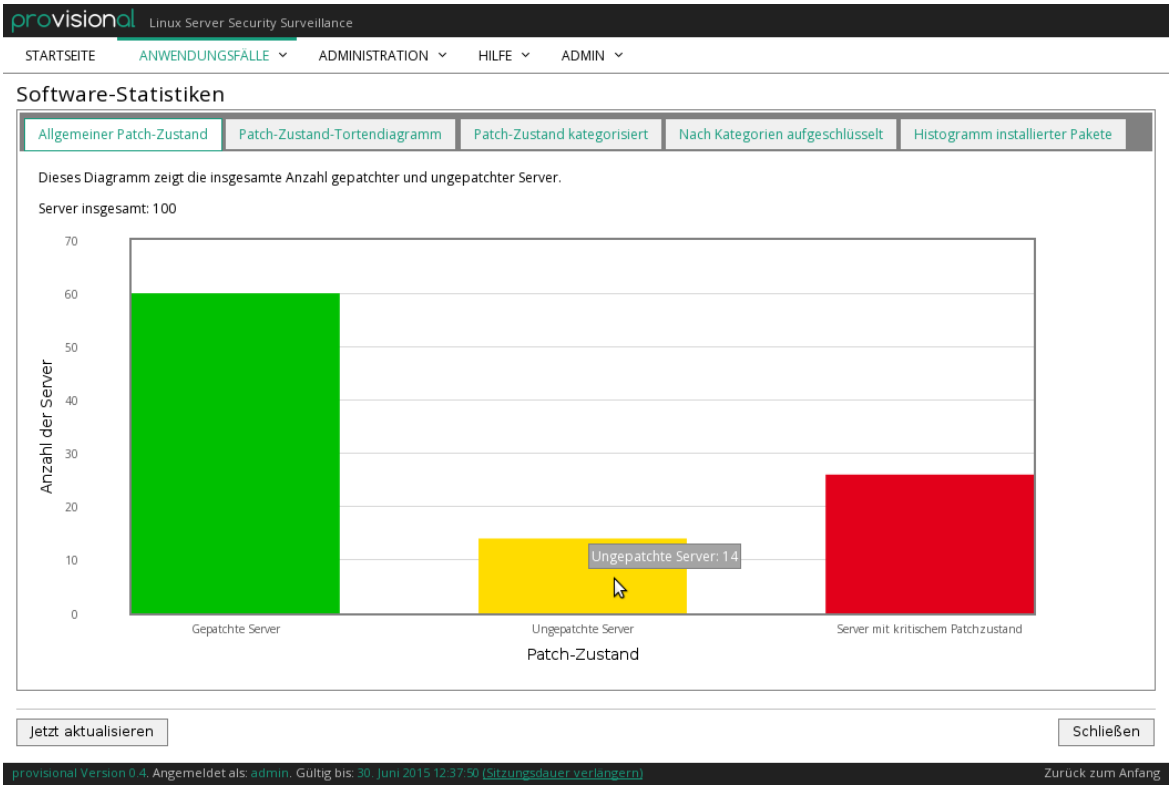


Abbildung 7.41.: Säulendiagramm mit unterschiedlichen Patch-Zuständen der Server.

Weise gehen aber Informationen über die Server der ausgeblendeten Kategorien verloren, die durchaus sicherheitsrelevant sind. Um dennoch eine hohe Anzahl von Serverkategorien für den Betrachter unterscheidbar zu machen, beinhalten die Tooltip-Texte der einzelnen Stapel den Namen der Kategorie.

In Abbildung 7.43 ist die Implementierung dieser Visualisierung im der prototypischen Anwendung zu sehen. Zu Demonstrationszwecken wählt der Server hier nicht nur die Zahlen, sondern auch eine begrenzte Menge an Kategorien nach dem Zufallsprinzip aus. Die Anzahl an Kategorien ist überschaubar und die einzelnen Stapel sind gut voneinander unterscheidbar.

Die Tooltip-Texte helfen nicht nur bei der Unterscheidung der Kategorien, sondern auch beim Ablesen der Werte der einzelnen Stapelelemente. Während bei einem Säulendiagramm ohne Offset die Werte einfach an den Oberkanten der Säulen ablesbar sind, ist dies bei gestapelten Säulen ab dem zweiten Stapelelement von unten schwieriger, da die Differenz zwischen oberer und unterer Kante gebildet werden muss. Die Tooltip-Texte beinhalten die exakten werte, sodass der Betrachter nicht anfangen muss, zu rechnen. In Abbildung 7.43 ist ein Beispiel für einen unterstützenden Tooltip-Text zu sehen.

In Abschnitt 6.1.2 wurde auch eine alternative Form von gestapelten Säulendiagrammen vorgeschlagen, um die Unterscheidung von Serverkategorien zu berücksichtigen. Anstatt die Kategorien farblich zu kodieren, werden die bewährten Signalfarben wieder eingeführt und jeder Kategorie wird eine eigene Säule gewidmet. Abbildung 7.44 zeigt ein Bildschirmfoto der Implementierung dieser Visualisierung im Prototyp.

Der Vorteil dieser Darstellungsform ist die hervorragende Unterscheidbarkeit der verwendeten Signalfarben. Zudem ist auf einen Blick ein Vergleich der Zustände unterschiedlicher Kategorien möglich. Bei einer zu großen Anzahl von Kategorien kann das Diagramm aber sehr breit werden und muss schließlich gescrollt werden. Aus diesem Grund ist auch hier eine Vorauswahl denkbar, die dem Betrachter allerdings sicherheitsrelevante Informationen ausgeblendeter Systeme vorenthält.

Neben dem Patch-Zustand der einzelnen Systeme ist es für den Administrator auch interessant, wie die Verteilung der Anzahl installierter Pakete auf den einzelnen Server ist. Dies liegt daran, dass mit der Anzahl installierter Pakete auch die potenzielle Verwundbarkeit eines Servers zusammenhängt: Je weniger Software

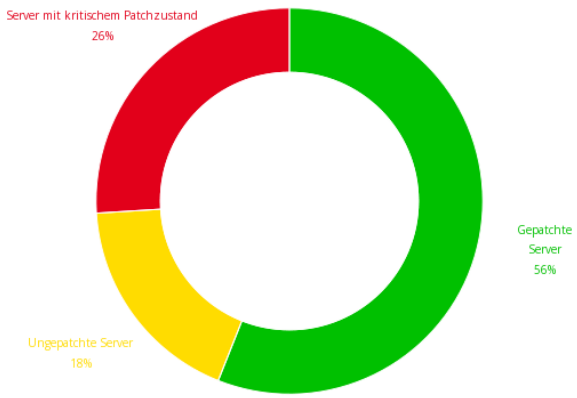


Abbildung 7.42.: Kreisdiagramm mit unterschiedlichen Patch-Zuständen der Server.

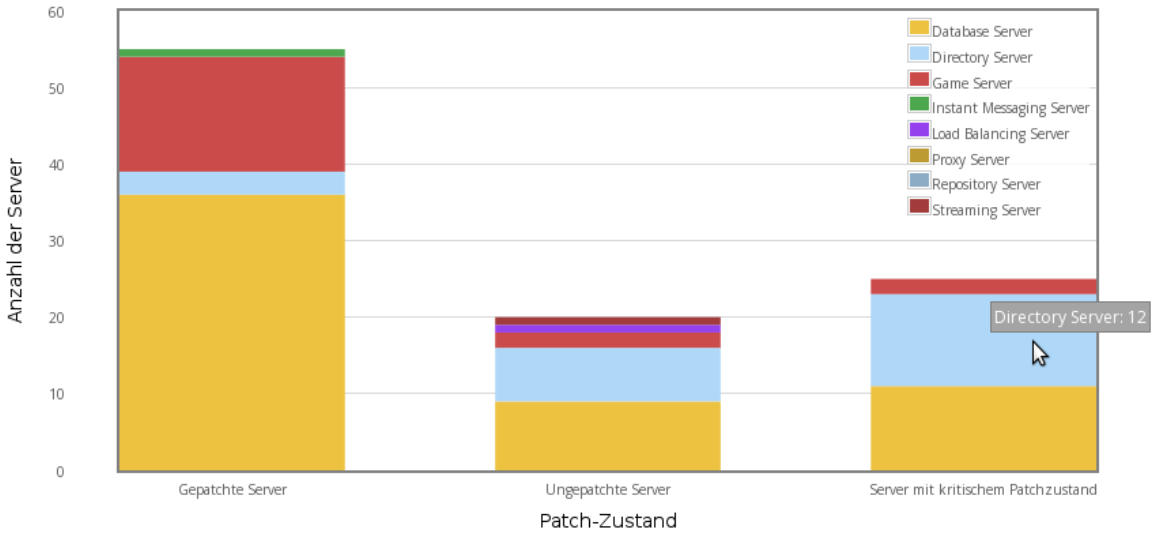


Abbildung 7.43.: Gestapeltes Säulendiagramm mit unterschiedlichen Patch-Zuständen unter Berücksichtigung von Serverkategorien.

vorhanden ist, desto geringer ist im Allgemeinen die Angriffsfläche. Um diese Verteilung grafisch darzustellen, wurde im Prototyp ein Histogramm implementiert. Ein Beispiel hiervon ist in Abbildung 7.45 dargestellt.

Die JavaScript-Bibliothek, die zum Zeichnen des Diagramms verwendet wird, unterstützt Histogramme nicht out of the box. Aus diesem Grund handelt es sich um ein an die Anforderungen an ein Histogramm angepasstes Säulendiagramm. Die Aufteilung der Verteilung in Buckets wird durch eine selbst programmierte JavaScript-Funktion realisiert.

Ergänzend zum Histogramm werden die folgenden statistischen Informationen zu der Verteilung eingeblendet:

- Minimale Paketanzahl
- Maximale Paketanzahl
- Median
- Mittelwert
- Standardabweichung

Das *Histogramm installierter Pakete* bietet dem Anwender im Gegensatz zu den vorherigen Schaubildern

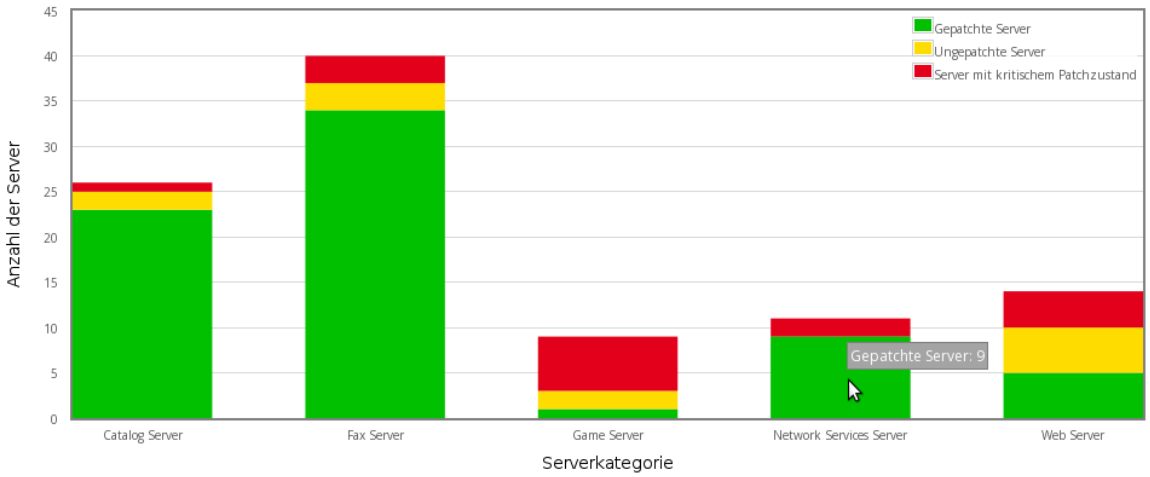


Abbildung 7.44.: Gestapeltes Säulendiagramm mit Patch-Zuständen nach Serverkategorien aufgeschlüsselt.

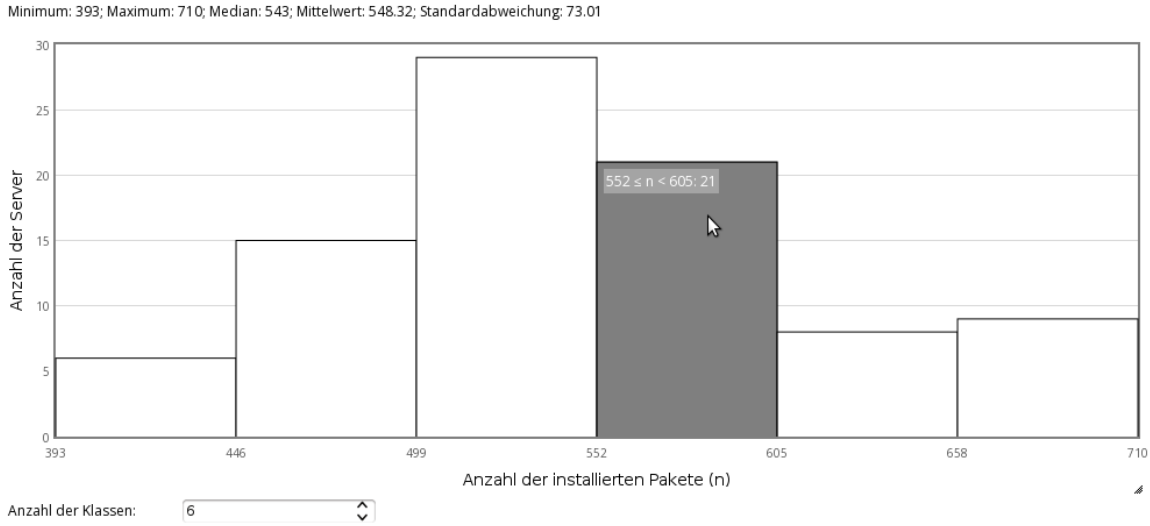


Abbildung 7.45.: Histogramm der Anzahl installierter Pakete auf den Servern.

eine zusätzliche Interaktionsmöglichkeit: Durch die Wahl der Anzahl der Buckets beeinflusst er die Größe des Wertebereichs der einzelnen Histogrammklassen. Die Auswahlmöglichkeit dieser Anzahl ist wichtig, da sie von der Anzahl der gesamten Anzahl der Server abhängt. In einem Produktivsystem beträgt die Anzahl der Server nicht immer exakt 100, sondern kann stark davon abweichen und auch mit der Zeit variieren. Die Aussagekraft eines Histogramms ist nur dann hoch, wenn die Anzahl der Klassen nicht zu groß gewählt wurde.

In Abbildung 7.46 wurde zu Demonstrationszwecken die Anzahl der Buckets im Vergleich zu 7.45 verdoppelt. Der initiale Wert in dem Eingabefeld stammt aus den persönlichen Einstellungen des Benutzers.

Beim Histogramm wurde auf Farben vollständig verzichtet, da dies auch der üblichen Darstellungsform von Statistikprogrammen wie zum Beispiel R entspricht und die Farben in diesem Anwendungsfall keine zusätzlichen Informationen bereitstellen. Die standardmäßig von der JavaScript-Bibliothek ausgewählten Säulenfarben wurden bei dieser Implementierung wieder entfernt, da sie in keinem Zusammenhang zu den abgebildeten Informationen stehen.

In einer möglichen Weiterentwicklung des Prototyps zu einem Wirksystem ist es denkbar, die einzelnen Säulen des Histogramms mit Event Handlern auszustatten. Durch einen klick darauf würde sich dann eine Übersicht

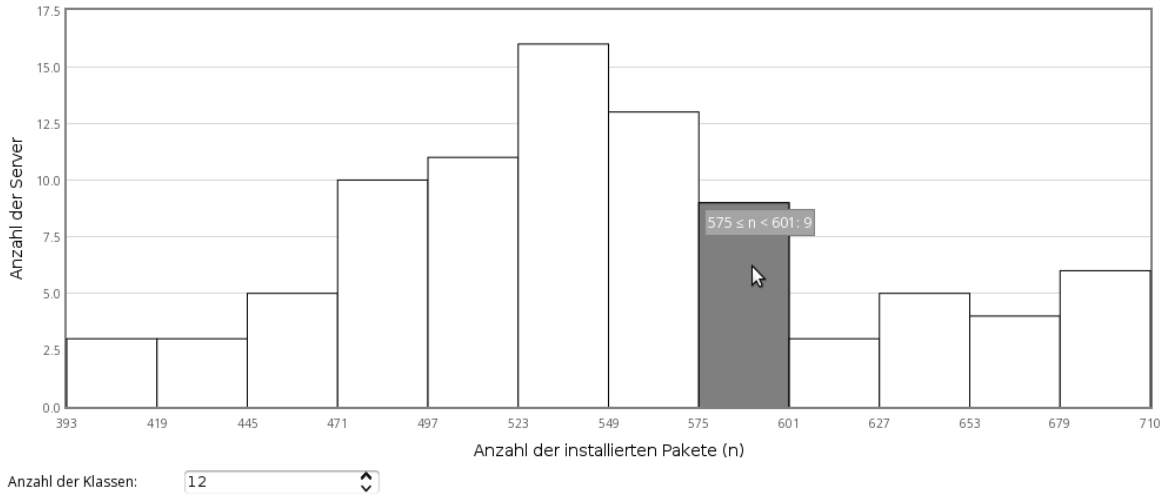


Abbildung 7.46.: Histogramm mit einer höheren Anzahl an Buckets.

der Server öffnen, die aufgrund der Anzahl installierter Pakete in die gewählte Kategorie fallen. Dies ist zum Beispiel nützlich, um Systeme unmittelbar näher betrachten zu können, auf denen verdächtig viel Software installiert ist.

7.4.3. Interaktive zoombare Visualisierung von Serverzuständen

Im Abschnitt 6.2.6 wurde eine *zoombare Gesamtübersicht mit Filterung und Sortierung* aller verfügbaren Server vorgeschlagen. Die zoombare Oberfläche erfüllt den Zweck eines digitalen Sortiertischs: Zunächst werden alle Server in Form kleiner Piktogramme eingeblendet und nach Sortierkriterien, die der Anwender frei wählen kann, entlang der Abszisse und der Ordinate des Koordinatensystems angeordnet. So erhält der Anwender zunächst einen groben Gesamtüberblick über die Zustände der einzelnen Server und kann bereits Outlier erkennen, bei denen Handlungsbedarf existiert.

Durch sukzessives Herausfiltern uninteressanter Systeme bleiben nur noch die Server in der Übersicht übrig, die für den Administrator von Interesse sind. Er kann in die Übersicht hineinzoomen, um die Systeme näher zu betrachten und bei Bedarf Detailinformationen zu bestimmten Systemen abfragen. Dieser Workflow entspricht Ben Shneidermans *Information Seeking Mantra* (siehe Abschnitt 6.1.2).

Um die technische Realisierbarkeit und den praktischen Nutzen dieses Bedienkonzepts zu demonstrieren, wurde im Prototyp eine Implementierung für den Anwendungsfall Software-Verwaltung (siehe Abschnitt 6.1) integriert. Abrufbar ist die Implementierung unter dem Software-Menüpunkt *Pan & Zoom*.

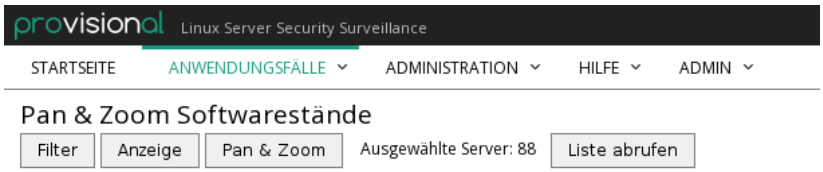


Abbildung 7.47.: Schaltflächen zum Aufruf der Steuerdialoge in der Pan&Zoom-Ansicht.

Unmittelbar nach dem Aufrufen dieser Webseite ist ist eine leere, rechteckige Bildfläche mit einigen Schaltflächen darüber zu sehen. Die rechteckige Bildfläche dient beim Demonstrator als Zeichenfläche für die dynamisch generierte zoombare Visualisierung der Server. Im Ausgangszustand handelt es sich dabei lediglich um einen `<div>`-Container, der per CSS mit einem grauen Rahmen versehen wurde.

Standardmäßig ist die Größe des Containers auf einen Wert von 800×800 Pixeln eingestellt. Mit Hilfe des Anfasspunkts rechts unten können Anwender die Zeichenfläche beliebig vergrößern und verkleinern und das Seitenverhältnis an ihre Bedürfnisse anpassen.

Die Schaltflächen über der Zeichenfläche (siehe Abbildung 7.47) öffnen jeweils eigene nichtmodale Dialogfenster, die dem Anwender folgende Möglichkeiten bieten, um die Visualisierung zu beeinflussen:

Filter Filterung der angezeigten Server nach unterschiedlichen Kriterien

Anzeige Beeinflussung des Ausgabeformats

Pan & Zoom Zoom und Bewegung innerhalb der Visualisierung

Liste abrufen Tabellarische Auflistung der angezeigten Server

Nachdem die Webseite aufgerufen wurde, ruft sie automatisch per AJAX Zustandsinformationen aller aktiven Linux-Server von der Server-Komponente des Prototyps ab, die dem Benutzer zugeordnet wurden. Es erfolgt aber noch keine Visualisierung, da sich der Anwender zunächst für die Sortierkriterien von Abszisse und Ordinate entscheiden muss. Dies geschieht über das Dialogfenster *Anzeige*.

Das Anzeige-Fenster ist in zwei Tabs unterteilt. Auf dem Tab mit der Bezeichnung *Anordnung* beeinflussen die beiden Dropdown-Menüs *X-Achse anordnen nach* und *Y-Achse anordnen nach* die räumliche Anordnung der Server in der Zeichenfläche. Standardmäßig sind die beiden Felder leer. Die folgenden Sortierkriterien sind für beide Dropdownlisten im Demonstrator implementiert:

- Server-Priorität
- Anzahl installierter Pakete
- Letztes Update

Sobald der Anwender in beiden Dropdownlisten eine Auswahl getroffen hat, wird die erste Visualisierung basierend auf der ungefilterten Menge aller aktiven Server gezeichnet. Falls der Anwender sowohl für die X-Achse als auch für die Y-Achse dasselbe Sortierkriterium auswählt, wird lediglich ein Testbild ausgegeben, da eine Visualisierung dieser Kombinationen nicht viel Sinn ergibt. Alle Server würden dann dicht aneinandergedrängt auf einer Ursprungsgeraden angeordnet werden während der Großteil der Zeichenfläche ungenutzt bliebe.

In Abbildung 7.48 ist eine erste Visualisierung zu sehen. Als Sortierkriterien wurden hier die Server-Priorität bei der Abszisse und die Anzahl installierter Pakete bei der Ordinate ausgewählt. Der Einfachheit halber und um wenig Platz zu beanspruchen werden die einzelnen Server als kleine Kreise dargestellt. Je höher die Priorität eines Servers ist, die vom Administrator vergeben wurde, desto weiter rechts wird er in der Visualisierung eingeblendet. Je mehr Pakete auf ihm installiert sind, desto weiter oben ist er in der Zeichenfläche anzutreffen. Aufgrund dieser Sortierung lassen sich bereits evtl. vorhandene Outlier feststellen.

Die Skala für die Server-Priorität beginnt bei 0 und endet bei 10. Da es bei der Anzahl der installierten Pakete einerseits keine Obergrenze gibt und andererseits keine Bildschirmfläche verschwendet werden soll, beginnt die Skala bei dem Wert, der als Minimum aller Server ermittelt wurde. Analog dazu endet die Skala beim Maximum. Die zur Verfügung stehende Zeichenfläche wird dadurch optimal ausgenutzt.

Im Tab *Anordnung* des Anzeige-Fensters befindet sich neben den Dropdownlisten jeweils eine Checkbox mit der Beschriftung *Reihenfolge umkehren*. Sobald eine dieser Checkboxen aktiviert wurde, erfolgt dementsprechend eine Spiegelung der Visualisierung in X- oder Y-Richtung. Die Kombination beider Spiegelungen ist ebenfalls möglich. Sobald der Anwender eines der Felder im Anzeige-Dialog per Mausklick ändert, wirkt sich dies unmittelbar auf die Visualisierung aus, die in diesem Fall neu gezeichnet wird.

Die in Abbildung 7.48 gezeigte Visualisierung ist noch nicht besonders aussagekräftig, da abgesehen von der Position der Server keine weiteren visuellen Attribute zum Einsatz kommen. Dies lässt sich ändern, indem eine weitere Server-Kennzahl auf den Radius der einzelnen Piktogramme abgebildet wird. Dies geschieht ebenfalls im Darstellungs-Tab des Anzeige-Fensters.

In Abbildung 7.49 wurde das verbleibende der drei implementierten Sortierkriterien auf den Radius übertragen. Standardmäßig bildet das Programm kleinere Werte auf näher am Ursprung gelegene Positionen oder auf kleinere Radien in der Visualisierung ab. Beim Sortierkriterium *Letztes Update* greift der Prototyp auf einen

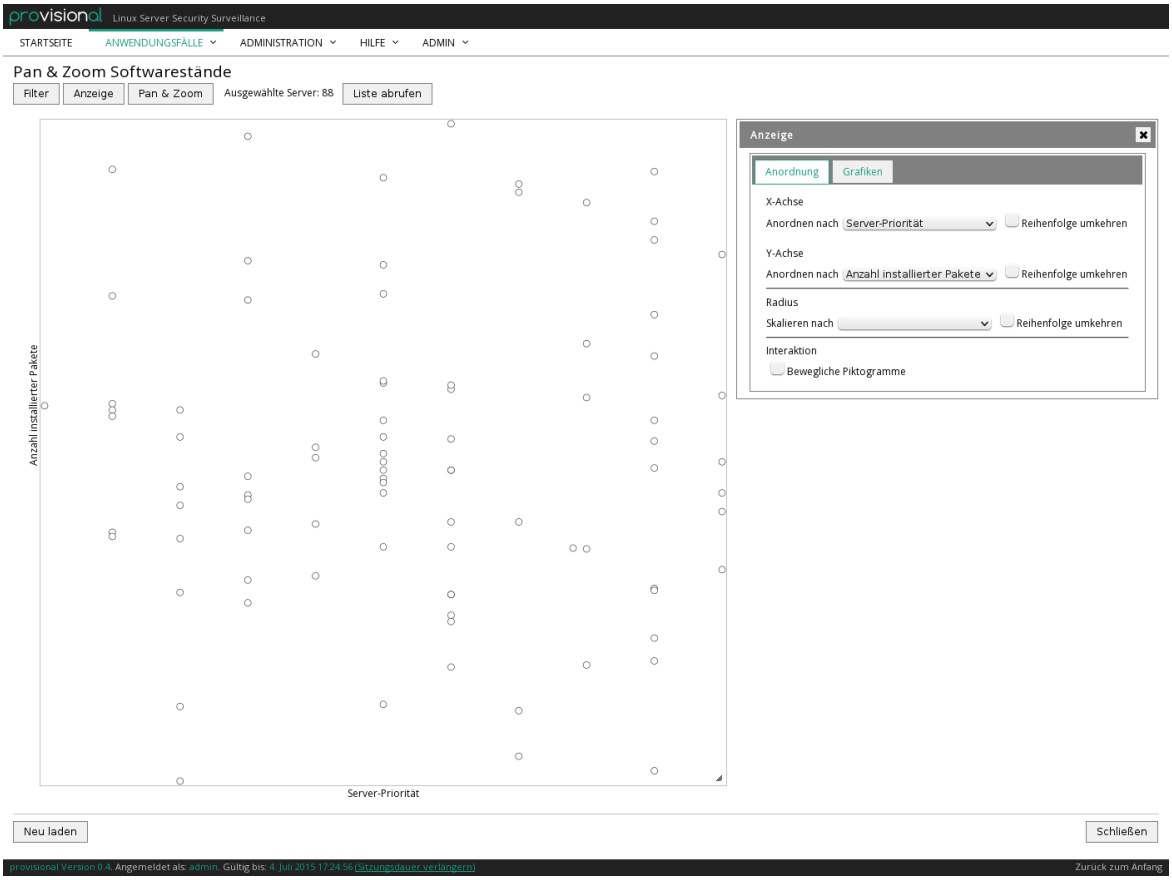


Abbildung 7.48.: Visualisierung der Server nach Priorität und Anzahl installierter Pakete geordnet.

Zeitstempel zurück. Länger zurückliegende Updates würden somit zu Piktogrammen mit einem kleineren Radius führen als kürzlich erfolgte Updates. Da Server mit einem alten Updatestand potenziell sicherheitsgefährdeter sind, ist es wünschenswert, dass sie möglichst groß dargestellt werden, um sofort ins Auge zu stechen. Aus diesem Grund ist beim Radius die Checkbox *Reihenfolge umkehren* aktiviert.

Im Beispielbild in Abbildung 7.49 wird ein weiteres Problem sichtbar, das bei Hinzunahme des Radius als visuelles Attribut stärker in Erscheinung tritt, nämlich Überlappungen. In Abhängigkeit von der Reihenfolge, in der die einzelnen Serverpiktogramme eingezeichnet werden und je nach Radius können einzelne Symbole auch vollständig überdeckt werden und sind dann nicht mehr sichtbar. Sicherheitsrelevante Informationen gehen auf diese Weise verloren.

Um diesem Effekt entgegenzuwirken, ermöglicht der Demonstrator die Aktivierung transparenter Objekte. Dies geschieht im zweiten Tab des Darstellungs-Dialogs, der den Titel *Grafiken trägt*. Durch die Aktivierung der Checkbox mit der Bezeichnung *Transparenz* werden die Server-Piktogramme durchsichtig und verdeckte Server werden sichtbar. Die Auswirkungen auf die Visualisierung sind im Linken Teilbild von Abbildung 7.50 zu sehen.

In Abschnitt 6.1.2 wurden die Signalfarben Grün, Gelb und Rot zur Kennzeichnung des allgemeinen Patch-Zustands eines Servers vorgeschlagen: Die Farbe Grün symbolisiert Systeme, die auf dem aktuellen Stand sind, die Farbe Gelb steht für Server, die Updates benötigen und rote Server besitzen Pakete mit kritischen Sicherheitslücken. Durch die präattentive Wahrnehmung von Signalfarben wirken sie sich enorm auf die Aussagekraft von Visualisierungen aus. Aus diesem Grund wurde auch im Demonstrator eine Option zum Aktivieren dieser Signalfarben implementiert.

Die Hinzunahme von Signalfarben geschieht im Grafiken-Tab des Anzeige-Fensters über die Checkbox mit der Bezeichnung *Signalfarben anzeigen*. Die Auswirkung dieser Option ist im rechten Teilbild von Abbildung 7.50

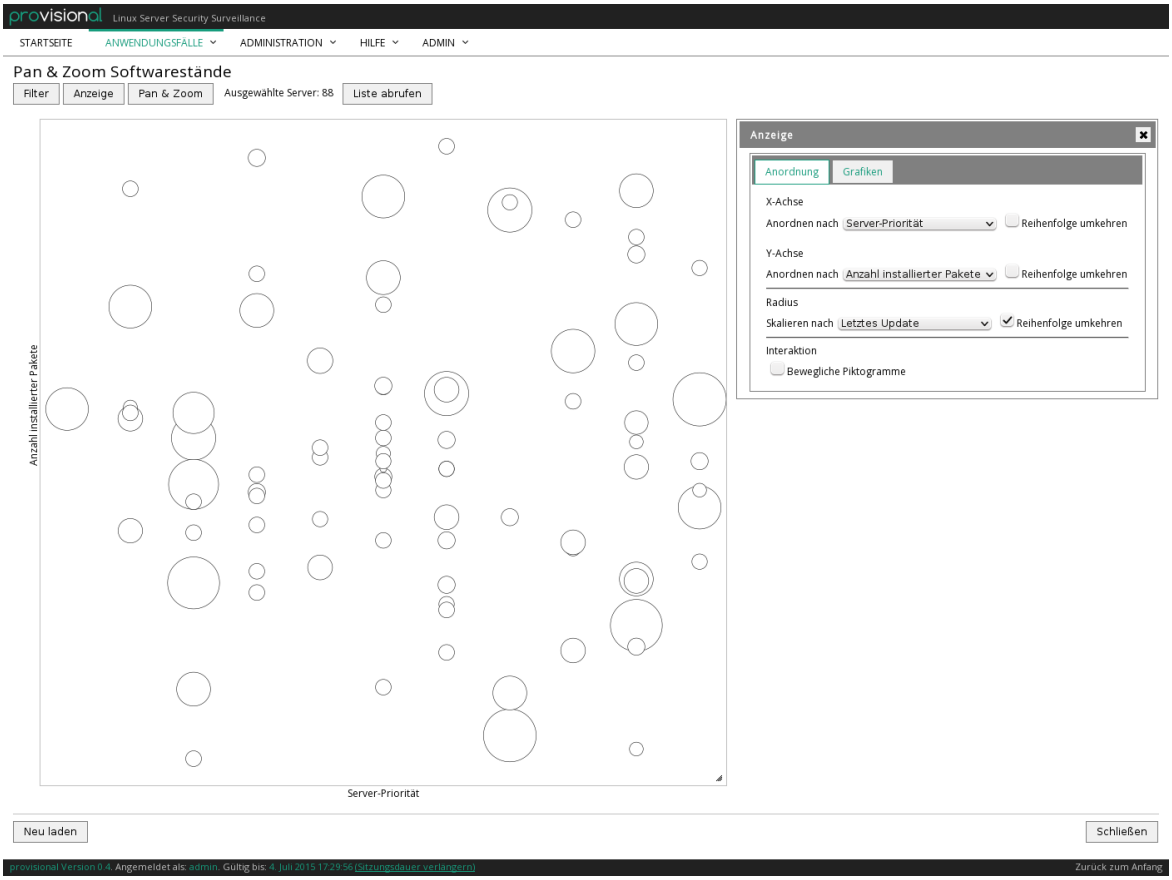


Abbildung 7.49.: Visualisierung der Server nach Priorität und Anzahl installierter Pakete geordnet mit zusätzlicher Abbildung des Update-Zeitpunkts auf den Radius.

sichtbar. Durch die gegenüberliegende Anordnung dieser beiden Teilbilder fällt bei einem Vergleich sofort der deutliche Informationsgewinn auf, da es eine Handvoll Server mit sicherheitskritischem Patch-Zustand gibt. Ohne die Signalfarben wäre eine sequenzielle Suche notwendig, die sehr zeitraubend ist.

