INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Masterarbeit

Penetrationstesting als Service für einen IT-Provider am Beispiel des Web-Hosting- und IaaS-Services des Leibniz-Rechenzentrum (LRZ)

Manuel Kauschinger

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Masterarbeit

Penetrationstesting als Service für einen IT-Provider am Beispiel des Web-Hosting- und IaaS-Services des Leibniz-Rechenzentrum (LRZ)

Manuel Kauschinger

Aufgabensteller: Prof. Dr. Helmut Reiser

Betreuer: Dipl.-Inform. Stefan Metzger
Simon Graf

Abgabetermin: 07. August 2017

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

München, den 07. August 2017

. . . . . .

(Unterschrift des Kandidaten)

Die Sicherheit von IT-Systemen wird vor allem in Unternehmen und Organisationen ein immer wichtigeres Thema. Um die Sicherheit der IT-Systeme zu erhöhen, werden Penetrationstests durchgeführt. Diese können dabei helfen, Schwachstellen zu identifizieren und deren potenzielle Gefährlichkeit zu erkennen. Da professionelle Penetrationstests einer externen Firma sehr teuer und bei einer großen Anzahl betriebener Webanwendungen nicht wirtschaftlich sind, soll eine Möglichkeit geschaffen werden, Penetrationstests bei einer Vielzahl von Webservern kostengünstiger zu gestalten.

In dieser Arbeit wurden die Anforderungen für einen Service für Penetrationstests analysiert. Dazu wurde eine Umfrage erstellt, bei der insgesamt 67 Kunden und Mitarbeiter des LRZ teilgenommen haben. Im Anschluss wurde ein Konzept für einen solchen Service ausgearbeitet. Im weiteren Verlauf der Arbeit wurde der Service prototypisch implementiert und an einigen Testobjekten evaluiert.

Für die Implementierung des Service wurden verschiedene Sicherheitstools angebunden, um Schwachstellen zu identifizieren und im Anschluss automatisiert auszunutzen. Die Evaluation des Service an einer Auswahl von Webanwendungen hat ergeben, dass der Prototyp gut funktioniert. Der Service konnte die meisten Schwachstellen automatisch erkennen und auch das Ausnutzen der Schwachstellen war in den meisten Fällen erfolgreich.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

In der heutigen Zeit ist die Anzahl der Webserver, die im Internet zur Verfügung stehen, fast jährlich steigend, wie es in Abbildung 1.1 zu sehen ist. Viele Firmen, die ihren Kunden früher nur die Möglichkeit gegeben haben, Waren in den lokalen Filialen zu erwerben, bieten nun Online-Shops, um ihre Produkte auch auf diesem Wege zu verkaufen. Bei solchen Online-Shops fallen viele personenbezogene Daten von den Nutzern an, um die Kaufabwicklung abschließen zu können und das Produkt an den Kunden zu versenden. Diese Daten fallen aber nicht nur in Online-Shops an, sondern auch bei vielen anderen Diensten im Internet, wie zum Beispiel bei sozialen Netzwerken.

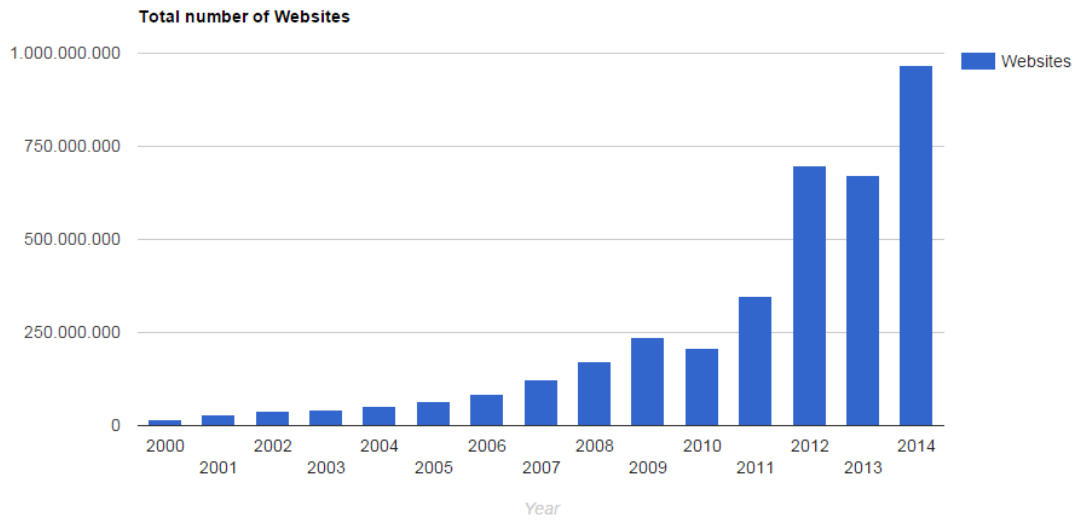


Abbildung 1.1.: Statistik über die Anzahl an Webseiten im Internet [ILS16]

Aufgrund dieser Entwicklung sind solche Dienste ein attraktives Ziel für Kriminelle, um an sensible Daten zu gelangen und diese auf Schwarzmärkten im Internet zu verkaufen. Hierzu gibt es eine Vielzahl von Fällen, die fast täglich in den Nachrichten zu lesen sind. Einige gute Beispiele hierfür sind massive Diebstähle von Nutzerdaten bei Dropbox, LinkedIn, Yahoo!, Tumblr und Myspace. Die Daten, welche bereits in den Jahren 2012 bis 2013 von Kriminellen entwendet wurden, werden nun auf Schwarzmärkten im Internet verkauft [SMD16] [ZEI16] [DRP16] [LFB16] [XBL16]. Aufgrund der erlangten Passwörter und der schlechten Angewohnheiten vieler Nutzer, die gleichen Zugangsdaten für verschiedene Dienste im Internet wiederzuverwenden, geht von einem sochen Diebstahl sensibler Daten eine große Gefahr aus. Dies zeigt eine Reihe von Einbrüchen auf die Twitterkonten einiger Prominenter wie Mark Zuckerberg und Richard Keith kurz nach der Veröffentlichung der geklauten LinkedIn Daten [MLO16].

Des Weiteren ist der neueste Trend sogar Dinge wie Haushaltsartikel und Überwachungskameras an das Internet zu bringen. Diese Geräte werden Internet of Things (IoT) genannt und von diversen Firmen in hoher Stückzahl verkauft. Oft wird dabei die Sicherheit der Geräte von den Herstellern außen vor gelassen, so dass sie viele Sicherheitslücken aufweisen. Ein kürzlich bekannt gewordener Fall ist das Mirai Botnetz. Die Mirai Schadsoftware scannt das Internet auf der Suche nach verwundbaren IoT Geräten, um diese zu kapern und sie zu einem großen Netz befallener Geräte zusammenzuschließen, die auf Anweisungen des Angreifers hören. Dadurch war es möglich, große Firmen wie den Hostingprovider OVH und den DNS Provider Dyn mit einer enormen Bandbreite anzugreifen.

Auch an Hochschulen und Universitäten steigt die Anzahl der in Betrieb genommenen Webserver jährlich an. Jede angebotene Vorlesung betreibt ihre eigene Webseite, auf der sie Informationen zu den jeweiligen Kursen veröffentlichen. Auch Studenten bekommen bei der Anmeldung einen eigenen Platz auf einem Webserver zugewiesen, um eine eigene Homepage zu betreiben. Aufgrund der Menge an Personen, die ihre Webauftritte innerhalb des Münchener Wissenschaftsnetzes betreiben, bietet sich für Angreifer eine große Angriffsfläche, wodurch das Wissenschaftsnetz zu einem lukrativen Ziel wird.

1.1. Motivation

Das Leibniz-Rechenzentrum (LRZ) ist der IT-Dienstleister der Ludwig-Maximilians-Universität (LMU), der Technischen Universität München (TUM) und der Münchener Hochschulen. Es bietet ihren Kunden eine große Auswahl an IT-Diensten wie Email- und Webservices bishin zu online Lehrangeboten. Diese Dienste werden den Universitäten und Hochschulen bereitgestellt, jedoch sind die Betreiber dieser Dienste für die Inhalte und den Betrieb selbst verantwortlich. Zudem steht den Kunden frei, bei Bedarf weitere neue Webserver für ihre Internetauftritte in Betrieb zu nehmen.

Ein Rechenzentrum, das eine so hohe Anzahl an Diensten bereitstellt, bietet Angreifern dadurch auch eine sehr große Angriffsfläche, was die Arbeit von Sicherheitsbeauftragten des LRZ erschwert. Da die Kunden ihre Webserver auch selbst in Betrieb nehmen können, hat das LRZ nur wenig Einfluss auf diese Server. Es kann schnell dazu kommen, dass wichtige Sicherheitspatches von den Kunden nicht eingespielt werden oder ihnen ein Fehler bei der Konfiguration der Dienste unterläuft. Aus diesem Grund beauftragt das LRZ in regelmäßigen Abständen externe Firmen, um professionelle Penetrationstests auf die verschiedenen Dienste durchzuführen. Des Weiteren setzt das LRZ ein Tool namens Dr. Portscan ein, das die einzelnen Dienste im gesamten Hochschulnetz auf offene Ports überprüft. Dies hilft, um ungewollt laufende Serverdienste zu erkennen und die zuständigen Personen über diese potentiell ungewollten Serverdienste zu informieren. Eine detaillierte Beschreibung zur Funktionsweise von Dr. Portscan wird in Kapitel 2 ausgeführt.

Um die Sicherheit der laufenden Webserver im LRZ zu erhöhen, ist es notwendig, in bestimmten Zeitintervallen nach Sicherheitslücken oder veralteter Software in Webservern zu suchen und die zuständigen Personen über die Ergebnisse zu informieren. Dies kann in Verbindung mit Dr. Portscan geschehen, indem zuerst das Netz nach offenen Ports durchsucht wird, um im Anschluss für alle Server mit einem offenen Port 80 (Http) oder Port 443 (Https) einen automatischen Penetrationstest durchzuführen.

Auch die Betreiber von Webservern in den verschiedenen Lehrstühlen der Universitäten und Hochschulen sollten bei der Absicherung ihrer Webauftritte unterstützt werden, um ein gutes Sicherheitsniveau schaffen zu können. Um dies zu realisieren, fehlt dem LRZ ein Service für Penetrationstests. Ein solcher Service kann den Administratoren der Webserver dabei helfen, um selbst gezielte Penetrationstests an ihren Webseiten durchzuführen.

1.2. Aufgabenstellung und Ziel der Arbeit

Die Durchführung von Penetrationstests ist eine wichtige Disziplin, um die Sicherheit der IT-Infrastruktur zu erhöhen. Aus diesem Grund hat das Bundesamt für Sicherheit in der Informationstechnik (BSI) einen Leitfaden und eine Studie für Penetrationstests veröffentlicht, welche die praxisorientierte Vorgehensweise eines Penetrationstests beschreibt. In diesem Dokument werden auch die Rahmenbedingungen, gesetzliche Aspekte und der genaue Ablauf eines Penetrationstests erläutert.

Um die IT-Sicherheit im Münchner Wissenschaftsnetz (MWN) auch in diesem Kontext zu verbessern, soll im Rahmen dieser Arbeit ein Service für automatische Penetrationstests entwickelt werden, um die Kunden und Sicherheitsverantwortlichen des LRZ bei der Erkennung von Schwachstellen in Webservern und Webapplikationen zu unterstützen. Zunächst sollen die Anforderungen der Kunden analysiert werden, um eine grobe Einschätzung zu bekommen, auf welche Sicherheitstests der Fokus liegen soll. Des Weiteren soll ein besonderes Augenmerk darauf liegen, dass der Service leicht anpass- und erweiterbar ist, da er für den praktischen Einsatz im LRZ gedacht ist und somit auch weiterentwickelt werden soll.

Ziel dieser Arbeit ist es nun, nach einer umfassenden Anforderungsanalyse einen Service für einen Penetrationstest zu entwerfen und zu implementieren. Um die Anforderungen an diesen Service zu erarbeiten, wird eine Umfrage für die Kunden, Sicherheitsverantwortlichen und Systemadministratoren des LRZ erstellt. Diese soll Aufschluss geben, welche Sicherheitstests von den einzelnen Parteien am meisten benötigt werden. Im Anschluss wird ein Konzept für einen solchen Service entwickelt. Dabei ist es wichtig zu bedenken, dass Kunden auf ihre eigenen Netze beschränkt werden müssen, um zu vermeiden, dass andere Parteien, die sich nicht im Netzbereich des jeweiligen Kunden befinden, angegriffen werden. Des Weiteren muss auch geklärt werden, wie oft ein solcher Scan durchgeführt wird und vor allem, wie gescannt werden soll. Dies ist nötig, um den reibungslosen Betrieb der Webserver nicht zu beeinträchtigen.

Bei einem Service für Penetrationstests werden im Laufe der Zeit neue Sicherheitsscans hinzukommen und andere, die sich nicht bewährt haben, werden abgeschaltet. Um dies zu administrieren, sollten Rollen eingeführt werden, um eine Hierarchie an Privilegien für die verschiedenen Benutzer zu schaffen. Somit können verschiedene Rollen vergeben werden, die jeweils die benötigten Rechte besitzen. Ein Kunde muss zum Beispiel bei einem Scan auf sein eigenes Netz beschränkt werden. Einem Sicherheitsverantwortlichen des LRZ muss es jedoch ermöglicht werden, auch größere Bereiche des MWN zu scannen. Eine weitere Rolle könnte auch ein Administrator sein, der neue Scanner hinzufügt, alte Scanner deaktiviert oder Kundenkonten verwaltet.

Des Weiteren muss nach jedem Scan ein Bericht für die verschiedenen Zielgruppen erstellt werden. Es muss einem Kunden mit wenig Hintergrundwissen in der IT-Sicherheit ermöglicht werden, diesen Service sinnvoll zu nutzen, indem ein ausführlicher Bericht über gefundene Schwachstellen, detaillierte Lösungsvorschläge und deren Risikopotential erstellt wird. Wenn jedoch Sicherheitsverantwortliche des LRZ einen Scan durchführen, genügt eine übersichtliche Zusammenfassung der Schwachstellen und das daraus resultierende Risiko.

Da das LRZ bereits das Tool Dr. Portscan einsetzt, um laufende Dienste auf Servern zu erkennen, liegt es nahe, dass es Dr. Portscan ermöglicht werden sollte, auf den Service für Penetrationstests zuzugreifen. Um diese zwei Tools zu verbinden, muss eine API als Schnittstelle breitgestellt werden, um den Penetrationstest auch automatisiert steuern zu können. Diese API kann auch von einem Kunden verwendet werden, falls dieser den Service gerne skriptgesteuert verwenden will. Des Weiteren muss es aber für die Betreiber von Webservern auch möglich sein, diesen Service leicht zu verwenden. Um dies zu erreichen, muss zusätzlich zu einer API ein intuitives Graphical User Interface (GUI) für die manuelle Verwendung des Service bereitgestellt werden.

Aufgrund der vielen Dienste, die im MWN betrieben werden, sollte es auch möglich sein, eine Menge von IP-Adressen, die gescannt werden sollen, einzubinden. Hier bietet es sich an, eine Import-Funktion für CVS und XML Dateien bereitzustellen.

1.3. Struktur der Arbeit

Die Arbeit ist strukturiert in acht Kapitel. Das Kapitel 2 befasst sich mit den Grundlagen der IT-Sicherheit. Hier wird vor allem darauf eingegangen, was Sicherheit in IT-Systemen bedeutet und was ein Penetrationstest ist. Im Anschluss werden die verschiedenen Sicherheitslücken genau beschrieben, um ein Grundverständnis zu schaffen. Des Weiteren folgt eine Beschreibung der Sicherheitsrisiken von Webservern und die Ausführung der meist verwendeten Angriffe auf Webanwendungen, welche von OWASP aufgestellt wurden.

In Kapitel 3 wird zunächst die Ausgangssituation am LRZ beschrieben und dabei vor allem auf die Sicherheit in Hochschulen und öffentlichen Einrichtungen eingegangen. Danach erfolgt eine Abgrenzung dieser Arbeit zu einem klassischen Schwachstellenscan. Im weiteren Verlauf dieses Kapitels werden die Anforderungen, die an einen Service für Penetrationstests gestellt werden, analysiert und beschrieben, um einen Überblick zu schaffen, auf welche Funktionen besonders Wert gelegt wird.

Im vierten Kapitel werden themenverwandte Arbeiten betrachtet und vorgestellt, um sie mit den Anforderungen an das zu entwickelnde Konzept zu vergleichen und die Notwendigkeit dieser Arbeit zu untermauern.

Nachfolgend beschreibt Kapitel 5 das Konzept des zu implementierenden Service und in Kapitel 6 wird beschrieben, wie die Implementierung durchgeführt wurde.

In Kapitel 7 wird aufgezeigt, wie die Evaluation des Prototyps durchgeführt wurde und wie genau das Endprodukt Schwachstellen in Webanwendungen aufspüren konnte. Die Ergebnisse der prototypischen Anwendung

werden am Ende dieses Kapitels in einem Resüme zusammengefasst.

In Kapitel 8 wird diese Arbeit zusammengefasst und weitere Funktionen und Erweiterungen vorgeschlagen, die für den Produktiveinsatz eines Service für Penetrationstests wichtig sind.

2. Grundlagen

2.1. Informationssicherheit

Die Sicherheit von IT-Infrastrukturen wird zunehmend wichtiger, da Cyber-Angriffe auf Organisationen und Unternehmen steigen. Jedoch gibt es nicht nur Angriffe von außen. Es besteht auch ein Gefahrenpotential von Mitarbeitern, die unter anderem Sabotage oder Spionage betreiben könnten. In der Informationssicherheit (IS) gibt es keine Maßeinheit für Sicherheit. Daher wird versucht, anhand von Teilzielen die Informationssicherheit zu definieren. Diese Teilziele sind Vertraulichkeit, Integrität und Verfügbarkeit, was in englischer Literatur auch als CIA bezeichnet wird, was für Confidentiality, Integrity und Availability steht. Nach Prof. Hartmut Pohl gibt es auch noch ein viertes Teilziel, die Verbindlichkeit [POH01]. Im Folgenden werden diese Begriffe kurz erklärt und mit jeweils einem Beispiel untermauert.

In IT-Systemen kommt es oft dazu, dass vertrauliche Informationen gespeichert werden. Zu vertraulichen Informationen zählen unter anderem Kreditkartennummern, Adressen und Namen von Kunden. Das Ziel **Vertraulichkeit** in der Informationssicherheit beinhaltet, dass solche vertraulichen Daten nur von Personen eingesehen werden können, die dazu berechtigt sind. Befindet sich jedoch eine Sicherheitslücke in einem System, das es dem Angreifer ermöglicht diese Informationen einzusehen, ist die Vertraulichkeit der Daten gefährdet. Hier ist auch zu beachten, dass die Vertraulichkeit auch geschützt werden muss, wenn sensible Daten über ein unsicheres Medium übertragen werden. Eine Möglichkeit, die Vertraulichkeit zu schützen, ist das Verschlüsseln der Daten.

Ein weiteres Ziel der Informationssicherheit ist die **Integrität**. Nach dem Glossar des IT-Grundschutzes des BSI wird die Integrität als die „Sicherstellung der Korrektheit (Unversehrtheit) von Daten und der korrekten Funktionsweise von Systemen“definiert [BSI13a]. Dies bedeutet, dass Daten auf dem System so zu schützen sind, dass sie nicht unerlaubt verändert werden können. Ein Beispiel für den Verlust von Integrität ist, wenn ein Angreifer Zugriff auf einen Webserver erlangt und die Webseite so manipulieren kann, dass sie Schadcode im Browser des Benutzers ausführt.

Des Weiteren ist auch die **Verfügbarkeit** des Systems ein Ziel der Informationssicherheit. Die Verfügbarkeit besagt, dass ein Dienst, IT-System oder auch Information von Benutzern zu jeder Zeit wie gewohnt abrufbar sein muss. Wenn die Verfügbarkeit nicht gewährleistet ist, kann dadurch auch großer Schaden entstehen. Ein Beispiel hierfür ist, wenn in einem Rechenzentrum häufig Fehler durch Angriffe auftreten und die IT-Infrastruktur von Kunden nicht mehr erreichbar ist. Hierdurch entsteht ein potentieller Schaden für den Kunden und bei häufigerem Auftreten wird sich dieser nach einem Dienstleister umsehen, der zuverlässiger ist.

Bei der **Verbindlichkeit** handelt es sich um die Möglichkeit, bestimmte Aktionen den ausführenden Personen eindeutig und gerichtsverwertbar nachweisen zu können. Dies ist zum Beispiel bei Online Shops von großer Bedeutung, damit bei einem Kauf eines Artikels ein rechtsverbindlicher Kaufvertrag zustande kommt. In diesem Beispiel kann die Verbindlichkeit gesichert werden, indem Log-Einträge vorgenommen werden, welche Person mit welcher IP-Adresse einen Artikel gekauft hat.

2.2. Dr. Portscan

In Rechnernetzen mit einer dezentralen Administration besteht die Gefahr, dass auf Servern ungewollte Dienste laufen, die vom Internet aus zugänglich sind und somit ein Sicherheitsrisiko darstellen. Um das eigene Netz abzusuchen und solche Dienste aufzuspüren, gibt es viele kostenlose aber auch kommerzielle Portscanning-Tools. Mit einer steigenden Anzahl an Servern in einem Netz wird ein solcher Scan jedoch sehr zeitintensiv

und unübersichtlich. Oft fehlt bei den Unternehmen auch das Personal um solche Scans durchzuführen und auszuwerten.

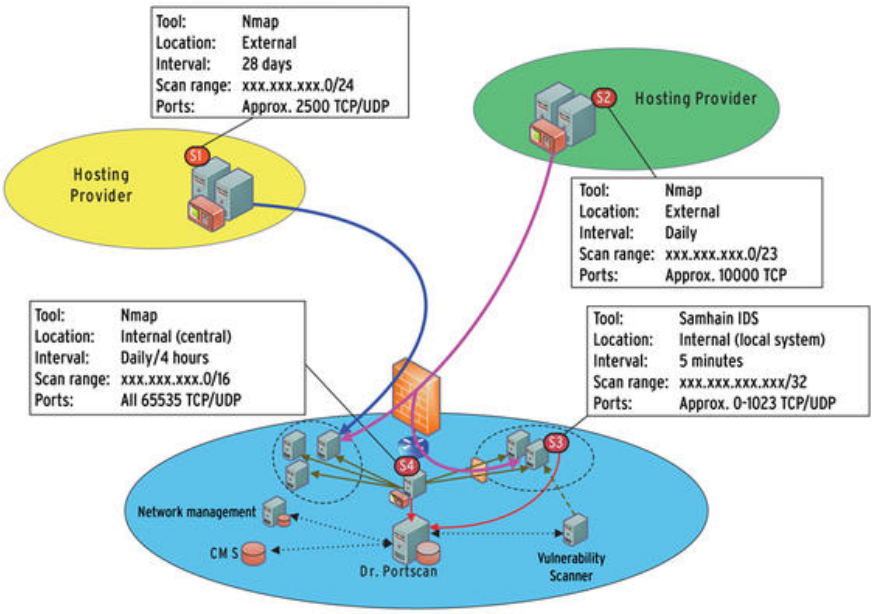


Abbildung 2.1.: Ausgangssituation vor der Entwicklung von Dr. Portscan

Für das MWN wurden in der Ausgangssituation von verschiedenen Systemen des DFN und einem externen Betreiber mit dem Open Source Portscanning Tool nmap das Netz auf offene Ports untersucht, wie in Abbildung 2.1 zu sehen ist. Das Problem hierbei war, dass es sich als schwierig erwies, die Resultate mit teils unterschiedlichen Ausgabeformaten auszuwerten. Aus diesem Grund wurde am LRZ das Delta Reporting Portscan Tool Dr. Portscan entwickelt, um eine automatisierte Portscan Anwendung zu schaffen, die alle Resultate verschiedener Analysetools in einem übersichtlichen Report zusammenfasst.

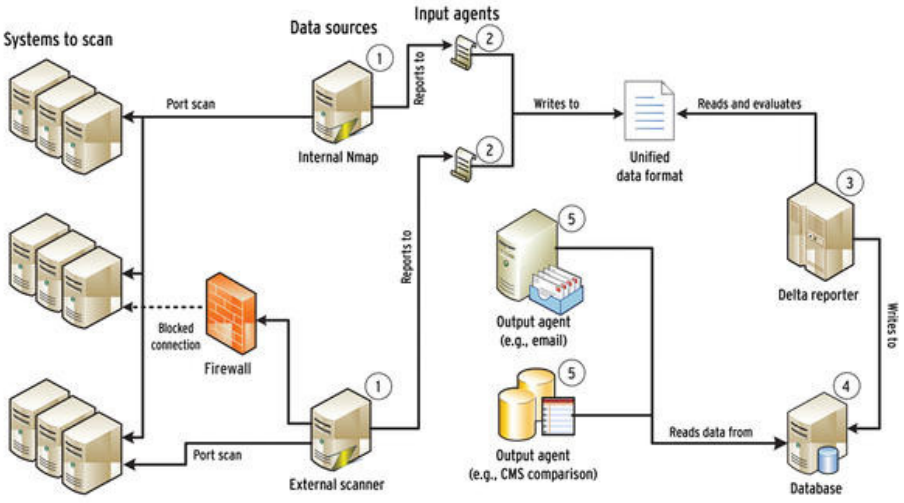


Abbildung 2.2.: Aufbau von Dr. Portscan

Dr. Portscan ist modular aufgebaut und besteht somit aus verschiedenen Komponenten. Dazu gehören die Portscanner, der Input- und Outputagent, der Delta-Reporter und eine zentrale Datenbank, wie in Abbildung

2.2 zu sehen ist. Die Ergebnisse der Portscanner werden über die Inputagenten in ein einheitliches Datenformat konvertiert und an eine zentrale Dr. Portscan-Instanz, den Delta-Reporter, weitergegeben. Von hier aus werden die Ergebnisse der Scanner automatisch aggregiert und miteinander verglichen. Aus den entdeckten Veränderungen können nun Berichte für verschiedene Zielgruppen erstellt oder auch als Parameter an andere Programme weitergegeben werden. Einen Überblick über die möglichen Ergebnisse des Delta-Reporters zeigt Tabelle 2.1.

Nr		Beschreibung
0		Keine Veränderung seit dem letzten Scan-Lauf
1		Neue Maschine wurde gefunden
2		Eine zuvor gefundene Maschine läuft nicht mehr
4		Der DNS-Name einer Maschine hat sich geändert
8		Es wurde ein neuer offener Port gefunden
16		Ein zuvor geöffneter Port ist nun geschlossen
32		Ein Port, der geschlossen wurde, ist nun wieder offen

Tabelle 2.1.: Veränderungstypen des Delta-Reporters

Die aggregierten Ergebnisse des Delta-Reporters werden nun zur weiteren Verwendung in einer zentralen Datenbank gespeichert, von der die Outputagenten diese Informationen wieder auslesen können. Die Outputagenten sorgen dafür, dass die Ergebnisse der Scans übersichtlich eingesehen werden können oder ein Bericht per Mail an die zuständigen Personen verschickt wird. [PSC13]

2.3. Penetrationstest

In vielen IT-Anwendungen, IT-Systemen oder Rechnernetzen verbergen sich Schwachstellen, die dem Betreiber nicht bekannt sind. Dies stellt ein großes Sicherheitsrisiko dar, denn ein Angreifer, der sich mit dem System beschäftigt, findet unter Umständen eine Schwachstelle, die ihm unerlaubten Zugriff auf das System gewährt. Um dieses Risiko zu mindern gibt es Organisationen, die Penetrationstests für ihre Kunden durchführen. Bei einem Penetrationstest handelt es sich um die gezielte Suche und Ausnutzung von Schwachstellen, um zu identifizieren, welche Schwachstellen in einem System existieren und wie weit ein Angreifer durch diese in ein System eindringen kann. Bei einem Penetrationstest werden verschiedene Methoden verwendet, die auch ein Angreifer verwenden würde. Das Ziel eines Penetrationstests ist nicht nur die Identifikation von Schwachstellen, sondern auch das Aufdecken von Fehlern, die bei einer fehlerhaften Bedienung einer Anwendung auftreten können sowie die Bestätigung der IT-Sicherheit durch einen externen Dritten [BSI03]. Letzteres ist wichtig, um bei einem solchen Test das Übersehen von Schwachstellen (Betriebsblindheit) zu vermeiden.

2.3.1. Kriterien für Penetrationstests

Bei einem Penetrationstest gibt es eine Vielzahl von verschiedenen Zielsetzungen, die vor dem Test festgelegt werden müssen. Somit kann bei einem Penetrationstest ein realistischer Angriff simuliert werden, aber auch ein Angriff von Insidern, die das Firmennetzwerk von ihrer täglichen Arbeit sehr gut kennen. Hierfür gibt es verschiedene Kriterien, die vor einem Test berücksichtigt werden müssen. Im Nachfolgenden werden diese Kriterien nach der Studie für Penetrationstests des BSI [BSI03] beschrieben.

Informationsbasis

Bei der Informationsbasis muss entschieden werden, wie viel Information der Tester über das anzugreifende Ziel erhalten soll. Hier unterscheidet man zwischen Black-Box-, Grey-Box- und White-Box-Test.

Bei einem **Black-Box-Test** bekommt der Tester nur sehr wenig bis zu keiner Information über das Angriffsziel. Dieser Test simuliert einen realistischen Angriff, da der Tester sich erst mit dem zu testenden System auseinan-

dersetzen muss, um Details zu recherchieren, wie zum Beispiel welche Dienste mit welchen Versionsnummern dort laufen. Dies ist für den Tester sehr aufwendig und zeitintensiv.

Im Gegensatz zu einem Black-Box-Test bekommt der Tester bei einem **White-Box-Test** mehr Informationen zu dem Angriffsziel. Ein solcher Test soll zeigen, wie weit ein Insider mit sehr viel Wissen über die IT-Infrastruktur des Unternehmens in das Ziel eindringen kann. Hierfür bekommt der Tester den vollen Umfang an Informationen wie IP-Adressen, verwendete Netzwerkprotokolle und den Source Code von Anwendungen, die auf dem Zielsystem laufen.

Wird von einem **Grey-Box-Test** gesprochen, handelt es sich dabei um eine Kombination des Black-Box- und des White-Box-Tests. Dieser gibt Aufschluss über die Gefahren, die von Dienstleistern oder kurzzeitigen Mitarbeitern ausgehen, die nur begrenztes Wissen über die IT-Infrastuktur des Unternehmens haben.