Wie der Visualisierung unmittelbar zu entnehmen ist, besitzen drei dieser besonders anfälligen Server die maximale Priorität, sie sind also für das Tagesgeschäft unentbehrlich. Am Radius ist zu erkennen, dass ihre letzte Aktualisierung im Mittel längere Zeit zurückliegt.[38] Hier herrscht scheinbar dringender Handlungsbedarf. Möglicherweise ist der Aufwand eines Updates aber nicht allzu hoch, da laut der Visualisierung alle der als sicherheitskritisch bewerteten Server bei der Anzahl installierter Pakete maximal im Mittelfeld liegen.

Obwohl im diesem Beispiel Signalfarben zur Verbesserung der Übersichtlichkeit aktiviert wurden, enthält die Visualisierung immer noch zwei Piktogramme ohne Füllfarbe. Es handelt sich dabei um Server, für die aktuell keine Bewertung des Patch-Zustands verfügbar ist, da es sich hierbei um einen aggregierten Wert handelt, der von der Serverkomponente noch nicht ausgerechnet wurde. Möglicherweise liegt dem Informationssystem für diese beiden Server auch keine aktuelle Referenzpaketliste vor, sodass ein Versionsvergleich überhaupt nicht möglich ist. Auch in diesem Fall liefert die Visualisierung einen Anhaltspunkt für die Bewertung des Patch-Zustands durch den Betrachter, da immerhin die Anzahl installierter Pakete und das Alter des letzten Updates aus der Visualisierung hervorgehen.

In den bisherigen Beispielbildern war immer eine Totalaufnahme aller verfügbarer Server zu sehen. Aufgrund möglicher Überlappungen und der Winzigkeit einiger Piktogramme ist es wünschenswert, in die Visualisie-

[38] Es gibt auch einige Server, die sich im grünen Bereich befinden, obwohl längere Zeit kein Update mehr eingespielt wurde, wie am großen Radius abzulesen ist. Ein Widerspruch besteht hier nicht, da es sich um Server mit Oldstable-Distributionen handeln könnte, deren Pakete seltener aktualisiert werden. Es ist auch denkbar, dass dort andere Programme als auf den Servern mit kritischem Software-Zustand installiert sind, für die momentan keine kritischen Lücken bekannt sind.

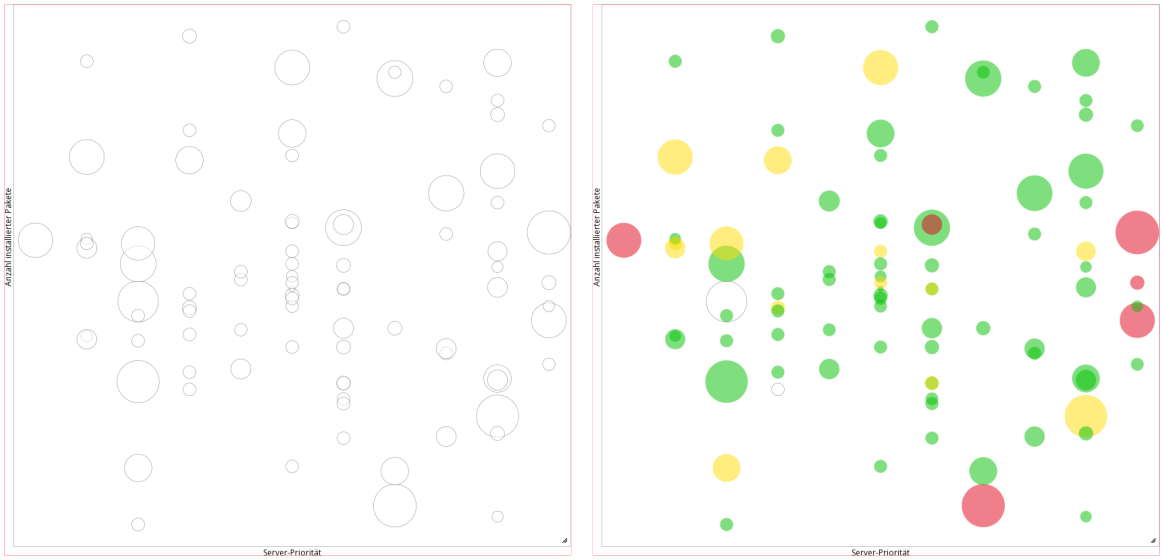


Abbildung 7.50.: Sichtbarmachen verdeckter Piktogramme durch Transparenz (links) und Hinzunahme von Signalfarben (rechts).

rung hineinzoomen zu können. Einzelne Server sind dann besser voneinander unterscheidbar. Dies erleichtert den Prozess, von den richtigen Systemen im Anschluss Detailinformationen abzurufen, ohne versehentlich daneben zu klicken.

Um einen Teil der Visualisierung zu vergrößern, bewegt der Anwender am einfachsten den Mauszeiger zu dem Objekt, das er näher betrachten will und dreht das Scrollrad der Maus nach oben. Umgekehrt kann er auch wieder hinauszoomen. Durch Klickziehen in einem beliebigen Bereich innerhalb der Visualisierung ist eine Verschiebung des Bildausschnitts möglich. Damit kann der Anwender Server, die von Interesse sind, ins Blickfeld rücken, ohne zuerst hinaus- und dann wieder hineinzoomen zu müssen. Da der Vergrößerungsfaktor bei dieser sog. Pan-Operation konstant bleibt, gilt das auch für den Maßstab der Koordinatenachsen und den Radius. Einzelne Server sind so einfacher zu bewerten und miteinander zu vergleichen als bei wiederholtem Hinaus- und Hineinzoomen.

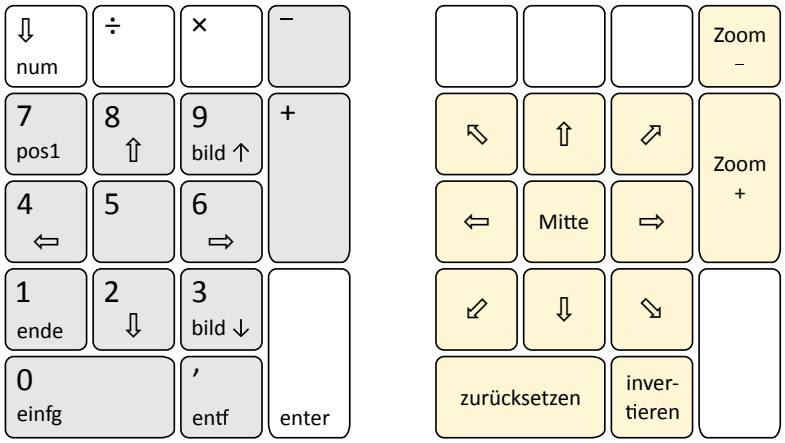


Abbildung 7.51.: Numerischer Ziffernblock einer Tastatur (links) und Tastenbelegung zur Navigation (rechts).

Alternativ zur direkten Interaktion mit der Visualisierung mit Hilfe der Maus stellt der Demonstrator auch ein Dialogfenster bereit, das zum Zoomen und zum Bewegen des Ausschnittfensters dient. Es ist über die Schaltfläche *Pan & Zoom* abrufbar. Die Schaltflächen in diesem Fenster sind aufgrund ihrer Symbole intuitiv bedienbar, zudem gibt es Tooltip-Texte bei jeder Schaltfläche. Eine Legende ist somit überflüssig. Die Check-

box mit der Beschriftung *Invertieren* bewirkt, dass nicht das Betrachtungsfenster, sondern der Bildinhalt in Pfeilrichtung bewegt wird. Der Anwender kann so die Bewegungsrichtung auswählen, die ihm mehr zusagt. Wer es umständlich findet, die Schaltflächen in dem Dialogfenster zur Navigation zu nutzen, kann zu diesem Zweck auch auf den numerischen Ziffernblock der Tastatur zurückgreifen. Die dazugehörige Tastenbelegung ist in Abbildung 7.51 dargestellt. Voraussetzung zur Nutzung der Tastatur ist, dass zuvor das Fenster geöffnet wurde.

Neben den vier Hauptbewegungsrichtungen stehen auch Schaltflächen zum Diagonalen Bewegen des Bildausschnitts bereit. Am einfachsten ist eine diagonale Bewegung als Kombination einer senkrechten und einer waagerechten Bewegung realisierbar. Dies geschieht durch die Addition der beiden Bewegungsvektoren. Dieser diagonale Vektor ist (außer in Grenzfällen) länger als einer der beiden Ausgangsvektoren. Damit sich die Bewegungen über den Ziffernblock oder das Fenster möglichst natürlich anfühlen, wurden die diagonalen Bewegungsvektoren normalisiert, damit sie dieselbe Länge besitzen wie die senkrechten und waagerechten.

Abbildung 7.52 zeigt eine Folge von Bildern, die während der Vergrößerung eines Bildausschnitts aufgenommen wurden. Das Navigationsfenster ist jeweils oben links eingeblendet.

Abbildung 7.52.: Schrittweises Hereinzoomen in die Visualisierung.

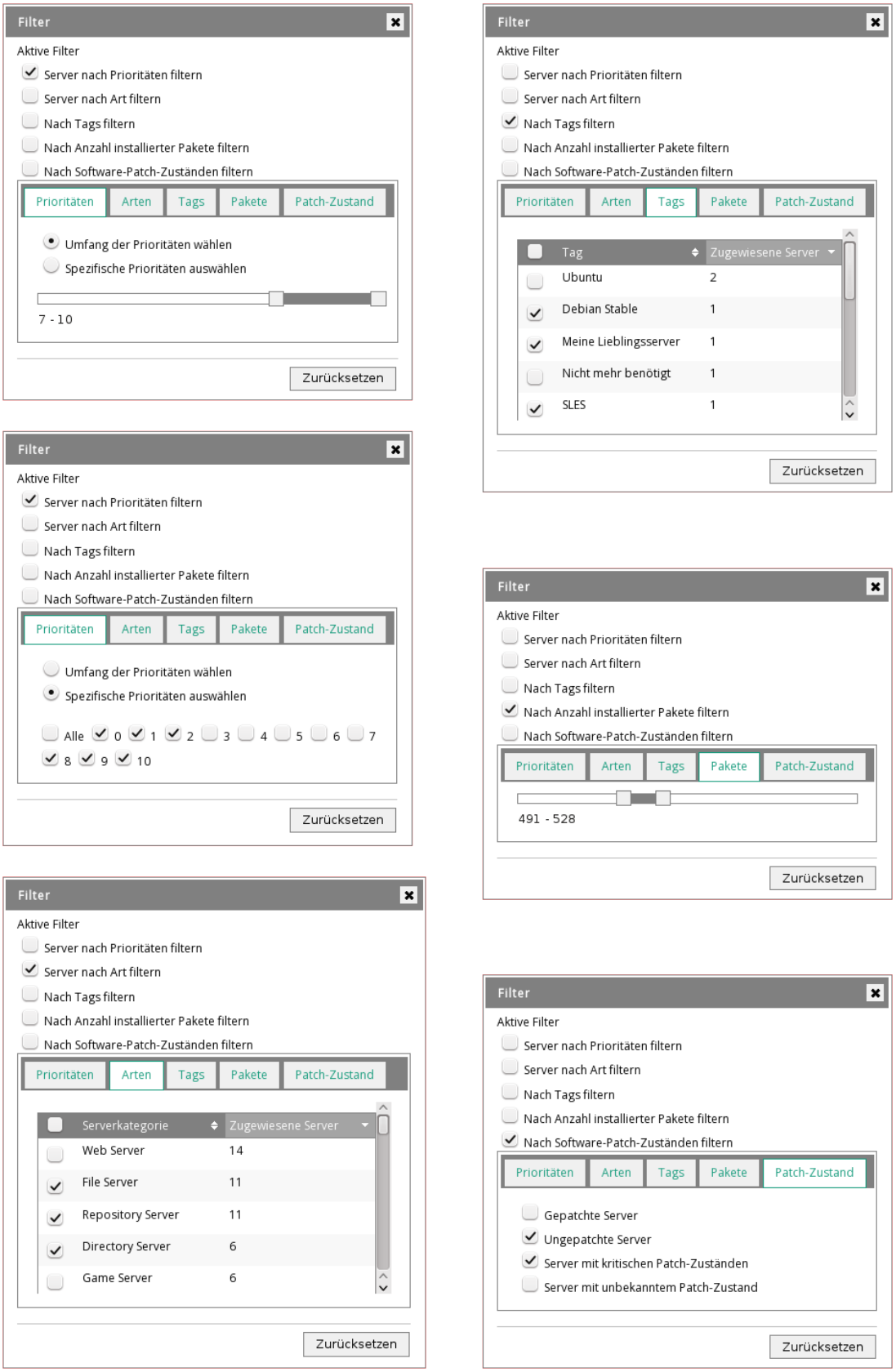


Abbildung 7.53.: Die eingeblendeten Server können anhand verschiedener Kategorien gefiltert werden.

Sobald der Anwender das Objekt von Interesse ausreichend vergrößert hat, kann er Detailinformationen zum entsprechenden Server abfragen. Dies geschieht im einfachsten Fall, indem er mit der Maus darauf zeigt und den Hostnamen des Servers in einem Tooltip-Text erhält. Durch einen Mausklick öffnet sich ein Fenster mit dem Titel *Server-Details*, das deutlich mehr Informationen bereitstellt. Bei der prototypischen Implementierung sind zu Demonstrationszwecken die folgenden Werte vorhanden:

- Hostname
- ID des Servers in der Datenbank
- Server-Kategorie
- Beschreibung
- Priorität
- Vom Anwender zugewiesene Tags
- Anzahl installierter Pakete
- Software-Status (kategorisch)
- Letztes Update

Dieses Detail-Fenster kann an einen freien Bereich des Bildschirms geschoben werden, um die Visualisierung nicht zu verdecken. Sobald der Anwender auf einen anderen Server klickt, zeigt dasselbe Fenster die Informationen des neuen Servers an. Das erste Teilbild in Abbildung 7.54 zeigt, wie der Anwender Zustandsinformationen vom Server mit dem Hostnamen `trall` abfragt.

Wie bereits zuvor erwähnt wurde, kann es vorkommen, dass Piktogramme von anderen überlagert werden. Durch die Aktivierung der Objekttransparenz werden diese Objekte zwar sichtbar, den dazugehörigen Hostnamen und die detaillierten Zustandsinformationen kann man aber nur dann abfragen, wenn zwischen ihnen und dem Mauszeiger kein anderes Objekt ist.

Für dieses grundlegende Problem wurde in Abschnitt 6.1.2 bereits vorgeschlagen, die Überlappung von Objekten durch einen Point-Displacement-Algorithmus zu lösen. In der prototypischen Implementierung wird dieses Problem pragmatischer gelöst: Die Piktogramme lassen sich bei Bedarf verschieben, um darunterliegende freizulegen.

Um die Server-Piktogramme per Klickziehen verschiebbar zu machen, genügt es, im Anordnungs-Tab des Anzeige-Fensters die Checkbox mit der Bezeichnung *Bewegliche Piktogramme* am unteren Rand des Fensters zu aktivieren. Die Interaktion mit der Visualisierung funktioniert dann weiterhin wie bisher beschrieben wurde, beim Klickziehen auf einem Piktogramm verschiebt sich dann aber nicht mehr der Bildausschnitt, sondern das Piktogramm selbst. Der Anwender muss sich darüber bewusst sein, dass die Information verschobener Piktogramme aus der Position dann nicht mehr stimmt. Sobald er aber irgendeine Anzeigeeinstellung ändert oder die Seite neu lädt, wird das Bild wieder neu gezeichnet und die Objektpositionen sind wieder korrekt.

Ab dem rechten oberen Teilbild in Abbildung 7.54 ist eine Verschiebesequenz dargestellt, die ein kleines grünes Server-Piktogramm freilegt: Zuerst wird das störende gelbe Piktogramm nach links verschoben, dann das grüne nach rechts. Im letzten Teilbild kann der Anwender schließlich auf den freigelegten Server mit dem Hostnamen `upolu` klicken, um seine Detail-Informationen abzurufen.

Der Vollständigkeit halber wird an dieser Stelle noch ein Beispiel mit der dritten Kombinationsmöglichkeit für die visuellen Attribute gezeigt. In Abbildung 7.55 sind zwei Visualisierungen zu sehen, die das Alter des letzten Updates entlang der X-Achse und Sie Server-Priorität entlang der Y-Achse anordnen. Der Radius wächst mit der Zahl der installierten Pakete.

Beide Teilbilder zeigen dieselbe Totalansicht der Server. Während im linken Teilbild wie gewohnt eine weiße Hintergrundfarbe verwendet wird, ist sie im rechten Teilbild schwarz. Die Option zur Verdunkelung des Hintergrunds wurde implementiert, um den Demonstrator auch gut in dunklen Umgebungen einsetzen zu können. Dort würde der weiße Hintergrund den Betrachter zu sehr blenden. Umgeschaltet wird zwischen den beiden Zuständen im Grafiken-Tab des Anzeige-Fensters mit Hilfe der Checkbox *Dunkler Hintergrund*.

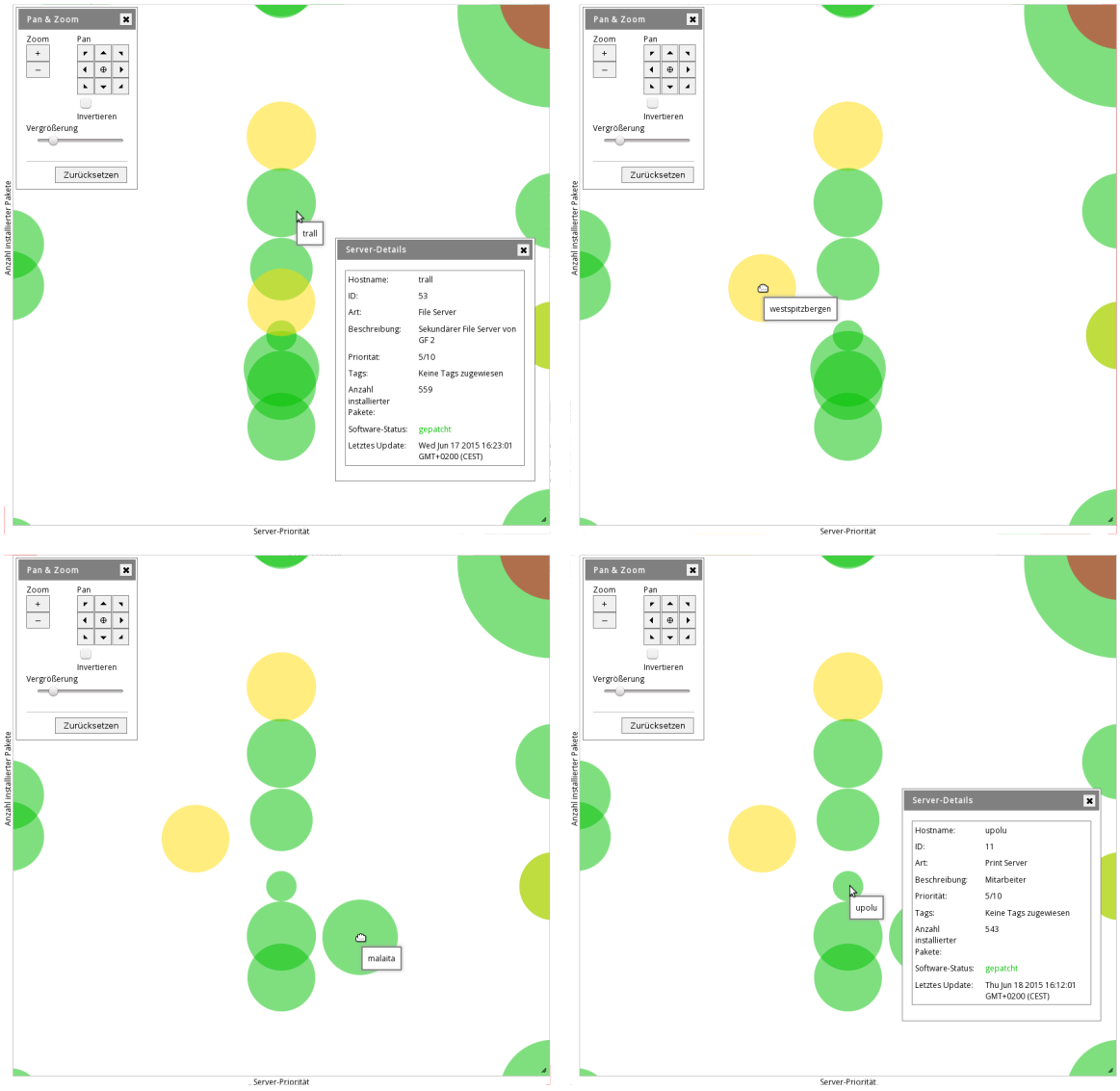


Abbildung 7.54.: Oben links: Abrufen von Details des Servers trall. Oben rechts, unten links, unten rechts: Freilegen des verdeckten Servers upolu und Abrufen seiner Details.

Durch das Zoomen und Verschieben erhält der Anwender gezielten Zugriff auf Informationen bestimmter Server. Signalfarben helfen ihm beim Unterscheiden sicherheitsrelevanter Zustände. Um ihn noch weiter zu unterstützen, verfügt der Demonstrator über verschiedene Filtermechanismen. Durch die sukzessive Hintereinanderschaltung von Filtern verringert sich die Anzahl der eingeblendeten Server.

Die Aktivierung von Filtern erfolgt über das Dialogfenster *Filter*, das über die gleichnamige Schaltfläche geöffnet werden kann. Abbildung 7.53 zeigt zu jedem Filter die dazugehörigen Einstellungsmöglichkeiten. Folgende Filter stehen dem Anwender zur Auswahl bereit:

- Priorität
- Serverkategorie
- Vom Benutzer festgelegte Tags
- Anzahl installierter Pakete
- Software-Patch-Zustand

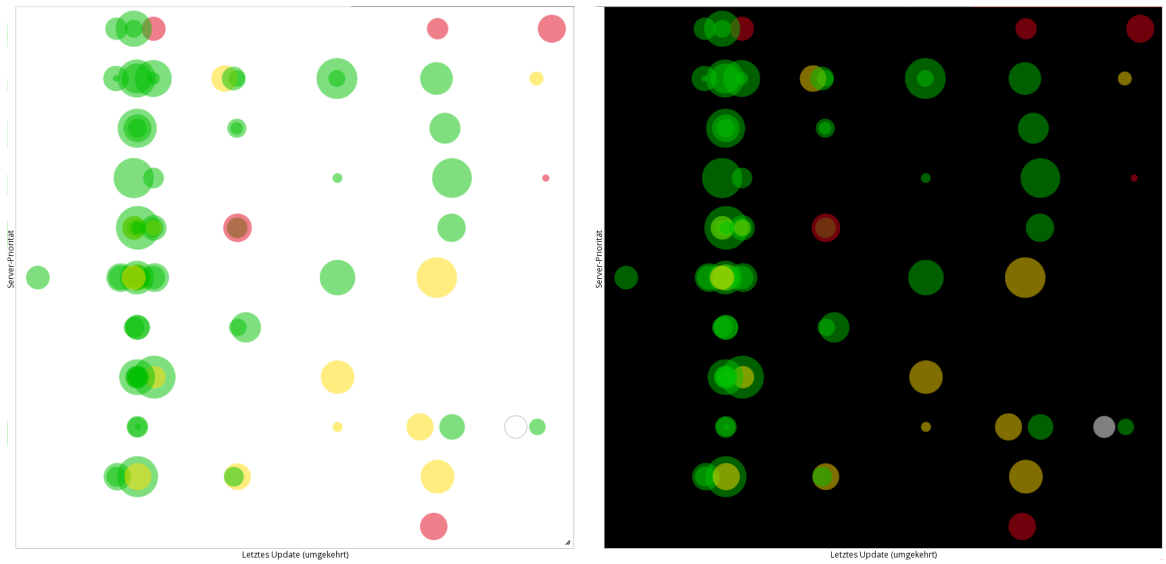


Abbildung 7.55.: Visualisierung der Server mit weißem (links) und schwarzem (rechts) Hintergrund.

Jeder dieser einzelnen Filter ist nach Belieben ein- und ausschaltbar. Die vom Anwender aktivierten Filter werden konjunktiv miteinander verknüpft: Jeder zusätzliche Filter schränkt die Anzahl der restlichen Server in der Visualisierung weiter ein oder lässt sie zumindest gleich. Jeder Filter verfügt über Konfigurationsmöglichkeiten in einem eigenen Tab unterhalb der Filter-Checkboxen. Um Bildschirmfläche zu sparen, werden die Einstellungsmöglichkeiten der einzelnen Filter nur dann eingeblendet, sobald mindestens ein Filter per Checkbox aktiviert wurde. Dadurch bleibt mehr Platz für die eigentliche Visualisierung.

Die Filterfunktion wurde mit Hilfe von Dynamic Queries (siehe Abschnitt 2.14.3) realisiert. Die grafischen Bedienfelder werden mit der Maus bedient, die syntaktische Korrektheit der Abfragen ist gewährleistet und Filteränderungen wirken sich unmittelbar nach jedem Mausklick auf die in der Visualisierung eingeblendeten Server aus, sofern der aktuelle Filter per Checkbox aktiviert wurde. Dies ermöglicht eine intuitive Bedienung, die den Anwender sogar dazu motiviert, die für ihn jeweils relevanten Filterkonfigurationen zu ermitteln.

Bei der Filterung der Server anhand ihrer Priorität hat der Anwender die Wahl zwischen der Angabe eines Wertebereichs oder der Auswahl beliebiger konkreter Werte. Die Wahl trifft er über die beiden Optionsfelder *Umfang der Prioritäten wählen* bzw. *Spezifische Prioritäten auswählen* im Filter-Tab *Prioritäten*. Entscheidet er sich für die zuerst genannte Option, wird ein mit Hilfe von jQuery UI generierter Bereichs-Slider eingeblendet, der über je einen Anfasspunkt für den minimalen und den maximalen enthaltenen Wert des Bereichs verfügt. Im zweiten Fall blendet die Anwendung Checkboxen ein, um jede erwünschte Priorität einzeln wählen zu können.

Der zweite Filter-Tab listet die verfügbaren Server-Kategorien auf. Wie die Tabellen, die bereits zuvor in diesem Kapitel auftauchten, ist auch diese Tabelle sortierbar. Die Sortierung nach der Anzahl zugewiesener Server erweist sich als sinnvoll, um die bedeutendsten Kategorien sofort im Blick zu haben. Einzelne Kategorien werden durch Aktivierung der dazugehörigen Checkbox ausgewählt. Die Checkbox in der Kopfzeile der Tabelle selektiert bzw. deselektiert alle Server-Kategorien auf einmal.

Ähnlich wie die Filterung nach Server-Kategorien erfolgt die Filterung der Server anhand benutzerdefinierter Tags in dem Filter-Tab *Tags*. Auch dort ist eine Tabelle enthalten. Die Konfiguration erfolgt analog zum zuvor genannten Filter.

Der Anwender kann die angezeigten Server anhand eines Bereichs einschränken, in dem sich die Anzahl der installierten Pakete befinden muss. Dies erfolgt in dem Filter-Tab mit dem Titel *Pakete* anhand eines Bereichsauswahl-Sliders. Bei der Implementierung im Demonstrator ist es empfehlenswert, den Bereich lediglich durch einfaches Klicken auf die gewünschten Bereiche des Sliders festzulegen und vom Klickziehen der Anfasspunkte abzusehen. Beim Klickziehen sendet der Client bei jedem Pixel, um das sich ein Anfasspunkt bewegt einen AJAX Request an den Server, der Server ermittelt die Zustandsinformationen der ausgewählten

Abbildung 7.56.: Sukzessive Filterung der Server am Beispiel uninteressanter Serverkategorien.

Linux-Server und sendet sie zum Client zurück, der sie daraufhin visualisiert. So entstehen sehr viele Requests innerhalb kurzer Zeit und die Visualisierung wird sehr häufig neu gezeichnet. Dies führt zu wahrnehmbaren Verzögerungen. Um dies in einem möglichen Produktivsystem zu verhindern, sollten die Requests nur beim Auslassen der Maustaste an den Server gesendet werden.

Als letzte Filtermöglichkeit im Demonstrator stehen die Patch-Zustände bereit. Damit ist es möglich, nach konkreten Signalfarben zu filtern. Dies ist nützlich, um von Anfang an nur Systeme mit bekanntem kritischem Software-Stand anzeigen zu lassen und um reaktiv tätig zu werden. Zur Anbahnung proaktiver Tätigkeiten macht es aber auch Sinn, die anderen Farben auch auszuwählen.

In Abbildung 7.56 ist eine Bildfolge zu sehen, in der die Anzahl der angezeigten Server durch Filterung sukzessive verringert wird. Die Sortierung und die visuellen Attribute sind die gleichen wie im vorherigen Beispiel.

Ein weiteres Beispiel für sukzessive Filterung ist in Abbildung 7.57 dargestellt. Zur Abwechslung wurden hier zunächst die Sortierkriterien geändert: Die X-Achse dient zur Abbildung des letzten Update-Zeitpunkts, wobei durch die Umkehrung der Skala die Server mit dem ältesten Update-Zeitpunkt rechts im Bild zu sehen sind. Die Y-Achse steht für die Anzahl installierter Pakete und der Radius für die Priorität.

In dem ersten Teilbild oben links überlagern sich die Server-Piktogramme stark, da scheinbar viele Systeme an einem gemeinsamen Patchday aktualisiert wurden. Da viele dieser Systeme im grünen Bereich sind, werden diese ausgeblendet, um die übrigen Server besser sichtbar zu machen. Dies führt zu der deutlich übersichtlicheren Darstellung im Teilbild oben rechts. Durch einen weiteren Filterschritt wurden unten links auch die gelben Systeme aus der Visualisierung entfernt. Das letzte Teilbild zeigt nur noch die kritischen Systeme an. Dort werden schließlich die Informationen für den Server `panay` abgerufen, da sein letztes Update lange Zeit zurückliegt und er über die maximale Priorität verfügt. Unter den übrigen Systemen besitzt er auch eine relativ hohe Anzahl installierter Pakete. Der Aufwand des Patchens ist im Vergleich zu den anderen roten Systemen also potenziell höher. Hier herrscht also dringender Handlungsbedarf und es muss sicherheitshalber etwas mehr Zeit einkalkuliert werden.

Abbildung 7.57.: Sukzessive Filterung der Server am Beispiel des Patch-Zustands und Abruf von Detailinformationen des Servers panay.