Aggressivität

Die Aggressivität eines Penetrationstests wird in passiv, vorsichtig, abwägend und aggressiv unterteilt. Bei einer passiven Aggressivitätsstufe werden die gefundenen Schwachstellen nur dokumentiert, aber nicht weiter ausgenutzt. Wird jedoch der vorsichtige Ansatz gewählt, werden Schwachstellen nur dann ausgenutzt, wenn ein Systemausfall aufgrund des Angriffs ausgeschlossen werden kann. Bei diesem Ansatz werden auch nur Angriffsmethoden gewählt, die sehr ressourcenschonend sind. Bei einem Test mit Aggressivitätsgrad "abwägend" wird versucht, das Zielsystem nur so zu testen, dass eine Beeinträchtigung des Systems unwahrscheinlich ist, jedoch aber vorkommen kann. Schon vor dem Test wird abgewägt wie wahrscheinlich es ist, erfolgreich zu sein und welche Konsequenzen entstehen können. Die letzte Aggressivitätsstufe ist aggressiv. Hierbei werden alle möglichen Schwachstellen ohne Rücksicht auf die Verfügbarkeit der Systeme getestet. Bei einem solchen Test kann es passieren, dass auch andere Systeme bis hin zur ganzen IT-Infrastruktur ausfallen können.

Umfang

Bei einem Penetrationstest sollten immer alle Systeme auf Schwachstellen untersucht werden. Liegt der Fokus nur auf bestimmten Komponenten, besteht weiterhin die Gefahr, dass es ein Einfalltor in das interne Netz gibt. Bekommt ein Angreifer einmal unerlaubten Zugriff in das innere Netz, bieten sich noch mehr Möglichkeiten, weitere Systeme zu befallen. Jedoch ist ein vollständiger Penetrationstest bei sehr großen Netzen nicht in kurzer Zeit machbar. Daher liegt der Fokus oft auf besonders gefährdeten Komponenten wie Systeme, die direkt an das Internet angebunden sind oder sehr sensible Daten enthalten. Daher existieren somit neben dem vollständigen Test auch der fokussierte- und der begrenzte Penetrationstest. Der fokussierte Test wird oft angewandt, wenn neue Systeme oder Anwendungen betrieben werden, um ein gleichmäßiges Sicherheitsniveau zu schaffen. Bei einem begrenzten Test liegt der Fokus auf einem bestimmten Teil der Infrastruktur.

Vorgehensweise

Die Vorgehensweise unterscheidet sich hauptsächlich in einem verdeckten und einem offensichtlichen Test. Das Ziel eines verdeckten Penetrationstests ist es, Sicherheitsanwendungen wie ein Intrusion Detection System (IDS) auf die Wirksamkeit zu prüfen oder auch die Mitarbeiter einer Organisation mittels Social Engineering zu testen. Bei einem verdeckten Test wird nur auf Methoden gesetzt, welche vom System nicht als Angriff gewertet werden. Fällt die Entscheidung jedoch auf einen offensichtlichen Test, so können je nach dem anzugreifenden System offensichtliche Sicherheitstests wie SQL-Injection oder Portscans durchgeführt werden.

Technik

Ein weiteres wichtiges Kriterium bei einem Penetrationstest ist die Technik. Soll ein realer Angriff von einem Cyberkriminellen simuliert werden, wird der Penetrationstest meist über das Netzwerk durchgeführt. Jedoch gibt es auch andere Einfallstore, die getestet werden sollten. Hat ein Angreifer zum Beispiel physischen Zugriff auf ein System, könnte es leichter fallen, bestimmte Schwachstellen auszunutzen, die über das Netzwerk wegen

einer existierenden Firewall nicht ausnutzbar sind. Des Weiteren besteht auch die Möglichkeit, Mitarbeiter des Unternehmens mit einem Social Engineering Angriff zur Herausgabe von Zugangsdaten zu bringen.

Ausgangspunkt

Der Ausgangspunkt bei einem Penetrationstest beschreibt, von wo der Angriff gestartet wird. Die meisten Organisationen betreiben eine Firewall, um den Zugriff nur auf gewisse Dienste zu unterbinden. Daher ist es oft schwer, das dahinterliegende System anzugreifen. Aus diesem Grund konzentriert sich ein Penetrationstest von außen auf die Konfiguration der eingesetzten Firewall, um zu testen, ob diese Konfigurationsfehler enthält, die es einem externen Angreifer ermöglicht, in das Innere eines Netzes einzudringen. Es ist aber auch wichtig, den Penetrationstest von innen durchzuführen, da hier in vielen Fällen keine Firewall übergangen werden muss, um die laufenden Dienste und Anwendungen auf ihre Sicherheit zu überprüfen. Ein Test von innen kann zeigen, wie gefährlich eine Schwachstelle in der Firewall wäre oder welche Möglichkeiten sich für einen Innentäter bieten würden.

2.3.2. Ablauf eines Penetrationstest

Entscheidet sich eine Organisation einen Penetrationstest durchführen zu lassen, muss dieser gut geplant werden. Das Vorgehen bei einem solchen Test wird in fünf Phasen unterteilt, die im Folgenden beschrieben werden [BSI03].

Phase 1: Vorbereitung

Um den Anforderungen des Auftraggebers gerecht zu werden, bedarf es einer gründlichen Vorbereitung. In dieser Phase muss geklärt werden, welche Komponenten getestet werden sollen und wie weit ein Penetrationstest gehen darf. Hier kann der Auftraggeber den Tester auf einen bestimmten Bereich begrenzen, der für einen Sicherheitstest besonders wichtig ist. Des Weiteren muss auch geklärt werden, welche Informationen der Tester über die IT-Infrastruktur des Unternehmens bekommt. Bei diesem Schritt wird entschieden, ob es sich um einen Black-Box-Test, Grey-Box-Test oder einen White-Box-Test handelt. Da es auch gesetzliche Bestimmungen gibt, die das Angreifen von Computersystemen und Netzwerken verbieten, müssen bei einem Penetrationstest alle durchzuführenden Tests und deren Risiken vertraglich vereinbart und dokumentiert werden, um spätere Schadenersatzansprüche zu vermeiden.

Phase 2: Informationsbeschaffung

Nachdem die Vorbereitungen abgeschlossen sind und alle wichtigen Eckpunkte vereinbart wurden, kann mit der Beschaffung von Information über die Zielsysteme begonnen werden. Um einen Überblick zu bekommen, welche Dienste erreichbar sind, wird ein Portscan gegen das Zielsystem durchgeführt. Des Weiteren benötigt der Tester Informationen über die eingesetzten Systeme und installierten Anwendungen, um einen detailreichen Überblick über die möglichen Angriffspunkte zu erlangen. Je nach Größe des Netzes oder der Menge zu testender Komponenten sollte in dieser Phase genug Zeit eingeplant werden. Beinhaltet der Test eine große Menge an Rechnern, kann die Informationsbeschaffung einige Wochen andauern.

Phase 3: Bewertung der Information

In dieser Phase werden die erlangten Informationen aus Phase 2 ausführlich zusammengetragen und das jeweilige Risiko bewertet. Um den Penetrationstest effizient durchführen zu können, werden anhand einer Risikobewertung der gesammelten Informationen entschieden, welche Komponenten in der nächsten Phase genauer betrachtet werden. Diese Reduktion der zu testenden Komponenten bedeutet natürlich auch eine Einschränkung des resultierenden Ergebnisses. Daher muss dies ausführlich dokumentiert und an den Auftraggeber weitergegeben werden.

Phase 4: Aktive Eindringversuche

In dieser Phase wird geprüft, wie sicherheitskritisch die ausgewählten Sicherheitsmängel von Phase 3 wirklich sind. Dies geschieht durch den Versuch, so weit wie möglich in ein System vorzudringen. Hier ist wichtig, jeden Schritt genau zu bedenken, da durch Eindringversuche die Zielsysteme auch beschädigt werden könnten. Wird dabei ein System getestet, das eine hohe Verfügbarkeit haben soll, so muss bedacht werden, wie der Test aufgebaut wird, um die Verfügbarkeit weiterhin zu gewähren. Eine weitere Möglichkeit, um die Verfügbarkeit der zu testenden Systeme sicherzustellen, ist das Verwenden von Schattensystemen. Dabei handelt es sich um eine exakte Kopie des zu testenden Systems. Der Vorteil bei der Verwendung von Schattensystemen ist, dass während des Penetrationstests sichergestellt ist, dass es zu keinen Ausfällen des originalen Systems kommt.

Phase 5: Abschlussanalyse

In der letzten Phase werden alle gefundenen Schwachstellen in einem Abschlussbericht zusammengefasst und deren Risiken genau erläutert. Ein solcher Bericht muss neben den Resultaten des Penetrationstests auch Möglichkeiten zur Behebung ausführen. Es ist wichtig, dass jede durchgeführte Aktion so beschrieben wird, dass sie für den Auftraggeber nachvollziehbar ist und gegebenenfalls wiederholt werden kann. Nach der Fertigstellung des Berichts sollte mit dem Auftraggeber ein Abschlussgespräch geführt werden, in dem noch einmal alle gefundenen Sicherheitsprobleme ausführlich besprochen werden.

2.3.3. Rechtliche Aspekte

Das Eindringen in IT-Systeme steht in Deutschland unter Strafe. So hat die Bundesregierung im Jahr 2007 ein Gesetz verabschiedet, das umgangssprachlich unter dem Namen Hackerparagraph bekannt ist. Hierbei handelt es sich um §203c des Strafgesetzbuches. Dieser Paragraph stellt das Vorbereiten des Ausspähens und des Abfangens von Daten unter Strafe. Dieses Gesetz beinhaltet auch das Erstellen und Benutzen von Programmen, die es ermöglichen, Daten abzufangen oder auszuspähen. Des Weiteren gibt es im Strafgesetzbuch noch weitere Paragraphen wie §202a StGB und §202b StGB. Ersteres stellt das Ausspähen von Daten unter Strafe und Letzteres verbietet das Abfangen von Daten [STG71].

Diese Gesetze sind sehr auf Kritik gestoßen, da sie im Strafgesetzbuch sehr vage beschrieben wurden und keine Sonderregelungen beinhalten, welche die legale Anwendung für gute Zwecke erlauben. Somit sind diese Gesetze durchaus ein Problem für Penetrationstester und IT-Sicherheitsspezialisten, da sie sich hier in einer gesetzlichen Grauzone bewegen. Aufgrund dieser Tatsache hat sich der Chefredakteur des iX Magazins im Jahr 2008 wegen der Vorbereitung des Abfangens und Ausspähens von Daten selbst angezeigt, da in diesem Magazin ein Artikel veröffentlicht wurde, der Administratoren dabei helfen soll, Systemeinbrüche in deren eigener IT-Systeme zu simulieren, um bestehende Schwachstellen aufzudecken [HEI08]. Des Weiteren wurde zu dem Magazin eine DVD beigelegt, die die damals besten Tools für den Einbruch in Systeme beinhaltete. Nach dieser Selbstanzeige gab es noch eine Reihe weiterer Personen, die sich wegen der gleichen Gründe bei der Staatsanwaltschaft selbst anzeigten. Die Selbstanzeige wurde später jedoch von der Staatsanwaltschaft eingestellt, da sich der Artikel mit dem Schutz vor Angriffen befasste und somit nicht mit der Vorbereitung einer Straftat zu werten sei [HEI09]. Auch andere Selbstanzeigen wurden später eingestellt.

Trotz der Tatsache, dass es bis heute in Deutschland keine Verurteilung wegen der gutartigen Verwendung solcher Tools gab, muss bei einem Penetrationstest vertraglich genau vereinbart werden, welche Aktionen bei einem Test durchgeführt werden [HEI13]. An diese vertraglichen Bestimmungen muss sich der Tester auch strikt halten, da es sonst dazu kommen kann, dass sein Handeln zu einer Strafanzeige führt. Diese vertraglichen Vereinbarungen werden in der zuvor beschriebenen Vorbereitungsphase genau geregelt.

2.3.4. Unterstützende Tools

Um Penetrationstests effizient durchführen zu können, bedarf es einiger automatisierter Tools, die dem Tester viel Arbeit abnehmen. Heute gibt es eine Vielzahl solcher Tools, von denen im Folgenden eine kleine Auswahl an Programmen vorgestellt werden sollen.

WPScan

Wordpress ist eine Webanwendung zur Erstellung und Verwaltung eigener Webseiten (Content Management System oder CMS). Das Projekt wurde im Jahr 2003 ins Leben gerufen und ist seitdem unter einer Open-Source-Lizenz verfügbar. Wordpress ist eines der meist benutzten CMS und ist daher auch ein attraktives Ziel für Angreifer [WPS17]. WPScan ist ein Tool, das Administratoren verwenden können, um eigene Wordpress-Webseiten auf Schwachstellen zu überprüfen. Das Tool überprüft alle Teile der Webseite auf bekannte Schwachstellen und gibt am Ende eine Zusammenfassung aller gefundenen Schwachstellen aus.

JoomlaScan

Joomla! ist ähnlich wie Wordpress ein Content Management System mit einer sehr hohen Beliebtheit. Damit auch möglich ist, Webseiten, welche mit Joomla! erstellt wurden, auf ihre Sicherheit zu untersuchen, wurde JoomlaScan ins Leben gerufen. Dieses Tool arbeitet ähnlich wie WPScan, indem es alle Teile der Webseite auf bekannte Schwachstellen untersucht und am Ende einen Bericht zu den Funden ausgibt.

SQLMap

SQLMap ist ein Tool zur automatischen Erkennung von Schwachstellen in Form von SQL-Injections. Zusätzlich ist es möglich, gefundene Schwachstellen auszunutzen und somit die Datenbank potentiell zu übernehmen. Da dieses Tool die Schwachstellen nicht nur aufdeckt und zu einem Bericht zusammenfasst, sondern die Webanwendung aktiv angreift, ist dies ein wichtiges Tool für einen effektiven Penetrationstest [SQL17].

Zed Attack Proxy

Zed ist ein Tool, das von OWASP ins Leben gerufen wurde. Es ist Open Source und viele freiwillige Softwareentwickler auf der ganzen Welt helfen bei der Entwicklung. Zed kann von Anfängern sowie von Sicherheitsexperten verwendet werden und hat viele Funktionen, die einen Penetrationstest einfacher gestalten. Die im Rahmen dieser Arbeit wichtigste Funktion ist der Intercepting Proxy, mit dem HTTP-Anforderungen sowie HTTP-Antworten abgefangen und verändert werden können. Des Weiteren ist in Zed ein Spider integriert, der es ermöglicht, alle Links einer Webseite in einer Datei zu speichern, damit die Information zu einem späteren Zeitpunkt verarbeitet werden kann [ZEN17].

Hydra

Hydra oder auch THC-Hydra ist ein Programm zur Durchführung von Wörterbuchattacken (Brute-Force-Attack). Bei einer Wörterbuchattacke wird versucht, ein Passwort anhand eines Wörterbuches zu erraten. Als Wörterbuch wird oft eine Liste der meist verwendeten Passwörter verwendet. Da Hydra ein kommandozeilenbasiertes Programm ist, lässt es sich auch gut für automatisierte Penetrationstests verwenden [HYD17].

Weevely Shell

Weevely ist eine durch Sicherheitstools schwer erkennbare Shell, die oft von Administratoren und Penetrationstestern verwendet wird, um sich mit einem Zielsystem zu verbinden und Systemkommandos auszuführen. Weevely bietet eine Kommandozeilen ähnliche Benutzeroberfläche und beinhaltet viele Funktionen, die es ermöglichen, das Zielsystem auf Schwachstellen zu untersuchen. Das Tool besteht aus zwei Komponenten - dem Client und dem Agent. Der Agent ist ein PHP-Skript, das auf dem Zielsystem eingeschleust wird. Die Kommandos, die der Agent ausführen soll, werden vom Client, der in Python implementiert wurde, über das Netzwerk gesendet. Damit Weevely funktioniert, muss das Zielsystem die Programmiersprache PHP installiert und für den Webserver konfiguriert haben [WEE17].

2.4. Schwachstellen-Scan

Bei einem Schwachstellen-Scan werden Werkzeuge eingesetzt, um einzelne Systeme oder auch ganze Netzwerke auf Schwachstellen zu untersuchen. Diese Werkzeuge sind meistens voll automatisiert und können daher nicht alle Schwachstellen aufspüren. Schwachstellen-Scanner bestehen oft aus einer Modulsammlung, bei der jedes Modul einen bestimmten Test ausführt. Die Anwender können somit automatische Tests zugeschnitten auf ihre Anforderungen erstellen.

Schwachstellen-Scans werden verwendet, um möglichst kostengünstig und in kurzer Zeit einen Überblick über mögliche Schwachstellen zu erhalten. Vor einem automatischen Test sollte jedoch eine gründliche Vorbereitung getroffen werden, da solche Tests oft große Mengen an Daten über das Netz senden und somit die Zielsysteme beeinträchtigen können. Wird ein Schwachstellen-Scan nicht überwacht und gegebenenfalls bei Problemen abgebrochen, können für Organisationen schwere Schäden entstehen.

2.5. Sicherheitsrisiken von Webservern

In der heutigen Zeit hat fast jedes Unternehmen und auch einige Privatpersonen eine eigene Webseite. Webseiten sind weltweit abrufbar und dienen als öffentlicher Auftritt, bei dem Kunden Informationen über das Unternehmen einholen können oder mit dem Unternehmen anhand von Kontaktformularen interagieren. Da Webserver weltweit erreichbar sind, können sie in der Regel leicht angriffen werden. Bei den meisten Webservern laufen neben der Webanwendung, die mit teils umfangreichen Funktionen ein großes Gefahrenpotential mitbringt, auch weitere Dienste wie ein Datenbankserver oder ein FTP-Server, die Schwachstellen beinhalten können. Da diese Dienste gewöhnlich nicht von einer Firewall geblockt werden, können sie für einen Einbruch in das System ausgenutzt werden. Dies macht den Webserver durch seine zahlreichen Angriffsvektoren zu einem leichten Ziel.

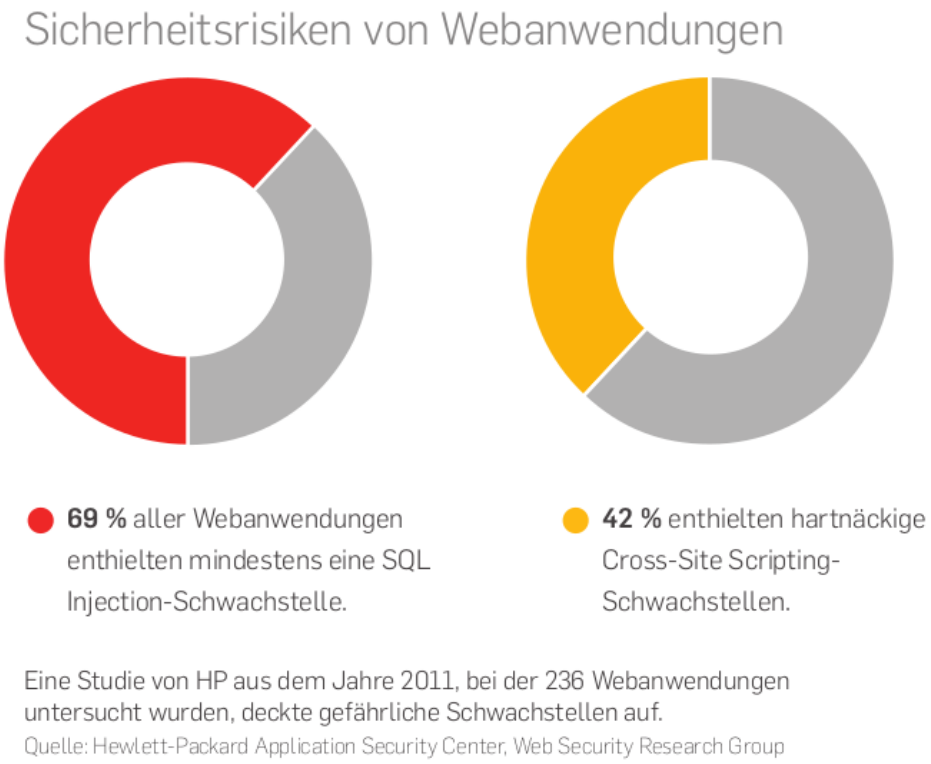


Abbildung 2.3.: Ergebnis einer Studie aus dem Jahr 2011 von HP zu den Risiken von Webservern [SOP12]

Nach einer Studie von Hewlett-Packard aus dem Jahr 2011 sind 80 Prozent aller Netzwerkattacken auf webbasierte Systeme abgezielt. Bei der Studie wurden 236 Webanwendungen auf Schwachstellen untersucht und viele gefährliche Schwachstellen gefunden. So enthielten 69 Prozent aller untersuchter Webseiten mindestens eine SQL-Injection Schwachstelle und bei 42 Prozent aller Webseiten waren hartnäckige XSS-Attacken möglich, wie in Abbildung 2.3 zu sehen ist [SOP12].

2.6. Schwachstellen

Eine einhundertprozentige Sicherheit ist in der IT nicht machbar, da jedes verwendete Programm Schwachstellen beinhalten kann, die bisher noch keiner gefunden hat oder die gefundene Schwachstelle nicht an den Hersteller weitergegeben wird. Besucht man einschlägige Webseiten wie die Exploit-DB[1] ist zu erkennen, dass fast täglich neue Schwachstellen in diversen Anwendugen gefunden werden. Diese Schwachstellen sind aber nur die, welche von Sicherheitsexperten veröffentlicht werden. Neben diesen einschlägigen Webseiten gibt es aber auch Schwarzmärkte, auf denen noch unbekannte Schwachstellen verkauft werden.

2.6.1. OWASP Top 10

OWASP ist ein Non-Profit Unternehmen, das sich auf die Sicherheit von Webanwendungen spezialisiert hat. Sie finanzieren sich hauptsächlich durch Spenden und Konferenzen. Das Ziel dieses Unternehmens ist es, die Sicherheit im World Wide Web zu verbessern, indem sie einige Projekte leiten, wie einen Leitfaden für Penetrationstests, den Zed Attack Proxy oder die OWASP Top 10, die für Sicherheitsbeauftragte frei zugänglich sind. Im Folgenden werden die Schwachstellen der OWASP Top 10, die in Tabelle 2.2 zusammengefasst sind, beschrieben. [OWA13]

A1	Injection
A2	Fehler in Authentifizierung und Session-Management
A3	Cross-Site Scripting (XSS)
A4	Unsichere direkte Objektreferenzen
A5	Sicherheitsrelevante Fehlkonfiguration
A6	Verlust der Vertraulichkeit sensibler Daten
A7	Fehlerhafte Autorisierung auf Anwendungsebene
A8	Cross-Site Request Forgery (CSRF)
A9	Nutzung von Komponenten mit bekannten Schwachstellen
A10	Ungeprüfte Um- und Weiterleitung

Tabelle 2.2.: OWASP Top 10 Sicherheitsbedrohungen

A1 - Injection

Eine Injection-Schwachstelle entsteht, wenn die Eingaben von einer nicht vertrauenswürdigen Quelle ungeprüft von der Anwendung weitergegeben und als Programmcode oder Kommandozeilenbefehl interpretiert werden. Injection-Schwachstellen gibt es in mehreren Varianten - Code Injection, OS Command Injection und SQL-Injection. Die SQL-Injection kommt bei Webanwendungen am häufigsten vor, indem ein Angreifer zum Beispiel in den URL-Parametern seinen eigenen SQL-Query mitgibt, der dann in der Datenbank ausgeführt wird. Eine SQL-Injection Schwachstelle ist sehr kritisch zu bewerten, da einem Angreifer ermöglicht wird, auf die Datenbank zu gelangen und die Inhalte zu lesen oder in manchen Fällen sogar zu verändern. Je nach Konfiguration des Servers kann ein Angreifer auch beliebige Dateien auf die Festplatte schreiben und dadurch diverse Systemkommandos ausführen. Um hierzu ein kleines Beispiel zu zeigen, kann ein PHP-Code mit einer SQL-Injection-Schwachstelle wie in Listing 2.2 aussehen.

[1] https://www.exploit-db.com

Listing 2.1: Ein verwundbarer PHP-Code mit einer SQL-Injection

```
<?php
echo "<h1>Meine kleine Homepage</h1>";
$page = $_GET['page'];
$db = new PDO('mysql:host=localhost;dbname=testdb', 'username', 'password');
$sql = "SELECT page_content FROM pages WHERE id=".$page;
$result = $db->query($sql);
$content = $result->fetchColumn(0);
echo $content;
?>
```

Angenommen die verwundbare Webseite verfügt über ein Anmeldeformular für Benutzer oder Administratoren, so kann ein Angreifer seinen eigenen SQL-Query in die Adressleiste eingeben und somit den Benutzernamen und das zugehörige Passwort aus der Datenbank auslesen, um unberechtigten Zugriff auf das jeweilige Benutzerkonto zu erlangen, wie in Abbildung 2.4 zu sehen ist.

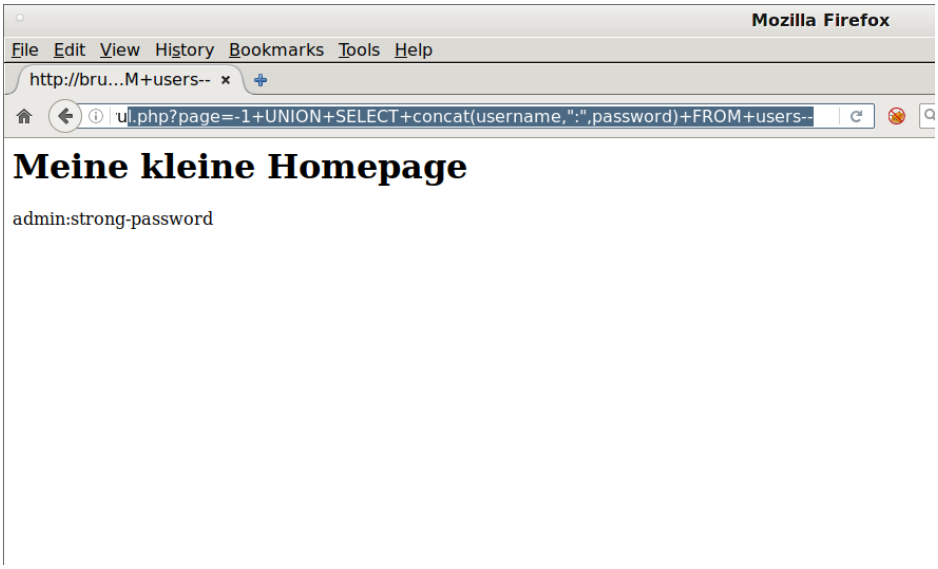


Abbildung 2.4.: Resultat einer SQL-Injection

Dieses Beispiel ist eine sehr einfache Variante der SQL-Injection. Es gibt je nach Anwendungsfall auch eine zeitbasierte SQL-Injection, bei der mittels eines Sleep-Befehls die Zeit gemessen werden kann, wie lange die Anwendung braucht, um zu laden. Eine weitere Variante ist die Blind-SQL-Injection, bei der ein Angreifer mittels True/False Anfragen Informationen aus der Datenbank extrahieren kann. Dies macht die Attacke zwar schwerer, jedoch gibt es automatisierte Tools, um schneller zu einem Ergebnis zu kommen. Ist es einem Angreifer nicht möglich, anhand der zuvor beschriebenen Attacken eine SQL-Injection auszunutzen, bleibt noch die Möglichkeit einer Error-Based-SQL-Injection. Bei dieser versucht der Angreifer Informationen aus der Datenbank anhand von Fehlermeldungen anzeigen zu lassen.

Das Vorgehen bei anderen Injection-Varianten ist ähnlich zu der in dem Beispiel beschriebenen Attacke. Der Unterschied hierbei ist lediglich, dass die Webanwendung die Eingabe des Benutzers, ohne diese zu prüfen, als Programmcode oder als Systemkommando ausführt.

A2 - Fehler in Authentifizierung und Session-Management

Viele Webanwendungen im Internet bieten ihren Nutzern die Möglichkeit, ein Benutzerkonto zu erstellen, um die persönlichen Daten zu speichern und sich bei einer erneuten Anmeldung an den jeweiligen Benutzer zu erinnern. Dies ist zum Beispiel der Fall bei Onlineshops oder auch sozialen Netzen. Damit die Webanwendung einen Benutzer während der Interaktion zuordnen kann und der Benutzer nicht bei jeder Aktion seine Zugangsdaten neu eingeben muss, sind Sessions nötig. Dies wird gewöhnlich mit Session-IDs realisiert. Sind diese

Session-IDs nicht ausreichend geschützt, so kann ein Angreifer an diese ID gelangen und somit die Session übernehmen.

Ein Beispiel für eine Schwäche im Session-Management, das jeder kennen sollte, ist, wenn ein Benutzer einen öffentlichen Computer verwendet und sich mit diesem auf eine Plattform wie Facebook einloggt und nach dem Verlassen des Computers vergisst, die Abmelde-Funktion zu verwenden. Wenn nun nach einiger Zeit ein anderer Benutzer diesen öffentlichen Computer verwendet und die Session des vorigen Benutzers noch aktiv ist, würde dieser bei dem Besuch von Facebook direkt auf das Profil des vorigen Benutzers gelangen. Hierzu gibt es noch eine Vielzahl anderer Varianten, wie ein Angreifer die Session eines Benutzers übernehmen könnte. Ein denkbares Szenario wäre das Erlangen von Cookies durch eine Cross-Site Scripting Schwachstelle in der Webanwendung.

Ein weiteres Risiko neben dem Session-Management ist ein Fehler in der Authentifizierung. Ein Szenario könnte sein, dass ein Angreifer in der Webanwendung eine SQL-Injection findet und diese ausnutzt, um die Benutzer samt Passwort auszulesen. Sind die Passwörter in der Datenbank im Klartext gespeichert, hat der Angreifer Zugriff auf alle Benutzerkonten und kann diese übernehmen. Aus diesem Grund sollten Passwörter immer in Form eines starken Hash gespeichert werden.