Nachdem der Administrator durch sukzessives Filtern nur noch Server in seiner Übersicht sieht, die gerade für ihn von besonderem Interesse ist, kann er wie bereits gezeigt wurde, per Mausklick detaillierte Zustandsinformationen zu den Servern abrufen. Wenn am Ende nur noch Server zu sehen sind, bei denen dringender Handlungsbedarf im Sinne einer Update-Installation besteht, muss der Administrator aber nicht alle Piktogramme in der Übersicht sequenziell anklicken, um sich ihre Hostnamen zu notieren. Um dem Administrator

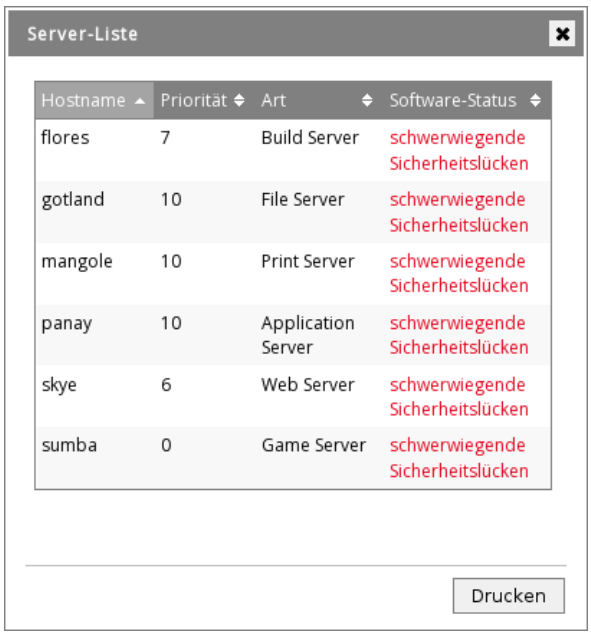


Abbildung 7.58.: Tabellarische Übersicht der Server in der Visualisierung noch vorhandenen Server.

die Arbeit zu erleichtern und Reaktionszeiten zu verkürzen, kann er per Mausklick auf die Schaltfläche mit der Bezeichnung *Liste abrufen* ein Fenster öffnen, das die wichtigsten Informationen zu den Servern, die noch in der Visualisierung sichtbar sind, bereitstellt. Ein Screenshot davon ist in Abbildung 7.58 zu sehen. Bei Bedarf kann der Administrator diese Liste ausdrucken und als Checkliste nutzen, während er den Servern administrativen Beistand leistet.

7.4.4. Paketversionen und Referenzliste

Ursprünglich war vorgesehen, reale Daten von Servern als Grundlage für die Visualisierungen im Demonstrator zu nutzen. Es stellte sich aber im Laufe der Zeit heraus, dass es sehr aufwendig ist, genügend Daten von Servern des LRZ zusammenzutragen und zu importieren. Im Kapitel 6 wurde ein speicherresistentes Programm bzw. ein per Cronjob periodisch ausgeführtes Skript vorgeschlagen, dass die notwendigen Serverdaten für jeden angesprochenen Anwendungsfall beschafft und zum zentralen Informationssystem sendet.

Um die Funktion der Linux-Server am LRZ nicht zu beeinflussen und aus datenschutzrechtlichen Gründen wäre für den Prototyp nur ein separates Testnetz mit einer Vielzahl eigens dafür einzurichtender virtueller Maschinen erforderlich geworden. Da der Schwerpunkt dieser Arbeit auf der Visualisierung sicherheitsrelevanter Serverdaten liegt und nicht auf deren Bereitstellung, greift der Prototyp zunächst nur auf frei erfundene Daten zurück.

Obwohl die Entscheidung getroffen wurde, fiktive Daten als Quelle für die Visualisierungen zu benutzen, wurde bereits damit begonnen, Funktionen im Prototyp zu implementieren, die für den Umgang mit echten Serverzustandsdaten geeignet sind. Die zu diesem Zweck bereits vorhandenen Elemente der Programmoberfläche werden im Anschluss noch kurz vorgestellt.

Für den Anwendungsfall der Softwareverwaltung (siehe Abschnitt 6.1) ist es erforderlich, dass das zentrale Informationssystem über eine Referenzpaketliste verfügt, die regelmäßig – am besten durch einen Automatismus – auf dem aktuellen Stand gehalten wird. Beim Einspielen dieser Referenzpaketliste sollten bereits vorhandene Versionen in der Datenbank nicht überschrieben werden, damit man später zum Beispiel für Audits auf historische Daten zurückgreifen kann.

Der Prototyp ermöglicht es bereits, die Referenzpaketliste eines openSUSE-Systems in die Datenbank hochzuladen. Dies geschieht über den Punkt *Paketliste* im Software-Untermenü bei den Anwendungsfällen. Ein

Screenshot vom aktuellen Entwicklungsstand ist in Abbildung 7.59 zu sehen.

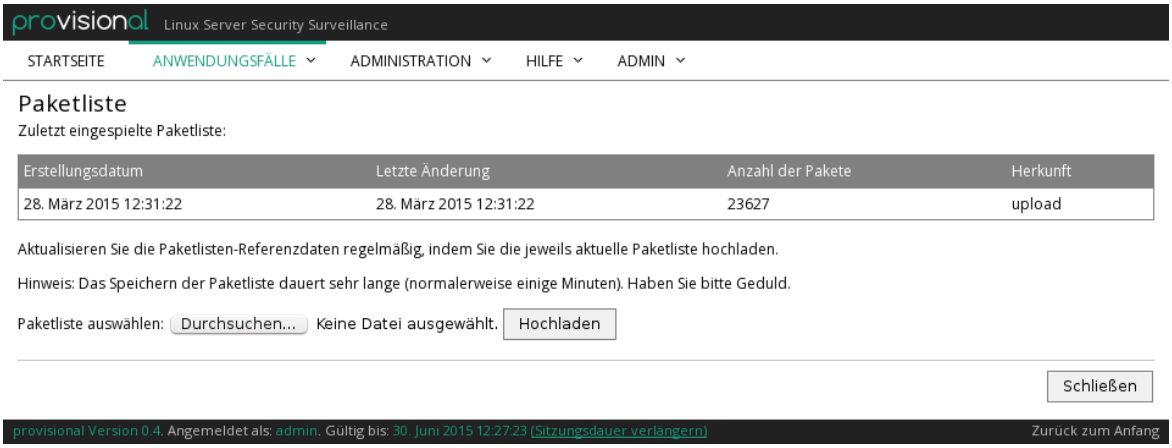


Abbildung 7.59.: Formular zum Importieren der Referenzpaketliste.

Das Bildschirmfoto zeigt eine Tabelle mit Informationen über die zuletzt hochgeladene Referenzpaketliste. Im abgebildeten Beispiel entspricht das Erstellungsdatum auch dem letzten Änderungsdatum, da sich nur eine Version in der Datenbank befindet. Diese Informationen dienen dazu, Angaben zu den zu diesem Zeitpunkt aktuellen Paketversionen zu erhalten. Diese werden zum Versionsvergleich mit den tatsächlich auf einem Server installierten Paketversionen benötigt, um die Aktualität der installierten Software bewerten zu können.

Die Paketliste in dem Beispielfoto umfasst Informationen über 23.627 Pakete. Der Wert *upload* in der Spalte mit der Bezeichnung *Herkunft* offenbart, dass diese Paketliste über die Bedienoberfläche des Demonstrators hochgeladen wurde. Zum Upload einer neuen Referenzdatei besitzt die Webseite unterhalb der Tabelle ein kleines Upload-Formular.[39]

Das Dateiformat, das die derzeitige Implementierung akzeptiert, entspricht der Bildschirmausgabe des Kommandozeilenbefehls `zypper packages`. Dieser Befehl listet die dem System bekannten Paketversionen auf. Da die Bildschirmausgabe verschiedene Versionen eines Pakets beinhaltet, ermittelt der Prototyp vor dem Import die jeweils höchste Versionsnummer.

Nach dem Importieren der Referenzpaketliste kennt der Prototyp die jeweils aktuellsten Paketversionen vom openSUSE-Repository. Eine Übersicht aller dem Prototyp bekannten Paketversionen inklusive der zuletzt importierten erhalten Anwender über den Punkt *Paketversionen* im Software-Untermenü der Anwendungsfälle. Abbildung 7.60 zeigt ein Bildschirmfoto davon.

Die Paketversionen sind in einer paginierbaren Tabelle aufgelistet. Zu den angezeigten Attributen zählen folgende:

Name Der Name des Pakets.

Version Versions-Zeichenkette in unterschiedlichen Formaten.

Index Interner Zähler für neuere Versionen des Programms.

Gefährlich Besitzt diese Paketversion sicherheitskritische Schwachstellen?

Ignorieren Soll diese Paketversion bei Versionsvergleichen ignoriert werden?

Kommentar Ein optionaler Beschreibungstext für das Paket.

Erstellungsdatum Zeitpunkt der erstmaligen Speicherung dieser Paketversion.

Letzte Änderung Zeitpunkt der letzten Änderung dieser Paketversion.

[39] Zu Beginn der Implementierungsphase wurde der Prototyp noch an eine SQLite-Datenbank angebunden. Diese verursachte beim Importieren der Referenzpaketliste mit den mehr als 23.000 Einträgen starke Performanceprobleme. Aus diesem Grund wurde bereits zur Implementierungsphase auf eine richtiges relationales DBMS umgestellt.

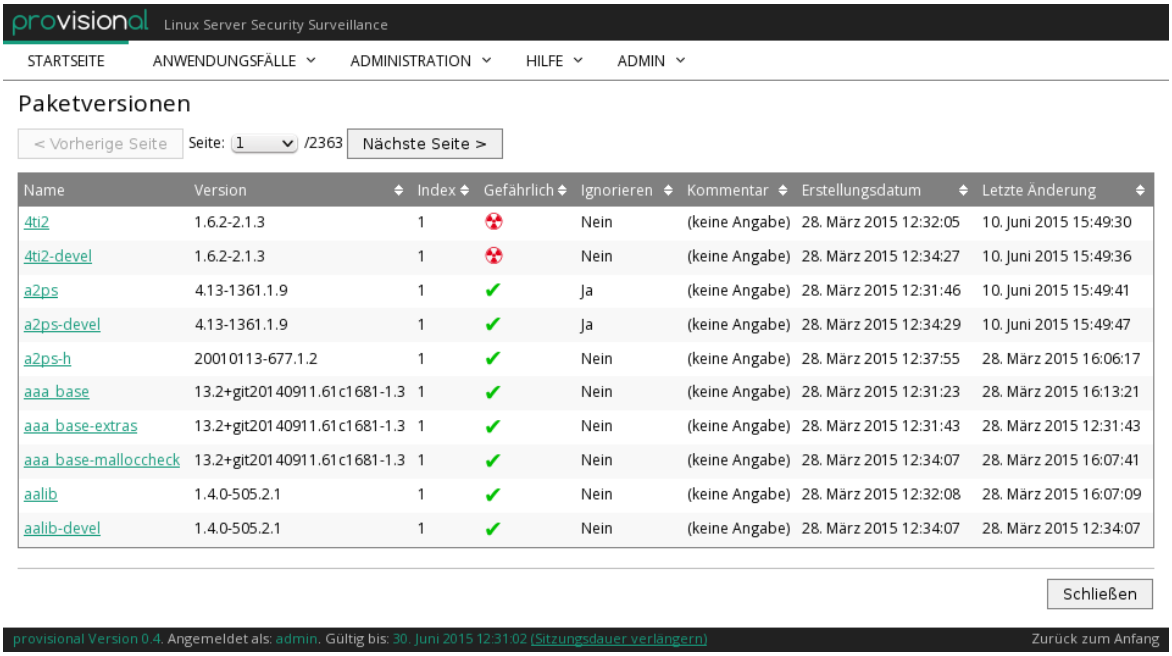


Abbildung 7.60.: Übersicht der dem Prototyp bekannten Pakete und Versionen.

Durch den Klick auf eines der Pakete können seine Attribute über das Eingabeformular in Abbildung 7.61 geändert werden. Diese Bedienoberfläche wurde geschaffen, um zu Demonstrationszwecken einzelne Pakete als gefährlich oder als ignorierbar einzustufen, damit sie später dementsprechend in den Visualisierungen besonders hervorgehoben oder gar nicht beachtet werden. In einem Produktivsystem sollten diese Zusatzinformationen besser automatisch bereitgestellt werden, da die regelmäßige manuelle Bearbeitung tausender Datensätze auf mehreren hundert Bildschirmseiten nicht praktikabel ist.[40]

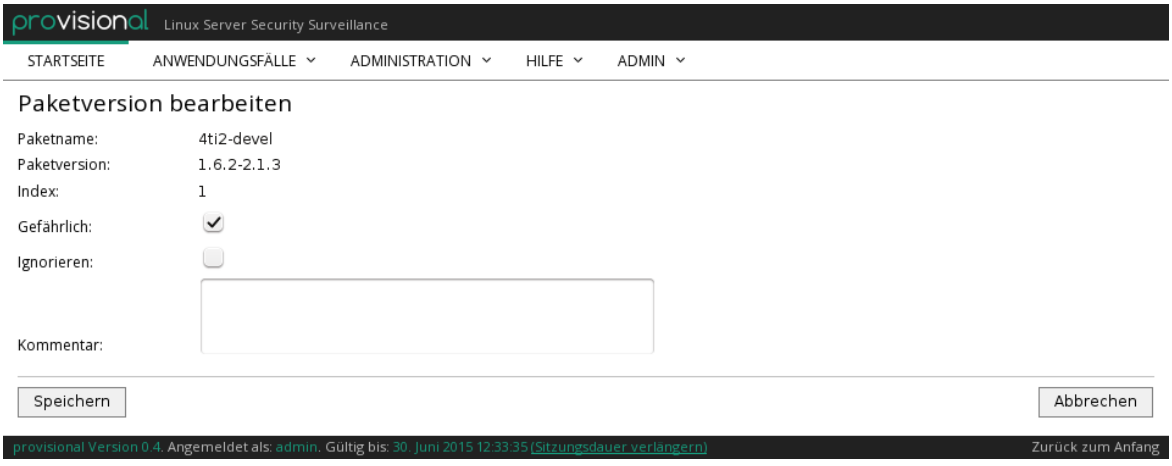


Abbildung 7.61.: Bearbeitung der Eigenschaften einer Paketversion.

[40] Selbst im unwahrscheinlichen Fall, dass sich jemand freiwillig zur Erledigung dieser Aufgabe zur Verfügung stellt, sollte die Arbeitszeit besser in die Entwicklung eines Automatismus gesteckt werden. Die Motivation dieser Masterarbeit besteht ja gerade darin, Administratoren von unüberschaubaren Textkolonnen zu verschonen und Zeit einzusparen. Die manuelle Bearbeitung tausender Datensätze ist schon aus dem Grund nicht zielführend, da dem Bearbeiter zwangsläufig Fehler unterlaufen. Mit falschen Eingangsdaten als Grundlage verlieren die Visualisierungen an Aussagekraft und führen bei den Betrachtern zu falschen Schlussfolgerungen, die sich gravierend auf die Informationssicherheit auswirken können.

7.5. Zusammenfassung

In diesem Kapitel wurde anhand des im Rahmen dieser Masterarbeit anzufertigenden Demonstrators gezeigt, dass sich die in Kapitel 6 vorgeschlagenen Visualisierungen in einer webbasierten Anwendung prinzipiell realisieren lassen. Durch die Verwendung etablierter Techniken aus der Webentwicklung und den Verzicht auf proprietäre Technologien wurde ein Grundgerüst geschaffen, das sowohl Grundlagen der Informationsvisualisierung implementiert, andererseits hohen Wert auf Informationssicherheit legt. Dieses Grundgerüst eignet sich für die Weiterentwicklung zu einem Produktivsystem.

Der Demonstrator nutzt neue Funktionen von HTML5. Durch die gezielte Verwendung der AJAX-Technologie beim Laden der zu visualisierenden Daten reduziert er die Wartezeit und sorgt an den Stellen, wo es Sinn macht, für eine flüssigere Bedienbarkeit des Programms. Gleichzeitig sinkt dadurch die Last sowohl im Netz als auch auf dem Server, da bei der Interaktion mit den dynamisch generierten Visualisierungen durch den Anwender nicht komplette Webseiten ständig erneut vom Server angefordert werden müssen. Dies spielt besonders bei einer möglichen Weiterentwicklung zu einem Produktivsystem eine wichtige Rolle. Mit Hilfe der beim Prototyp verwendeten Basistechnologien sind auch Erweiterungen des Systems denkbar, die noch im Abschnitt 9.2 beschrieben werden. Da das zoombare Bedieninterface aus einer zur Laufzeit dynamisch generierten Vektorgrafik besteht und die enthaltenen Elemente nach Belieben mit Event Handlern ausgestattet werden können, steht einer Erweiterung nichts im Weg.

Bei der Gestaltung der Bedienoberfläche des Prototyps wurden grundlegende Designprinzipien angewandt und sowohl auf Bedienfreundlichkeit als auch auf Übersichtlichkeit Wert gelegt. Die Ergebnisse der Evaluierung bereits vorhandener Administrationstools wurden dabei berücksichtigt, denn die gleichen Fehler sollten nicht wiederholt werden. Durch die Personalisierbarkeit diverser Anzeigeeinstellungen ist die Bedienoberfläche bereits in einer prototypischen Anwendung an die Bedürfnisse der Anwender individuell anpassbar.

Anhand der Implementierung der zoombaren Bedienoberfläche wurde gezeigt, dass diese Art der Visualisierung ohne Weiteres in einer webbasierten Anwendung zur Darstellung sicherheitsrelevanter Zustandsinformationen von Linux-Servern integriert werden kann und dass diese Art der Visualisierung in kurzer Zeit eine Bewertung des Sicherheitszustands der Server gewährleistet. Es wurde gezeigt, dass Ben Shneidermans Information Seeking Mantra zur Bewältigung von unüberschaubaren Datenmengen (siehe Abschnitt 6.1.2) in Kombination mit anderen Techniken der Informationsvisualisierung gewinnbringend auch im Bereich der Linux-Server-Administration anwendbar ist.

8. Evaluierung des Prototyps

Im Kapitel 5 wurden Administrationstools anhand visueller Attribute bewertet, die aus der Anforderungsanalyse im Kapitel 4 resultierten. In diesem Kapitel wird der im Rahmen der Masterarbeit entstandene Prototyp ebenfalls dieser Bewertung unterzogen. Das Testergebnis soll dabei helfen, die Frage zu beantworten, ob Methoden der Informationsvisualisierung im Kontext der Linux-Server-Administration gewinnbringend eingesetzt werden können.

Da der Schwerpunkt des Prototyps auf Visualisierung beruht, erfolgt die Bewertung der prototypischen Implementierung in erster Linie anhand der visuellen Anforderungen aus Kapitel 4. Da dort aber auch funktionale und nichtfunktionale Anforderungen an ein Administrationstool aufgestellt wurden, die zum Teil den Prototyp betreffen, werden diese auch berücksichtigt. Zur Bewertung der visuellen Eigenschaften werden wie auch in Kapitel 5 Noten von *sehr gut* bis *mangelhaft* herangezogen. Damit ist ein direkter Vergleich der visuellen Attribute des Demonstrators mit denen der existierenden Tools gewährleistet. Da es in dieser Ausarbeitung aufgrund der Aufgabenstellung für die funktionalen und nichtfunktionalen Anforderungen keine Vergleichsmöglichkeiten gibt, erfolgt deren Bewertung etwas weniger differenziert. Als mögliche Resultate kommen dort *erfüllt*, *teilweise erfüllt* und *nicht erfüllt* in Frage.

Fairerweise muss an dieser Stelle berücksichtigt werden, dass sich viele der funktionalen und nichtfunktionalen Anforderungen an ein Produktivsystem richten, das die Linux-Server-Administration durch Visualisierung verbessern soll. Aufgrund des visuellen Schwerpunkts des Prototyps wurden diese nur teilweise umgesetzt. Von den möglichen Anwendungsfällen wurde der Schwerpunkt auf die Software-Verwaltung gelegt, da dieser Anwendungsfall einerseits nahezu täglich eine Rolle spielt und andererseits einen starken Bezug zur Informationssicherheit hat. Aufgrund der gemeinsamen Visualisierungsmethoden für unterschiedliche Anwendungsfälle kann die Implementierung dieses Anwendungsfalls auch als Grundlage für andere Use Cases angesehen werden. Da die in Kapitel 6 erarbeiteten Lösungsvorschläge die Grundlage für die Erfüllung der Anforderungen darstellen, werden diese bei der Evaluierung ebenfalls berücksichtigt. In diesem Kapitel wurde schließlich gezeigt, wie die konkreten Anforderungen erfüllt werden können.

8.1. Bewertung der visuellen Anforderungen

In diesem Abschnitt erfolgt die Bewertung der prototypischen Anwendung anhand der visuellen Anforderungen aus Abschnitt 4.2.3. Im Folgenden wird jede Anforderung noch einmal kurz vorgestellt und danach jeweils ein Vergleich mit der Umsetzung im Demonstrator durchgeführt.

V1: Übersichtlichkeit

Die Bedienoberfläche darf nicht überfrachtet sein, da sonst die Übersichtlichkeit darunter leidet.

Beim Entwurf der Bedienoberfläche wurde großer Wert darauf gelegt, nur die wichtigsten Informationen auf den Bildschirm zu bringen, um die Übersichtlichkeit zu optimieren und die Aufmerksamkeit des Anwenders nicht zu strapazieren. Die Bedienoberfläche verfolgt deshalb einen konservativen aber bewährten Aufbau, der aus einer Titelleiste, einem Menü, einer Hauptanzeigefläche und einer Fußzeile besteht. Diese Elemente kommen in vielen webbasierten Anwendungen vor und kommen Anwendern zumindest prinzipiell vertraut vor. Sie wissen von Anfang an, wo sie nach etwas suchen müssen.

Die Titelleiste besitzt ein Programmlogo, das dem Anwender kennzeichnet, in welcher Anwendung er sich gerade befindet. Sie ist relativ flach gehalten, um selbst auf kleineren Bildschirmen keine kostbare Bildfläche zu verschwenden. Das Menü zeigt auf der obersten Ebene nur die notwendigsten Elemente an und hebt den aktuellen Menüpunkt optisch hervor. Dadurch weiß der Anwender immer, zu welcher Programmfunktion die

aktuell aufgerufene Webseite gehört. Die Anzeige bestimmter Menüelemente erfolgt nur dann, wenn der Anwender ein entsprechendes Zugriffsrecht besitzt. Dies gilt zum Beispiel für den Menüpunkt *Administration*. Da Anwender ohne Administratorrechte die URLs aus dem Administrationsmenü gar nicht aufrufen können und sie mit den Menüeinträgen auch nicht konfrontiert werden sollen, ist der Menüpunkt bei ihnen ausgeblendet. Dies steigert die Übersichtlichkeit des Prototyps zusätzlich.

Eine konsistente Gestaltung der einzelnen Webseiten trägt ebenfalls zur Übersichtlichkeit bei, da Schaltflächen mit ähnlichen Funktionen immer an einer ähnlichen Stelle auf dem Bildschirm befindlich sind. Dies gilt zum Beispiel für Buttons mit der Beschriftung *Schließen* oder *Abbrechen*. Diese befinden sich immer rechts unten auf einer Webseite. Schaltflächen zum Erzeugen neuer Datensätze sind immer links unten vorzufinden.

Informationen, die nicht immer sichtbar sein müssen, wurden in Dialogfenster ausgelagert, die der Anwender bei Bedarf öffnen, frei auf dem Bildschirm anordnen und wieder schließen kann. Dadurch vergrößert sich die Übersichtlichkeit ebenfalls. Um auch abseits der Webseiten, die Visualisierungen beinhalten, große Datenmengen möglichst übersichtlich darzustellen, kommt das Tablesorter-Plug-in zum Einsatz (siehe Abschnitt 7.2.1).

Da der Prototyp diese Anforderung bestens erfüllt, erhält er hier die Teilbewertung *sehr gut*.

V2: Farben

In der Bedienoberfläche müssen Farben sparsam und zielgerichtet eingesetzt werden.

Die Anforderung, Farben bei der Bedienoberfläche möglichst sparsam und zielführend einzusetzen, wurde beim Prototyp konsequent umgesetzt. Dabei wurde Ben Shneidermans Rat befolgt, zunächst die Oberfläche für Monochrom-Displays zu designen und erst danach gezielt Farben hinzuzufügen (siehe Abschnitt 2.13.3). Aus diesem Grund setzt die Anwendung bei der Bedienoberfläche auf eine Handvoll Graustufen bzw. Weiß und Schwarz.

Zur Hervorhebung wichtiger Oberflächenelemente nutzt die Anwendung lediglich eine einzige Signalfarbe. Diese Signalfarbe wird beispielsweise zur Kennzeichnung aktiver Menüpunkte oder Schaltflächen verwendet. Bei der standardmäßig eingestellten Signalfarbe wurde darauf geachtet, dass sie einerseits nicht aufdringlich wirkt, andererseits dass sie nur in der Bedienoberfläche selbst und nicht in den Visualisierungen zum Einsatz kommt, um Verwechslungen auszuschließen. Um sämtliche Geschmäcker zufriedenzustellen, können Anwender in den persönlichen Einstellungen des Demonstrators eine individuelle Signalfarbe auswählen.

Da es kaum möglich ist, noch sparsamer und dennoch zielführender mit Farben in der Bedienoberfläche umzugehen, erhält der Prototyp in dieser Teilbewertung die Note *sehr gut*.

V3: Kontrast

In der Bedienoberfläche müssen die Kontrastverhältnisse hoch sein.

Eine weitere Anforderung ist, die Kontrastverhältnisse der Bedienoberfläche möglichst hoch zu halten, um die Lesbarkeit zu erleichtern. Diese Forderung setzt der Prototyp konsequent um. Die meisten Oberflächentexte bestehen aus schwarzer Farbe auf weißem oder zumindest hellem Hintergrund. Helle Hintergrundfarben sind gerechtfertigt, um einzelne Bedienelemente besser vom Webseitenhintergrund hervorzuheben. Dies ist zum Beispiel bei Schaltflächen der Fall.

An einigen Stellen besitzt die Bedienoberfläche hellen Text auf dunklem Hintergrund. Dies ist zum Beispiel beim Menü der Fall. Eine Umkehrung des Konzepts *Schwarz auf Weiß* wurde hier gewählt, um eine bessere Unterscheidbarkeit des Menüs von anderen Bildschirminhalten zu gewährleisten. Der Kontrast zwischen einem aufgeklappten Menü und dem Hintergrund des Hauptanzeigebereichs ist dadurch deutlich höher als es bei einem Menü mit weißem Hintergrund und schwarzem Text der Fall wäre.

Das Prinzip *Hell auf Dunkel* wird neben dem Menü auch bei den Titelleisten der Dialogfenster und Tabellen eingesetzt. Auch hier dient es zu einer Besseren Unterscheidbarkeit vom Hintergrund und von den Tabelleninhalten. Um auch die einzelnen Tabelleninhalte voneinander besser unterscheiden zu können, besitzen die Tabellen zwei sich abwechselnde helle Hintergrundfarben. Zur deutlichen Hervorhebung interessanter Tabellenzeilen verdunkelt sich ihre Hintergrundfarbe, sobald der Anwender den Mauszeiger darauf bewegt. Dadurch verbessert sich der Kontrast zwischen der interessanten Zeile und den umgebenden Zeilen.

Aufgrund der konsequenten Optimierung der Kontrastverhältnisse auf der Bedienoberfläche erhält der Demonstrator bei der Bewertung dieser Anforderung die Teilnote *sehr gut*.

V4: Objektgröße

In der Bedienoberfläche müssen Oberflächenelemente eine ausreichende Größe besitzen.

Oberflächenelemente einerseits ausreichend groß zu gestalten, damit sie gut sichtbar und klickbar sind und andererseits möglichst klein zu halten, um keine unnötige Bildschirmfläche zu verschwenden, ist eine Gratwanderung. Mit Ausnahme der Titelleiste und der Fußzeile besitzen alle Oberflächentexte eine Größe von mindestens *9pt*. Bei der Titelleiste und der Fußzeile ist die geringere Textgröße zu verschmerzen, da dort lediglich sehr allgemeine oder unveränderliche Informationen angezeigt werden, die nicht in den Vordergrund treten sollen.

Im Bereich der Visualisierungen lassen sich alle Grafiken in der Größe nach den eigenen Anforderungen anpassen. Die einzelnen Säulen der Diagramme sind genügend breit, um gut abgelesen werden zu können und skalieren mit der Breite des Diagramms mit. Lediglich die Textgröße bei den Diagrammbeschriftungen könnte etwas größer sein, da hier noch auf die etwas zu geringe Standardschriftgröße der verwendeten Diagramm-Bibliothek zurückgegriffen wird.

Die zoombare Oberfläche bietet den Vorteil, dass auch kleine Objekte beliebig vergrößert werden können, um sie besser sehen zu können. Hier kann jeder Anwender nach den eigenen Bedürfnissen eine für sich akzeptable Objektgröße individuell einstellen.

Im Großen und Ganzen erfüllt der Prototyp die Merkmale dieser Anforderung. Da einige Texte eine grenzwertige Schriftgröße besitzen, wird hier die Teilnote *gut* vergeben.[1]

V5: Visualisierungsformen

Die Arten der Visualisierungen müssen zu den zugrundeliegenden Daten und Dimensionalitäten passen.

Der Prototyp setzt unterschiedliche Arten von Visualisierungen ein, die zur Repräsentation der zugrundeliegenden Daten passend ausgewählt wurden. Dies beginnt bereits auf den Webseiten aus dem Bereich Administration. Um sich häufig wiederholende Texte in umfangreichen Tabellen zu vermeiden, kommen stattdessen Piktogramme zum Einsatz. Sie sparen im Vergleich zu Texten Platz und stechen dank ihrer Signalfarbe ins Auge. Optional einblendbare Legenden helfen unerfahrenen Anwendern bei der Interpretation der Piktogramme.

Um die Lesbarkeit der Tabellen weiter zu verbessern, beinhalten einige davon Fortschrittsbalken zur grafischen Repräsentation numerischer Werte. Dies ist im Administrationsbereich beim Attribut Server-Priorität der Fall. Als besonders nützlich erweisen sich diese Fortschrittsbalken in Kombination mit der Sortierfunktion der Tabellen. Wenn die Sortierung anhand der Priorität erfolgt, stechen die Fortschrittsbalken deutlich besser ins Auge als Zahlen. Der Administrator sieht unmittelbar, welche Server besonders wichtig sind.

Der im Demonstrator implementierte Anwendungsfall *Software-Verwaltung* nutzt noch weitere Visualisierungsformen: Ampelpiktogramme zur Kennzeichnung der allgemeinen Server-Zustände ermöglichen einen schnellen Überblick und sind intuitiv interpretierbar. Optional einblendbare Legenden erleichtern ungeübten Anwendern die Interpretation der Ampelfarben.

Die Balkendiagramme der Software-Statistiken eignen sich hervorragend zur Visualisierung einer geringen Menge absoluter Werte und zum Vergleich dieser Werte. Für Anwender, die an relativen Werten interessiert sind, hält die Anwendung ein Kreisdiagramm bereit. Bei den Visualisierungen der Patch-Zustände, die auch die Server-Kategorien berücksichtigen, entscheidet der Anwender, über welche visuellen Attribute die Server-Kategorien im Diagramm untergebracht werden. Er kann die für ihn jeweils beste Ansicht wählen. Beim Histogramm der installierten Pakete kann der Anwender die Anzahl der Säulen frei wählen und so selbst einen Wert bestimmen, der zur Gesamtanzahl der betrachteten Server passt.

Im Kapitel 6 wurde als anwendungsfallübergreifende Methode eine Visualisierung vorgeschlagen, die auf einer zoombaren Übersicht mit Dynamic Queries und Filterung basiert. Es überträgt Ben Shneidermans Information Seeking Mantra (siehe 6.1.2) auf den Bereich der Linux-Server-Administration, da dort umfangreiche

[1] Es soll auch nicht der Eindruck entstehen, dass in dieser Masterarbeit zugunsten des Demonstrators mit zweierlei Maß gemessen wird. Hundertprozentige Konformität zu erzielen, ist wohl nur in Nordkorea möglich [ERD 14].

Datenmengen anfallen, die beherrschbar gemacht werden sollen. In dem Kapitel wurde gezeigt, dass sich diese Form der Visualisierung eignet, um die Patch-Zustände einer großen Menge von Linux-Servern zu überwachen und sowohl reaktiv als auch proaktiv tätig zu werden. Der Prototyp implementiert die wichtigsten Funktionen dieses Visualisierungsschemas, um die technische Umsetzbarkeit und Sinnhaftigkeit zu demonstrieren. Im Kontext der Linux-Server-Administration ist diese Art der Visualisierung innovativ und zielführend: Der Anwender entscheidet selbst, welche Serverdaten auf die unterschiedlichen visuellen Attribute gemappt werden und wie sie auf dem Bildschirm sortiert werden. Er kann sukzessive Filter zusammenstellen, um gezielt nach bestimmten Systemen zu suchen, deren Details er schließlich abrufen kann. Um sich auf die Demonstration des Bedienkonzepts an sich zu konzentrieren, verwendet der Demonstrator anstatt von Server-Piktogrammen Kreise. Gegenüber von Glyphen in Form von Server-Piktogrammen besitzen sie zwar weniger Attribute, für die im Demonstrator bereitgestellten Daten sind diese aber erstens ausreichend und zweitens fällt der Größenvergleich einfacher, sofern ein Attribut auf den Radius gemappt wurde.

Zusammenfassend ist über den Demonstrator zu sagen, dass er einerseits herkömmliche Visualisierungsformen unter Berücksichtigung der Grundlagen der Informationsvisualisierung umsetzt, andererseits den Nutzen einer in diesem Kontext innovativen Methode der Visualisierung zeigt. Dies führt zur Teilnote *sehr gut* bei diesem Anforderungspunkt.

V6: Beschriftungen

Visualisierungen müssen mit den notwendigen Beschriftungen versehen sein, damit sie nicht fehlinterpretiert werden können.

Alle herkömmlichen Visualisierungen im Prototyp wurden mit einem vernünftigen Maß an Beschriftungstexten versehen. Die zuvor genannten Fortschrittsbalken verfügen sowohl über permanent sichtbare numerische Werte als auch über Tooltip-Texte. Die Tooltip-Texte mögen zwar redundant wirken, sie sind dennoch vorhanden, da bereits an mögliche Weiterentwicklungen der Anwendung gedacht wurde: Für das Tablesorter-Plug-in gibt es Erweiterungen, die es dem Anwender gestatten, selbst die sichtbaren Spalten zu beeinflussen. Wenn ein Anwender nur die Fortschrittsbalken sehen möchte, kann er die Beschriftung ausblenden. Über die Tooltip-Texte kann er dennoch die genauen Werte ablesen, der sich hinter diesen Visualisierungen verbergen.

Auch im Bereich der Visualisierung der Software-Patch-Zustände verfügen die Diagramme über die nötigen Beschriftungen. Besonders bei den gestapelten Diagrammen erweisen sich die Tooltip-Texte mit den zugrundeliegenden Werten als hilfreich. Die Beschriftungen in der zoombaren Oberfläche beschränken sich auf ein Minimum, da hier die Demonstration der interaktiven Visualisierungsmethode im Vordergrund steht. Eine Skala auf den Koordinatenachsen, die sich dynamisch an den gewählten Vergrößerungsgrad und angezeigten Bildbereich anpasst, wurde im Demonstrator nicht mehr implementiert, da dies Änderungen an der JavaScript-Bibliothek jQuery Panzoom (siehe Abschnitt 7.2.1) nach sich gezogen hätte.

Da im Prototyp bis auf die Achsenbeschriftungen der zoombaren Oberfläche genügend aussagekräftige Beschriftungen vorhanden sind, wird der Erfüllungsgrad dieser Anforderung mit der Note *gut* bewertet.

V7: Visual Clutter

Visualisierungen müssen frei von Visual Clutter sein, um die Übersichtlichkeit zu verbessern.

Bei der Gestaltung der Visualisierungen im Prototyp wurde konsequent darauf geachtet, Visual Clutter zu verhindern. Lediglich bei der zoombaren Bedienoberfläche besteht die Gefahr von Visual Clutter. Die Ursache hierfür liegt an der großen Anzahl an Servern, die zu Überlappungen in der Oberfläche führen können. Das ist unvermeidlich. Um dennoch eine gute Unterscheidbarkeit der einzelnen Piktogramme zu ermöglichen, stellt der Demonstrator verschiedene Funktionen bereit: Mit Hilfe der Filterfunktion können Anwender die Menge der auf dem Bildschirm angezeigten Server reduzieren. Die optional hinzuschaltbare Objekttransparenz macht auch alle verdeckten Piktogramme deutlich sichtbar. Noch besser unterscheidbar werden sie durch Aktivierung der optionalen Farbskala und durch den beliebig wählbaren Vergrößerungsgrad. Sollten dennoch einzelne Objekte bei näherer Betrachtung von anderen verdeckt werden, sodass ihre Detailinformationen nicht abrufbar sind, ermöglicht der Demonstrator das Freilegen verdeckter Objekte durch das Verschieben der darüber liegenden Piktogramme.

Da der Demonstrator einerseits Visualisierungen bereitstellt, die völlig frei von Visual Clutter sind und andererseits in der zoombaren Oberfläche wirksame Werkzeuge zur Elimination von systembedingtem Visual Clutter bereitstellt, erhält er in dieser Anforderungskategorie die Note *sehr gut*.

V8: Farbskalen

Farben in Visualisierungen müssen sparsam und zielgerichtet eingesetzt werden. Farbige Datenreihen müssen gut voneinander unterscheidbar sein.

Bei der Zusammensetzung der Farbskalen im Prototyp wurden die Grundlagen der Informationsvisualisierung befolgt. Dies fängt bereits bei der Farbauswahl der Benutzeroberfläche an. Dort gibt es lediglich eine einzige Akzentfarbe, die zur Hervorhebung wichtiger Elemente in der Bedienoberfläche gewählt wurde. Damit sich der Anwender bestmöglich auf die Visualisierungen im Prototyp konzentrieren kann, wird in sämtlichen Farbskalen in den Visualisierungen auf diese Akzentfarbe verzichtet.

Sowohl in in den Statistik-Diagrammen als auch in der zoombaren Oberfläche des Anwendungsfalls Software-Verwaltung bestehen die Farbskalen größtenteils aus den gut zu unterscheidenden Signalfarben Grün, Gelb und Rot. Bei den Diagrammen, deren Säulen in Server-Kategorien unterteilt sind, findet eine Farbkodierung mit einer überschaubaren Anzahl gut unterscheidbarer Farben statt. Bei dem Histogramm der installierten Pakete wurde hingegen auf Farbe komplett verzichtet, um eine unverfälschte Interpretation der angezeigten Säulen zu gewährleisten. Standardmäßig hätte die Diagramm-Bibliothek dort Farben verwendet.

Aufgrund der guten Umsetzung der Grundlagen der Informationsvisualisierung bei den Farbklassen erhält der Prototyp in dieser Kategorie die Teilwertung *sehr gut*.

V9: Kontraste in Visualisierungen

Kontrastverhältnisse in Visualisierungen müssen hoch sein, um die Lesbarkeit zu optimieren.

Um den Kontrast innerhalb der Visualisierungen zu optimieren, wurde wie auch in der Bedienoberfläche ein weißer Hintergrund gewählt. Die kräftigen Farben der zuvor erwähnten Farbskalen heben sich gut sichtbar vom Hintergrund ab. Die Tooltip-Texte der Visualisierungen heben sich durch ihren dunklen Hintergrund und der weißen Textfarbe wiederum gut von den Diagrammen ab. Die Achsbeschriftungen sind abgesehen von der zuvor angesprochenen grenzwertigen Textgröße aufgrund ihres Kontrasts zum Hintergrund gut erkennbar. Eine gute Interpretation der Diagramme ist so gewährleistet.

Das Histogramm der installierten Pakete unterscheidet sich von den anderen Diagrammen: Es wird dort keine Farbskala verwendet, um die Bedeutung der einzelnen Säulen nicht unterschwellig zu verzerren. Um bei der Betrachtung des Histogramms die einzelnen Säulen dennoch gut voneinander unterscheiden zu können, wird ihre Hintergrundfarbe beim Überfahren mit dem Mauszeiger verdunkelt und so ein hoher Kontrast zum Rest der Visualisierung geschaffen.

Die zoombare Visualisierung der Software-Patch-Zustände stellt dem Anwender Funktionen bereit, um den Kontrast nach den eigenen Bedürfnissen zu optimieren: Standardmäßig besitzen die Kreise, die die einzelnen Server kennzeichnen, keine Hintergrundfarbe. Ein Schwarzer Rand sorgt hier für eine scharfe Trennung der Objekte. Bei Bedarf kann der Anwender Signalfarben hinzuschalten. Da schwarze Rahmen um Farbflächen die Kontrastwirkung verringern (siehe 2.13.2), wird der schwarze Rahmen bei aktivierten Signalfarben entfernt. Um die einzelnen Serverpiktogramme auch bei Überlappungen gut voneinander unterscheiden zu können, können Anwender optional die Objekttransparenz aktivieren.

Da die zoombare Oberfläche eine sehr großflächige Visualisierung ist und je nach der Verteilung der Server viel freie Hintergrundfläche beinhalten kann, ist es möglich, dass manche Anwender sich von dem weißen Hintergrund ein wenig geblendet fühlen. Aus diesem Grund erlaubt es der Prototyp, die Hintergrundfarbe der zoombaren Visualisierung bei Bedarf zu verdunkeln. Dadurch ist weiterhin ein akzeptabler Kontrast zu den Server-Piktogrammen gewährleistet. Vor allem in dunklen Arbeitsumgebungen kann der schwarze Hintergrund entspannender für die Augen sein [SCHEU 08].

Zusammenfassend wird der Kontrast in den Visualisierungen des Prototyps als *sehr gut* bewertet.

V10: Chartjunk und Non-Data-Ink

Chartjunk lenkt den Betrachter ab und ist unerwünscht. Non-Data-Ink muss auf ein Minimum reduziert werden.

Die Visualisierungen des Demonstrators sind komplett frei von Chartjunk, Eye Candy und ähnlichen unnötigen Elementen wie Farbverläufen, die keine Zusatzinformationen beinhalten. Die für die zweidimensionalen

Visualisierungen verwendete JavaScript-Bibliothek Flot (siehe Abschnitt 7.2.1) besitzt in der Standardkonfiguration Elemente, die primär der Verschönerung dienen sollen. Dazu zählen zum Beispiel transparente Diagrammsäulen und zusätzliche Gitternetzlinien. Diese Elemente wurden allerdings im Prototyp durch geeignete Konfigurationsparameter entfernt.

Über den Non-Data-Ink-Anteil (siehe Abschnitt 2.7.1) lässt sich streiten, sofern man Edward Tufte als Diskussionspartner gegenüber sitzt. Er vertritt das strenge Dogma, dass Füllflächen von Diagrammsäulen überflüssig sind, da sie Non-Data-Ink darstellen. Bei Liniendiagrammen mag das vielleicht gut zutreffen, da die Füllfläche in der Regel keine Zusatzinformation repräsentiert. Die Diagramme des Demonstrators hingegen setzen hauptsächlich Signalfarben ein, die von der Verkehrsampel stammen. Hier ist es explizit erwünscht, dass die Diagrammsäulen damit eingefärbt sind, um besser ins Auge zu stechen. Eine rote Fläche, die kritische Lücken in Softwarepaketen repräsentiert wirkt alarmierender auf den Betrachter als eine dünne rote Linie. Ebenso wirkt eine grüne Fläche, die den Optimalzustand repräsentiert, entspannender auf den Betrachter als eine dünne grüne Linie. Zusammen mit der voreingestellten Säulenbreite ergibt sich so ein gut erkennbares Gesamtbild, das seinen Zweck erfüllt. Wie an diesem Beispiel zu sehen ist, kommt es gelegentlich auch auf den Kontext an, wie sehr bestimmte Grundlagen der Informationsvisualisierung befolgt werden.

Es gibt aber auch Visualisierung im Prototyp, bei denen der Non-Data-Ink-Anteil stark reduziert werden kann, ohne dass sie an Aussagekraft verlieren. Dies gilt zum Beispiel für das Tortendiagramm. Da dort nur relative Verhältnisse zwischen den signalfarbenen Segmenten veranschaulicht werden sollen, kann Farbe eingespart werden. Dies geschieht, indem erst ab einem bestimmten Radius Farbe verwendet wird, das Zentrum der Grafik bleibt weiß.

Die zoombare Bedienoberfläche kommt ebenfalls vollkommen ohne Chartjunk aus. Das Data-Ink-Verhältnis ließe sich hier nach Edward Tufte auch noch weiter verbessern, es macht aber auch hier keinen Sinn, da sonst die Aussagekraft der Visualisierung darunter leiden würde. Wenn die kreisförmigen Server-Piktogramme nur eine farbige Linie besäßen, wären sie für den Betrachter kaum noch zu unterscheiden. Dabei ist es aber die Absicht der zoombaren Bedienoberfläche, die präattentive Wahrnehmung (siehe Abbildung 2.11) von Signalfarben auszunutzen. Im Gegensatz zum zuvor genannten Kreisdiagramm macht es in der zoombaren Oberfläche des Prototyps aber keinen Sinn, die Färbung der Piktogramme erst ab einem bestimmten Radius zu beginnen. Da in dieser Visualisierung Überlappungen sehr wahrscheinlich sind, würden die resultierenden weißen Innenkreise irritierend wirken und als Visual Clutter wahrgenommen werden, besonders nachdem die Objekttransparenz aktiviert wurde. Aus praktischer Sicht besitzt die zoombare Bedienoberfläche also einen guten Data-Ink-Anteil.

Da die Visualisierungen im Prototyp absolut frei von Chartjunk sind und die Abweichung von optimalen Data-Ink-Verhältnissen gut begründet werden kann, erhält der Demonstrator hier die Teilnote *sehr gut*.

V11: Lie Factor

Der Lie Factor muss optimal sein, um die Aussage der Visualisierungen nicht zu verzerren und Fehlinterpretationen zu verhindern.

Besonders im Bereich der Linux-Server-Administration ist es unerlässlich, sicherheitsrelevante Entscheidungen anhand unverfälschter Daten zu treffen. Aus diesem Grund wurde bei der Implementierung des Demonstrators großer Wert darauf gelegt, dass die Visualisierungen die zugrundeliegenden Daten verzerrungsfrei darstellen. Aus diesem Grund sind alle Achsskalierungen im Prototyp linear und besitzen keinen versteckten Offset.

Eine verzerrte Wahrnehmung der Informationen ist nicht nur durch nichtlineare Achsskalierungen und Offsets möglich, sondern kann auch durch die Wirkung der verwendeten Farben beeinflusst werden. Um klare Aussagen zu vermitteln benutzt der Demonstrator deshalb in den meisten Visualisierungen ausschließlich die Signalfarben *Grün*, *Gelb* und *Rot* zur Kennzeichnung optimaler, suboptimaler und kritischer Zustände. Aufgrund der Geläufigkeit dieser Signalfarben sind Fehlinterpretationen unwahrscheinlich. In Zweifelsfällen hilft ein Blick auf die Legende.

Aufgrund seiner verzerrungsfreien Visualisierungen erhält der Prototyp in dieser Kategorie die Note *sehr gut*.

In Tabelle 8.1 werden die einzelnen Teilergebnisse der Bewertung der visuellen Anforderungen noch einmal zusammengefasst.

Nr.	Kurzbeschreibung	Bewertung
V1	**Übersichtlichkeit:** Die Bedienoberfläche darf nicht überfrachtet sein, da sonst die Übersichtlichkeit darunter leidet.	sehr gut
V2	**Farben:** In der Bedienoberfläche müssen Farben sparsam und zielgerichtet eingesetzt werden.	sehr gut
V3	**Kontrast:** In der Bedienoberfläche müssen die Kontrastverhältnisse hoch sein.	sehr gut
V4	**Objektgröße:** In der Bedienoberfläche müssen Oberflächenelemente eine ausreichende Größe besitzen.	gut
V5	**Visualisierungsformen:** Die Arten der Visualisierungen müssen zu den zugrundeliegenden Daten und Dimensionalitäten passen	sehr gut
V6	**Beschriftungen:** Visualisierungen müssen mit den notwendigen Beschriftungen versehen sein, damit sie nicht fehlinterpretiert werden können.	gut
V7	**Visual Clutter:** Visualisierungen müssen frei von Visual Clutter sein, um die Übersichtlichkeit zu verbessern.	sehr gut
V8	**Farbskalen** Farben in Visualisierungen müssen sparsam und zielgerichtet eingesetzt werden. Farbige Datenreihen müssen gut voneinander unterscheidbar sein.	sehr gut
V9	**Kontraste in Visualisierungen:** Kontrastverhältnisse in Visualisierungen müssen hoch sein, um die Lesbarkeit zu optimieren.	sehr gut
V10	**Chartjunk und Non-Data-Ink:** Chartjunk lenkt den Betrachter ab und ist unerwünscht. Non-Data-Ink muss auf ein Minimum reduziert werden.	sehr gut
V11	**Lie Factor:** Der Lie Factor muss optimal sein, um die Aussage der Visualisierungen nicht zu verzerren und Fehlinterpretationen zu verhindern.	sehr gut
	Gesamtbewertung	**sehr gut**

Tabelle 8.1.: Bewertung der visuellen Anforderungen beim Prototyp.

8.2. Bewertung der funktionalen und nichtfunktionalen Anforderungen

Die erste funktionale Anforderung aus Abschnitt 4.2.1 lautet folgendermaßen:

> **F1: Beherrschbarmachen großer Datenmengen**
>
> Große Datenmengen sollen automatisch beschafft, konsolidiert und durch geeignete Methoden der Informationsvisualisierung beherrschbar gemacht werden.

Während des Fortschreitens der Implementierung des Prototyps stellte sich heraus, dass es umständlich ist, echte Paketlisten von einer dreistelligen Anzahl von Servern zu erhalten und manuell in den Demonstrator einzuspeisen. Da der Schwerpunkt dieser Arbeit in der Visualisierung von sicherheitsrelevanten Serverdaten liegt und zur Beschaffung der Daten in Kapitel 6 bereits Lösungsvorschläge für eine Implementierung gemacht wurden, fällt dies aber nicht negativ ins Gewicht.

Der Demonstrator ist bereits darauf vorbereitet, in einer potenziellen Weiterentwicklung mit echten Serverdaten befüllt zu werden. In Abschnitt 7.2.4 wurden bereits einige Relationen aus dem Datenbankschema vorgestellt, die zu diesem Zweck schon vorbereitet wurden. Bedienoberflächen Webseiten zu diesem Zweck wurden ebenfalls vorbereitet und in Abschnitt 7.4.4 vorgestellt.

Insgesamt gilt diese Anforderung als *teilweise erfüllt*.

Die Einhaltung der folgenden funktionalen Anforderungen F2 bis F5 kann gemeinsam beantwortet werden. Aus diesem Grund werden sie hier zunächst zusammen aufgezählt:

> **F2: Softwarestände überwachen**
>
> Auf den Servern installierte Paketversionen sollen zentral abrufbar sein. Anhand einer Referenzliste soll die Aktualität und die Sicherheit der Pakete bestimmt werden.

F3: Firewall-Konfigurationen überwachen

Die Wirksamkeit und Sicherheit der Firewall-Konfigurationen auf den Servern soll an zentraler Stelle ersichtlich sein. Einzelne Konfigurationsdateien sollen bei Bedarf abrufbar sein.

F4: Sicherheit privater X.509-Schlüssel überwachen

Die Sicherheit privater X.509-Schlüssel aller Server soll bewertet und an zentraler Stelle ersichtlich sein. Ein Zugriff auf die Schlüssel ist nicht erlaubt.

F5: Sicherheit von TLS-Konfigurationen überwachen

Die Sicherheit und Wirksamkeit der TLS-Konfigurationen aller Server soll bewertet und zentral angezeigt werden.

Der Demonstrator beinhaltet zwar keine exakte Implementierung der funktionalen Anforderungen von F2 bis F5, aber es wurde wertvolle Vorarbeit geleistet: Für jede dieser Anforderungen wurden in Kapitel 6 detaillierte Lösungsvorschläge für die Implementierung in einem Produktivsystem erarbeitet. Diese Lösungsvorschläge berücksichtigen einerseits die Visualisierung der für jeden Anwendungsfall relevanten Daten, andererseits auch die Beschaffung der dazu erforderlichen Quelldaten. Aufgrund dieser wertvollen Vorarbeit und da sich der Demonstrator einfach um diese Implementierungen erweitern lässt, werden diese Anforderungen im Großen und Ganzen als *erfüllt* betrachtet.

F6: Datenschutzrechtliche Aspekte

Das Tool muss Datenschutzrechtliche Regelungen umsetzen. Datensparsamkeit ist essentiell.

Die Datenbank des Prototyps beinhaltet im Auslieferungszustand lediglich frei erfundene Serverdaten aus dem Bereich der Software-Verwaltung, um die Funktionalität und Tauglichkeit der dazugehörigen Visualisierungen zu demonstrieren. Daten mit Personenbezug erfasst die Datenbank des Prototyps nur in sehr begrenztem Umfang. Dazu zählen der Benutzername, Vor- und Nachname und die E-Mail-Adresse der Anwender. Im Ereignisprotokoll, das die Anwendung generiert, tauchen nur die im Idealfall pseudonymisierten Benutzernamen auf. Dies ist zum Beispiel der Fall, wenn ein Benutzerkonto aufgrund zu vieler Falscheingaben des Passworts automatisch gesperrt wurde. Hier stellt diese Information ein Sicherheitsmerkmal dar, um Angreifer erkennen zu können. Sollte dies aber bereits zu viel des Guten sein, kann der Benutzername im Programmcode des Servers ohne Weiteres wieder entfernt werden. Eine Protokollierung konkreter Nutzdaten, die ein Anwender im Prototyp eingibt, findet nicht statt. Ebenso wenig werden dort IP-Adressen gespeichert.[2]

In einem potenziellen Wirksystem, das unter anderem echte Logdateien von hunderten Servern abruft, um sie zu aggregieren und zu visualisieren, stellt der Datenschutz ein ernstzunehmendes Problem dar. Personenbezug muss dort durch Maßnahmen wie zum Beispiel Anonymisierung unkenntlich gemacht werden. Dies geschieht im Idealfall schon auf den Servern selbst, bevor die Daten zum zentralen Informationssystem übertragen werden. Da der Prototyp durch diese Problematik nicht konfrontiert ist, gilt diese Anforderung weitestgehend als *erfüllt*.

Die erste nichtfunktionale Anforderung aus Abschnitt 4.2.2 lautet folgendermaßen:

NF1: Bedienfreundlichkeit

Bedienfreundlichkeit soll erreicht werden, indem das Tool dem Administrator das häufige Wechseln zu anderen Tools oder der Kommandozeile erspart. Es muss einfach und geräteübergreifend zu bedienen sein. Mehrsprachigkeit ist erwünscht.

Die bei diesem Anforderungspunkt geforderte Ersparnis, regelmäßig zu anderen Tools wechseln zu müssen, kann der Prototyp nicht erfüllen, da er keine realen Daten von den Linux-Servern automatisch bezieht. Da er aber möglicherweise zu einem Produktivsystem weiterentwickelt wird, wurde von Anfang an auf eine bedienfreundliche Oberfläche wert gelegt. Bedienfreundlichkeit ist unter anderem durch die Übersichtlichkeit der Programmoberfläche und die Einhaltung anderer visueller Anforderungen gewährleistet, die bereits zuvor im Abschnitt 8.1 detailliert besprochen wurden. Die optionale mehrsprachige Bedienoberfläche wurde

[2] Sollte der Prototyp später als Produktivsystem nicht vom Django-Entwicklungsserver, sondern von einem richtigen Webserver bereitgestellt werden, fallen dort zusätzliche Protokolleinträge an. Der Informationsgehalt dieser Einträge hängt von der individuellen Konfiguration des Webservers ab.

mit Hilfe des Template-Systems von Django und gettext-Sprachdateien realisiert. Durch die Auslagerung der Oberflächentexte in separate Sprachdateien gestalten sich die Änderung existierender und die Erstellung neuer Übersetzungen besonders einfach.

Die Tatsache, dass es sich beim Prototyp um eine webbasierte Anwendung handelt, bildet das Fundament für eine geräteübergreifende Einsetzbarkeit. Durch den Verzicht auf proprietäre und häufig verwundbare Technologien wie zum Beispiel Adobe Flash und durch die Verwendung etablierter und vereinheitlichter JavaScript-Bibliotheken sowie HTML5 ist ein hoher Grad an Plattformunabhängigkeit gewährleistet. Die platzsparenden Oberflächenelemente und die auf mehrere Bildschirmseiten aufgeteilten Tabellen ermöglichen auch den Betrieb auf Geräten mit geringerer Bildschirmauflösung. Zudem sind alle vom Prototyp generierten Visualisierungen per Drag&Drop in ihrer Größe an das jeweilige Ausgabegerät anpassbar.

Zusammenfassend ist festzustellen, dass der Prototyp diese nichtfunktionale Anforderung *teilweise erfüllt*.

NF2: Reaktionszeit

Als zentraler Anlaufpunkt soll das Tool sicherheitsrelevante Daten durch Informationsvisualisierung augenblicklich begreifbar machen und so eine geringe Reaktionszeit ermöglichen.

Die Reaktionszeit von Administratoren zu verringern setzt die Weiterentwicklung des Prototyps zu einem Produktivsystem voraus, das unter Berücksichtigung von Informationssicherheit und Datenschutz echte Daten von den knapp tausend Linux-Servern bezieht und visualisiert. Da der Prototyp den Schwerpunkt auf die Demonstration der Visualisierungen an sich und nicht auf die Beschaffung der Server-Zustandsinformationen legt, kann dieser Teil der Anforderung überhaupt nicht erfüllt werden.

Viel wichtiger ist zunächst jedoch, dass der Prototyp unter anderem anhand seiner interaktiven Visualisierungsmethode in zoom- und filterbaren Server-Übersicht beweist, dass sicherheitsrelevante Serverdaten durch Methoden der Informationsvisualisierung augenblicklich begreifbar gemacht werden können. Wie dem als *sehr gut* eingestuften Gesamtergebnis der visuellen Anforderungen aus Abschnitt 8.1 zu entnehmen ist, führt die Einhaltung von Grundlagen der Informationsvisualisierung zu übersichtlichen und informativen Visualisierungen, die schnell und intuitiv interpretiert werden können. Die Visualisierungen des Prototyps besitzen trotz ihrer Einfachheit eine Qualität, die die in Kapitel *sec:existing* bewerteten Administrationstools größtenteils überragt.

Zusammenfassend werden die Punkte dieser Anforderung vom Prototyp *teilweise erfüllt*. Die Teilanforderung der zielführenden Visualisierung, die die Voraussetzung für eine kurze Reaktionszeit ist, wird jedoch vollständig *erfüllt*.

NF3: Basistechnologien

Durch den Verzicht auf proprietäre Basistechnologien sollen Unabhängigkeit geschaffen, Einstiegsbarrieren gesenkt und die Wartbarkeit der Anwendung erhöht werden.

Der geforderte Verzicht auf proprietäre Basistechnologien sowie weitere Merkmale des Prototyps, die sich positiv auf die Sicherheit auswirken, wurden bereits erläutert. Aus diesem Grund gilt diese Anforderung als *erfüllt*.

NF4: Informationssicherheit

Das Tool soll grundlegende Sicherheitsfunktionen wie ein differenziertes Rechtesystem und LDAP-Authentifizierung implementieren. Webbasierte Anwendungen sollen insbesondere Schutzmaßnahmen gegen häufig durchgeführte Angriffstechniken bieten. Die Sicherheit der gespeicherten Daten ist zu gewährleisten.

Aufgrund der Aufgabenstellung darf auch die Informationssicherheit im Demonstrator nicht vernachlässigt werden. Aus diesem Grund wurde das Programm mit einem differenzierten Rechtesystem ausgestattet, das zwischen Administratoren und eingeschränkten Anwendern unterscheidet. Rechte können gruppenspezifisch, rollenspezifisch und individuell an Benutzer verteilt werden. Dasselbe gilt für die Server, die sicherheitsrelevante Informationen zur Visualisierung bereitstellen. Der Prototyp wurde von Anfang an so geplant, dass Anwender nur eine beschränkte Ansicht auf die zu überwachenden Server erhalten.

Ein weiterer Aspekt dieser Anforderung war eine Benutzerauthentifizierung über einen zentralen Verzeichnisdienst. Diese Anforderung wurde im Prototyp implementiert und erfolgreich anhand eines angebundenen

Active-Directory-Servers getestet. Die zentralen Benutzerkonten stellen eine Ergänzung zu lokalen Konten dar. Die LDAP-Authentifizierung ist eine Option, die der Administrator bei der Konfiguration eines Benutzerkontos einstellen kann. Bei der Implementierung der LDAP-Authentifizierung erwies sich die Wahl des serverseitigen Frameworks Django als vorteilhaft, da es die einfache Einbindung eigener sog. *Authentication Backends* ermöglicht.

Die Einbindung von Authentication Backends geschieht in der zentralen Konfigurationsdatei `settings.py`. Ein Ausschnitt davon ist in Listing 8.1 zu sehen. Der Eintrag in der dritten Zeile stammt vom Framework selbst und bezieht sich auf die Benutzerkonten, deren Passwort-Hashwerte in der lokalen Datenbank des Prototyps gespeichert sind. Die zweite Zeile lädt das selbst entwickelte Authentication Backend für Active-Directory-Server. Zu Demonstrationszwecken steht nicht immer ein Active-Directory-Server bereit. Dies ist zum Beispiel der Fall, wenn das Demonstrationssystem mit keinem Netz verbunden ist. Anmeldeversuche würden dann scheitern, weil das Programm keine Verbindung zum eingestellten Authentifizierungsserver herstellen kann. Um dies zu vermeiden genügt es, die zweite Zeile auszukommentieren. Der Prototyp betrachtet dann nur noch die lokale Benutzerdatenbank.

Listing 8.1: Konfiguration der Authentifizierungs-Backends im Prototyp.

```
1 AUTHENTICATION_BACKENDS = (
2     'provisional.auth_ad.ActiveDirectoryBackend',
3     'django.contrib.auth.backends.ModelBackend',
4 )
```

Im Abschnitt 3.1 wurden einige Angriffstechniken angesprochen, die sich speziell gegen Webanwendungen richten. Bei der Implementierung des Prototyps wurden von Anfang an potenzielle Sicherheitsrisiken angegangen, indem auf die im Django-Framework integrierten Schutzmechanismen zurückgegriffen wurde. Diese umfassen einen Schutz gegen XSS-, CSRF- und Clickjacking-Angriffe sowie Maßnahmen gegen SQL-Injektionen. Die serverseitigen Schutzmechanismen, zu denen auch das Aussperren potenziell unsicherer Webbrowser zählt, können in Abschnitt 7.2.2 nachgelesen werden.

Wie der Demonstrator unter Verwendung des Webservers Apache mit durch eine geeignete TLS-Konfiguration abgesichert werden kann, ist in den Abschnitten A.3.6 und 6.4.3 nachzulesen. Auch zu Demonstrationszwecken ohne einen Apache-Webserver kann durch eine verschlüsselte Datenübertragung zwischen dem Client und dem Entwicklungsserver Vertraulichkeit gewährleistet werden. Um dies zu erzielen, kann ein TLS Proxy oder ein SSH-Tunnel verwendet werden. Eine Hilfestellung zur Einrichtung dieser beiden Systeme befindet sich in den Abschnitten A.3.7.2 und A.3.7.1.

Die Datenbank des Demonstrators kann problemlos auf einen dedizierten Datenbankserver ausgelagert werden. Durch die Konfiguration einer verschlüsselten Verbindung zum Server werden die Nutzdaten und Serverinformationen verschlüsselt zwischen dem Demonstrator und dem Datenbankserver übertragen. Die Datenbank auf dem zentralen Server kann zusammen mit anderen Datenbanken regelmäßig gesichert werden. Damit ist eine hohe Sicherheit der Daten gewährleistet.

Zusammenfassend ist festzustellen, dass der Prototyp sämtliche Teilanforderungen aus der Kategorie Informationssicherheit erfüllt bzw. die notwendigen Schnittstellen bereitstellt. Aus diesem Grund gilt die Anforderung als *erfüllt*.

NF5: Integrierbarkeit in bestehende Umgebung

Das Tool soll die am häufigsten verwendeten Linux-Server-Distributionen unterstützen. Schnittstellen zu vorhandenen DBMSs müssen vorhanden sein. Die Integrierbarkeit bzw. Kompatibilität zu vorhandenen Tools ist erwünscht.