A3 - Cross-Site Scripting (XSS)

Cross-Site Scripting ist eine sehr häufig auftretende Schwachstelle in Webanwendungen, die von vielen Administratoren noch vor einigen Jahren nicht sehr ernst genommen wurde, da sie dachten, dass von der Schwachstelle keine große Gefahr ausgehe [SWA06][AOD10]. Jedoch ist eine XSS Schwachstelle eine große Gefahr für die Nutzer einer Webseite, da es dadurch möglich ist, die Cookies des Benutzers auszulesen und gegebenenfalls dessen Session zu übernehmen. Ein weiteres Szenario könnte sein, den Benutzer durch gefälschte Links auf eine gefälschte Webseite weiterzuleiten, um dessen Zugangsdaten zu erlangen. XSS Schwachstellen treten immer dann auf, wenn eine Webanwendung Skriptcode ungeprüft in die Webseite übernimmt. Von Cross-Site Scripting gibt es drei verschiedene Typen - persistent, reflektiert und DOM-basiert. Persistentes XSS zeichnet sich dadurch aus, dass der Skriptcode gespeichert und bei jedem Benutzer ausgegeben und im Browser ausgeführt wird. Reflektierte XSS sind hingegen das Gegenteil einer persistenten XSS und werden somit nur temporär über Formulare oder URLs in die Webseite eingeschleust. Dom-basierte XSS oder auch lokale XSS sind hingegen der anderen zwei Varianten seltener und sind auf Fehler im client-seitigen Programmcode zurückzuführen. Nachfolgend wird anhand eines Beispiels gezeigt, wie ein reflektiertes Cross-Site Scripting aussehen kann.

Listing 2.2: Ein verwundbarer Programmcode mit einer reflektierten XSS Schwachstelle

```
<html>
<head>
<title> XSS Beispiel </title>
</head>
<body>
<h1>Login</h1>
<form method="post" action="">
Username: <input type="text" name="username" value="<?php echo $_GET['username']; ?>"><br>
Password: <input type="password" name="password" value=""><br>
<input type="submit" name="submit">
</form>
</body>
</html>
```

In diesem Beispiel wird einem Login-Formular der Benutzername mit der HTTP-GET Methode übergeben und ohne zu prüfen in das HTML input-Feld eingetragen. Dies kann ein Angreifer nun ausnutzen, indem er der Webseite durch die URL JavaScript Programmcode übergibt, der dann im Browser des Benutzers ausgeführt wird. Abbildung 2.5 zeigt wie der Progammcode im Browser ausgeführt wird.

In diesem Beispiel wird nur ein Popup-Fenster im Browser des Benutzers ausgegeben, jedoch kann ein Angreifer der URL auch Schadcode anhängen oder die Webseite um weitere Funktionen erweitern.

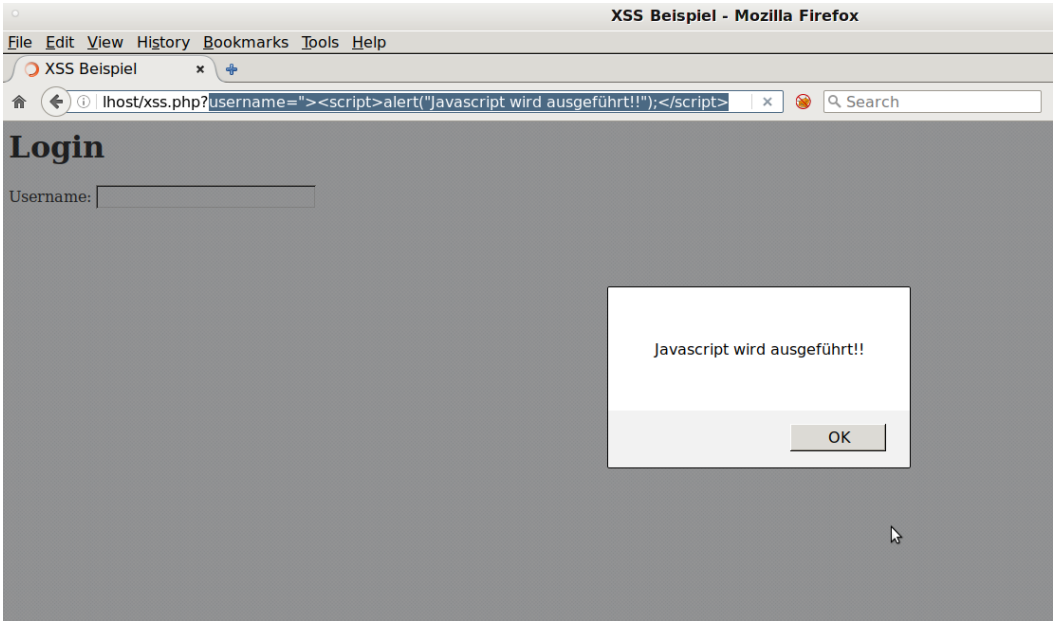


Abbildung 2.5.: Resultat einer XSS Schwachstelle

A4 - Unsichere direkte Objektreferenz

Eine Objektreferenz ist bei Webanwendungen oft ein Schlüssel, Benutzername oder andere eindeutige Werte, über die auf Ressourcen zugegriffen werden kann. Oft handelt es sich bei der Ressource um eine Datenbank, von der Informationen zu dem jeweiligen Benutzer abgefragt werden sollen.

Dies wird zu einem Problem, wenn die Anwendung nicht prüft, welcher Benutzer auf bestimmte Informationen zugreifen darf und einen Datensatz eines anderen Benutzers zurückgibt, auf den eigentlich kein Zugriff erfolgen sollte. So kann die Objektreferenz zum Beispiel die in der Datenbank hinterlegte eindeutige ID eines Benutzers sein. Ein Angreifer kann dies ausnutzen, indem er die ID seines Benutzerprofils verändert und somit auch Informationen zu anderen Benutzern abrufen kann, für die er keine Berechtigung besitzt.

Um eine Schwachstelle wie die unsichere direkte Objektreferenz zu verhindern, ist es nötig, eine Zugriffskontrolle zu implementieren, die entscheidet, welcher Benutzer auf welche Ressource zugreifen darf. Eine weitere Möglichkeit, solche Angriffe zu verhindern, besteht darin, die indirekte Objektreferenz zum Abfragen der Daten zu nutzen. Die indirekte Objektreferenz unterscheidet sich darin, dass die Referenz ein zufällig und möglichst lang gewählter Wert ist, der nur für einen bestimmten Zeitraum gültig ist. Dieser begrenzte Zeitraum könnte zum Beispiel die Dauer einer Session oder ein statisch festgelegter Wert sein. Da die Objektreferenz möglichst lang sein soll und auch nur eine bestimmte Zeit existiert, ist es fast unmöglich, anhand von Durchprobieren eine gültige Objektreferenz eines anderen Benutzers zu finden.

A5 - Sicherheitsrelevante Fehlkonfiguration

Platz fünf der OWASP Top 10 nimmt die sicherheitsrelevante Fehlkonfiguration ein, bei der es eine Vielzahl von möglichen Szenarien gibt. Zu diesen Fehlkonfigurationen zählen unter anderem lesbare Log-Dateien, die sensible Daten beinhalten, Fehlermeldungen von Server und Webanwendungen oder auch Serverbackups, die vom Internet aus abrufbar sind. Ein Angreifer kann so viele Informationen erlangen, die ihm den Weg, weitere Schwachstellen zu finden, erleichtern.

Ein Beispiel für eine sicherheitsrelevante Fehlkonfiguration kann sein, wenn auf einem Webserver das Directory Listing aktiv ist. Die Directory Listing Funktion bedeutet, dass ein Webserver den Inhalt eines Ordners anzeigt, wenn in diesem keine Index-Datei vorhanden ist. In diesem Fall könnte ein Angreifer verschiedene Verzeichnisse

auf dem Server durchprobieren, die seiner Meinung nach sicherheitskritische Dateien enthalten können. Ist in diesen Verzeichnissen keine Index-Datei vorhanden, wird der Webserver den Inhalt dieses Ordners anzeigen.

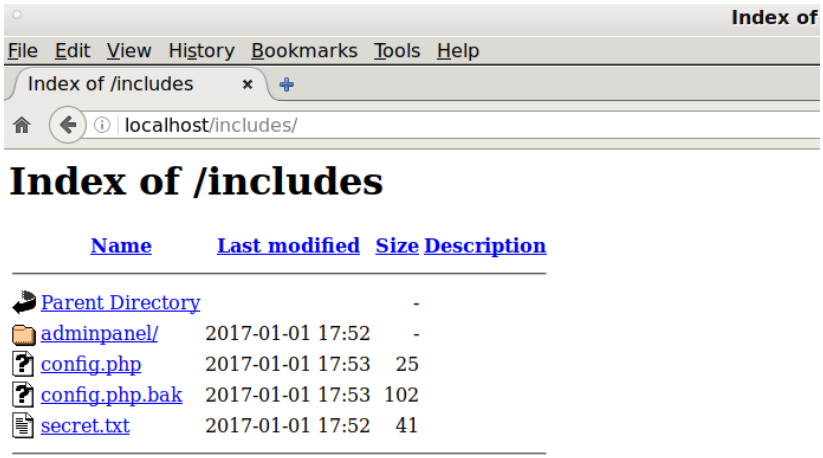


Abbildung 2.6.: Fehlkonfiguriertes Directory Listing

In Abbildung 2.6 ist zu erkennen, dass der Angreifer hier viel Information bekommt, die er nicht haben sollte. Einerseits sieht er, dass sich in diesem Verzeichnis eine Administrationsoberfläche befindet, die er als nächstes Ziel aussuchen kann. Zum anderen aber befindet sich in diesem Ordner eine Konfigurationsdatei, die möglicherweise Passwörter zu der Datenbank enthalten kann. Da in diesem Beispiel ein Backup der Konfigurationsdatei gemacht wurde, könnte der Angreifer diese problemlos herunterladen.

A6 - Verlust der Vertraulichkeit sensibler Daten

Der Verlust der Vertraulichkeit sensibler Daten kommt immer dann vor, wenn sensible Daten nicht verschlüsselt gespeichert werden oder mit einem als nicht mehr sicher geltenden Algorithmus verschlüsselt wurden. Jedoch gibt es auch andere Möglichkeiten, die Vertraulichkeit von sensiblen Daten zu verlieren. Speichert zum Beispiel die Webanwendung Daten verschlüsselt in einer Datenbank und entschlüsselt diese Daten auch wieder bei einem Abruf, könnte ein Angreifer, der in der Webanwendung eine SQL-Injection gefunden hat, trotz der Verschlüsselung auf die sensiblen Daten zugreifen.

Ein weiteres Szenario, das öfter vorkommt, ist, wenn eine Webanwendung Dateien zum Herunterladen anbietet, jedoch die Eingabe des Benutzers nicht genügend überprüft. In Listing 2.3 wird gezeigt, wie eine solche verwundbare Webanwendung aussehen kann.

Listing 2.3: Ein Beispiel für eine Arbitrary File Download Vulnerability

```
<?php
$file = $_GET['file'];

header('Content-Description: File Transfer');
header('Cache-Control: public');
header("Content-Transfer-Encoding: binary");
header('Content-Disposition: attachment; filename='. basename($file));
header('Content-Length: '.filesize($file));

ob_clean();
flush();
readfile($file);
?>
```

Dieses Download-Skript übernimmt den Inhalt der HTTP-GET Methode in die Variable *$file* ohne eine entsprechende Überprüfung, welche Datei heruntergeladen wird. Somit kann ein Angreifer jede beliebige Datei herunterladen, auf die der Benutzer mit dem der Webserver ausgeführt wird, Leseberechtigung besitzt. Beispielsweise kann der Angreifer die URL so manipulieren, wie es in Listing 2.4 gezeigt wird und somit die passwd Datei herunterladen, in der alle verfügbaren Benutzer des Systems eingetragen sind.

Listing 2.4: Beispiel eines Angriffes der Arbitrary File Download Vulnerability

```
http://victim.org/filedownload.php?file=../../../../../../../../../../etc/passwd
```

A7 - Fehlerhafte Autorisierung auf Anwendungsebene

Bei vielen Webanwendungen gibt es Benutzer mit verschiedenen Rollen, wie Administratoren, Mitarbeiter und normale Benutzer. Eine fehlerhafte Autorisierung auf Anwendungsbasis tritt dann auf, wenn die Webanwendung ohne Session-Management arbeitet und sich somit nur auf Eingabeparameter des jeweiligen Benutzers verlässt. Ähnlich wie bei dem schon zuvor beschriebenen Sicherheitsrisiko - unsichere direkte Objektreferenz - kann ein Angreifer die Parameter mit verschiedenen Werten durchprobieren und sehen, wie die Webanwendung darauf reagiert. Wenn die Webanwendung auf eine solche Schwachstelle anfällig ist, kann der Angreifer seine Rechte erweitern und somit potentiell auch Administratorrechte erlangen.

Ein Szenario, wie eine solche Schwachstelle aussehen kann, ist, wenn eine Webanwendung die verschiedenen Rollen der Benutzer ausschließlich über ein Cookie identifiziert, das bei der Anmeldung clientseitig gespeichert und bei jedem Seitenaufruf neu an den Server übermittelt wird. Gelingt es einem Angreifer nun, das Cookie mittels eines XSS-Angriffs zu entwenden oder den Inhalt des Cookies zu erraten, kann er auf der Webseite Aktionen ausführen, die normalerweise nur einem höher privilegierten Benutzer zur Verfügung stehen.

A8 - Cross-Site Request Forgery

Bei einer Cross-Site Request Forgery Attacke (CSRF) wird ein Benutzer dazu gebracht, beispielsweise einen Link in einer Phishing-Mail[1] oder auf einer vom Angreifer kontrollierten Website zu verwenden. Der Angreifer versucht, durch diesen gefälschten Link gewisse Aktionen auf der Zielwebseite mit den Rechten des Benutzers auszuführen. Diese Aktionen können unter anderem das Ändern der Email des Benutzers oder das Löschen gewisser Daten sein. Da der Benutzer selbst den Link klickt, wird durch den Browser alles Nötige zur Identifizierung des Benutzers an den betreffenden Webserver gesendet, wie Session-Cookies und die IP des Benutzers. Daher funktioniert dieser Angriff nur dann, wenn der Benutzer zu diesem Zeitpunkt auf der Webseite angemeldet ist und diese keine CSRF-Gegenmaßnahmen implementiert hat. Neben der klassischen CSRF gibt es auch die stored CSRF, bei der die Attacke durch die Webseite selbst ausgeführt wird. Dies wird erreicht, indem der Angreifer eine weitere Schwachstelle in der Webanwendung findet, um diese zu manipulieren.

Soll die Webanwendung gegen CSRF geschützt werden, verwenden viele einen CSRF-Token. Dabei handelt es sich um einen vom Server zufällig generierten Wert, der bei jedem Aufruf mittels HTTP-POST oder HTTP-GET mitgegeben wird. Da der Angreifer diesen Wert nicht wissen kann, wird es für ihn fast unmöglich, Erfolg zu haben. Damit der Angriff auch durch die Brute-Force-Methode soweit erschwert wird, dass er unpraktikabel ist, sollte der CSRF-Token eine ausreichende Länge haben.