Zu den am häufigsten am LRZ verwendeten Linux-Server-Distribution zählt SLES bzw. openSUSE. Mit Abstand auf Platz zwei folgt Debian. Um eine hohe Kompatibilität des Demonstrators zu den Software- und System-Entwicklung (SuSE)-Servern zu gewährleisten, wurde als Entwicklungsplattform die zur Zeit aktuelle openSUSE-Distribution gewählt (siehe Abschnitt 7.2.1). Um Entwicklern potenzieller Erweiterungen das Aufsetzen der eigenen Entwicklungsumgebung zu vereinfachen, wurde zur Installation der erforderlichen Abhängigkeiten bevorzugt das Paketsystem von openSUSE verwendet. Die im Prototyp für ein mögliches Entwicklungssystem bereits vorbereiteten Funktionen zur Speicherung echter Paketlisten sind ebenfalls auf das

spezifische Format ausgerichtet, das das openSUSE-Distributionstool `zypper` bereitstellt. Der Import von Referenzlisten in diesem Format ist bereits im Prototyp möglich. Weiterhin wird gefordert, dass Schnittstellen zu bereits vorhandenen DBMSs existieren. Dass der Prototyp dies unterstützt, wurde zuvor schon festgestellt.

Als wünschenswerte Option wurde die Kompatibilität zu bereits vorhandenen Tools genannt. Da am LRZ selbstgebaute Tools zur Überwachung bestimmter Server-Zustandsinformationen zum Einsatz kommen, liegt es nahe, den Prototyp dazu kompatibel zu machen. Da allerdings nicht feststeht, wie es um die Zukunft dieser Tools aufgrund ihrer hohen Spezifität und des historischen Wachstums steht, ist die Erfüllung dieser funktionalen Anforderung zum aktuellen Zeitpunkt als hinfällig zu betrachten. Es hindert allerdings niemanden daran, den Prototyp dennoch in diese Richtung weiterzuentwickeln oder Teile davon in bereits vorhandene Systeme zu übernehmen.

In Anbetracht der Realisierbarkeit und Sinnhaftigkeit der geforderten Punkte, werden diese bei der prototypischen Anwendung als *erfüllt* angesehen.

Nr.	Kurzbeschreibung	Bewertung
F1	**Beherrschbarmachen großer Datenmengen:** Große Datenmengen sollen automatisch beschafft, konsolidiert und durch geeignete Methoden der Informationsvisualisierung beherrschbar gemacht werden.	teilweise erfüllt
F2	**Softwarestände überwachen** Auf den Servern installierte Paketversionen sollen zentral abrufbar sein. Anhand einer Referenzliste soll die Aktualität und die Sicherheit der Pakete bestimmt werden.	erfüllt
F3	**Firewall-Konfigurationen überwachen:** Die Wirksamkeit und Sicherheit der Firewall-Konfigurationen auf den Servern soll an zentraler Stelle ersichtlich sein. Einzelne Konfigurationsdateien sollen bei Bedarf abrufbar sein.	erfüllt
F4	**Sicherheit privater X.509-Schlüssel überwachen:** Die Sicherheit privater X.509-Schlüssel aller Server soll bewertet und an zentraler Stelle ersichtlich sein. Ein Zugriff auf die Schlüssel ist nicht erlaubt.	erfüllt
F5	**Sicherheit von TLS-Konfigurationen überwachen:** Die Sicherheit und Wirksamkeit der TLS-Konfigurationen aller Server soll bewertet und zentral angezeigt werden.	erfüllt
F6	**Datenschutzrechtliche Aspekte:** Das Tool muss Datenschutzrechtliche Regelungen umsetzen. Datensparsamkeit ist essentiell.	erfüllt
NF1	**Bedienfreundlichkeit:** Bedienfreundlichkeit soll erreicht werden, indem Tool dem Administrator das häufige Wechseln zu anderen Tools oder der Kommandozeile erspart. Es muss einfach und geräteübergreifend zu bedienen sein. Mehrsprachigkeit ist erwünscht.	teilweise erfüllt
NF2	**Reaktionszeit:** Als zentraler Anlaufpunkt soll das Tool sicherheitsrelevante Daten durch Informationsvisualisierung augenblicklich begreifbar machen und so eine geringe Reaktionszeit ermöglichen.	erfüllt
NF3	**Basistechnologien:** Durch den Verzicht auf proprietäre Basistechnologien sollen Unabhängigkeit geschaffen, Einstiegsbarrieren gesenkt und die Wartbarkeit der Anwendung erhöht werden.	erfüllt
NF4	**Informationssicherheit:** Das Tool soll grundlegende Sicherheitsfunktionen wie ein differenziertes Rechtesystem und LDAP-Authentifizierung implementieren. Webbasierte Anwendungen sollen insbesondere Schutzmaßnahmen gegen häufig durchgeführte Angriffstechniken bieten. Die Sicherheit der gespeicherten Daten ist zu gewährleisten.	erfüllt
NF5	**Integrierbarkeit in bestehende Umgebung:** Das Tool soll die am häufigsten verwendeten Linux-Server-Distributionen unterstützen. Schnittstellen zu vorhandenen DBMSs müssen vorhanden sein. Die Integrierbarkeit bzw. Kompatibilität zu vorhandenen Tools ist erwünscht.	erfüllt
	Gesamtbewertung	**erfüllt**

Tabelle 8.2.: Bewertung der funktionalen und nichtfunktionalen Anforderungen beim Prototyp.

8.3. Zusammenfassung

Die Bewertung der visuellen Anforderungen in Abschnitt 8.1 resultierte beim Prototyp in einem insgesamt sehr guten Ergebnis. Unabhängig davon, welchen Zweck die in Kapitel 5 bewerteten Administrationstools erfüllen, ist die Qualität der Bedienoberfläche und der Visualisierungen beim Prototyp höher. Dieses Ziel wurde erreicht, indem Grundlagen der Informationsvisualisierung bis auf wenige begründete Fälle konsequent umgesetzt wurden. Aufgrund der sehr guten Bewertung der visuellen Attribute bildet der Prototyp eine hervorragende Grundlage, um ihn zu einem Produktivsystem weiterzuentwickeln und am LRZ zur Überwachung der Linux-Server einzusetzen.

Neben seinen visuellen Qualitäten erfüllt der Prototyp auch einen Großteil der funktionalen und nichtfunktionalen Anforderungen, die in Kapitel 4 an ein Administrationstool zur Überwachung von Linux-Servern gestellt wurden. Hervorzuheben sind die Implementierungsdetails des Prototyps, die zur Verbesserung der Informationssicherheit beitragen (siehe Abschnitt 8.2). Sie sprechen ebenfalls für den Prototyp als Basis für ein mögliches Produktivsystem.

Der Prototyp zeigt, dass sich Ben Shneidermans Information Seeking Mantra zur Bewältigung unüberschaubarer Datenmengen (siehe Abschnitt 6.1.2) auch auf die Administration von Linux-Servern gewinnbringend anwenden lässt. Das interaktive zoombare Interface des Demonstrators ermöglicht einerseits die schnelle Bestimmung von Servern, bei denen Handlungsbedarf besteht, und lässt sich andererseits auch für proaktive Zwecke einsetzen. Das ist eine wichtige Eigenschaft, um die Informationssicherheit der Server zu optimieren, bevor konkrete Probleme auftreten.

9. Fazit und Ausblick

In diesem Kapitel werden die Ergebnisse zusammengefasst, die im Rahmen dieser Masterarbeit erzielt wurden. Basierend auf den erarbeiteten Resultaten und in Hinblick auf eine Produktivsetzung des entwickelten Prototyps werden anschließend weitere Ausbaustufen sowie Themen für Folgearbeiten diskutiert.

9.1. Zusammenfassung der Ergebnisse

Zu den ersten Ergebnissen, die aus dieser Masterarbeit resultierten, zählt der Grundlagenteil. Zunächst wurde in Kapitel 2 eine Zusammenfassung wichtiger Grundlagen der Informationsvisualisierung geschaffen. Dabei wurde Rücksicht auf die Verwendbarkeit dieser Zusammenfassung im weiteren Vorgehen bei der Anfertigung der Masterarbeit genommen: Es wurden einerseits Design-Grundregeln vorgestellt, die für Diagramme und dynamisch generierte Visualisierungen gelten (siehe Abschnitt 2.7), andererseits wurden Richtlinien für das Design von grafischen Bedienoberflächen insbesondere in Hinblick auf die zielführende Verwendung von Farben angesprochen (siehe Abschnitt 2.13). Zur Visualisierung von serverbezogenen Daten ist es ebenfalls wichtig, Methoden zum Umgang mit multivariaten Daten zu beherrschen. Diese wurden im ersten Grundlagenteil im Abschnitt 2.12 angesprochen. Wichtige Konzepte zur Interaktion mit Visualisierungen wurden in Abschnitt 2.14 und Methoden zur Präsentation umfangreicher Datenmengen in Abschnitt 2.17 vorgestellt. Diese Grundlagen erwiesen sich später als nützlich.

Nach dem Grundlagenteil, der sich mit Informationsvisualisierung befasst, folgt ein weiterer Grundlagenteil mit dem Schwerpunkt Informationssicherheit im Kapitel 3. Die Motivation für dieses Kapitel liegt darin begründet, dass Informationen über Server-Zustände, die als Grundlage zur Visualisierung dienen, einen starken Bezug zur IT-Sicherheit haben. Ein Informationssystem, das auf diese Daten zugreift, muss einen vertraulichen Umgang mit diesen Informationen sicherstellen. Ebenso dürfen auch die anderen Grundpfeiler der Informationssicherheit, nämlich Integrität und Verfügbarkeit, nicht vernachlässigt werden. Kapitel 3 geht in Abschnitt 3.1 auf die essentiellen Arten von Bedrohungen und Angriffen ein, die von Kriminellen ausgehen. Da der im Rahmen dieser Masterarbeit zu entwickelnde Prototyp eine Webanwendung werden sollte, wurden dort auch spezielle Bedrohungen angesprochen, denen webbasierte Applikationen am häufigsten ausgesetzt sind. Dieses Kapitel befasst sich weiterhin mit potenziellen Bedrohungen und Angriffen (siehe Abschnitt 3.2), denen das LRZ potenziell ausgesetzt ist und stellt sowohl Sicherheitsziele in Abschnitt 3.3 als auch Maßnahmen in Abschnitt 3.4 vor, um die Sicherheitsziele effektiv in die Tat umzusetzen.

Am LRZ wird bereits eine Vielzahl an unterschiedlichen Administrationstools eingesetzt. Häufig handelt es sich nicht um zentrale Systeme, sondern um Insellösungen, die von einigen Administratoren bevorzugt zur Verwaltung ihrer Server verwendet werden. Zur Aufgabenstellung dieser Masterarbeit zählte es, Methoden zur Visualisierung sicherheitsbezogener Server-Zustandsinformationen zu entwickeln und Ergebnisse in einem Prototyp zu realisieren. Dies setzte eine Analyse der Anforderungen voraus, die speziell am LRZ an ein Administrationstool gestellt werden. Auch wenn der Prototyp nicht als Produktivsystem vorgesehen war, sollten die Ergebnisse dieser Anforderungsanalyse soweit wie möglich in seinen Entwurf und die Implementierung einfließen. Die Anforderungsanalyse, die im Rahmen dieser Masterarbeit entstand und in Kapitel 4 dieser Ausarbeitung eingeflossen ist, gliedert sich in zwei Teile: Zunächst wurde in Abschnitt 4.1 anhand des zum Zeitpunkt der Anfertigung aktuellen Jahresberichts eine Übersicht der vom LRZ bereitgestellten Dienste und der zur Diensterbringung verwendeten Server erstellt. Im Anschluss wurden konkrete Anforderungen an ein Administrationstool zur Überwachung von Linux-Servern zusammengetragen, die schließlich kategorisiert wurden. Das Resultat fand seinen Weg in Abschnitt 4.2.

Wie bereits zuvor erwähnt wurde, werden am LRZ viele unterschiedliche Administrationstools eingesetzt. Sie greifen zum Teil auf Methoden der Informationsvisualisierung zurück, um dem Administrator Serverzustände

zu vermitteln. Bevor jedoch eigene Methoden zur Visualisierung sicherheitsbezogener Zustandsinformationen von Servern in dieser Arbeit entwickelt wurden, fand zunächst eine Betrachtung bereits existierender Tools statt (siehe Kapitel 5). Neben einigen Anwendungen wie *Ansible* und *Splunk*, die am LRZ Verwendung finden, wurden aus unterschiedlichen Kategorien repräsentative Administrationstools anhand ihrer visuellen Qualitäten evaluiert. Es wurde abschließend ein Überblick erstellt, der die Verwendbarkeit der Tools aufgrund ihrer Visualisierungsmethoden bewertet. Bei der Evaluierung wurde auf die Anforderungen an ein Administrationstool zurückgegriffen, die zuvor im Kapitel 4 zusammengetragen wurden. Da sich diese Arbeit hauptsächlich mit Visualisierung befasst, wurden die Anwendungen auch nur anhand ihrer visuellen Qualitäten bewertet.

Das Ergebnis der Evaluation fiel insgesamt folgendermaßen aus: Die Qualität der Visualisierungen und grafischen Bedienoberflächen der Tools ist sehr unterschiedlich. Es gibt Tools, die Grundlagen der Informationsvisualisierung gut umsetzen und übersichtliche Bedienoberflächen besitzen. Diese Tools eignen sich gut für den Einsatz in der Linux-Server-Administration, da sie auf effiziente Art und Weise sicherheitsrelevante Informationen visualisieren. Andererseits gibt es Tools, deren Aufgabe es ist, sicherheitsrelevante Parameter von Servern zu überwachen, sie erschweren dem Administrator aber die Arbeit aufgrund visueller Defizite. Häufige Kritikpunkte sind unübersichtliche, überladene Bedienoberflächen und Diagramme, eine schlechte Wahl von Farben, zu geringe Kontraste und fehlende Beschriftungen. In einem Fall waren Diagramme dermaßen unübersichtlich, dass sie überhaupt nicht mehr zuverlässig interpretierbar waren. Im sicherheitsrelevanten Kontext der Linux-Server-Administration ist dies fatal, da der Administrator wichtige Informationen nicht visuell vermittelt bekommt oder er sie falsch abliest. Es ist auch nicht davon auszugehen, dass ein Administrator gerne regelmäßig mit einem Tool arbeitet, das eine qualitativ schlechte Oberfläche und unübersichtliche Visualisierungen besitzt. Wenn er das Tool nur ungern nutzt, ist auch keine Grundlage für den zielführenden Einsatz und für das Treffen vernünftiger Entscheidungen gegeben. Bei einigen Programmen, die bei der Evaluation gut abschnitten, stellte sich während der Recherche heraus, dass dort Grundlagen der Informationsvisualisierung umgesetzt wurden, die in der Vorgängerversion noch fehlten. Es wurde also bewusst Arbeit in übersichtlichere Visualisierungen und nicht nur in neue Programmfunktionen gesteckt. Dennoch gibt es auch bei den Anwendungen mit den übersichtlichsten Oberflächen und Visualisierungen noch Optimierungspotenzial, denn die beste Gesamtnote war lediglich *gut*.

In Kapitel 6 wurden schließlich Anwendungsfälle aus der Linux-Server-Administration ausgewählt, die einerseits häufig vorkommen, andererseits einen starken Bezug zur Informationssicherheit aufweisen. Ziel war es von Anfang an, ein gemeinsames Schema zur Visualisierung der Serverzustände zu entwickeln, das für alle Anwendungsfälle eingesetzt werden kann. Zunächst wurde für den praxisrelevanten Anwendungsfall der Software-Verwaltung erörtert, welche Informationen für den Administrator essentiell sind und sich durch Methoden der Informationsvisualisierung möglichst einfach aber auch wirksam darstellen lassen. Zur Visualisierung aggregierter Zustandsinformationen wurde auf bewährte Diagrammtypen zurückgegriffen. Zusätzlich wurde ein für den Kontext der Linux-Server-Administration innovatives und interaktives Visualisierungskonzept entwickelt, das auf einer *zoombaren Bedienoberfläche* (siehe Abschnitt 2.17.2) basiert und mit Hilfe von *Dynamic Queries* (siehe Abschnitt 2.14.3) und *konstanter Informationsdichte* (siehe Abschnitt 2.12.7) einen digitalen Sortiertisch für Linux-Server realisiert. Es wurde gezeigt, dass Ben Shneidermans bewährtes Information Seeking Mantra (siehe Abschnitt 6.1.2) auch auf die unüberschaubaren Datenmengen hunderter Linux-Server übertragen werden kann und dass Administratoren das resultierende Visualisierungsschema nicht nur reaktiv, sondern auch proaktiv zielführend einsetzen können, um einen Beitrag zur Informationssicherheit am Rechenzentrum zu leisten. Beim Entwurf der Visualisierungen in Kapitel 6 wurde auf die Grundlagen zurückgegriffen, die zuvor im ersten Teil dieser Ausarbeitung zusammengetragen wurden.

Eine weitere Leistung, die in der Aufgabenstellung gefordert wurde, war die Implementierung einer prototypischen Webanwendung. Sinn und Zweck des Prototyps sollte die Demonstration der Umsetzbarkeit und Wirkung der im theoretischen Teil dieser Arbeit entwickelten Visualisierungsmethoden sein. Als Anschauungsobjekt wurde in Abschnitt 7.1 der Anwendungsfall der Software-Verwaltung aufgrund der Alltagsrelevanz für Administratoren gewählt. Die Entwicklung und Implementierung begann nicht zeitlich nach der Fertigstellung des theoretischen Teils, sondern begleitete das Voranschreiten der Masterarbeit schon deutlich früher. Zunächst wurden Basistechnologien ausgewählt, die sich einerseits zur Implementierung eines Prototyps eignen, andererseits auch bei der Weiterentwicklung zu einem Produktivsystem weiterverwendet werden können. Ein auf Übersichtlichkeit und Erweiterbarkeit ausgelegtes Datenmodell wurde schließlich entwickelt (siehe Abschnitt 7.2.4). Mit Hilfe des Python-basierten Frameworks *Django* auf der Serverseite und auf *jQuery* basierenden JavaScript-Bibliotheken auf der Clientseite wurde zunächst ein Grundgerüst für eine webbasierte

Anwendung geschaffen, das für den Mehrbenutzerbetrieb mit differenzierten Zugriffsrechten ausgelegt ist. Dabei wurde hoher Wert auf Wiederverwendbarkeit und Barrierefreiheit gelegt. Nachdem das Grundgerüst fertiggestellt war, wurde es Schritt für Schritt erweitert, um repräsentative Visualisierungsmethoden aus Kapitel 6 zu demonstrieren. Beim Bau des Prototyps erwies sich die Evaluation existierender Administrationstools in Kapitel 5 als nützlich, da die betrachteten Programme sowohl vorbildhafte als auch nicht nachahmenswerte Vorlagen lieferten. Da der Schwerpunkt des Themas in der Visualisierung von Serverdaten liegt und nicht in der Beschaffung, wurde der Entwicklungsaufwand mehr in die Visualisierungskomponenten als in die Beschaffung der zugrundeliegenden Daten gesteckt. Der Prototyp greift deshalb auf Zufalls- und vordefinierte Beispieldaten zurück, Vorarbeit für die Integration realer Daten wurde aber sowohl theoretisch im zuvor genannten Kapitel als auch praktisch in Form des Datenmodells und durch einsatzbereite Bedienoberflächen im Prototyps geleistet. Nach der Implementierung der Visualisierungen im Prototyp erfolgte eine Dokumentation der Programmfunktionen, die relevant für Informationssicherheit sind und der Funktionen, die die eigentlichen Visualisierungen beinhalten.

Im Anschluss erfolgte eine weitere Evaluation. Diesmal rückten aber nicht fremde Systeme, sondern der Prototyp selbst in den Vordergrund. Als Grundlage für die Evaluierung wurde erneut auf die Ergebnisse der Anforderungsanalyse aus Kapitel 4.2 zurückgegriffen. Neben der Umsetzung von Anforderungen, die aus den Grundregeln der Informationsvisualisierung resultieren, wurden dort auch funktionale und nichtfunktionale Anforderungen betrachtet.

Neben visuellen Referenzimplementierungen von Standarddiagrammen beinhaltet der Demonstrator mit der zuvor vorgeschlagenen zoombaren Bedienoberfläche ein interaktives Visualisierungsschema, das in keinem der anderen Administrationstools in vergleichbarer Form vorhanden ist. Anhand der Implementierung wurde gezeigt, dass Ben Shneidermans Information Seeking Mantra zur Bewältigung unüberschaubarer Datenmengen (siehe Abschnitt 6.1.2) auch auf den Bereich der Linux-Server-Administration übertragbar ist und dass es Sinn macht, Visualisierungen für diesen sicherheitsrelevanten Einsatzzweck zu entwickeln. Die zoombare Oberfläche greift auf die aussagekräftigsten visuellen Attribute zurück, die Visualisierungen bereitstellen können: Dazu zählen die Position von Objekten und präattentiv wahrnehmbare Signalfarben. Durch die Objektposition lassen sich Werte besonders präzise visualisieren, die präattentiv wahrnehmbaren Farben lenken die Aufmerksamkeit des Betrachters schnellstmöglich zu wichtigen Objekten. Übertragen auf die Linux-Server-Administration können Systeme, bei denen mehr oder weniger dringender Handlungsbedarf besteht, derart farblich kodiert werden, dass ein augenblicklicher Gesamteindruck über die Serversicherheit entsteht. Zu einer genauen Vergleichbarkeit von Zustandsinformationen nutzt der Demonstrator zweidimensionale Positionen. Sukzessive Filteroperationen ermöglichen dem Administrator den Fokus auf die Server, die ihn gerade am meisten interessieren. Mit Hilfe der Vergrößerungsfunktion behält er die Objekte besser im Blick. Die stufenlosen und absolut flüssigen Zoom-Operationen in der Implementierung tragen zu einem Immersionseffekt bei, der die Aufmerksamkeit des Anwenders wirksam auf die Visualisierung lenkt. Durch die gezielte Abfrage der Detailinformationen zu einzelnen Servern bleibt der Administrator von unnötigen Daten verschont. Im Endeffekt erhält er genau die und ausschließlich die Informationen, die er von seinen Servern benötigt, um darauf basierend weitere Schritte einzuleiten. Damit spart er wertvolle Zeit.

Als einziges Programm, das im Rahmen dieser Arbeit betrachtet wurde, erhielt der Prototyp für die Umsetzung der visuellen Anforderungen die Gesamtnote *sehr gut*. Daran ist zu erkennen, dass die Qualität der Visualisierungen und der Bedienoberflächen von Administrationstools zugunsten des Administrators optimiert werden kann, indem konsequent Grundlagen der Informationsvisualisierung umgesetzt werden. Dass gegen bestimmte Grundregeln in wohlbegründeten Fällen auch verstoßen werden darf, zeigte sich bei der Evaluation ebenfalls. Tools mit einer hohen visuellen Qualität leisten dem Administrator bessere Dienste im Bereich der Linux-Server-Administration, da sie sicherheitsrelevante Informationen in einer effizienten und schnell ablesbaren Form auf den Bildschirm bringen und den Anwender nicht von den wesentlichen Fakten ablenken.

Mit Hilfe der Ergebnisse dieser Masterarbeit konnte belegt werden, dass es sowohl möglich als auch sinnvoll ist, Methoden der Informationsvisualisierung im Kontext der Linux-Server-Administration einzusetzen. Die Grundvoraussetzungen hierfür sind durch die große Menge an Rohdaten und die automatisierbare, algorithmische Verarbeitung zu Visualisierungen gegeben. Bewährte Konzepte aus dem Bereich der Informationsvisualisierung wie zum Beispiel Ben Shneidermans Information Seeking Mantra zur Bewältigung unüberschaubarer Datenmengen (siehe Abschnitt 6.1.2) können erfolgreich auch in diesem Kontext angewendet werden, um potenziell die Sicherheit bei der Linux-Server-Administration zu erhöhen. Dass der Einsatz von Visualisierungen in diesem Kontext in der Praxis möglich ist, wurde durch die für die Evaluation ausgewählten

Tools und durch den selbst entwickelten Prototyp gezeigt. Sinnvoll ist der Einsatz ebenfalls, da die Administrationstools sicherheitsrelevante Informationen durch Visualisierungen im Allgemeinen übersichtlicher und prägnanter darstellen.

Die bloße Existenz von Visualisierungen führt allerdings nicht automatisch zu einem besseren Administrationstool und damit zu einer höheren Sicherheit.[1] Der Grund liegt darin, dass Visualisierungen nicht automatisch übersichtlich sind und dass aufgrund der Vielfalt der Möglichkeiten, die visuelle Darstellungen bieten, Fehlgriffe möglich sind. Es ist erstrebenswert, auf etablierten Grundlagen der Informationsvisualisierung zurückzugreifen, um sowohl die Bedienfreundlichkeit von Benutzeroberflächen als auch den Informationsgehalt von Visualisierungen zu optimieren. Einerseits macht es Sinn, altbekannte Visualisierungsformen wie zum Beispiel Säulendiagramme bezüglich ihrer Darstellungsqualität zu optimieren, da dies zu einer höheren Übersichtlichkeit und einer effizienteren Vermittlung sicherheitsrelevanter Informationen von Linux-Servern führt. Andererseits wurde auch gezeigt, dass durch die Anwendung dieser Grundlagen Visualisierungen geschaffen werden können, die im Kontext der Linux-Server-Administration neu sind und durch ihre Interaktionsmöglichkeiten einen Mehrwert gegenüber den herkömmlichen Visualisierungsformen bieten.

9.2. Mögliche Ausbaustufen und Folgearbeiten

Der Demonstrator, der im Rahmen dieser Masterarbeit implementiert wurde, legt den Schwerpunkt auf die Umsetzung einer im Kontext der Linux-Serveradministration neuen Visualisierungsmethode. Bei den Daten, die als Grundlage zur Visualisierung dienen, handelt es sich um fiktive Werte, die in der Datenbank abgelegt wurden und zum Teil auch um Zufallswerte, die bei jedem Aufruf vom Server neu generiert werden. Nachdem gezeigt wurde, dass sich die implementierte Technik der Visualisierung im sicherheitsrelevanten Kontext der Linux-Serveradministration gewinnbringend einsetzen lässt, ist es eine logische Konsequenz, in einer weiteren Ausbaustufe den Prototyp mit echten Daten zu beliefern.

Beim Software Design wurden diese mögliche weitere Ausbaustufen stets im Hinterkopf behalten. Der Client Code muss prinzipiell nicht geändert werden, um echte Daten von Servern zu visualisieren. Die AJAX-Schnittstellen sind ebenfalls in ihrer bestehenden Form dazu geeignet, auch echte Zustandsinformationen von Linux-Servern zu übermitteln. Falls das Produktivsystem zusätzliche Informationen visualisieren soll, sind Änderungen am Programmcode hauptsächlich in Form von Ergänzungen zu tätigen, die sich an den bestehenden Implementierungen orientieren können. Dies gilt auch für das erarbeitete Datenbankschema.

In einem Produktivsystem mag es dennoch Gründe geben, Änderungen am Datenbankschema des Prototyps vorzunehmen. Ein Grund hierfür besteht in der Einschränkung des objektrelationalen Mappers von Django, keine zusammengesetzten Primärschlüssel und keine primärschlüssellosen Tabellen out of the box zu unterstützen. Falls die Implementierung mit Basistechnologien erfolgt, die diese Einschränkungen nicht besitzen, ist eine Anpassung des Schemas sogar ratsam, da das DBMS dann selbst in der Lage ist, bestimmte Prüfungen durchzuführen, die ansonsten im Programm selbst erfolgen müssten. Ein weiterer Grund für eine Änderung des Schemas mögen implementierungsspezifische Performanceoptimierungen darstellen.

Wenn ein Produktivsystem entwickelt werden soll, das echte Zustandsinformationen von den Servern erhält, stellt sich auch die Frage, wie sie ins System gelangen. Das manuelle Eintippen der Informationen über die Oberfläche des Programms ist nicht zielführend und vermutlich würde sich auch niemand freiwillig dazu bereit erklären. In Kapitel 6 wurden bereits Vorschläge gemacht, wo genau die sicherheitsrelevanten Informationen zu suchen sind. In diesem Zusammenhang wurde auch von einem Programm auf den Servern gesprochen, das entweder speicherresistent ist oder per Cronjob regelmäßig gestartet wird, um Informationen zu liefern. Egal für welche Variante sich die Entwickler eines Produktivsystems entscheiden, müssen sie unbedingt gewährleisten, dass eine Datenübertragung nur verschlüsselt und nach erfolgreicher Authentifizierung erfolgt. Die Untersuchung dieser Datenbeschaffung und ihrer Automatisierbarkeit unter Berücksichtigung von Informationssicherheit wäre ein geeignetes Thema für eine andere Abschlussarbeit.

[1] Dass die bloße Existenz eines Werkzeugkastens automatisch zur Problemlösung führt, scheint auch bei Politikern ein verbreiteter Irrglaube zu sein. Immer wieder erfährt man zum Beispiel von aktionistischen Volksvertretern, die sich allein durch die Bereitstellung von Computern an Schulen einen höheren Lernerfolg versprechen. Dass solche Bestreben auch nach hinten losgehen können, zeigt ein aktueller Fall aus Kalifornien [BKR 14].

Die visualisierenden Maßnahmen des Prototyps basieren auf den Vorschlägen aus Abschnitt 6.1, wo die Software-Verwaltung angesprochen wurde. Im Da im Kapitel 6 noch drei weitere sicherheitsrelevante Anwendungsfälle diskutiert und Vorschläge zur Beschaffung der benötigten Informationen und ihrer Visualisierung unterbreitet wurden, liegt es nahe, diese Anwendungsfälle auch in einem Wirksystem zu implementieren. Zuvor muss allerdings abgewägt werden, welche dieser Use Cases im Wirksystem vorrangig benötigt werden.

Die zoombare Bedienoberfläche mit Dynamic Queries, die bei jedem der in dieser Arbeit diskutierten Anwendungsfälle wiederkehrt, ist prinzipiell so universell ausgelegt, dass sie sich auch zur Visualisierung beliebiger anderer sicherheitsrelevanter Serverdaten eignet. Es ist denkbar, zusätzliche Filter und weitere Sortierkriterien für die Koordinatenachsen einzuführen. Ebenso ist es denkbar, zusätzliche visuelle Attribute in die Visualisierung zu integrieren, da der Demonstrator der Einfachheit halber nur auf Kreise zurückgreift.

Bei der prototypischen Implementierung wurde das Konzept verfolgt, dass einzelne Server von einem Administrator als *inaktiv* gekennzeichnet werden können. Sie tauchen dann gewissermaßen nur noch als Karteileichen in der Datenbank auf, in den Visualisierungen sind sie nicht mehr sichtbar. In einem Wirksystem ist ggf. nicht nur für den Administrator des Informationssystems, sondern auch für seine Anwender von Interesse, welche Server nicht mehr im Einsatz sind. Dementsprechend macht es Sinn, auch deaktivierte Server optional in die Visualisierungen aufzunehmen.

Die räumliche Anordnung der Server-Piktogramme in der zoombaren Oberfläche muss in einem Produktivsystem nicht zwangsweise nach zwei an den Bildachsen angetragenen Sortierkriterien erfolgen. Eine logische Gruppierung ist genauso denkbar wie eine benutzerspezifische räumliche Organisation der Server. Der Benutzer sollte in der Lage sein, individuell zusammengestellte Ansichten und auch Filteroptionen abzuspeichern und bei Bedarf wieder abzurufen. Dadurch findet er sich besser zurecht und kommt schneller ans Ziel.

Durch die Einbindung von Direct Manipulation (siehe Abschnitt 2.14.2) sind weitere Interaktionen mit den einzelnen Servern denkbar, als lediglich Zustandsinformationen abzufragen. Die Anwendung ließe sich dadurch um Funktionen bereichern, die bereits von anderen Tools aus dem Bereich Konfigurationsmanagement umgesetzt werden. Dazu zählt zum Beispiel die Installation von Updates oder das Neustarten eines Servers. Durch die Integration solcher Funktionen und den Ausbau zu einem Konfigurationsmanagement-Tool lässt sich der ständige Wechsel zwischen mehreren Anwendungen vermeiden und der Anwender muss nicht ständig umdenken. Zudem sind kürzere Reaktionszeiten gewährleistet, die aus Sicht der Informationssicherheit entscheidende Vorteile bieten können. Zur Interaktion mit einer größeren Menge von Servern sind Stellvertreterobjekte eine mögliche Lösung.