A9 - Nutzung von Komponenten mit bekannten Schwachstellen

Platz neun der OWASP Top Ten nimmt die Nutzung von Komponenten mit bekannten Schwachstellen ein. Dieses Risiko tritt auf, da viele Webseitenbetreiber auf fertige Applikationen wie Programm-Bibliotheken oder Content Management Systeme (CMS) setzen, mit deren Hilfe sie ihre Webseiten aufbauen. In diesen Komponenten werden von Sicherheitsexperten oft Schwachstellen entdeckt und veröffentlicht, nachdem der Hersteller darüber informiert wurde. Es ist oft der Fall, dass Webseitenbetreiber diese Komponenten nicht auf dem aktuellen Stand halten und somit ihre Webseite für Angriffe anfällig ist. Die bekannten Schwachstellen in

[1] Phishing Mails sind Emails, die aussehen, als würden sie von einer vertrauenswürdigen Webseite kommen und dadurch den Benutzer dazu verleiten, einen Link in dieser Email zu klicken.

weit verbreiteten Webanwendungen sind für Angreifer ein attraktives Ziel, da sie auf sehr vielen Webservern betrieben werden und somit die Ausnutzung einer Schwachstelle auf einer Vielzahl von Servern möglich ist.

A10 - Ungeprüfte Um- und Weiterleitung

Einige Webapplikationen ermöglichen dem Benutzer durch eine automatische Weiterleitung auf andere Inhalte oder Webseiten geleitet zu werden. Diese Weiterleitungen werden oft durch den HTTP-GET-Parameter realisiert. Wenn eine Webanwendung nicht überprüft, wohin der Benutzer weitergeleitet wird, kann ein Angreifer den GET-Parameter manipulieren und somit einen Benutzer auf eine Webseite umleiten, die unter der Kontrolle des Angreifers ist. In diesem Fall kann der Angreifer den Benutzer auf eine Phishing-Webseite umleiten und somit sensible Daten abgreifen. Eine dem Nutzer bekannte Webseite wird oft Vertrauen entgegengebracht, wodurch eine Attacke dieser Art oft dazu führt, dass ein Phishing-Angriff für den Angreifer erfolgreich ist. Aus diesem Grund sollte eine ungeprüfte Weiterleitung auf keinen Fall möglich sein.

Ein weiteres Szenario könnte sein, wenn die Webseite den Benutzer auf andere interne Inhalte umleitet und nicht prüft, auf welche Unterseiten weitergeleitet wird. In diesem Fall könnte ein Angreifer den Parameter so manipulieren, dass er auf eine Seite umgeleitet wird, auf die er normalerweise keinen Zugriff haben sollte, wie zum Beispiel ein Admin-Menü.

2.6.2. Weitere zu beachtende Risiken

Die OWASP Top 10 zeigt die zehn wichtigsten Risiken für Webentwickler. Jedoch gibt es noch eine Reihe weiterer Risiken, die bei der Entwicklung und dem Betrieb von Webanwendungen beachtet werden sollten. Im Folgenden werden weitere Risiken erläutert, die zu beachten sind.

Remote File Upload

Viele Webanwendungen bieten die Möglichkeit, Dateien, wie Bilder oder andere Anhänge, auf den Server hochzuladen. Wird von der Webseite der Dateityp nicht ordungsgemäß geprüft, können beliebige Dateien hochgeladen werden. Dies kann ein Angreifer ausnutzen, um beispielsweise eine eigene PHP-Datei hochzuladen und diese auf dem Webserver auszuführen.

Clickjacking

Bei einer Clickjacking-Attacke versucht der Angreifer durch ein HTML-Frame eine legitime Webseite einzubinden, die zum Beispiel ein Loginformular enthält. Somit kann der Angreifer Buttons oder auch Inputfelder überlagern und seine eigenen meist bösartigen Aktionen ausführen, sobald der Nutzer auf den veränderten Inhalt klickt. Diese Attacke ist ähnlich der CSRF und kann für den Nutzer selbst sehr gefährlich sein. Solch eine Methode kann auch verwendet werden, wenn der Betreiber der Webseite zwar einen Schutz gegen CSRF mittels Tokens implementiert hat, aber das Einbinden der Webseite über ein HTML-Frame nicht verhindert. In diesem Fall könnte man den Schutz gegen CSRF mit einer Clickjacking-Attacke umgehen. Um zu verdeutlichen, wie eine Clickjacking-Attacke funktioniert, wird im Folgenden ein Beispiel aufgeführt [CLK13].

In diesem Beispiel beinhaltet die anzugreifende Webseite ein Formular, das mit einem CSRF-Token ausgerüstet ist (vgl. Listing 2.5). Aufgrund des Schutzes gegen eine CSRF-Attacke kann diese Webseite mit dieser Methode nicht angegriffen werden. Es muss also eine Möglichkeit gefunden werden, um den Nutzer zu verleiten, das Formular abzusenden.

Listing 2.5: Durch Clickjacking verwundbare Webseite

```
<html>
<head>
<title>Clickjacking Beispiel</title>
</head>
<body>
<form action="" method="post">
<input type="text" name="amount" value="300">
<input type="hidden" name="token" value="[csrf-token]">
<input type="submit" name="submit" value="Bezahlen">
</form>
</body>
</html>
```

Im nächsten Schritt erstellt der Angreifer eine Webseite, die mittels eines Iframes die anzugreifende Webseite einbindet, wie Listing 2.6 zeigt. Damit der Benutzer das Iframe mit der originalen Webseite nicht sieht, wird es unsichtbar gemacht, wie in Listing 2.7 zu sehen ist.

Listing 2.6: Webseite des Angreifers

```
<html>
<head>
<title>Clickjacking Angriff</title>
</head>
<body>
<button id="click">Klick mich und dich erwarten tolle Preise</button>
<iframe src="http://opferwebseite.de/bezahlungsformular.html" id="opfer-website"></iframe>
</body>
</html>
```

Jetzt muss der Angreifer mittels einer CSS-Datei den erzeugten Button so positionieren, dass er sich an der gleichen Position wie bei der originalen Webseite befindet. Das unsichtbare Iframe liegt jedoch über dem erzeugten Button, ist aber für den Benutzer unsichtbar.

Listing 2.7: CSS Datei

```
#click {
    position:absolute;
    top:0px;
    left:0px;
    color: green;
}

#opfer-website {
    width: 100%;
    height: 100%;
    opacity: 0.0;
}
```

Will der Benutzer jetzt den angekündigten Preis gewinnen und klickt den Button, wird im Hintergrund aber nicht der vom Angreifer erzeugte Button ausgeführt, sondern das Formular der originalen Webseite.

2.6.3. Common Vulnerability Scoring System

Eine Schwierigkeit in der Informationssicherheit ist es, Schwachstellen richtig einschätzen und bewerten zu können. Aus diesem Grund wurde im Jahr 2005 vom National Infrastructure Advisory Council ein einheitliches Verfahren geschaffen, um Schwachstellen miteinander vergleichen zu können, egal aus welcher Quelle sie kommen. Inzwischen wird dieses Verfahren aber von dem Forum of Incident Response and Security Teams (FIRST) weiterentwickelt. Dieses Verfahren nennt sich Common Vulnerability Scoring System oder kurz CVSS.

CVSS verwendet einen Algorithmus, der Werte für den Base Score, Temporal Score und Environmental Score ausgibt, um das Bedrohungspotential einer Schwachstelle ermitteln zu können. Der CVSS-Score ist ein Wert, welcher in fünf Wertungsstufen unterteilt ist, wie Tabelle 2.3 zeigt. Der Wert des CVSS-Score ist numerisch und kann eine Zahl zwischen 0.0 und 10.0 annehmen, wobei nach der aktuellen Version 3.0 der Wert 0.0 für eine nicht bewertete Schwachstelle steht und der Wert 10.0 eine besonders kritische Schwachstelle angibt [CVS15].

Score	Wertung
0.0	None
0.1 - 3.9	Low
4.0 - 6.9	Medium
7.0 - 8.9	High
9.0 - 10.0	Critical

Tabelle 2.3.: Werte und zugehörige Einstufung eines CVSS-Scores

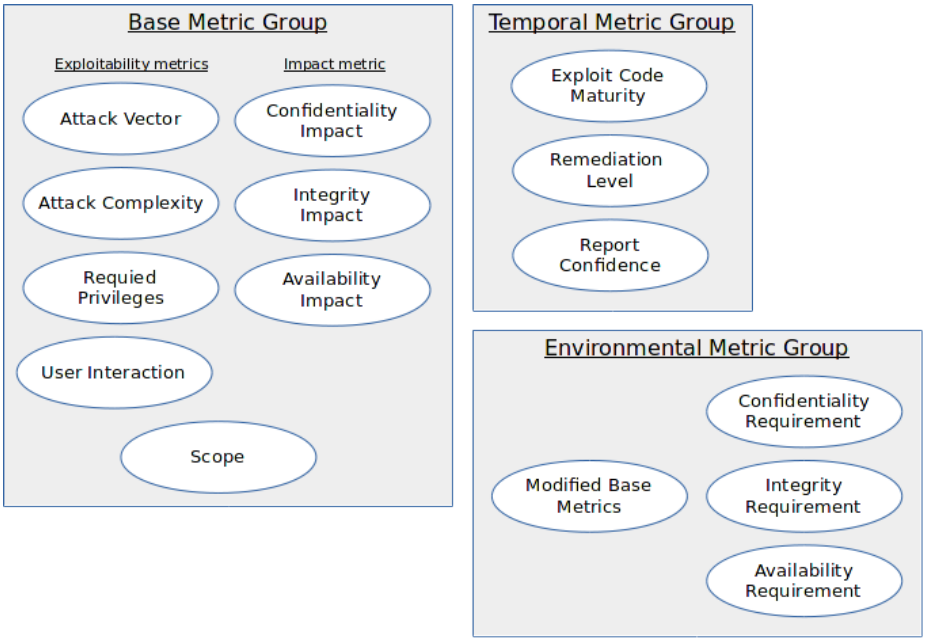


Abbildung 2.7.: CVSS v3 Metric Groups

Der CVSS-Score besteht aus 3 Hauptgruppen - der Base Metric Group, Temporal Metric Group und Environmental Metric Group - mit jeweils einer Reihe an Metriken, wie es in Abbildung 2.7 dargestellt ist. Die Base Metric Group stellt die grundlegenden Eigenschaften der Verwundbarkeit dar und ist aufgeteilt in die Bereiche Exploitability Metrics und Impact Metrics. Bei der Temporal Metric Group handelt es sich um die zeitabhängigen Eigenschaften der Verwundbarkeit, wie zum Beispiel, ob die Schwachstelle bereits bestätigt wurde. Die Environmental Metric Group behandelt die szenarienspezifischen Eigenschaften der Verwundbarkeit, wie die Auswirkung auf das Unternehmen [CVS15]. Im Folgenden wird die Bedeutung der einzelnen Metriken kurz erklärt.

Base Metric Group:

- Attack Vector: Kann die Schwachstelle nur lokal ausgenutzt werden oder auch über das Netzwerk?
- Attack Complexity: Wie schwer ist es und wie viel Detailwissen wird benötigt, um die Schwachstelle auszunutzen?
- Required Privileges: Muss sich ein Angreifer authentifizieren, um die Schwachstelle auszunutzen?

- User Interaction: Dies gibt an, ob der Angriff nur mit dem Mitwirken eines Benutzers durchgeführt werden kann. Das wäre der Fall, wenn der Benutzer auf einen Link in einer Phishing Mail klicken muss, damit der Angriff erfolgreich ist.
- Confidentiality Impact: Welche Auswirkung hat ein Ausnutzen der Schwachstelle für die Vertraulichkeit?
- Integrity Impact: Welche Auswirkung hat die Schwachstelle auf die Integrität?
- Availability Impact: Welche Auswirkung hat die Schwachstelle auf die Verfügbarkeit?
- Scope: Im Scope wird die Komponente mit der Schwachstelle sowie die Komponente auf der ein Schadcode ausgeführt wird definiert.

Temporal Metric Group:

- Exploit Code Maturity: Ist ein Exploit, Proof of Concept, etc. im Umlauf?
- Remediation Level: Ist ein offizieller Bugfix oder Workaround bekannt?
- Report Confidence: Wie viel Information über die Schwachstelle ist öffentlich bekannt?

Environmental Metric:

- Modified Base Metrics: Misst das Potential für den Schaden an physikalischen Einheiten wie das Equipment.
- Confidentiality Requirement: Dieser Wert kann an die jeweiligen Anforderungen angepasst werden.
- Integrity Requirement: Dieser Wert kann an die jeweiligen Anforderungen angepasst werden.
- Availability Requirement: Dieser Wert kann an die jeweiligen Anforderungen angepasst werden.

Damit die CVSS Scores berechnet werden können, bedarf es einer Reihe an Formeln, die im Folgenden erklärt werden sollen. Der Base Score ist eine Funktion aus dem Exploitability- und dem Impact Score. Der Base Score ist wie folgt definiert [CVS15]:

$$f(Impact, Exploitability) \begin{cases} 0, & \text{if Impact sub score } <= 0 \\ \lceil min((Impact + Exploitability), 10) \rceil, & \text{if Scope Unchanged} \\ \lceil min(1.08 * (Impact + Exploitability), 10) \rceil, & \text{if Scope Changed} \end{cases}$$

dabei ist der Impact sub score (ISC) definiert als:

$$\text{Scope unchanged } 6.42 * ISC_{Base}$$

$$\text{Scope changed } 7.52 * [ISC_{Base} - 0.029] - 3.25 * [ISC_{Base} - 0.02]^{15}$$

mit:

$$ISC_{Base} = 1 - [(1 - Impact_{Conf}) * (1 - Impact_{Integ}) * (1 - Impact_{Avail})]$$

Der Exploitability sub score ist definiert als:

$$ExploitabilitySubScore = 8.22 * AttackVector * AttackComplexity * PrivilegeRequired * UserInteraction$$

Damit der Temporal Score berechnet werden kann, wird eine Funktion verwendet, die wie folgt definiert ist:

$$TemporalScore = \lceil BaseScore * ExploitCodeMaturity * RemediationLevel * ReportConfidence \rceil$$

2.6.4. Common Vulnerability Exposures (CVE)

Common Vulnerability and Exposure (CVE) ist ein Verzeichnis von öffentlich bekannten Schwachstellen in Software und Hardware. Der Name besteht aus ”vulnerability”, also die Verwundbarkeit von Anwendungen und ”exposure”, das durch Fehlkonfiguration oder Fehler in Software einem Angreifer Informationen bietet, die ihm das Eindringen in ein Computersystem erleichtern. Ein CVE-Eintrag besteht aus einer eindeutigen Nummer, einem Schwachstellentyp und einer kurzen Beschreibung. Die Identifikationsnummer besteht aus der Jahreszahl, in der die Schwachstelle gemeldet wurde und einer fortlaufenden Nummer. Der Schwachstellentyp gibt an, um welche Art von Schwachstelle es sich handelt, wie zum Beispiel eine XSS-Schwachstelle. Früher wurden sie unterteilt in Candidates und Entries. Dabei wurde eine Schwachstelle erst als Candidate ausgewiesen und später nach einigen Votes als Entry übernommen. [CVE17]

CVE wird gepflegt von MITRE - eine Non-Profit-Organisation, die Forschungsinstitute betreibt - und weiteren CVE-Numbering-Authorities (CNA), meist große Software- und Hardwarekonzerne wie Microsoft oder Oracle. CVE bietet den Vorteil, dass sich Unternehmen und Institute über bekannte Schwachstellen austauschen und diese mit einer eindeutigen CVE-Nummer identifiziert werden können.