Um noch mehr Informationen in der zoombaren Oberfläche unterzubringen, ohne unnötig viel Visual Clutter zu schaffen, ist es auch denkbar, den Detailgrad der angezeigten Piktogramme von der momentan ausgewählten Vergrößerungsstufe abhängig zu machen. Dies macht auch aus Performance-Gründen Sinn, da der Computer in einer Totalansicht dann nicht tausende Details vom Server abfragen und auf die Ausgabefläche zeichnen muss, die so klein sind, dass man sie ohnehin nicht entziffern kann. Durch das sukzessive Einblenden von Zusatzinformationen in Abhängigkeit vom Vergrößerungsgrad durch ästhetische Animationen wird auch der Immersionseffekt verstärkt. Dies führt dazu, dass der Administrator das Tool wirksamer einsetzen kann. Bei der Implementierung einer Visualisierung mit konstanter Informationsdichte können sich die Entwickler an einem Vorschlag von Allison Woodruff et al. ein Beispiel nehmen [WOOD+ 98].

Obwohl der Prototyp von Anfang an auf eine leichte Erweiterbarkeit ausgelegt wurde, ist es nicht auszuschließen, dass bei der Entwicklung eines davon inspirierten Produktivsystems auf andere Basistechnologien zurückgegriffen wird. Dazu zählen zum Beispiel die verwendeten Programmiersprachen. Ein Grund hierfür mag sein, dass die Entwickler bereits viel Erfahrung mit anderen Basistechnologien gesammelt haben oder es gibt strenge Vorgaben vom Auftraggeber, die einzuhalten sind.[2] Solange das Produktivsystem ebenso wie der Prototyp eine Webanwendung werden soll, ist selbst im Fall geänderter Basistechnologien die Wiederverwendung von Code möglich: Die dynamisch generierten Visualisierungen können weiterhin auf denselben JavaScript-Bibliotheken aufbauen. Der bereits vorhandene Client-Programmcode kann den HTML-Template-Dateien entnommen und in die eigenen Templates oder JavaScript-Dateien aufgenommen werden. Das liegt daran, dass HTML5 der De-Facto-Standard auf der Client-Seite ist und jQuery zu den populärsten JavaScript Frameworks zählt.

[2] Es gibt tatsächlich Auftraggeber, die den Einsatz von Visual Basic fordern, um ein Beispiel zu nennen.

Die Kommunikation zwischen Client und Server findet beim Prototyp im Sinne einer klassischen Webanwendung statt: Der Anwender fordert eine Webseite an, der Server generiert sie mit den entsprechenden Inhalten und sendet sie an den Client zurück. Auf Webseiten, die Visualisierungen beinhalten, nutzt der Prototyp AJAX, um Zustandsinformationen der Server dynamisch nachzuladen, ohne komplette Webseiten erneut anzufordern. Verzögerungen in der Bedienoberfläche werden so minimiert und gleichzeitig sinkt die Last im Netz und auf dem Server.

Falls ein Produktivsystem entwickelt werden soll, ist es eine Überlegung wert, den Client noch AJAX-lastiger zu machen, um die Datenmenge, die für die Übertragung von redundantem HTML Code benötigt wird, auf ein Minimum zu reduzieren und nur noch Nutzdaten in einem effizienten Format wie JSON zu übertragen. Durch den hohen AJAX-Anteil ist eine nahezu verzögerungsfreie Bedienoberfläche realisierbar. Dieses Konzept setzt die Implementierung einer REST-Schnittstelle auf Client- und Serverseite voraus. Dadurch werden Client und Server stark voneinander entkoppelt und werden so austauschbar. Nach der Spezifikation der REST-Schnittstelle können unterschiedliche Entwickler parallel am Client und am Server arbeiten. Dieser Ansatz würde einen im Vergleich zum Demonstrator hohen JavaScript-Anteil voraussetzen, um die Bedienoberfläche komplett auf dem Client zu generieren. Ggf. ist hierzu die Verwendung eines Frameworks notwendig, das auf diesen Anwendungsfall spezialisiert ist, wie zum Beispiel *Sencha Ext JS*[3].

Der Prototyp ist für den Betrieb auf einem Webserver mit der HTTP-Protokollversion 1.1 ausgelegt. Bei möglichen Erweiterungen können Entwickler hier über den Tellerrand blicken und auf Funktionen der vor kurzem standardisierten Version 2.0 zurückgreifen [BRI 15]. Bei HTTP/2 ist der Server in der Lage, Push-Nachrichten an den Client zu schicken, ohne dass der Client zuvor einen Request an ihn gesendet hat. Die Push-Funktion lässt sich beispielsweise nutzen, um Änderungen an den ermittelten Serverzuständen unmittelbar an die eingeloggten Anwender weiterzuleiten. Per Server Push kann der Client auch dazu bewegt werden, unmittelbar den Logout-Bildschirm aufzurufen, nachdem die Benutzersitzung aus der Ferne von einem Administrator über die Sitzungsverwaltung (siehe Abschnitt 7.3.3) beendet wurde. Potenziell sicherheitskritische Informationen verschwinden dann sofort vom Bildschirm der betroffenen Person.

Ein weiterer denkbarer Ansatz für ein Produktivsystem basiert nicht auf dem Ausbau des Prototyps selbst, sondern auf der Integration der informationsvisualisierenden Maßnahmen in ein bereits bestehendes Administrationstool. An einem Standort wie dem LRZ werden bereits solche Tools eingesetzt. Teilweise handelt es sich um Eigenentwicklungen. Sofern ein solches Tool in der Lage ist, in ausreichendem Maß Daten zu liefern, die als Grundlage zur Visualisierung verwendet werden können, spricht nichts gegen eine Integration der im Demonstrator vorgestellten Visualisierungen in das bereits vorhandene System. Speziell die im Prototyp verwendeten JavaScript-Bibliotheken zeichnen sich dadurch aus, dass sie gut in vorhandene Webseiten integriert werden können. Durch die konsequente Verwendung des jQuery-Namensraums sind Konflikte mit anderen eingebundenen Frameworks oder Bibliotheken unwahrscheinlich. Sollte es dennoch zu Problemen mit fremden Bibliotheken kommen, schlagen die jQuery-Entwickler auf ihrer Website einige Lösungsmöglichkeiten vor.[4]

[3] Ext-JS-Website: `https://www.sencha.com/products/extjs/`

[4] jQuery-Konflikte mit anderen Bibliotheken vermeiden: `https://learn.jquery.com/using-jquery-core/avoid-conflicts-other-libraries/`

A. Anhang

A.1. Abbildungen

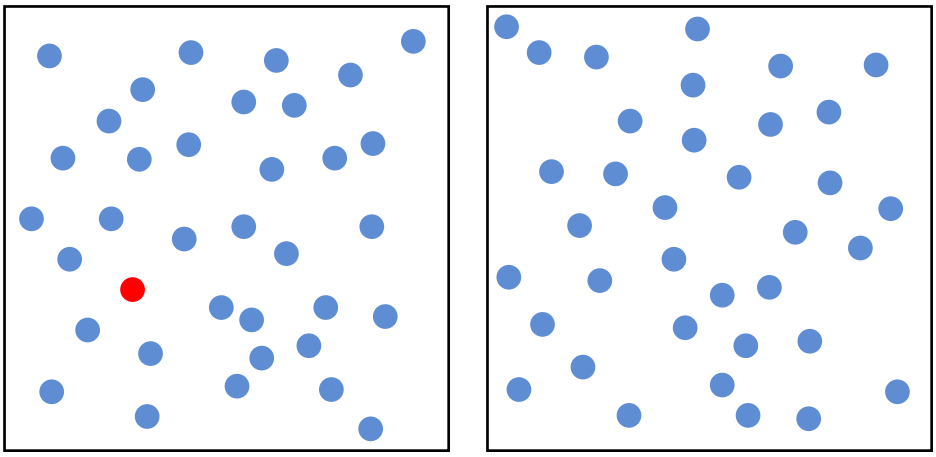


Abbildung A.1.: Farbe wird präattentiv wahrgenommen.

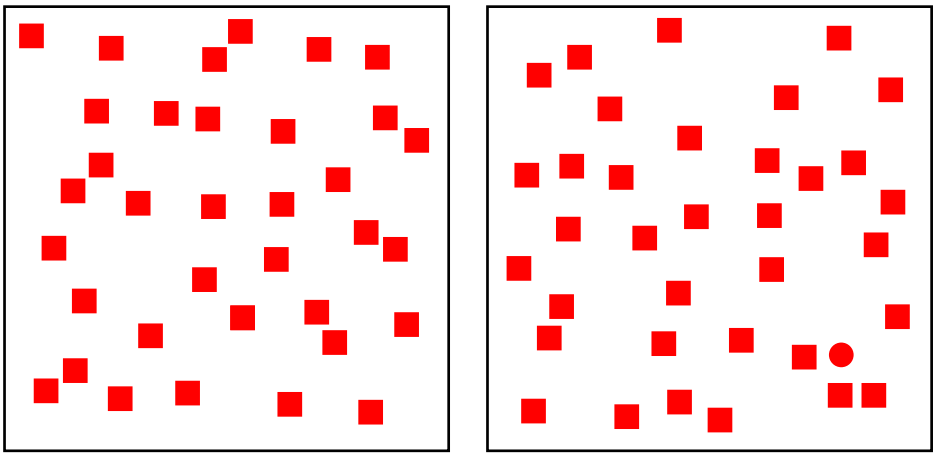


Abbildung A.2.: Ist ein roter Kreis in dem Bild?

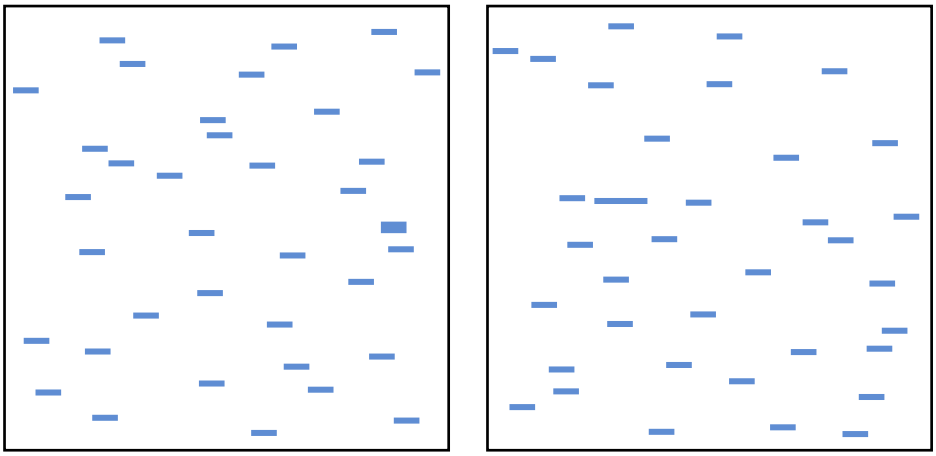


Abbildung A.3.: Längen und Breiten werden präattentiv wahrgenommen.

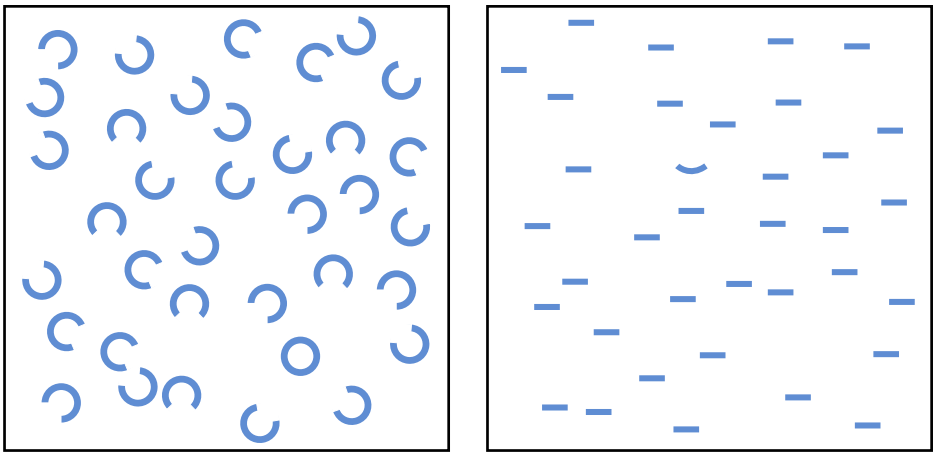


Abbildung A.4.: Abgeschlossenheit und Krümmungen werden präattentiv wahrgenommen.

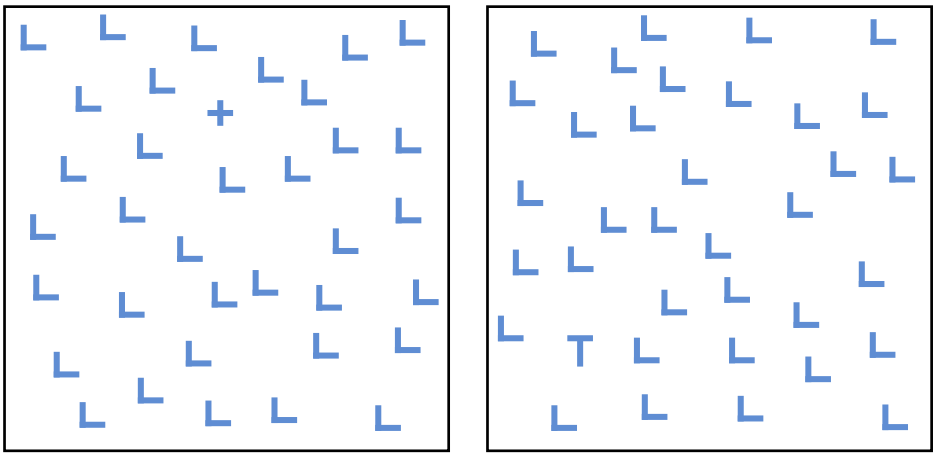


Abbildung A.5.: Überschneidungen und Abschlüsse werden präattentiv wahrgenommen.

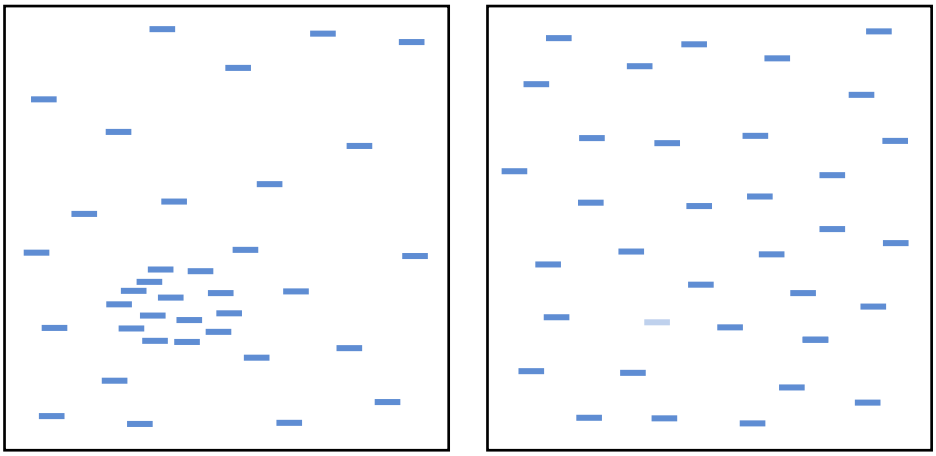


Abbildung A.6.: Häufungen und Helligkeit werden präattentiv wahrgenommen.

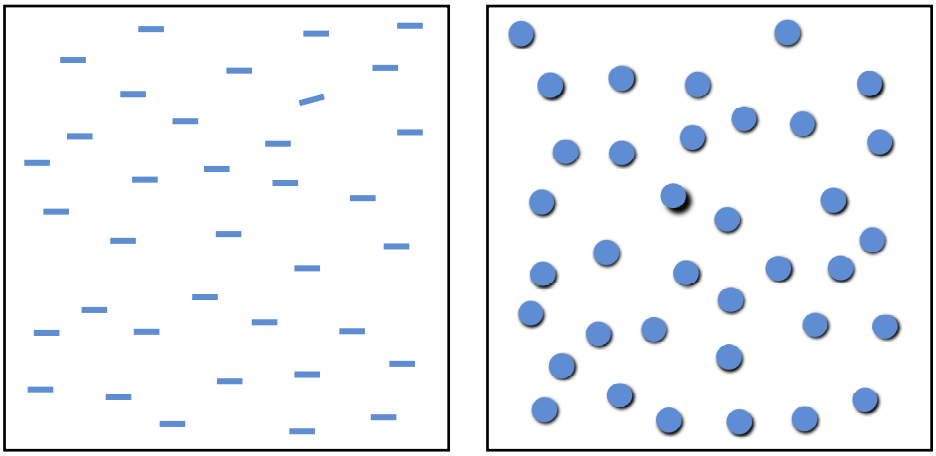


Abbildung A.7.: Die Wahrnehmung von Ausrichtungen und räumlicher Tiefe erfolgen präattentiv.

A.2. Quelltexte

Listing A.1: Basis-Template für alle Webseiten

```
1  {% load i18n %}
2  {% load staticfiles %}
3  <!DOCTYPE html>
4  <html>
5          <head>
6                  <title>{{ application_name }}</title>
7                  <meta charset="UTF-8" />
8                  <link rel="stylesheet" type="text/css" href="{% static '
                       provisional/css/main.css' %}" />
9                  <link rel="stylesheet" type="text/css" href="{% static '
                       provisional/js/jquery-ui-1.11.2.custom/jquery-ui.css' %}" />
10                 <link rel="stylesheet" type="text/css" href="{% static '
                       provisional/js/jquery-tablesorter-2.0/themes/blue/style.css'
                       %}" />
11                 <script type="text/javascript" src="{% static 'provisional/js/
                       jquery-2.1.3/jquery-2.1.3.js' %}"></script>
```

```
12                  <script type="text/javascript" src="{% static 'provisional/js/
                        jquery-ui-1.11.2.custom/jquery-ui.js' %}"></script>
13                  <script type="text/javascript" src="{% static 'provisional/js/
                        jquery-tablesorter-2.0/jquery.tablesorter.js' %}"></script>
14          </head>
15          <body>
16                  {% block base_content%}{% endblock %}
17
18                  <!-- ... -->
19          </body>
20  </html>
```

Listing A.2: Template für die Benutzerkontenübersicht

```
1   {% extends "provisional/frontend.html" %}
2   {% load i18n %}
3
4   {% block content %}
5   <table class="tablesorter" id="users_table">
6           <thead>
7                   <tr>
8                           <th>{% trans "Username" %}</th>
9                           <th>{% trans "First name" %}</th>
10                          <th>{% trans "Last name" %}</th>
11                          <th>{% trans "E-mail" %}</th>
12                          <th>{% trans "Active" %}</th>
13                  </tr>
14          </thead>
15          <tbody>
16                  {% for user in users %}
17                  <tr>
18                          <td><a href="{% url 'admin_users_edit' user.id %}" title=
                                "{% trans 'Edit user account' %}">{{ user.username }}
                                </a></td>
19                          <td>{{ user.first_name }}</td>
20                          <td>{{ user.last_name }}</td>
21                          <td><a href="mailto:{{ user.email }}?Subject=provisional"
                                 title="{% trans 'Send an email to this user' %}">{{
                                user.email }}</a></td>
22                          <td>{{ user.is_active|yesno }}</td>
23                  </tr>
24                  {% endfor %}
25          </tbody>
26  </table>
27  <input id="add_user_button" type="button" value="{% trans 'Create a new user
        account' %}" />
28  <script type="text/javascript">
29  $(document).ready(function(){
30          $('#users_table').tablesorter({
31                  sortList: [[0,0]]
32          });
33          $('#add_user_button').click(function(){
34                  window.location="{% url 'admin_users_add' %}";
35          });
36  });
37  </script>
38  {% endblock %}
```

Listing A.3: Ausschnitt der deutschen Sprachdatei

```
1   #: myforms.py:6 myforms.py:11 myforms.py:21
```

```
#: templates/provisional/admin_users.html:8
msgid "Username"
msgstr "Benutzername"

#: myforms.py:14 myforms.py:24 templates/provisional/admin_users.html:9
msgid "First name"
msgstr "Vorname"

#: myforms.py:15 myforms.py:25 templates/provisional/admin_users.html:10
msgid "Last name"
msgstr "Nachname"

#: myforms.py:16 myforms.py:26
msgid "Email"
msgstr "E-Mail"

#: myforms.py:17 myforms.py:27 templates/provisional/admin_users.html:12
msgid "Active"
msgstr "Aktiv"

#: views.py:153
msgid "A user with the supplied name already exists."
msgstr "Es existiert bereits ein Benutzerkonto mit dem angegebenen Namen."
```

Listing A.4: Auszug aus dem URL Mapping

```
from django.conf.urls import patterns, include, url

from django.contrib import admin
admin.autodiscover()

urlpatterns = patterns('',
    # Login, logout, home etc.
    url(r'^$', 'provisional.views.home', name='root'),
    url(r'^home/$', 'provisional.views.home', name='home'),
    url(r'^login/$', 'provisional.views.login_view', name='login_view'),
    url(r'^logout/$', 'provisional.views.logout_view', name='logout_view'),
    url(r'^settings/', 'provisional.views.settings_view', name='settings_view'),
    url(r'^accessdenied/$', 'provisional.views.accessdenied', name='accessdenied'
        ),

    # Administration
    url(r'^admin/users/$', 'provisional.views.admin_users', name='admin_users'),
    url(r'^admin/users/add/$', 'provisional.views.admin_users_add', name='
        admin_users_add'),
    url(r'^admin/users/edit/(?P<user_id>\d+)/$', 'provisional.views.
        admin_users_edit', name='admin_users_edit'),

    # Help
    url(r'^help/about/$', 'provisional.views.help_about', name='help_about'),

    # Show a "404" page if all other URLs didn't match.
    url(r'^.*/$', 'provisional.views.notfound', name='notfound'),
)
```

Listing A.5: Auszug aus den Formularklassen

```
from django import forms
from django.utils.translation import ugettext_lazy

# Login form
```

```
5  class LoginForm(forms.Form):
6      username = forms.CharField(max_length=30, required=True, label=ugettext_lazy(
           'Username'))
7      password = forms.CharField(max_length=128, required=True, widget=forms.
           PasswordInput, label=ugettext_lazy('Password'))
8
9  # Administration: User add form
10 class AdminUserAddForm(forms.Form):
11     username = forms.CharField(max_length=30, required=True, label=ugettext_lazy(
           'Username'))
12     password = forms.CharField(max_length=128, widget=forms.PasswordInput, label=
           ugettext_lazy('Password'))
13     password2 = forms.CharField(max_length=128, widget=forms.PasswordInput, label
           =ugettext_lazy('Repeat password'))
14     first_name = forms.CharField(max_length=30, required=True, label=
           ugettext_lazy('First name'))
15     last_name = forms.CharField(max_length=30, required=True, label=ugettext_lazy
           ('Last name'))
16     email = forms.EmailField(max_length=128, required=True, label=ugettext_lazy('
           Email'))
17     is_active = forms.BooleanField(required=False, label=ugettext_lazy('Active'))
```

Listing A.6: Die View für das Anmeldeformular

```
1  # Login form
2  # This function must not be called "login" due to possible collision
3  # with django.contrib.auth.login
4  @cache_control(no_cache=True, no_store=True)
5  def login_view(request):
6      context = {}
7
8      if request.method == 'POST':
9          form = myforms.LoginForm(request.POST)
10         if form.is_valid():
11             user = authenticate(username=form.cleaned_data['username'], password=
                   form.cleaned_data['password'])
12
13             # Wrong credentials?
14             if user is None:
15                 context.update({
16                     'popup_message': 'Die Zugangsdaten sind nicht korrekt.',
17                     'form': form
18                 })
19                 return render_to_response('provisional/login.html', context,
                       context_instance=RequestContext(request, processors=[
                       general_context_processor]))
20             # User account inactive?
21             if not user.is_active:
22                 context.update({
23                     'popup_message': 'Das Benutzerkonto wurde deaktiviert.',
24                     'form': form
25                 })
26                 return render_to_response('provisional/login.html', context,
                       context_instance=RequestContext(request, processors=[
                       general_context_processor]))
27
28             # Log in the user
29             login(request, user)
30
31             # Redirect to the home page or to a specific view, if supplied
32             if request.GET and 'next' in request.GET:
```

```
33                 return redirect(request.GET['next'])
34             else:
35                 return redirect(reverse(provisional.views.home))
36     else:
37         form = myforms.LoginForm()
38 
39     context.update({
40         'form': form,
41     })
42 
43     return render_to_response('provisional/login.html', context, context_instance
           =RequestContext(request, processors=[general_context_processor]))
```

Listing A.7: Das HTML-Template des Anmeldeformulars

```
1  {% extends "provisional/base.html" %}
2  {% load i18n %}
3  
4  {% block base_content %}
5  <div id="login_dialog" title="{{ application_name }} Login">
6          <form id="login_form" action="" method="post">
7                  {% csrf_token %}
8                  {% for field in form %}
9                  <div class="fieldWrapper">
10                         {{ field.errors }}
11                         {{ field.label_tag }}
12                         {{ field }}
13                 </div>
14                 {% endfor %}
15         </form>
16 </div>
17 <script type="text/javascript">
18 $(document).ready(function(){
19         /* ... */
20 });
21 </script>
22 {% endblock %}
```

Listing A.8: Ausschnitt einer als JSON kodierten Antwort des Servers.

```
1  {
2      "data": {
3          "server_categories_dataset": [
4              {
5                  "data": [
6                      [
7                          0,
8                          24
9                      ],
10                     [
11                         1,
12                         22
13                     ]
14                 ],
15                 "label": "Gepatchte Server"
16             }, /* ... */
17         ],
18         "server_categories_names": [
19             "Database Server",
20             "Terminal Server"
21         ],
```

```
22          "total_servers_count": 100
23      },
24      "success": true
25  }
```

Listing A.9: Datenstruktur mit Formatierungsoptionen für Flot (oben) und Zeichnen der Visualisierung (ab Zeile 32).

```
1  var categoricalOptions = {
2      xaxis: {
3          axisLabel: "{%trans 'Patch state' %}",
4          axisLabelUseCanvas: true,
5          axisLabelPadding: 16,
6          ticks: categoricalTicks,
7          tickLength: 0
8      },
9      yaxis: {
10         axisLabel: "{%trans 'Number of servers' %}",
11         axisLabelUseCanvas: true,
12         axisLabelPadding: 16
13     },
14     series: {
15         bars: {
16             show: true,
17             barWidth: 0.6,
18             align: "center",
19             fill: 1.0,
20             lineWidth: 0
21         },
22         stack: true
23     },
24     grid: {
25         hoverable: true,
26         borderWidth: 2,
27         borderColor: "#808080",
28         backgroundColor: "#FFFFFF"
29     }
30 };
31
32 function updateCategoricalData(data) {
33     // Show the total number of servers
34     $('#total_servers_count_categorical').text(data['total_servers_count']);
35
36     // Plot the diagram
37     var plot = $.plot($('#placeholder_categorical'), data['
           server_categories_dataset'], categoricalOptions);
38
39     // Make the diagram resizable
40     $('#placeholder_categorical').resizable({
41         minWidth: 400,
42         minHeight: 400
43     });
44 }
```

A.3. Installationsanleitung

Dieser Abschnitt beschreibt die Installation des Demonstrators auf einem Testsystem. Es wird davon ausgegangen, dass dort bereits eine Neuinstallation von openSUSE 13.2 mit dem standardmäßig vorausgewählten

Paketumfang durchgeführt wurde. Für die Installation sind an manchen Zwischenschritten Administratorrechte notwendig.

A.3.1. Django und Python-Module installieren

Vor der eigentlichen Installation des Demonstrators ist es zunächst erforderlich, Django und einige Python-Zusatzmodule zu installieren. Dabei ist zu beachten, dass der Demonstrator für Python 3.4 entwickelt wurde, da gegenüber Python 2 einige Altlasten und Unsauberkeiten entfernt wurden. Die Installation von Django erfolgt mit diesem Befehl:

```
sudo zypper install python3-Django
```

Zusätzliche Python-Module können bei openSUSE nicht über das Distributions-Repository heruntergeladen werden. Dazu ist ein anderes Tool wie zum Beispiel `pip3.4` oder `easy_install-3.4` notwendig. Sollte sich das Installationssystem hinter einem Web Proxy befinden, muss das verwendete Tool dementsprechend konfiguriert werden. Zu diesem Zweck wird auf frei zugängliche Anleitungen im WWW verwiesen.

Die folgenden Perl-Module müssen nun mit `pip3.4` oder `easy_install-3.4` installiert werden:

- `mysql-connector-python`
- `pytz`

A.3.2. Anlegen der Datenbank

In dieser Anleitung wird davon ausgegangen, dass das DBMS der Einfachheit halber auf demselben Rechner betrieben wird wie auch der Demonstrator. Bei openSUSE 13.2 ist MySQL bereits vorinstalliert, der Dienst wird aber nicht automatisch beim Hochfahren gestartet. Dies wird am einfachsten über das Distributionstool Yet another Setup Tool (YaST) unter dem Punkt “Dienste-Verwaltung” nachgeholt. Abbildung A.8 zeigt, wie die erforderliche Konfiguration aussieht.

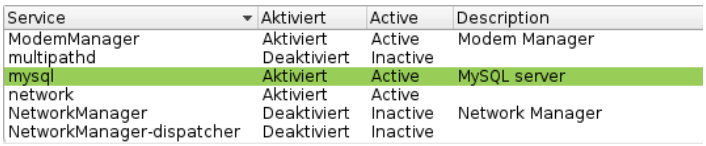


Abbildung A.8.: Aktivierung des MySQL-Systemdienstes unter YaST.

Sobald der MySQL-Dienst läuft, muss ein Passwort für den Datenbankanwender `root` angelegt werden. Dies geschieht mit dem folgenden Befehl:

```
sudo mysql_secure_installation
```

Das o. g. Programm fragt nicht nur nach einem neuen `root`-Passwort, sondern auch, ob bereits vorhandene Testdaten gelöscht werden sollen. Im Normalfall werden diese nicht benötigt.

Nachdem das `root`-Passwort eingestellt wurde, kann eine Verbindung zu dem DBMS hergestellt werden:

```
mysql -h localhost -u root -p
```

Über den Kommandozeilen-Client erfolgt dann das Erzeugen einer zunächst leeren Datenbank für den Demonstrator sowie eines Benutzers, der Zugriffsrechte darauf erhält. Sowohl für den Benutzernamen als auch für den Datenbanknamen wird hier `provisional` verwendet, das Standardpasswort in dieser Dokumentation ist `provisional123`. Abweichungen hiervon müssen später beim Anpassen der Konfigurationsdatei des Demonstrators berücksichtigt werden:

```
CREATE DATABASE provisional;
CREATE USER 'provisional'@'localhost' IDENTIFIED BY 'provisional123';
GRANT ALL PRIVILEGES ON provisional.* TO 'provisional'@'localhost';
quit
```

Anschließend empfiehlt es sich, den Zugriff des neu angelegten Benutzers auf die Datenbank zu überprüfen. Dies geschieht folgendermaßen:

```
mysql -h localhost -u provisional -p
(Verbindung zum DBMS wurde hergestellt)
CONNECT provisional;
quit
```

Vor dem Beenden des Clients durch den Befehl in der letzten Zeile dürfen keine Fehlermeldungen auftauchen.

A.3.3. Dateizugriffsrechte festlegen

Um den Demonstrator auf einem openSUSE-Server zu installieren, muss er zunächst in das Verzeichnis `/opt/provisional/` kopiert werden. Für den späteren Betrieb mit dem Webserver Apache werden die Dateizugriffsrechte dieses Verzeichnisses rekursiv angepasst:

```
sudo chown -R wwwrun:wwwrun /opt/provisional/
```

Falls zu Testzwecken kein Apache-Server, sondern der Entwicklungsserver von Django eingesetzt werden soll, müssen keine besonderen Dateizugriffsrechte eingestellt werden. Es genügt dann auch, den Demonstrator innerhalb des eigenen Home-Verzeichnisses zu platzieren.

A.3.4. Anpassung der Konfigurationsdatei

Vor der Inbetriebnahme des Demonstrators müssen zunächst Einstellungen in der Datei `settings.py` vorgenommen werden. Um sicherzustellen, dass keine Debug-Ausgaben im Fehlerfall generiert werden, muss der Wert der Variablen `DEBUG` in `False` geändert werden.

Die Konfiguration der Datenbankverbindung erfolgt in dem Dictionary `DATABASES`. Da auf eine MySQL-Datenbank zurückgegriffen wird, muss bei `ENGINE` der Wert `mysql.connector.django` eingegeben werden. Die Bedeutung der Attribute `NAME`, `USER`, `PASSWORD`, `HOST` und `PORT` ist selbsterklärend.

Der Prototyp kann den Zugriff über Browser verhindern, die vom Administrator als gefährlich eingestuft werden. Dies geschieht anhand der Erkennung des Agent Strings im HTTP Header. Um der Anwendung einen unsicheren Browser bekannt zu machen, muss das Tupel `DISALLOWED_USER_AGENTS` um eine Zeile der folgenden Form erweitert werden. Der Funktionsparameter ist ein regulärer Ausdruck, der einen bestimmten Browser herausfiltern soll:

```
re.compile(r'MSIE 6.0'),
```

Damit Benutzer des Demonstrators sich gegenüber einem Active Directory authentisieren können, müssen die folgenden Einstellungsparameter angepasst werden:

AD_SERVER_ADDRESS Fully Qualified Domain Name (FQDN) des Active Directory Servers.

AD_SERVER_PORT Port des Active Directory Servers. Standardmäßig ist Port 636 für LDAPS voreingestellt.

AD_SERVER_USE_SSL Sichere SSL-Verbindung herstellen. Standard ist `True`.

AD_SEARCH_DN Distinguished Name (DN) des Suchverzeichnisses mit den Einträgen der Benutzer.

AD_DOMAIN_NAME Der Name der Windows-Domäne, die der Active Directory Server bedient.

AD_SEARCH_FIELDS Suchfelder für Benutzernamen, zum Beispiel `sAMAccountName` für den Windows-Benutzernamen.

Außerdem muss zu diesem Zweck auch gewährleistet sein, dass das Authentication Backend für das Active Directory geladen wird (siehe Listing 8.1).

Der Demonstrator ist für das Versenden von Benachrichtigungs-Mails vorbereitet, auch wenn diese Funktion noch nicht genutzt wird. Zu diesem Zweck sind die folgenden Parameter vorgesehen:

EMAIL_ADDRESS Absenderadresse des Demonstrators.

EMAIL_HOST Der FQDN des Simple Mail Transfer Protocol (SMTP) Servers.

EMAIL_USE_AUTHENTICATION Auf `True` stellen, falls der SMTP Server eine Authentifizierung verlangt.

EMAIL_USER Der Name des Demonstrator-Postfachs.

EMAIL_PASSWORD Das dazugehörige Passwort, falls Authentifizierung verwendet wird.

EMAIL_PORT Der SMTP Port.

EMAIL_USE_TLS Auf `True` stellen, falls sichere TLS-Verbindungen zum Server aufgebaut werden sollen.

Die folgenden Einstellungen umfassen das Management der Benutzersitzungen:

SESSION_SAVE_EVERY_REQUEST Wenn diese Einstellung auf `True` gesetzt ist, werden Benutzersitzungen bei jeder Aktion des Benutzers bis zur maximalen Sitzungsdauer verlängert.

SESSION_COOKIE_AGE Die maximale Sitzungsdauer in Sekunden.

Die Anwendung ist in der Lage, ein Benutzerkonto nach der mehrmaligen Falscheingabe des Passworts zu sperren. Die maximale Anzahl zulässiger Fehlversuche wird bei der Variablen `FAILED_LOGIN_THRESHOLD` festgelegt. Der Standardwert ist 5.

Einige Ansichten in der Weboberfläche des Demonstrators besitzen einen Paginator, um umfangreiche Datenmengen auf mehrere Bildschirmseiten zu verteilen. Der Standardwert für die Anzahl angezeigter Elemente pro Seite wird bei der Variablen `DEFAULT_PAGINATOR_SIZE` festgelegt. Dieser Wert wird immer dann verwendet, wenn ein Anwender keinen benutzerdefinierten Wert in den Programmeinstellungen eingestellt hat. Die Einstellungen `MINIMUMPAGINATZOR_SIZE` und `MAXIMUM_PAGINATOR_SIZE` sind für den unteren und oberen Grenzwert der Auswahlfelder der Seitengröße vorgesehen.

Es gibt noch zwei weitere Konfigurationsparameter, die die Anzeige der Weboberfläche beeinflussen: Der Wert von `DEFAULT_REFRESH_INTERVAL` gibt die Zeit in Sekunden an, nach der bestimmte Visualisierungen automatisch aktualisiert werden. `DEFAULT_HISTOGRAM_CLASSES` gibt die Anzahl der Säulen vor, die standardmäßig in einem Histogramm angezeigt werden. Anwender des Prototyps können beide Standardwerte in den Programmeinstellungen an die eigenen Bedürfnisse anpassen.

A.3.5. Initialisierung der Datenbank

Nachdem ein Datenbankanwender und eine dazugehörige Datenbank erstellt sowie die Verbindungsparameter in die Datei `settings.py` eingetragen wurden, muss die Datenbank des Demonstrators mit initialen Werten gefüllt werden. Dies ist im einfachsten Fall durch den Import eines SQL Dumps möglich. Ein solcher Dump wird in Form der Datei `provisional.sql` mit dem Quellcode mitgeliefert. Er ist für einen MySQL-Server ausgelegt und beinhaltet auch Beispieldaten zur Visualisierung.

Falls eine andere Datenbank verwendet wird, müssen die Initialwerte anders importiert werden. Aus diesem Grund werden sie zusätzlich als JSON-Dateien bereitgestellt. Diese Dateien werden mit Hilfe des Django-Skripts `manage.py` in die Datenbank geladen. Der objektrelationale Mapper des Frameworks übernimmt die

erforderliche Übersetzung in das zum eingestellten Datenbankhersteller passende Format. Beispielwerte für die Linux-Server sind allerdings nur in dem MySQL-Dump enthalten.

Die Initialisierung besteht aus zwei grundlegenden Schritten:

- Anlegen der Tabellen
- Füllen der Tabellen mit Initialwerten

Zunächst ist es erforderlich, in das Projektverzeichnis des Demonstrators zu wechseln. Danach werden die Tabellen erzeugt:

```
1 cd /opt/provisional
2 python3.4 manage.py syncdb
```

Die Initialwerte für die Datenbank befinden sich in JSON-Dateien innerhalb des Ordners `fixtures` im Applikationsverzeichnis. Die einzelnen Dateien haben folgende Bedeutung:

AdminAccount.json Legt das Benutzerkonto `admin` zusammen mit einem Initialpasswort an.

Permission.json Zugriffsrechte und Beschreibungen.

ServerType.json Beinhaltet vorgefertigte Serverkategorien.

Für jede in die Datenbank zu ladende Datei muss ein Befehl nach dem folgenden Schema eingegeben werden:

```
1 python3.4 manage.py loaddata provisional/fixtures/<dateiname>.json
```

A.3.6. Konfiguration des Webservers Apache

Es wird davon ausgegangen, dass auf dem Zielsystem bereits eine frische Installation des Webservers Apache durchgeführt wurde. Zur Gewährleistung der Informationssicherheit ist die Einrichtung eines Virtual Hosts für HTTPS-Verbindungen unerlässlich. Dies setzt voraus, dass ein Server-Zertifikat und ein privater Schlüssel auf dem Server an passender Stelle gespeichert und TLS für diesen Virtual Host konfiguriert wurde. Zu diesem Zweck muss außerdem das Apache-Modul `ssl` aktiviert sein.

Um den Rahmen dieser Masterarbeit nicht zu sprengen, wird an dieser Stelle auf externe Fachliteratur zur Schlüsselerzeugung, zur Generierung eines Certificate Signing Request (CSR) und der Grundkonfiguration eines Virtual Hosts für HTTPS verzichtet. Zu diesen Themen gibt es genügend frei verfügbare Literatur im WWW.

Einige wichtige Hinweise zur Absicherung von Webservern wurden aber bereits in dieser Masterarbeit genannt: Tipps für das sichere Speichern des privaten Schlüssels sind in Abschnitt 6.3 nachzulesen. In Abschnitt 6.4 gibt es Hinweise zur sicheren Konfiguration von TLS bei Apache.

Der Webserver des Demonstrators sollte so konfiguriert werden, dass grundsätzlich Webseitenaufrufe über das sichere HTTPS-Protokoll erfolgen. Fordert der Client eine Ressource über unverschlüsseltes HTTP an, erfolgt automatisch eine Umleitung zur entsprechenden URL via HTTPS. Um dies zu bewerkstelligen, wird auf das Rewrite-Modul `mod_rewrite` zurückgegriffen, das zuvor mit dem Befehl `a2enmod rewrite` aktiviert werden muss. Listing A.10 zeigt die dafür notwendigen Konfigurationszeilen.

Listing A.10: Apaches Rewrite Engine schreibt HTTP- in HTTPS-Requests um

```
1 RewriteEngine   on
2 RewriteCond     %{HTTPS} off
3 RewriteRule     (.*) https://%{HTTP_HOST}%{REQUEST_URI}
```

Zur Bereitstellung des Prototyps über Apache ist die Installation und Aktivierung des Moduls `mod_wsgi`[1] erforderlich. Damit ist der Webserver in der Lage, WSGI-Applikationen bereitzustellen, zu denen auch der Prototyp zählt.

Die restlichen Installationsschritte können der offiziellen Django-Dokumentation entnommen werden, da die für den Prototyp spezifischen Schritte bereits aufgezählt wurden. Als nächstes muss eine eigene Apache-Konfigurationsdatei für die WSGI-Anwendung angelegt werden. Eine Anleitung hierzu gibt es auf der Django-Website.[2]

Nachdem Apache für die Bereitstellung des Demonstrators konfiguriert wurde, muss der Dienst des Webservers neu gestartet werden, damit sämtliche Konfigurationsänderungen übernommen werden:

```
1 sudo service apache2 restart
```

A.3.7. Alternativ: Start der Entwicklungsumgebung

Zu Testzwecken muss der Prototyp nicht zwingend über einen Webserver wie Apache bereitgestellt werden, hier genügt der Entwicklungsserver von Django vollkommen. Vor allem wenn an dem Quellcode weiterentwickelt wird, ist der Entwicklungsserver schneller einsatzbereit und liefert zudem Debug-Informationen während der Laufzeit. Der Entwicklungsserver wird aus dem Verzeichnis des Projekts mit dem folgenden Kommando gestartet:

```
1 python3.4 manage.py runserver
```

Dieser Befehl bewirkt, dass der Server-Prozess auf dem TCP-Port 8000 lauscht und nur auf Requests reagiert, die vom selben Rechner abgesendet wurden. Soll der Zugriff auch von anderen Rechnern aus erlaubt sein, muss der Server folgendermaßen gestartet werden (die Angabe eines abweichenden Ports ist möglich):

```
1 python3.4 manage.py runserver 0.0.0.0:8000
```

Nach dem Start des Entwicklungsservers kann vom selben Rechner aus die Weboberfläche des Prototyps über die URL `http://localhost:8000` abgerufen werden. Beim Zugriff von einem anderen Rechner aus ist `localhost` durch den Namen oder die IP-Adresse des Entwicklungssystems zu ersetzen.

A.3.7.1. Verschlüsselte Kommunikation via SSH-Tunnel

Beim Betrieb des Entwicklungsservers ist zu beachten, dass er keine verschlüsselte Datenübertragung via HTTPS unterstützt. Sollte zu Demonstrationszwecken dennoch eine verschlüsselte Übertragung erwünscht sein, damit der Datenverkehr beispielsweise zwischen dem Entwicklungsrechner und einem Konferenzraumrechner sicher übertragen wird, besteht die Möglichkeit, einen SSH-Tunnel zu verwenden.

Um einen SSH-Tunnel aufbauen zu können, muss auf dem Entwicklungsrechner ein SSH-Dienst laufen und es werden die Zugangsdaten von einem Benutzer benötigt, der sich per SSH dort einloggen kann. Auf dem entfernten Rechner muss ein SSH Client wie beispielsweise *PuTTY*[3] vorhanden sein. Die Vorgehensweise zur Einrichtung eines SSH-Tunnels mit PuTTY wird auf einer Webseite von `netzmafia.de`[4] gut beschrieben. Obwohl dort auf eine ältere Version zurückgegriffen wird, sind die essentiellen Schritte dieselben.

Zum Öffnen des Tunnels muss sich der Anwender mit den zuvor erwähnten Zugangsdaten am entfernten System authentisieren. Dies geschieht in einem herkömmlichen Terminal-Fenster (siehe Abbildung A.9). Der Tunnel bleibt solange bestehen, bis das Terminal-Fenster geschlossen wird.

[1] mod_wsgi-Dokumentation: `https://code.google.com/p/modwsgi/`
[2] WSGI-Konfiguration: `https://docs.djangoproject.com/en/1.6/howto/deployment/wsgi/modwsgi/`
[3] PuTTY-Website: `http://www.chiark.greenend.org.uk/~sgtatham/putty/`
[4] SSH-Tunnel mit PuTTY: `http://www.netzmafia.de/skripten/internet/putty-tunnel.html`

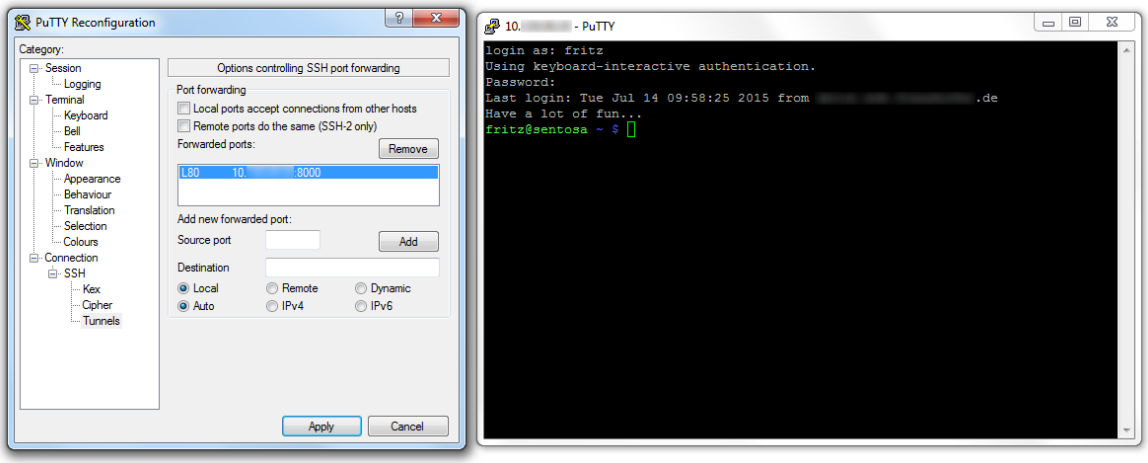


Abbildung A.9.: Erstellung eines SSH-Tunnels zum Entwicklungssystem mit PuTTY.

Der Tunnel bewirkt, dass auf dem lokalen Rechner ein TCP-Port geöffnet wird, zu dem der lokale Browser eine Verbindung aufbauen kann. In der Regel handelt es sich hier um Port 80. Der Browser kommuniziert beim Aufruf von `http://localhost` dann allerdings nicht mit dem lokalen System, sondern mit dem Entwicklungsrechner über den geöffneten SSH-Tunnel. Da SSH den Datenverkehr verschlüsselt, werden auch die Webseiten und Benutzereingaben trotz fehlender HTTPS-Verbindung verschlüsselt zwischen den beiden Rechnern übertragen.

A.3.7.2. Verschlüsselte Kommunikation via TLS-Proxy

Sollte auf dem Rechner, von dem aus der Zugriff auf die Weboberfläche des Demonstrators erfolgen soll, kein SSH Client vorhanden sein und auch nicht ohne Weiteres installiert werden können, kann alternativ zum Tunnel ein TLS-Proxy-Server verwendet werden. Dabei handelt es sich im einfachsten Fall um einen zu diesem Zweck konfigurierten `nginx`-Webserver[5] mit einem selbst signierten Zertifikat. Der Zugriff auf den Prototyp vom entfernten Rechner aus erfolgt dann nicht direkt über den Entwicklungsserver, sondern über den Proxy-Prozess, der stellvertretend für den Entwicklungsserver die TLS-Terminierung übernimmt. Eine detaillierte Dokumentation zur Konfiguration eines TLS-Proxy-Servers ist auf der Homepage von nginx abrufbar.[6]

Dass die Konfiguration von nginx zu einem TLS-Proxy-Server für Demonstrationszwecke relativ schnell von der Hand geht, zeigt die beispielhafte Konfigurationsdatei in Listing A.11, deren Umfang überschaubar ist.

Listing A.11: Minimale Beispielkonfiguration für einen TLS-Proxy-Server mit nginx.

```
1  server {
2      listen 443;
3      server_name provisional;
4
5      ssl on;
6      ssl_certificate /etc/ssl/certs/provisional.crt;
7      ssl_certificate_key /etc/ssl/private/provisional.key;
8      ssl_protocols TLSv1.2;
9      ssl_ciphers 'ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:
           ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-
           GCM-SHA384:DHE-RSA-AES128-GCM-SHA256';
10     ssl_prefer_server_ciphers on;
11
12     access_log /var/log/nginx/provisional.access.log;
13
```

[5] nginx-Website: `http://nginx.org/`
[6] TLS-Proxy-Server mit nginx: `http://wiki.nginx.org/SSL-Offloader`

```
14      location / {
15          proxy_pass http://localhost:8000;
16      }
17  }
```

A.3.8. Einloggen

Die Installation ist nun abgeschlossen. Ab sofort kann die Weboberfläche des Prototyps über den Browser aufgerufen werden. Sofern sich der Browser auf demselben Testsystem wie der Prototyp befinden, muss im Browser die Adresse `https://localhost/` aufgerufen werden.

Daraufhin erscheint das Anmeldefenster des Prototyps (siehe Abbildung A.10). Der Login erfolgt zunächst mit dem Benutzerkonto `admin`, das zuvor in die Datenbank geladen wurde, und seinem Standardpasswort `admin123`. Das Initialpasswort kann nach dem ersten Login geändert werden.

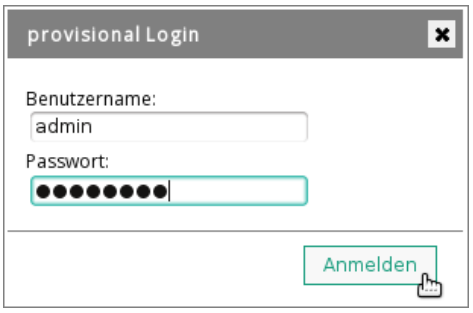


Abbildung A.10.: Der Anmeldedialog des Prototyps.

A.4. Verwendete Software

Name	Quelle
Adobe Acrobat X Pro 10.1.13	http://www.adobe.com/de/
Corel PHOTO-PAINT X6	http://www.corel.com/de/
Debian GNU/Linux 8	https://www.debian.org/
Django 1.6	https://www.djangoproject.com/
Eclipse Luna 4.4.1	https://eclipse.org/
Gimp 2.8.14	http://www.gimp.org/
Inkscape 0.48.5	https://inkscape.org/de/
MariaDB 10.0.13	https://mariadb.org/
Microsoft Excel 2010	http://www.microsoft.com/de-de/default.aspx
Microsoft Visio 2010	http://www.microsoft.com/de-de/default.aspx
Microsoft Windows 7 Enterprise	http://www.microsoft.com/de-de/default.aspx
Mozilla Firefox 38	https://www.mozilla.org/de/firefox/new/
nmap 6.47	http://nmap.org/
openSUSE 13.2	https://de.opensuse.org/
phpMyAdmin 4.2.13	http://www.phpmyadmin.net/
Poedit 1.5.4	https://poedit.net/
PuTTY 0.64	http://www.putty.org/
Python 3.4	https://www.python.org/
SciTE 3.4.4	http://www.scintilla.org/SciTE.html
SparxSystems Enterprise Architect 9.2	http://www.sparxsystems.de/
TeX Live 2013	https://www.tug.org/texlive/
Texmaker 4.3	http://www.xm1math.net/texmaker/
WinSCP 5.5.3	http://winscp.net/eng/docs/lang:de

Tabelle A.1.: Verwendete Software.

A.5. Inhalt des beigelegten Datenträgers

Auf dem beigelegten Datenträger befinden sich folgende Dateien und Ordner im Stammverzeichnis:

code/ Dieser Ordner beinhaltet den Programmcode des Prototyps. Die Verzeichnisstruktur des Prototyps kann in Abschnitt 7.2.2 nachgelesen werden.

latex/ LaTeX-Quellen der Ausarbeitung und Portable Document Format (PDF)-Version. Die Verzeichnisstruktur ist auf der Webseite `http://www.nm.ifi.lmu.de/teaching/Ausschreibungen/Layout/` beschrieben.

liesmich.txt Eine Textdatei mit der Beschreibung des Datenträgerinhalts.

Abkürzungsverzeichnis

A

ACL . . . . . Access Control List
AD . . . . . Active Directory
ADSL . . . . . Asymmetric Digital Subscriber Line
AES . . . . . Advanced Encryption Standard
AJAX . . . . . Asynchronous JavaScript and XML
API . . . . . Application Programming Interface
API . . . . . A priori importance
APT . . . . . Advanced Packaging Tool
ASA . . . . . Adaptive Security Appliance
ASLR . . . . . Address Space Layout Randomization
AWS . . . . . Amazon Web Services

B

BDSG . . . . . Bundesdatenschutzgesetz
BfV . . . . . Bundesamt für Verfassungsschutz
BITBLT . . . . . Bit-boundary block transfer
BMWi . . . . . Bundesministerium für Wirtschaft und Energie
BSI . . . . . Bundesamt für Sicherheit in der Informationstechnik
btrfs . . . . . B-tree File System

C

CA . . . . . Certificate Authority
CIA . . . . . Central Intelligence Agency
CIDR . . . . . Classless Inter-Domain Routing
CIFS . . . . . Common Internet File System
CPU . . . . . Central Processing Unit
CRIME . . . . . Compression Ratio Info-leak Made Easy
CSR . . . . . Certificate Signing Request
CSRF . . . . . Cross-Site Request Forgery
CSS . . . . . Cascading Style Sheets
CVE . . . . . Common Vulnerabilities and Exposures
CVSS . . . . . Common Vulnerability Scoring System

D

DAC . . . . . Digital-to-Analog Converter
DBMS . . . . . Database Management System
DDoS . . . . . Distributed Denial of Service
DEP . . . . . Data Execution Prevention
DES . . . . . Data Encryption Standard
DFN . . . . . Deutsches Forschungsnetz
DH . . . . . Diffie-Hellman
DHCP . . . . . Dynamic Host Configuration Protocol
DHE . . . . . Diffie-Hellman Ephemeral
DIY . . . . . Do it yourself

DMI . . . . . Direct Manipulation Interface
DMZ . . . . . Demilitarized Zone
DN . . . . . Distinguished Name
DNS . . . . . Desoxyribonukleinsäure
DNS . . . . . Domain Name System
DNSSEC . . . . . Domain Name System Security Extensions
DOM . . . . . Document Object Model
DoS . . . . . Denial of Service
DRY . . . . . Don't repeat yourself

E

ECDHE . . . . . Elliptic curve Diffie-Hellman Ephemeral
eduroam . . . . . Education Roaming
EE . . . . . Enterprise Edition
EKG . . . . . Elektrokardiogramm
ESM . . . . . Enterprise Security Manager
ESXi . . . . . Elastic Sky X integrated

F

FHS . . . . . Filesystem Hierarchy Standard
FIM . . . . . File Integrity Monitoring
FQDN . . . . . Fully Qualified Domain Name

G

GCM . . . . . Galois/Counter Mode
GNU . . . . . GNU's Not Unix
GOP . . . . . Guardians of Peace
GPL . . . . . GNU General Public License
GPU . . . . . Graphics Processing Unit
GUI . . . . . Graphical User Interface

H

HBA . . . . . Host Bus Adapter
HCI . . . . . Human-Computer Interaction
Hi-Fi . . . . . High Fidelity
HIDS . . . . . Host-based Intrusion Detection System
HSM . . . . . Hardware Security Module
HSTS . . . . . HTTP Strict Transport Security
HTML . . . . . Hypertext Markup Language
HTTP . . . . . Hypertext Transfer Protocol
HTTPS . . . . . Hypertext Transfer Protocol Secure

I

ICMP . . . . . Internet Control Message Protocol
ID . . . . . Identifier
IDE . . . . . Integrated Development Environment
IDS . . . . . Intrusion Detection System
IEEE . . . . . Institute of Electrical and Electronics Engineers
IETF . . . . . Internet Engineering Task Force
Iframe . . . . . Inline Frame
IIS . . . . . Internet Information Services
IP . . . . . Internet Protocol
IPAM . . . . . IP Address Management

IPsec . . . . . Internet Protocol Security
IPv6 . . . . . Internet Protocol Version 6
IT . . . . . Informationstechnik

J

JSON . . . . . JavaScript Object Notation

K

KDE . . . . . K Desktop Environment

L

LAMP . . . . . Linux Apache MySQL PHP
LAN . . . . . Local Area Network
LDAP . . . . . Lightweight Directory Access Protocol
LDAPS . . . . . Lightweight Directory Access Protocol over Secure Sockets Layer
LMU . . . . . Ludwig-Maximilians-Universität München
LRZ . . . . . Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften
LRZ-SIM . . . . . Leibniz-Rechenzentrum Secure Identity Management
LTS . . . . . Long Term Support

M

MAC . . . . . Mandatory Access Control
MAC . . . . . Message Authentication Code
MAC . . . . . Media Access Control
MD5 . . . . . Message-Digest Algorithm 5
MIME . . . . . Multipurpose Internet Mail Extensions
MIT . . . . . Massachusetts Institute of Technology
MPI . . . . . Message Passing Interface
MSDN . . . . . Microsoft Developer Network
MVC . . . . . Model View Controller
MVP . . . . . Most Valuable Professional
MWN . . . . . Münchner Wissenschaftsnetz

N

NAS . . . . . Network Attached Storage
NASL . . . . . Nessus Attack Scripting Language
NFS . . . . . Network File System
NFV . . . . . Network Functions Virtualization
NSA . . . . . National Security Agency
NTP . . . . . Network Time Protocol

O

OS . . . . . Operating System
OSSIM . . . . . Open Source Security Information Management
OTRS . . . . . Open Ticket Request System
OWASP . . . . . Open Web Application Security Project

P

PARC . . . . . Palo Alto Research Center
PC . . . . . Personal Computer
PCA . . . . . Policy Certification Authority

PCIe . . . . . Peripheral Component Interconnect Express
PDF . . . . . Portable Document Format
PEM . . . . . Privacy Enhanced Mail
PFS . . . . . Perfect Forward Secrecy
PHP . . . . . PHP Hypertext Preprocessor, ursprünglich Personal Home Page Tools
PIN . . . . . Personal Identification Number
POSIX . . . . . Portable Operating System Interface
PPI . . . . . Pixels per Inch
PRACE . . . . . Partnership for Advanced Computing in Europe

Q

QoS . . . . . Quality of Service

R

RA . . . . . Registration Authority
RADIUS . . . . . Remote Authentication Dial-In User Service
RAID . . . . . Redundant Array of Independent Disks
RAM . . . . . Random Access Memory
RC4 . . . . . Ron's Code 4
REST . . . . . Representational State Transfer
RHEL . . . . . Red Hat Enterprise Linux
RPM . . . . . RPM Package Manager, ursprünglich Red Hat Package Manager

S

SAN . . . . . Storage Area Network
SCCM . . . . . System Center Configuration Manager
SCP . . . . . Secure Copy
SDN . . . . . Software-defined networking
SELinux . . . . . Security-Enhanced Linux
Setuid . . . . . Set User ID
SHA . . . . . Secure Hash Algorithm
SiBe . . . . . Sicherheitsbevollmächtigter
SIEM . . . . . Security Information and Event Management
SLA . . . . . Service-Level Agreement
SLES . . . . . SUSE Linux Enterprise Server
SMB . . . . . Server Message Block
SMTP . . . . . Simple Mail Transfer Protocol
SNMP . . . . . Simple Network Management Protocol
SOAP . . . . . ursprünglich Simple Object Access Protocol
SPL . . . . . Search Processing Language
SQL . . . . . Structured Query Language
SSH . . . . . Secure Shell
SSID . . . . . Service Set Identifier
SSL . . . . . Secure Sockets Layer
SuSE . . . . . ursprünglich Software- und System-Entwicklung
SVG . . . . . Scalable Vector Graphics
SYN . . . . . Synchronize

T

TCP . . . . . Transmission Control Protocol
TLS . . . . . Transport Layer Security
TPM . . . . . Trusted Platform Module
TUM . . . . . Technische Universität München

U

UA . . . . . User Agent
UDP . . . . . User Datagram Protocol
UEFI . . . . . Unified Extensible Firmware Interface
UI . . . . . User Interface
URL . . . . . Uniform Resource Locator
USB . . . . . Universal Serial Bus
USM . . . . . Unified Security Management
UX . . . . . User Experience

V

VFW . . . . . Virtual Firewall
VLAN . . . . . Virtual Local Area Network
VM . . . . . Virtual Machine
VMCI . . . . . Virtual Machine Communication Interface
VPN . . . . . Virtual Private Network
VS-NfD . . . . . Verschlusssache - Nur für den Dienstgebrauch

W

W3C . . . . . World Wide Web Consortium
WAN . . . . . Wide Area Network
WiN . . . . . Deutsches Wissenschaftsnetz
WLAN . . . . . Wireless Local Area Network
WSGI . . . . . Web Server Gateway Interface
WSUS . . . . . Windows Server Update Services
WWW . . . . . World Wide Web
WYSIWYG . . . . . What you see is what you get

X

XML . . . . . Extensible Markup Language
XSS . . . . . Cross-Site Scripting

Y

YAML . . . . . YAML Ain't Markup Language, ursprünglich Yet Another Markup Language
YaST . . . . . Yet another Setup Tool

Z

ZFS . . . . . Zettabyte File System

Literaturverzeichnis

[ADR+ 15] ADRIAN, DAVID, KARTHIKEYAN BHARGAVAN, ZAKIR DURUMERIC, PIERRICK GAUDRY, MATTHEW GREEN, J. ALEX HALDERMAN, NADIA HENINGER, DREW SPRINGALL, EMMANUEL THOMÉ, LUKE VALENTA, BENJAMIN VANDERSLOOT, ERIC WUSTROW, SANTIAGO ZANELLA-BÉGUELIN und PAUL ZIMMERMANN: *Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice*, 2015, https://weakdh.org/imperfect-forward-secrecy.pdf . Aufgerufen am 30.05.2015. 177

[ALE 13] ALEXANDER, KERSTIN: *Kompendium der visuellen Information und Kommunikation*. Springer, Zweite Auflage, 2013. 2

[ALI 15] ALIENVAULT, INC.: *AlienVault Delivers New Layer of Security for IT Teams on Amazon Web Services*, 2015, https://www.alienvault.com/who-we-are/press-releases/alienvault-delivers-new-layer-of-security-for-it-teams-on-amazon-web-services . Aufgerufen am 18.04.2015. 113

[ALT 13] PROF. DR. FLORIAN ALT: *Vorlesung Informationsvisualisierung*, 2013, http://www.medien.ifi.lmu.de/lehre/ws1314/iv/ . Aufgerufen am 31.10.2014. 5

[APA 15] AUSTRIA PRESSE AGENTUR: *Kinderpornos auf Website der KZ-Gedenkstätte Mauthausen platziert*, 2015, http://www.nachrichten.at/oberoesterreich/Kinderpornos-auf-Website-der-KZ-Gedenkstaette-Mauthausen-platziert;art4,1789709 . Aufgerufen am 11.05.2015. 63

[BALL+ 04] BALL, ROBERT, GLENN A. FINK und CHRIS NORTH: *Home-Centric Visualization of Network Traffic for Security Administration*. In: *Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, VizSEC/DMSEC '04, Seiten 55–64. ACM, 2004, http://doi.acm.org/10.1145/1029208.1029217 . Aufgerufen am 30.04.2015. 162

[BAU+ 01] BAUDISCH, PATRICK, NATHANIEL GOOD und PAUL STEWART: *Focus Plus Context Screens: Combining Display Technology with Visualization Techniques*. In: *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, UIST '01, Seiten 31–40, New York, NY, USA, 2001. ACM, http://doi.acm.org/10.1145/502348.502354 . Aufgerufen am 10.07.2015. 55

[BDSG 15] BUNDESMINISTERIUM FÜR JUSTIZ UND VERBRAUCHERSCHUTZ: *Bundesdatenschutzgesetz (BDSG)*, 2015, http://www.gesetze-im-internet.de/bundesrecht/bdsg_1990/gesamt.pdf . Aufgerufen am 15.05.2015. 83

[BECK+ 87] BECKER, RICHARD A., WILLIAM S. CLEVELAND und ALLAN R. WILKS: *Dynamic Graphics for Data Analysis*. Statistical Science, 2(4):355–395, 1987, https://projecteuclid.org/euclid.ss/1177013104 . Aufgerufen am 11.07.2015. 41