3. Anforderungsanalyse

Damit die Anforderungen des im Rahmen dieser Arbeit zu entwickelnden Service herausgearbeitet werden können, muss zunächst die Ausgangssituation geklärt werden. Dazu wird zuerst erläutert, was ein Hochschulrechenzentrum ist und wo der Zuständigkeitsbereich liegt, um im Folgenden genauer auf das LRZ und dessen Dienstleistungen in Bezug auf das Webhosting und den Infrastructure as a Service (IaaS) einzugehen. Im Anschluss wird diese Arbeit von einem klassischen Schwachstellenscan abgegrenzt und ein kurzer Einblick in die Ergebnisse der Umfrage gegeben, die im Rahmen dieser Arbeit durchgeführt wurde. Im weiteren Verlauf dieses Kapitels werden die Anforderungen an einen Service für Penetrationstests basierend auf der Umfrage hergeleitet und beschrieben.

3.1. Hochschulrechenzentren

Hochschulrechenzentren sind Zentraleinrichtungen, die als standortbezogener IT-Dienstleister für Hochschulen und Universitäten fungieren. Der Zuständigkeitsbereich eines Hochschulrechenzentrums ist ausschließlich für die wissenschaftlichen Einrichtungen, ihre Mitarbeiter, Studierenden und ihre Partner. Die Aufgaben für Einrichtungen dieser Art beginnen mit der Beschaffung von Hard- und Software und erstrecken sich von der Planung und dem Betrieb der IT-Infrastruktur bis hin zu Diensten, wie das Anbieten von Kommunikationsdiensten und Serversystemen. Des Weiteren fällt auch die Organisation und der Betrieb von externen Datenanschlüssen, wie an das Deutsche Forschungsnetz (DFN), unter den Aufgabenbereich eines Hochschulrechenzentrums.

3.2. Ausgangssituation am LRZ

3.2.1. Leibniz-Rechenzentrum

Das Leibniz-Rechenzentrum (LRZ) ist das Rechenzentrum der Bayerischen Akademie der Wissenschaft und somit der IT-Dienstleister der Hochschulen, Universitäten und einigen weiteren wissenschaftlichen Einrichtungen im Großraum München. Es wurde im Jahr 1962 gegründet und ist seither zu einem der größten wissenschaftlichen Rechenzentren Europas gewachsen [LRZ].

Das LRZ betreibt neben dem Rechenzentrum auch das Münchener Wissenschaftsnetz (MWN), an das die Ludwig-Maximilians-Universität, die Technische Universität, diverse Münchener Hochschulen und das Studentenwerk München mit ihren Wohnheimen angebunden sind. Des Weiteren wurden kürzlich auch die Münchener Museen an das MWN angebunden [LRZ15]. Die Weitläufigkeit dieses Netzes ist in Abbildung 3.1 dargestellt.

Derzeit sind über 200.000 Endgeräte, wie Arbeitsrechner für Studenten und Mitarbeiter, an das MWN angebunden [LRZ12]. Für die Anbindung an andere Netze wurde ein Backbone aus zwei Glasfaserleitungen von der Deutschen Telekom AG und von M-net langfristig angemietet. Für die Kunden bietet das LRZ neben der Anbindung an das MWN auch Dienste wie Webhosting, IaaS, virtuelle Maschinen und Mailserver an.

3.2.2. Managed Webhosting

Das LRZ bietet seinen Kunden zwei Produkte des betreuten Webhosting an. Das erste Produkt ist für eine persönliche Homepage, bei der aber nur statische Inhalte erlaubt sind und somit Webanwendungen auf Basis von PHP nicht eingesetzt werden können. Der Benutzer bekommt bei diesem Angebot einen Speicher von zwei

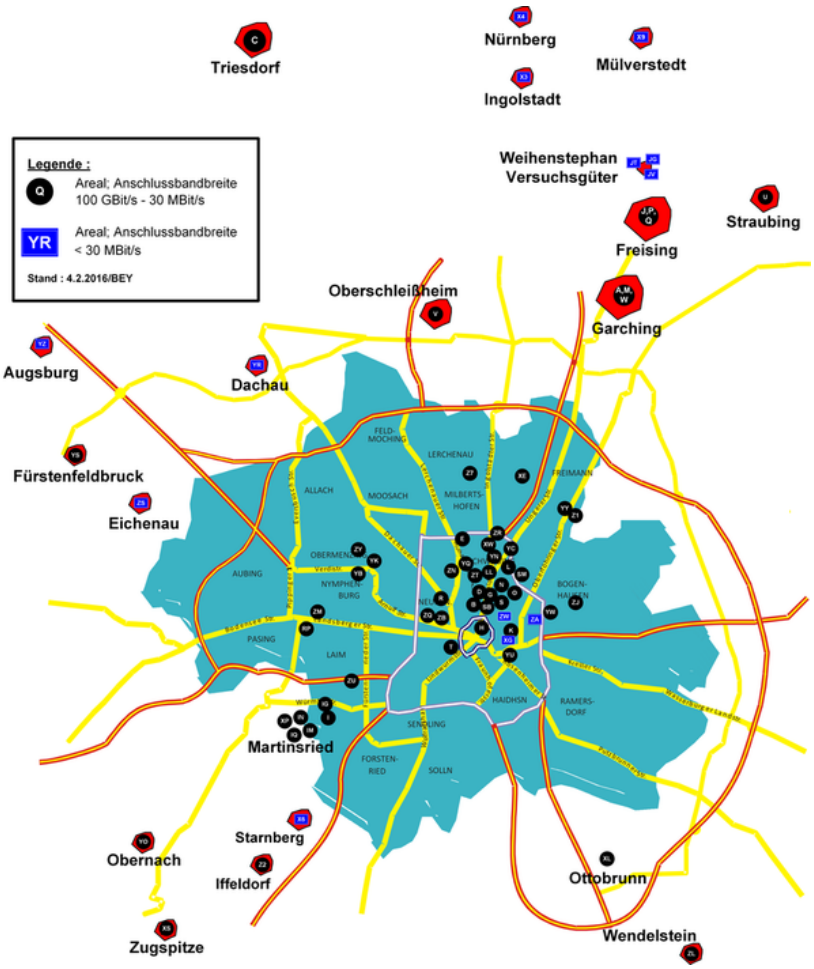


Abbildung 3.1.: Die Kartenansicht des MWN (nicht maßstabsgerecht) [LRZ15]

Gigabyte und eine Subdomain mit einem frei wählbaren Wunschnamen. Um gegebenenfalls Datenverlusten entgegenzuwirken, wird für den Nutzer täglich ein automatisches Backup seiner Daten vorgenommen. Auf die Backupdaten hat der Nutzer jedoch keinen direkten Zugriff und muss bei Fällen von Datenverlust den Service-Desk des LRZ kontaktieren.

Das zweite Produkt, welches durch das LRZ angeboten wird, ist ein betreutes Hosting von Webanwendungen für Lehrstühle, Institute und öffentliche Einrichtungen. Bei diesem Produkt bekommt der Kunde 10 Gigabyte Speicherplatz und kann die gewünschte Domain selbst wählen. Wenn der Kunde für seine Webanwendung mehr als den angebotenen Speicherplatz benötigt, kann dies beim LRZ beantragt werden. Bei diesem Produkt, im Gegensatz zu der oben beschriebenen persönlichen Homepage, wird es dem Kunden ermöglicht, auch Webanwendungen auf Basis von PHP zu verwenden. Derzeit bietet das LRZ ihren Kunden eine eigene Typo3 Instanz mit teiladministrativem Zugang an. Es können jedoch auch andere CMS verwendet werden [LRZ16a]. Das managed Webhosting kann ganz an die Anforderungen des Kunden angepasst werden. Somit ist es auch möglich, für die Webanwendung eine Datenbank einzurichten. Der Einsatz eigener Skripte oder das Installieren von Erweiterungen ist jedoch nicht möglich. Dieses Produkt beinhaltet, wie auch das schon oben beschriebene, ein tägliches Backup, um Datenverlust vorzubeugen. Der Kunde erhält hier Zugriff auf das Backup seiner Daten oder der Datenbank. Für die Einrichtung der Webserver, die Pflege des Betriebssystems sowie das Installieren von Sicherheitspatches muss sich der Kunde nicht selbst kümmern, da dies vom LRZ übernommen wird. Der Kunde muss sich lediglich um die Inhalte und die Struktur der Webseite kümmern sowie redaktionelle Rollen, wie beispielsweise Autoren, vergeben.

Da Sicherheitsrisiken in statischen Webseiten kaum auftreten, liegt der Fokus dieser Arbeit auf Webanwendungen mit dynamischen Inhalten. Das Webhosting für statische Webseiten wird demnach nicht weiter betrachtet.

3.2.3. Virtuelle Maschinen

Für Kunden, deren Projekte nicht in das Webhosting-Angebot des LRZ passen, da sie zum Beispiel eigene Skripte einsetzen, die nicht in PHP geschrieben sind, werden virtuelle Maschinen angeboten. Diese virtualisierten Serverinstanzen werden vom LRZ auf eigenen Servern betrieben. Das Angebot beinhaltet neben dem Betrieb der virtuellen Maschine die Bereitstellung eines hochverfügbaren Festplattenspeichers, einer weltweit erreichbaren IP-Adresse und eine Möglichkeit der Datensicherung auf Basis des Tivoli Storage Manager (TSM). Der vom Kunden benötigte Speicherplatz kann flexibel angepasst werden.

Bei der Bestellung wird die virtuelle Maschine in einer Grundkonfiguration ausgeliefert. Hierbei kann der Kunde zwischen verschiedenen Betriebssystemen wie Linux oder Windows wählen. Nachdem der Server an den Kunden ausgeliefert wurde, wird vonseiten des LRZ nur das installierte Betriebssystem auf dem neuesten Stand gehalten. Der Kunde muss sich dann um die Installation der zu verwendenden Software, die Konfiguration und das Einspielen von Sicherheitspatches der eigen installierten Anwendungen selbst kümmern [LRZ17a]. Aus diesem Grund ist es wichtig, dass vorab geklärt wird, wer für die Konfiguration der Server und die Pflege der installierten Software zuständig ist. Dies ist sicherheitstechnisch eine große Herausforderung für das LRZ, da die Pflege der Server und das fortlaufende Untersuchen und Beheben von Schwachstellen nicht überprüft werden kann.

3.2.4. Infrastructure as a Service (IaaS)

Neben den klassischen Webhosting bietet das LRZ seinen Kunden auch eine Infrastructure as a Service (IaaS) an. Dabei handelt es sich um die Möglichkeit, mit wenigen Mausklicks kurzfristig beliebig viele virtuelle Maschinen zu bestellen, um die eigene Rechenleistung dynamisch zu erhöhen. Dieses ermöglicht dem Kunden schnell Rechenkapazitäten für beispielsweise einmalige Anwendungen zu bestellen.

Der Vorteil dabei ist, dass die Kunden keine eigene und oft kostenintensive Infrastruktur vorhalten müssen, um eine einmalige gegebenenfalls rechenintensive Anwendung betreiben zu können. Das LRZ gibt den Kunden dabei die Freiheit, ihre bestellten virtuellen Maschinen nach ihren Bedürftnissen zu konfigurieren und nach dem Einsatz wieder zu zerstören.

Das Zusammenfassen vieler Dienste und die somit entstehende Komplexität der Infrastruktur bringt auch Sicherheitsrisiken mit. Da die Kunden ihre virtuellen Maschinen und die darauf laufenden Anwendungen selbst konfigurieren können, entstehen weitere Risiken durch beispielsweise unsichere Konfigurationen der zu betreibenden Anwendungen.

3.2.5. Selbst verwaltete Netze

Das MWN ist ein sehr großes Netz, das mit der Zeit immer weiter gewachsen ist. Das LRZ verwaltet dabei die Anbindung verschiedener Fakultäten und Institutionen an das MWN. Jedoch gibt es auch Teilnetze, die nicht unter den Zuständigkeitsbereich des LRZ fallen. So werden beispielsweise die Campusnetze der Informatik der Technischen Universität, der Hochschule München, verschiedener Studentenheime des Studentenwerks und in der Medizin selbst verwaltet [LRZ12]. In diesen selbst verwalteten Netzen werden zum Teil auch Webserver betrieben, auf die das LRZ keinen Einfluss hat. Da hier nicht sichergestellt werden kann in welchem Zustand sich die Webserver befinden und ob auch regelmäßig Updates eingespielt werden, ist dies auch hier aus sicherheitstechnischer Sicht ein großes Problem.

3.3. Abgrenzung zu einem Schwachstellenscan

Der Begriff Penetrationstest und die Methoden zur Durchführung geht zurück auf einen im Jahr 1995 entwickelten Schwachstellen-Scanner ”SATAN”. Dieses Programm, das für das UNIX-Betriebssystem entwickelt wurde, war damals das erste Tool, mit welchem die automatisierte Untersuchung von Schwachstellen ermöglicht wurde [BSI03].

Aus heutiger Sicht muss man jedoch differenzieren zwischen einem automatisierten Schwachstellen-Scan und einem Penetrationstest. Ein automatisierter Scan der Zielsysteme kann durchgeführt werden, um kostengünstig Schwachstellen in Systemen zu entdecken, jedoch ersetzt das nicht einen manuellen Penetrationstest eines Experten, da automatisierte Anwendungen ab einem bestimmten Punkt ihre Grenzen erreichen. Die eingesetzten Algorithmen eines automatischen Scans können zwar technische Schwachstellen finden, jedoch scheitern sie oft bei der Interpretation der Ergebnisse. Ein Beispiel, bei dem ein automatischer Scan scheitern könnte, ist die Untersuchung auf Schwachstellen im Session Management. Oft lassen sich bei Webanwendungen die Benutzerrechte ausweiten, die eigentlich nur höher privilegierten Nutzern zustehen. Damit eine Schwachstelle im Session Management einer Webanwendung gefunden und ausgenutzt werden kann, bedarf es in vielen Fällen eines Experten, der die Ausgabe der Webanwendung interpretieren kann und mit etwas Kreativität Wege findet, diese Schwachstelle erfolgreich auszunutzen.

In dieser Arbeit soll jedoch einen Schritt weitergegangen werden als ein klassischer Schwachstellenscan. Die Anwendung soll den Nutzer bei einem Penetrationstest unterstützen, indem der Prozess so gut wie möglich automatisiert wird, jedoch der Benutzer selbst eingreifen kann, wenn der Service an seine Grenzen stößt. Der Benutzer durchläuft hierbei, angelehnt an einen klassischen Penetrationstest, mehrere Phasen. Beginnend mit einer Vorbereitungsphase, in der Genehmigungen für einen solchen Test eingeholt werden, gefolgt von einem Asset Discovery Scan zur Informationsbeschaffung, welche Dienste auf dem Zielsystem laufen. Im Anschluss kann der Benutzer anhand der gesammelten Informationen gezielte Schwachstellen-Scans durchführen oder auch versuchen, mittels einer Toolsammlung aktiv in das System einzubrechen.