[BED 90] BEDDOW, JEFF: *Shape coding of multidimensional data on a microcomputer display*. In: *Proceedings of the 1st Conference on Visualization '90*, VIS '90, Seiten 238–246. IEEE Computer Society Press, 1990, http://dl.acm.org/citation.cfm?id=949569 . Aufgerufen am 03.02.2015. 29

[BER 13] BERTIN, JACQUES: *Sémiologie graphique: Les diagrammes, les résaux, les cartes (Neuauflage)*. Editions de l'Ecole des Hautes Etudes en Sciences Sociales, 2013. 17

[BFV 14] BUNDESAMT FÜR VERFASSUNGSSCHUTZ FÜR DIE VERFASSUNGSSCHUTZBEHÖRDEN IN BUND UND LÄNDERN: *Sicherheitslücke Mensch – Der Innentäter als größte Bedrohung für die Unternehmen*, 2014, http://www.verfassungsschutz.de/download/faltblatt-2014-04-sicherheitsluecke-mensch.pdf . Aufgerufen am 29.12.2014. 71, 86

[BKR 14] BECKER, LEO: *Schulbehörde von Los Angeles rückt offenbar vom iPad ab*, 2014, http://heise.de/-2243400 . Aufgerufen am 25.07.2015. 267

[BLE+ 05] BLESS, ROLAND, STEFAN MINK, ERIK-OLIVER BLASS, MICHAEL CONRAD, HANS-JOACHIM HOF, KENDY KUTZNER und MARCUS SCHÖLLER: *Sichere Netzwerkkommunikation – Grundlagen, Protokolle und Architekturen*. Springer, 2005. 60

[BOE 14] BÖCK, HANNO: *Speicher-Randomisierung unter Linux mangelhaft*, 2014, http://www.golem.de/1412/111010.html . Aufgerufen am 10.01.2015. 73

[BRI 15] BRIGHT, PETER: *HTTP/2 finished, coming to browsers within weeks*, 2015, http://arstechnica.com/information-technology/2015/02/http2-finished-coming-to-browsers-within-weeks/ . Aufgerufen am 15.07.2015. 269

[BSI 13] BUNDESAMT FÜR SICHERHEIT IN DER INFORMATIONSTECHNIK: *Absicherung eines Servers (ISi-Server)*, 2013, https://www.bsi.bund.de/DE/Themen/Cyber-Sicherheit/ISi-Reihe/ISi-Server/server_node.html . Aufgerufen am 27.12.2014. 60

[BSI 15] BUNDESAMT FÜR SICHERHEIT IN DER INFORMATIONSTECHNIK: *Technische Richtlinie TR-02102-2: Kryptographische Verfahren: Empfehlungen und Schlüssellängen Teil 2 – Verwendung von Transport Layer Security (TLS)*, 2015, http://goo.gl/qnXp9S . Aufgerufen am 28.04.2015. 71, 169, 175, 203

[BUTZ 12] PROF. DR. ANDREAS BUTZ: *Vorlesung Informationsvisualisierung*, 2012, http://www.medien.ifi.lmu.de/lehre/ws1213/iv/ . Aufgerufen am 08.12.2014. 5

[CARD+ 99] CARD, STUART K., JOCK D. MACKINLAY und BEN SHNEIDERMAN: *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann, 1999. 7, 15

[CASS 14] CASS, STEPHEN: *Top 10 Programming Languages – Spectrum's 2014 Ranking*, 2014, http://spectrum.ieee.org/computing/software/top-10-programming-languages . Aufgerufen am 21.02.2015. 185

[CHEN+ 98] CHENG, HEYNING und RON AVNUR: *Traffic Analysis of SSL Encrypted Web Browsing*, 1998, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.3.1201 . Aufgerufen am 03.02.2015. 61

[CHER 74] CHERNOFF, HERMAN: *The Use of Faces to Represent Points in k-Dimensional Space Graphically*. Journal of the American Statistical Association, 68(342):361–368, 1973, http://goo.gl/HLWUTp . Aufgerufen am 15.07.2015. 28

[CON 07] CONTI, GREG: *Security Data Visualization: Graphical Techniques for Network Analysis*. No Starch Press, 2007. 25

[DIE 13] DIEDRICH, OLIVER: *Servermarkt: x86 und Linux auf dem Vormarsch*, 2013, http://heise.de/-2063193 . Aufgerufen am 22.09.2014. 1

[DIE 15] DIEDRICH, OLIVER: *Ubuntu: Snappy Apps sollen Debian-Pakete ersetzen*, 2015, http://heise.de/-2629493 . Aufgerufen am 09.05.2015. 86

[DIX 03] DIX, ALAN: *Human-Computer Interaction*. Prentice Hall, Dritte Auflage, 2003. 40

[DON 78] DONELSON, WILLIAM C.: *Spatial Management of Information*. In: *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '78, Seiten 203–209. ACM, 1978, http://dl.acm.org/citation.cfm?id=807391 . Aufgerufen am 03.02.2015. 52

[DOS+ 04] SELAN DOS SANTOS und KEN BRODLIE: *Gaining understanding of multivariate and multidimensional data through visualization*. Computers & Graphics, 28(3):311–325, 2004, http://www.sciencedirect.com/science/article/pii/S0097849304000251 . Aufgerufen am 11.07.2015. 17

[EICK+ 92] EICK, STEPHEN G., JOSEPH L. STEFFEN und ERIC E. SUMMER, JR.: *SeeSoft – A Tool For Visualizing Line Oriented Software Statistics*. IEEE Transactions on Software Engineering, 18(11):957–968, 1992, http://dx.doi.org/10.1109/32.177365 . Aufgerufen am 11.07.2015. 50

[EIK 15] EIKENBERG, RONALD: *Details zur kritischen Lücke im Telnet-Server von Windows*, 2015, http://heise.de/-2518951 . Aufgerufen am 30.01.2015. 72

[EIK 15B] EIKENBERG, RONALD: *Flash-Player deaktivieren! Sicherheits-Update lässt kritische Lücke offen*, 2015, http://heise.de/-2526789 . Aufgerufen am 14.02.2015. 85

[EIK 15C] EIKENBERG, RONALD: *Security-Funktion HSTS als Supercookie*, 2015, http://heise.de/-2511258 . Aufgerufen am 15.05.2015. 177

[ELL+ 05] ELLIS, GEOFFREY, ENRICO BERTINI und ALAN DIX: *The Sampling Lens: Making Sense of Saturated Visualisations*, 2005, http://eprints.lancs.ac.uk/12648/1/CHI%2705_sampling_lens.pdf . Aufgerufen am 03.02.2015. 33

[ENG+ 67] ENGLISH, WILLIAM K., DOUGLAS C. ENGELBART und MELVYN L. BERMAN: *Display-Selection Techniques for Text Manipulation*. IEEE Transactions on Human Factors in Electronics, HFE-8(1):5–15, 1967, http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1698228&tag=1 . Aufgerufen am 10.07.2015. 41

[ERD 14] ERDMANN, LISA: *Wahlen in Nordkorea: Keine einzige Stimme gegen Kim*, 2014, http://www.spiegel.de/politik/ausland/kim-jong-un-bei-wahlen-in-nordkorea-mit-100-prozent-zustimmung-gewaehlt-a-957765.html . Aufgerufen am 20.07.2015. 254

[EVA 13] EVANS, CLAIRE L.: *A Eulogy for Skeuomorphism*, 2013, http://motherboard.vice.com/read/a-eulogy-for-skeumorphism . Aufgerufen am 08.12.2014. 43

[EXU 99] SAINT-EXUPÉRY, ANTOINE DE: *Terre des hommes (Neuauflage)*. Gallimard, 1999. 12

[FIS 12] FISHER, DENNIS: *CRIME Attack Uses Compression Ratioschir of TLS Requests as Side Channel to Hijack Secure Sessions*, 2012, https://threatpost.com/crime-attack-uses-compression-ratio-tls-requests-side-channel-hijack-secure-sessions-091312/77006 . Aufgerufen am 15.05.2015. 177

[FLU+ 81] FLURY, BERNHARD und HANS RIEDWYL: *Graphical Representation of Multivariate Data by Means of Asymmetrical Faces*. Journal of the American Statistical Association, 76(376):757–765, 1981, http://www.jstor.org/stable/2287565?seq=1#page_scan_tab_contents . Aufgerufen am 11.07.2015. 29

[FRE+ 95] FREEMAN, ERIC und SCOTT FERTIG: *Lifestreams: Organizing your Electronic Life*, 1995, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.48.6769 . Aufgerufen am 03.02.2015. 48

[FRIE 05] FRIENDLY, MICHAEL: *Milestones in the History of Data Visualization: A Case Study in Statistical Histography*. In: WEIHS, C. und W. GAUL (Herausgeber): *Classification: The Ubiquitus Challenge*, Seiten 34–52. Springer, New York, 2005, http://www.math.yorku.ca/SCS/Papers/gfkl.pdf . Aufgerufen am 11.07.2015. 11

[FUR 86] FURNAS, GEORGE W.: *Generalized Fisheye Views*. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '86, Seiten 16–23. ACM, 1986, http://dl.acm.org/citation.cfm?id=22342 . Aufgerufen am 03.02.2015. 55

[FUR+ 95] FURNAS, GEORGE W. und BENJAMIN B. BEDERSON: *Space-scale Diagrams: Understanding Multiscale Interfaces*. In: *CHI '95 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, Seiten 234–241. ACM Press/Addison-Wesley Publishing Co., 1995, http://dx.doi.org/10.1145/223904.223934 . Aufgerufen am 11.07.2015. 52

[GEE+ 05] GEE, ALEXANDER G., MIN YU, HONGLI LI und GEORGES G. GRINSTEIN: *Dynamic and interactive Dimensional Anchors for Spring-Based Visualizations*, 2005,

http://www.cs.uml.edu/%7Eagee/publications/cs2005/cs2005.pdf . Aufgerufen am 06.10.2014. 7

[HAV+ 00] HAVRE, SUSAN, BETH HETZLER und LUCY NOWELL: *ThemeRiver: Visualizing Theme Changes over Time*. In: *Proceedings of the IEEE Symposium on Information Vizualization 2000*, INFOVIS '00. IEEE Computer Society, 2000, http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=885098 . Aufgerufen am 03.02.2015. 48

[HUN 13] HUNSINGER, ED: *Go Splunk Yourself!*, 2013, http://www.geeked.info/go-splunk-yourself/ . Aufgerufen am 29.04.2015. 135

[HUNT 15] HUNT, TROY: *Do you really want "bank grade" security in your SSL? Here's how Aussie banks fare*, 2015, http://www.troyhunt.com/2015/05/do-you-really-want-bank-grade-security.html . Aufgerufen am 30.05.2015. 176

[IETF 15] SHEFFER, Y., R. HOLZ und P. SAINT-ANDRE: *RFC 7525: Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)*, 2015, https://www.rfc-editor.org/rfc/rfc7525.txt . Aufgerufen am 09.05.2015. 174, 175, 180, 203

[ING 75] INGALLS, DAN: *Inter-Office Memorandum*, 1975, http://bitsavers.trailing-edge.com/pdf/xerox/alto/BitBLT_Nov1975.pdf . Aufgerufen am 13.02.2015. 53

[JAQ 07] JAQUITH, ANDREQ: *Security Metrics – Replacing Fear, Uncertainty, and Doubt*. Addison-Wesley, 2007. 143, 155

[KAH 13] KAHLE, CHRISTIAN: *Neues NSA-Datenzentrum schon öfter durchgebrannt*, 2013, http://winfuture.de/news,78209.html . Aufgerufen am 27.07.2015. 75

[KAS 12] KASPERSKY LAB: *Pressemitteilung: Rund ein Viertel der Nutzer verwendet veraltete Internet Browser*, 2012, http://newsroom.kaspersky.eu/de/texte/detail/article/rund-ein-viertel-der-nutzer-verwendet-veraltete-internet-browser/ . Aufgerufen am 01.06.2015. 203

[KEIM+ 06] KEIM, DANIEL A., FLORIAN MANSMANN, JORN SCHNEIDEWIND und HARTMUT ZIEGLER: *Challenges in Visual Data Analysis*. In: *Proceedings of the Conference on Information Visualization*, IV '06, Seiten 9–16. IEEE Computer Society, 2006, http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1648235 . Aufgerufen am 03.02.2015. 7

[KEIM+ 94] KEIM, DANIEL A. und HANS-PETER KRIEGEL: *VisDB: Database Exploration Using Multidimensional Visualization*. IEEE Computer Graphics and Applications, 14(5):40–49, 1994, http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=310723 . Aufgerufen am 03.02.2015. 30

[KEIM+ 95] KEIM, DANIEL A., HANS-PETER KRIEGEL und MIHAEL ANKERST: *Recursive Pattern: A Technique for Visualizing Very Large Amounts of Data*. In: *Proceedings of the 6th Conference on Visualization '95*, VIS '95, Seiten 279–286. IEEE Computer Society Press, 1995, http://dl.acm.org/citation.cfm?id=833846 . Aufgerufen am 03.02.2015. 31

[KEIM+ 98] KEIM, DANIEL A. und ANNEMARIE HERRMANN: *The Gridfit Algorithm: An Efficient and Effective Approach to Visualizing Large Amounts of Spatial Data*. In: *Proceedings of the Conference on Visualization '98*, VIS '98, Seiten 181–188. IEEE Computer Society Press, 1998, http://dl.acm.org/citation.cfm?id=288245 . Aufgerufen am 03.02.2015. 34

[KLI 13] KLINGE, MARCUS: *Desktop, Ordner und Dateien. Strukturen und Metaphern der selbstorganisierten Arbeit am Computer*. GRIN Verlag, 2013. 39

[KNO 14] KNOKE, FELIX: *Windows 7 geht in Rente – und wird immer beliebter*, 2014, http://www.spiegel.de/netzwelt/web/microsoft-hat-windows-8-aufgegeben-windows-7-ist-immer-noch-vorreiter-a-1000678.html . Aufgerufen am 27.12.2014. 71

[KOH 95] KOHONEN, TEUVO: *Self-Organizing Maps*. Springer, 1995. 32

[KWON+ 11] KWON, BUM CHUL, WAQAS JAVED, NIKLAS ELMKVIST und JI SOO YI: *Direct Manipulation Through Surrogate Objects*. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, Seiten 627–636. ACM, 2011, http://doi.acm.org/10.1145/1978942.1979033 . Aufgerufen am 09.07.2015. 43

[LEV 92] LEVKOWITZ, HAIM und GABOR T. HERMAN: *Color Scales for Image Data*. Computer Graphics and Applications, 12(1):72–80, 1992, http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=135886&tag=1 . Aufgerufen am 11.05.2015. 35

[LRZ 14] LEIBNIZ-RECHENZENTRUM DER BAYERISCHEN AKADEMIE DER WISSENSCHAFTEN: *Jahresbericht 2013*. 2014, http://www.lrz.de/wir/berichte/JB/JBer2013.pdf . Aufgerufen am 31.01.2015. 78, 80

[LU+ 07] LU, L., R. SAFAVI-NAINI, J. HORTON und W. SUSILO: *Comparing and debugging firewall rule tables*, 2007, http://www.uow.edu.au/ wsusilo/IETPaper_LHSS.pdf . Aufgerufen am 14.04.2015. 164

[MAN+ 12] MANSMANN, FLORIAN, TIMO GÖBEL und WILLIAM CHESWICK: *Visual Analysis of Complex Firewall Configurations*. In: *Proceedings of the Ninth International Symposium on Visualization for Cyber Security*, VizSec '12, Seiten 1–8, New York, NY, USA, 2012. ACM, http://doi.acm.org/10.1145/2379690.2379691 . Aufgerufen am 13.07.2015. 139, 167

[MCKE 13] MCKENNA, BRIAN: *Splunk CEO: Unintended use cases important to machine data business*, 2013, http://www.computerweekly.com/news/2240182913/Splunk-CEO-unintended-use-cases-important-to-machine-data-business . Aufgerufen am 29.04.2015. 135

[MEI 88] MEIER, BARBARA J.: *ACE: A Color Expert System for User Interface Design*. In: *Proceedings of the 1st Annual ACM SIGGRAPH Symposium on User Interface Software*, UIST '88, Seiten 117–128. ACM, 1988, http://doi.acm.org/10.1145/62402.62424 . Aufgerufen am 12.05.2015. 37

[MIL+ 03] MILLER, TODD und JOHN STASKO: *InfoCanvas: A Highly Personalized, Elegant Awareness Display*, 2003, http://www.cc.gatech.edu/ stasko/papers/chi03-workshop.pdf . Aufgerufen am 24.04.2015. 58

[MOHR 11] MOHR, PETER: *Optische Rhetorik: Visualisierungen und Medien in Präsentationen wirkungsvoll einsetzen*. Books on Demand GmbH, 2011. 39, 123, 143

[MOZ 14] MOZILLA FOUNDATION: *Most popular Extensions*, 2014, https://addons.mozilla.org/en-US/firefox/extensions/?sort=users . Aufgerufen am 27.10.2014. 20

[MOZ 14b] MOZILLA FOUNDATION: *Phasing Out Certificates with SHA-1 based Signature Algorithms*, 2014, https://blog.mozilla.org/security/2014/09/23/phasing-out-certificates-with-sha-1-based-signature-algorithms/ . Aufgerufen am 16.05.2015. 177

[MSDN 11] MICROSOFT DEVELOPER NETWORK: *So wird's gemacht: Auswählen zwischen SVG und Canvas*, 2011, https://msdn.microsoft.com/de-de/library/gg193983 . Aufgerufen am 27.06.2015. 189

[NET 08] NETAPP: *Duke Institute for Genome Sciences Speeds Research and Boosts VMware Performance with Netapp*, 2008, http://www.netapp.com/us/system/pdf-reader.aspx?pdfuri=tcm:35-59798-16&m=duke.pdf . Aufgerufen am 04.04.2015. 152

[NETC 15] NETCRAFT: *January 2015 Web Server Survey*, 2015, http://news.netcraft.com/archives/2015/01/15/january-2015-web-server-survey.html . Aufgerufen am 30.05.2015. 173, 191

[NOR 99] NORMAN, DONALD A.: *The Invisible Computer: Why Good Products Can Fail, the Personal Computer Is So Complex, and Information Appliances Are the Solution*. The MIT Press, 1999. 37

[NSA 01] NATIONAL SECURITY AGENCY: *Defense in Depth: A practical strategy for achieving Information Assurance in today's highly networked environments*, 2001, https://www.nsa.gov/ia/_files/support/defenseindepth.pdf . Aufgerufen am 23.04.2015. 65, 161

[OWA 13] THE OWASP FOUNDATION: *OWASP Top 10 – 2013 – The Ten Most Critical Web Application Security Risks*, 2013, http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202013.pdf . Aufgerufen am 03.02.2015. 62

[PAL 14] PALMER, CHRIS und RYAN SLEEVI: *Google Online Security Blog: Gradually sunsetting SHA-1*, 2014, http://googleonlinesecurity.blogspot.co.uk/2014/09/gradually-sunsetting-sha-1.html . Aufgerufen am 16.05.2015. 177

[PER+ 93] PERLIN, KEN und DAVID FOX: *Pad: An Alternative Approach to the Computer Interface*. In: *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, Seiten 57–64. ACM, 1993, http://doi.acm.org/10.1145/166117.166125 . Aufgerufen am 03.02.2015. 52

[PIC+ 88] PICKETT, RONALD M. und GEORGES G. GRINSTEIN: *Iconographic Displays for Visualizing Multidimensional Data*. In: *Proceedings of the 1988 IEEE International Conference on Systems, Man, and Cybernetics*, Band 1, Seiten 514–519, 1988, http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=754351 . Aufgerufen am 03.02.2015. 29

[PRE 15] PREIM, BERNHARD und RAIMUND DACHSELT: *Interaktive Systeme: Band 2: User Interface Engineering, 3D-Interaktion, Natural User Interfaces (eXamen.press)*. Springer Vieweg, 2015. 37

[PURC+ 08] PURCHASE, HELEN C., NATALIA ADRIENKO, T. J. JANKUN-KELLY und MATTHEW WARD: *Theoretical Foundations of Information Visualization*. In: KERREN, ANDREAS, JOHN T. STASKO, JEAN-DANIEL FEKETE und CHRIS NORTH (Herausgeber): *Information Visualization: Human-Centered Issues and Perspectives*, Band 4950 der Reihe *Lecture Notes in Computer Science*, Seiten 46–64. Springer, Berlin Heidelberg, 2008. 7

[SAL+ 75] SALTON, GERARD A., A. WONG und C. S. YANG: *A Vector Space Model for Automatic Indexing*. Communications of the ACM, 18(11):613–620, 1975, http://doi.acm.org/10.1145/361219.361220 . Aufgerufen am 03.02.2015. 16

[SAR+ 92] SARKAR, MANOJIT und MARC H. BROWN: *Graphical Fisheye View of Graphs*. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '92, Seiten 83–91. ACM, 1992, http://doi.acm.org/10.1145/142750.142763 . Aufgerufen am 11.07.2015. 55

[SCHE 14] SCHERSCHEL, FABIAN: *Hacker legen Sony Pictures komplett lahm*, 2014, http://heise.de/-2462889 . Aufgerufen am 27.12.2014. 65

[SCHE 14B] SCHERSCHEL, FABIAN: *ShellShock: Standard-Unix-Shell Bash erlaubt das Ausführen von Schadcode*, 2014, http://heise.de/-2403305 . Aufgerufen am 13.03.2015. 140

[SCHE 14C] SCHERSCHEL, FABIAN: *Fragwürdige Hash-Funktion SHA-1 immer noch beliebt*, 2014, http://heise.de/-2106581 . Aufgerufen am 16.05.2015. 177

[SCHEU 08] SCHEUER, STEPHAN: *Weiße Schrift auf schwarzem Grund oder Schwarze Schrift auf weißem Grund?*, 2008, http://www.fit-fuer-usability.de/archiv/weisse-schrift-auf-schwarzem-grund-oder-schwarze-schrift-auf-weissem-grund/ . Aufgerufen am 28.05.2015. 120, 256

[SCHI 91] SCHISCHKOFF, GEORGI: *Philisophisches Wörterbuch*. Kröner, Stuttgart, 22. Auflage, 1991. 22

[SCHIN+ 13] SCHINNER, ALEXANDER und STEFAN KEMMERER: *T-Systems Top Stories: Denken und handeln wie ein Angreifer*, 2013, https://public.t-systems.de/oeffentlicher-sektor/denken-und-handeln-wie-ein-angreifer/1153342 . Aufgerufen am 23.04.2015. 62

[SCHIR 15] SCHIRRMACHER, DENNIS: *Spionage-Trojaner wütete im Netzwerk von Kaspersky*, 2015, http://heise.de/-2687375 . Aufgerufen am 25.07.2015. 63

[SCHM 14] SCHMIDT, JÜRGEN: *Der GAU für Verschlüsselung im Web: Horror-Bug in OpenSSL*, 2014, http://heise.de/-2165517 . Aufgerufen am 27.12.2014. 76, 140, 170, 180

[SCHM 14B] SCHMIDT, JÜRGEN: *Mängel beim Selbstschutz von Antiviren-Software*, 2014, http://heise.de/-2465869 . Aufgerufen am 10.01.2015. 73

[SCHM 14C] SCHMIDT, JÜRGEN: *BadUSB: Wenn USB-Geräte böse werden*, 2014, http://heise.de/-2281098 . Aufgerufen am 23.04.2015. 62

[SCHM 15] SCHMIDT, JÜRGEN: *Das BSI und der Elfenbeinturm*, 2015, http://heise.de/-2589893 . Aufgerufen am 28.04.2015. 71

[SCHN 06] SCHNEIER, BRUCE: *Schneier on Security: Security in the Cloud*, 2006, https://www.schneier.com/blog/archives/2006/02/security_in_the.html . Aufgerufen am 23.04.2015. 161

[SHN 04] SHNEIDERMAN, BEN: *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison Wesley, 2004. 38

[SHN 82] SHNEIDERMAN, BEN: *The future of interactive systems and the emergence of direct manipulation*. In: *Proceedings of the NYU Symposium on User Interfaces on Human Factors and Interactive Computer Systems*, Seiten 1–28. Ablex Publishing Corp., 1982, http://www.tandfonline.com/doi/abs/10.1080/01449298208914450 . Aufgerufen am 15.07.2015. 41

[SHN 96] SHNEIDERMAN, BEN: *The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations*. In: *Proceedings of the 1996 IEEE Symposium on Visual Languages*, VL '96, Seiten 336–343. IEEE Computer Society, 1996, http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=545307&tag=1 . Aufgerufen am 13.03.2015. 141

[SIEG+ 72] SIEGEL, J., E. FARRELL, R. GOLDWYN und H. FRIEDMAN: *The surgical implication of physiologic patterns in myocardial infarction shock*. Surgery, 72:126–141, 1972. 30

[SIN+ 08] SINHA, SUSHANT, MICHAEL BAILEY und FARNAM JAHANIAN: *Shades of Grey: On the effectiveness of reputation-based "Blacklists"*. In: *MALWARE 2008. 3rd International Conference on Malicious and Unwanted Software*, Seiten 57–64. IEEE, 2008, http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4690858 . Aufgerufen am 23.04.2015. 72

[SIN+ 10] SINHA, SUSHANT, MICHAEL BAILEY und FARNAM JAHANIAN: *Improving Spam Blacklisting Through Dynamic Thresholding and Speculative Aggregation*, 2010, http://www.internetsociety.org/sites/default/files/sinh.pdf . Aufgerufen am 23.04.2015. 72

[SIR 13] SIRRIX AG: *TrustedDisk erhält Zulassung für Verschlusssachen*, 2013, http://www.sirrix.de/content/news/65356.htm . Aufgerufen am 30.01.2015. 70

[SKOG+ 03] SKOG, TOBIAS, SARA LJUNGBLAD und LARS ERIK HOLMQUIST: *Between Aesthetics and Utility: Designing Ambient Information Visualizations*. In: *Proceedings of the Ninth Annual IEEE Conference on Information Visualization*, INFOVIS '03, Seiten 233–240. IEEE Computer Society, 2003, http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1249031 . Aufgerufen am 24.04.2015. 57

[SMI 02] SMITH, LINDSAY I.: *A tutorial on Principal Components Analysis*, 2002, http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf . Aufgerufen am 03.02.2015. 32

[SPE 05] SPENNEBERG, RALF: *Intrusion Detection und Prevention mit Snort 2 & Co. – Einbrüche auf Linux-Servern erkennen und verhindern*. Addison-Wesley, 2005. 74

[SPEN+ 95] SPENCE, BOB, LISA TWEEDIE, HUW DAWKES und HUA SU: *Visualisation For Functional Design*. In: GERSHON, NAHUM und STEVE EICK (Herausgeber): *Information Visualization*, Seiten 4–10. IEEE, 1995, http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=528680 . Aufgerufen am 11.07.2015. 26

[TES 07] TESCHL, GERALD und SUSANNE TESCHL: *Mathematik für Informatiker: Band 1: Diskrete Mathematik und Lineare Algebra*. Springer, 2007. 26

[THO 11] THOMA, JÖRG: *Linus Torvalds nennt Gnome 3 ein "Großes Durcheinander"*, 2011, http://www.golem.de/1108/85472.html . Aufgerufen am 08.12.2014. 42

[TOP 10] TOPRAK, MEHMET: *Übernahme: Intel kauft McAfee – Geniestreich oder Fehler des Chip-Giganten?*, 2010, http://www.netzwelt.de/news/83781-uebernahme-intel-kauft-mcafee.html . Aufgerufen am 29.04.2015. 121

[TOS 97] TOSHIBA CORPORATION: *2SK170 Field Effect Transistor Silicon N Channel Junction Type for Low Noise Audio Amplifier Applications, Datenblatt*, 1997, http://pdf1.alldatasheet.com/datasheet-pdf/view/30581/TOSHIBA/2SK170.html . Aufgerufen am 31.10.2014. 26

[TRAN+ 07] TRAN, TUNG, EHAB AL-SHAER und RAOUF BOUTABA: *PolicyVis: Firewall Security Policy Visualization and Inspection*. In: *Proceedings of the 21st Conference on Large Installation System Administration Conference*, LISA '07, Seiten 1–16, Berkley, Kalifornien, 2007. USENIX Association, https://www.usenix.org/legacy/event/lisa07/tech/full_papers/tran/tran_html/index.html . Aufgerufen am 13.07.2015. 139

[TRE+ 80] TREISMAN, ANNE M. und GARRY GELADE: *A feature-integration theory of attention*. Cognitive Psychology, 12(1):97–136, 1980, http://www.sciencedirect.com/science/article/pii/0010028580900055 . Aufgerufen am 11.07.2015. 19

[TUFT 01] TUFTE, EDWARD R.: *The Visual Display of Quantitative Information*. Graphics Press, Zweite Auflage, 2001. 11, 12, 29, 45

[W3C 14] HICKSON, IAN, ROBIN BERJON, STEVE FAULKNER, TRAVIS LEITHEAD, ERIKA DOYLE NAVARA, EDWARD O'CONNOR und SILVIA PFEIFFER: *HTML5 – A vocabulary and associated APIs for HTML and XHTML – W3C Recommendation 28 October 2014*, 2014, http://www.w3.org/TR/html5/Overview.html . Aufgerufen am 28.02.2015. 204

[WARE 04] WARE, COLIN: *Information Visualization: Perception for Design*. Morgan Kaufmann, Zweite Auflage, 2004. 36, 41

[WAT 02] WATTENBERG, MARTIN: *Arc Diagrams: Visualizing Structure in Strings*. In: *Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02)*, INFOVIS '02, Seiten 110–116. IEEE Computer Society, 2002, http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1173155 . Aufgerufen am 03.02.2015. 50

[WEB+ 01] WEBER, MARC, MARC ALEXA und WOLFGANG MÜLLER: *Visualizing Time-Series on Spirals*. In: *Proceedings of the IEEE Symposium on Information Visualization 2001 (INFOVIS'01)*, INFOVIS '01, Seiten 7–13. IEEE Computer Society, 2001, http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=963273 . Aufgerufen am 03.02.2015. 47

[WEIN+ 78] WEINHANDL, FERDINAND und FREIHERR VON EHRENFELS, CHRISTIAN: *Gestalthaftes Sehen. Ergebnisse und Aufgaben der Morphologie. Zum Hundertjährigen Geburtstag von Christain von Ehrenfels*. Wissenschaftliche Buchgesellschaft, Darmstadt, 1978. 22

[WEIS+ 96] WEISER, MARK und JOHN SEELY BROWN: *Beyond Calculation*. Kapitel The coming age of Calm Technology, Seiten 75–85. Copernicus, 1996, http://www.ubiq.com/hypertext/weiser/acmfuture2endnote.htm . Aufgerufen am 24.04.2015. 57

[WIED 10] WIED, JOCHEN: *Aufwand und Nutzen von Value Added Printing*. Diplomica Verlag GmbH, 2010. 39

[WIL+ 92] WILLIAMSON, CHRISTOPHER und BEN SHNEIDERMAN: *The Dynamic HomeFinder: Evaluating Dynamic Queries in a Real-Estate Information Exploration System*. In: *Proceedings of the 15th Annual International ACM SIGIR Conference on Research*

and Development in Information Retrieval, SIGIR '92, Seiten 338–346. ACM, 1992, http://dl.acm.org/citation.cfm?id=133216 . Aufgerufen am 03.02.2015. 44

[WILK 09] WILKENS, ANDREAS: *Oracle übernimmt Sun*, 2009, http://heise.de/-214120 . Aufgerufen am 31.03.2015. 156

[WIN 13] WINDECK, CHRISTOF: *Facebook nimmt schwedisches Rechenzentrum in Betrieb*, 2013, http://heise.de/-1886765 . Aufgerufen am 28.10.2014. 6

[WOLF+ 13] WOLF, WERNER, WALTER BERNHART und ANDREAS MAHLER: *Immersion and Distance: Aesthetic Illusion in Literature and Other Media (Studies in Intermediality)*. Rodopi, 2013. 146

[WOOD+ 98] WOODRUFF, ALLISON, JAMES LANDAY und MICHAEL STONEBRAKER: *Constant Density Visualizations of Non-uniform Distributions of Data*, 1998, http://www.allisonwoodruff.com/publications/1998-Woodruff-UIST98-Nonuniform.pdf . Aufgerufen am 03.02.2015. 33, 268

[ZEH 98] ZENHDER, CARL AUGUST: *Informationssysteme und Datenbanken*. Teubner, Stuttgart, 1998. 6