3.4. Definition der Anforderungen

In den vorherigen Abschnitten wurde der Ausgangspunkt am LRZ beschrieben und diese Arbeit von einem klassischen Schwachstellenscan abgegrenzt. Da diese zwei Punkte geklärt sind, können im Folgenden die Anforderungen an einen Service für Penetrationstests erarbeitet werden. Dabei muss darauf geachtet werden, dass besonders auf die Anforderungen der späteren Benutzer eingegangen wird und diese zu erfüllen sind. Damit die Anforderungen identifiziert werden können, wurde zunächst eine Umfrage erstellt, die die Meinungen von Sicherheitsbeauftragten, Administratoren und Kunden des LRZ widerspiegeln soll. Bei dieser Umfrage haben 67 Mitarbeiter und Kunden des LRZ teilgenommen. Zudem wurden auch eine Reihe von Gesprächen mit Mitgliedern der Rechnerbetriebsgruppen und Webserveradminstratoren geführt. Die Ergebnisse der Umfrage sowie der Gespräche wurden daraufhin analysiert und die Anforderungen an einen Service für Penetrationstests abgeleitet.

Die in der Umfrage und den Gesprächen erarbeiteten Anforderungen wurden zur besseren Übersicht jeweils einer der vier Phasen eines Penetrationstests zugewiesen. Da dieser Service auch Anforderungen besitzt, die sehr allgemein gehalten sind, gibt es zudem eine weitere Kategorie, die alle allgemeinen Anforderungen beinhaltet. Die Kategorien, in welche die Anforderungen aufgeteilt wurden, zeigt die folgende Aufzählung.

- Allgemeine Anforderungen
- Anforderungen zur Informationsbeschaffung
- Anforderungen zur Informationsbewertung
- Anforderungen zum aktiven Eindringen
- Anforderungen zur Abschlussanalyse

3.5. Einblick in die Ergebnisse der Umfrage

Bevor die Anforderungen an den Service für Penetrationstests erläutert werden, soll zunächst ein kurzer Einblick in die Ergebnisse der Umfrage geschaffen werden. Dazu werden die Statistiken ausgewählter Fragen etwas näher betrachtet. Die Ergebnisse werden im vollen Umfang am Ende dieser Arbeit angehängt.

Die Umfrage wurde in verschiedene Kategorien unterteilt. Folgende Liste zeigt die vier Kategorien:

1. Erfassung des allgemeinen Ist-Standes
2. Anforderungen und Wünsche zum Umfang des Service
3. Anforderungen und Wünsche für den Funktionsumfang
4. Priorisierung der geforderten Funktionen.

Im Folgenden wird auf die Ergebnisse der einzelnen Kategorien näher eingegangen.

3.5.1. Erfassung des Ist-Standes

Bei der Entwicklung eines Service für Penetrationstests ist es wichtig, den Ist-Zustand im Rechenzentrum zu kennen, damit der Service an die dort vorliegenden Gegebenheiten ausgerichtet werden kann.

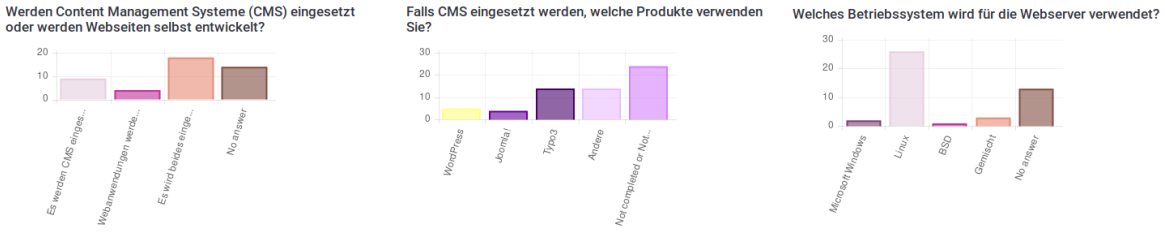


Abbildung 3.2.: Ein Auszug der Ergebnisse des Ist-Standes

Abbildung 3.2 zeigt die Ergebnisse zu den Fragen über eingesetzte Webanwendungen und Betriebssysteme. Das erste Diagramm soll Auskunft darüber geben, wie viele Webanwendungen fertige Lösungen sind und wie viele selbst entwickelt werden. Hier ist zu erkennen, dass bei den meisten Kunden Webanwendungen von externen Anbietern sowie selbstentwickelte Webseiten eingesetzt werden. Das zweite Diagramm zeigt, dass bei der Frage zu den installierten Webapplikationen sehr viele Typo3-Instanzen verwendet werden. Bei dieser Frage lag der Fokus auf den bekanntesten CMS Herstellern. Jedoch ist auch zu erkennen, dass teilweise CMS von anderen Herstellern verwendet werden, die in dieser Frage nicht genauer spezifiziert wurden. Bei der Frage, welches Betriebssystem zum Einsatz kommt, ist klar zu erkennen, dass die meisten Webserver unter dem Linux Betriebssystem betrieben werden.

3.5.2. Anforderungen und Wünsche zum Umfang des Service

In der nächsten Fragenkategorie, die sich mit den Anforderungen und Wünschen zum Umfang des Service befasst hat, wurden den Kunden Fragen gestellt, welche Webservices eingesetzt werden und welche Benutzeroberfläche gewünscht wird. Dabei geben die Fragen zu den Webservices Aufschluss, in welchem Umfang Tests entwickelt werden sollen. Hier steht beispielsweise die Frage im Raum ob Tests, die sich auf Web 2.0 Anwendungen spezialisiert haben, zum Einsatz kommen sollen.

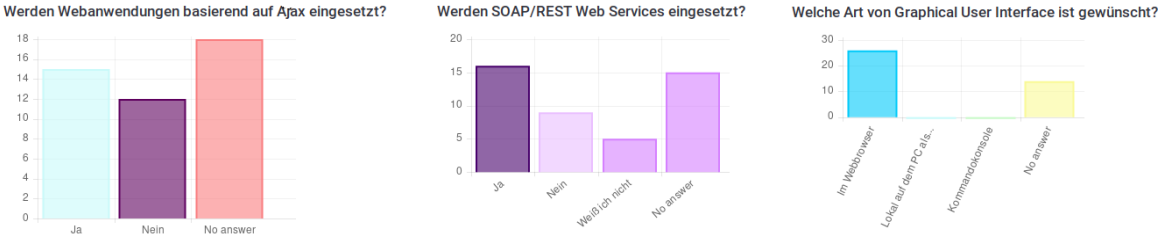


Abbildung 3.3.: Ein Auszug der Ergebnisse für die Wünsche zum Umfang des Service

In Abbildung 3.3 ist ein Auszug der Ergebnisse zu sehen. Hier ist zu erkennen, dass einige Kunden Webanwendungen basierend auf Ajax einsetzen. Zudem zeigen die Ergebnisse auch, dass eine Mehrzahl der Kunden auch Webservices wie SOAP und REST betreiben. Die letzte Frage in diesem Auszug bezieht sich auf die gewünschte Benutzeroberfläche. Hier ist klar zu erkennen, dass die Kunden ziemlich einstimmig eine Benutzeroberfläche im Browser wünschen.

3.5.3. Anforderungen und Wünsche für den Funktionsumfang

Eine weitere Fragenkategorie der Umfrage behandelt die Wünsche an den Funktionsumfang des Service. In dieser Kategorie wurden die Kunden befragt, welche Funktionen der Service anbieten soll. In Abbildung 3.4 sind die Ergebnisse zu Fragen über die Berichterstellung und die Analyse bereits erfolgter Angriffe zu sehen.

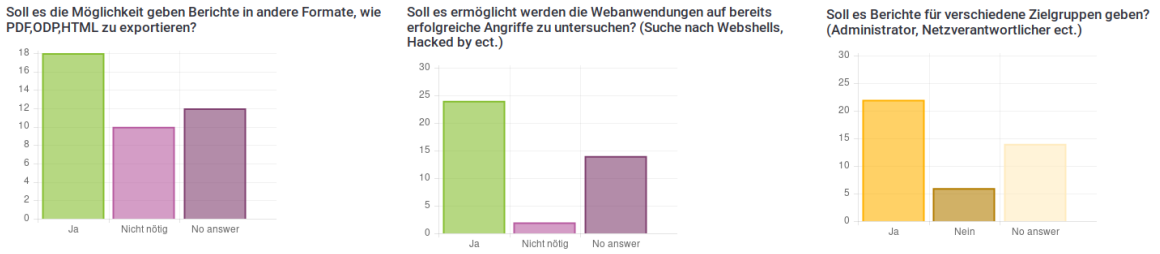


Abbildung 3.4.: Ein Auszug der Ergebnisse für den gewünschten Funktionsumfang

Bei dem ersten Diagramm wurde gefragt, ob es möglich sein soll, Berichte in andere Formate zu exportieren. Während einige Kunden diese Funktion nicht benötigen, wünschen sich andere aber, diese Funktion mit aufzunehmen. Die Mehrheit der Kunden haben jedoch dafür gestimmt, dass sie sich diese Funktion wünschen. Bei dem zweiten Diagramm wurde gefragt, ob es ermöglicht werden soll, Systeme auf bereits erfolgte Angriffe zu untersuchen. Hier ist das Ergebnis sehr deutlich ausgefallen. Fast alle Teilnehmer wünschen sich diese Funktion. Bei der Frage, ob es Berichte für verschiedene Zielgruppen geben soll, haben sich die meisten Teilnehmer dafür ausgesprochen, die Funktion zur Generierung verschiedener Berichte für verschiedene Zielgruppen mit aufzunehmen.

3.5.4. Priorisierung der geforderten Funktionen

In der letzten Fragenkategorie der Umfrage wurden die Kunden gefragt, bestimmte Funktionen zu priorisieren, um eine Einschätzung zu bekommen, wie wichtig einzelne Funktionen sind. Mit dieser Priorisierung kann erarbeitet werden, auf welchen Funktionen der Fokus liegen soll.

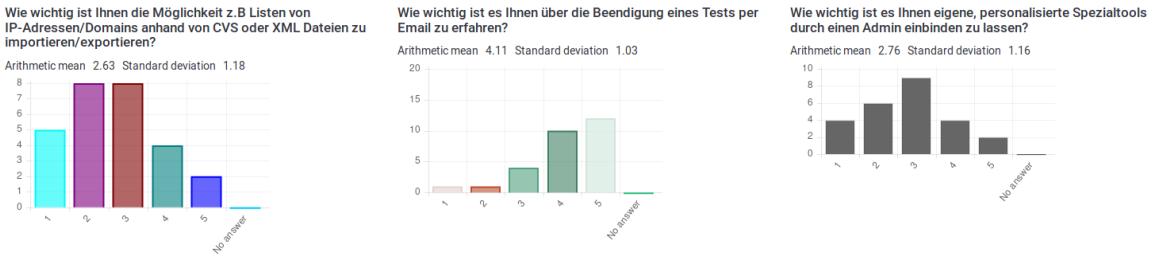


Abbildung 3.5.: Ein Auszug der Ergebnisse für die Priorisierung der geforderten Funktionen

In Abbildung 3.5 wird ein Auszug der Fragen für die Priorisierung gezeigt. Das erste Diagramm zeigt die Ergebnisse, wie wichtig die Kunden das Importieren von mehreren Zielsystemen einschätzen. Diese Funktion wird von den meisten Teilnehmern als nicht sehr wichtig eingeschätzt. Die nächste Frage, wie wichtig eine Benachrichtigung zur Beendigung eines Penetrationstests per Email ist, haben die meisten Teilnehmer mit einer

hohen Priorität bewertet. Bei der Frage, ob die Kunden das spätere Einbinden von Spezialtools für wichtig erachten, zeigen die Ergebnisse, dass hier die Priorität nur mittel ist.

3.6. Allgemeine Anforderungen

3.6.1. Automatischer Start eines Penetrationstests

In manchen Fällen ist es nützlich, dem Benutzer eines Service für Penetrationstests die Möglichkeit zu geben, einen Test automatisch zu einem festgelegten Zeitpunkt oder auch periodisch zu starten. Dies hat auch die Umfrage ergeben, in der fast alle Beteiligten diese Funktion gewünscht haben.

3.6.2. Importieren und Exportieren von CSV-Dateien

Die Kunden des LRZ betreiben teilweise mehrere Webserver, die regelmäßig getestet werden sollten. Damit es dem Kunden oder auch Sicherheitsbeauftragten des LRZ ermöglicht wird, eine Vielzahl von Servern zu testen, soll eine Import- und Exportfunktion von CSV-Dateien bereitgestellt werden, um Listen von IP-Adressen oder Domainnamen in den Service einzubringen.

3.6.3. Benachrichtigung über die Beendigung eines Tests

Wird ein Penetrationstest auf einem Webserver durchgeführt, kann der Tester je nach Bedarf einstellen, welche Komponenten überprüft werden und in welchem Detail diese getestet werden sollen. Somit kann es passieren, dass ein Penetrationstest eine längere Zeit in Anspruch nimmt, bis alle Tests vollständig durchgelaufen sind. In diesem Fall soll es die Möglichkeit geben, eine Benachrichtigung über die Fertigstellung eines Tests per Email zu erhalten. Da Emails ein unsicheres Übertragungsmedium sind, soll hier nur die Benachrichtigung über einen fertiggestellten Test versendet werden, die keine weitere Information über Schwachstellen oder den zu testenden Webserver enthält.

3.6.4. Benutzerverwaltung und Administration durch Superusers

Bei dem Einsatz eines solchen Service in einer Hochschule gibt es eine Vielzahl von Benutzern in verschiedenen Lehrstühlen und Organisationen, die ihre Webanwendungen auf Schwachstellen testen wollen. Um diese Vielzahl von Benutzern verwalten zu können, bedarf es eines Superusers, der die Benutzer und deren Rechte verwalten und pflegen kann.

3.6.5. Hierarchische Benutzerrechte innerhalb einer Abteilung

Innerhalb eines Lehrstuhles soll der zu entwickelnde Service von verschiedenen Personen verwendet werden. Zum einen soll ein Sicherheitsbeauftragter in einem Lehrstuhl alle Webserver testen können, zum anderen sollen aber auch die Entwickler, die an einer Webanwendung arbeiten, diese auf ihre Sicherheit hin testen können. Damit realisiert werden kann, dass zum Beispiel ein Entwickler nur den Server testen darf, auf dem er auch arbeitet, muss es verschiedene Nutzerrechte geben, um den Penetrationstest auf die zulässigen Server zu begrenzen. Die folgende Liste zeigt, welche Benutzer es geben sollte.

- Superuser
- Sicherheitsverantwortlicher
- Administrator
- Entwickler
- IT-Manager

3.6.6. Mehrbenutzer-Betrieb

Der Penetrationtesting-Service wird hauptsächlich von Kunden und Sicherheitsbeauftragten des LRZ verwendet. Als Nutzer kommen daher verschiedene Lehrstühle und Organisationen in Frage. Da der Service auch von mehreren Benutzern zur gleichen Zeit verwendet werden könnte, muss es möglich sein, Penetrationstests auch parallel durchführen zu können.

3.6.7. Sicherheit der Anwendung

Da der zu entwickelnde Service, wie schon oben beschrieben, von vielen Personen benutzt wird, welche auch identifiziert werden müssen, werden personenbezogene Daten für die Anmeldung am System gespeichert. Daher müssen die Anforderungen hinsichtlich des Datenschutzes, der Datensicherheit und des Zugriffsschutzes erfüllt werden. Der Service muss daher die Empfehlungen des Bundesamtes für Sicherheit in der Informationstechnik zur sicheren Entwicklung von Webanwendungen erfüllen [BSI13b].

3.6.8. Mandantenfähigkeit

Ein Service für Penetrationstests kann von den Nutzern für gute Zwecke eingesetzt werden, indem sie die eigenen Server auf Schwachstellen untersuchen und die Gefährlichkeit dieser Schwachstellen anhand eines Einbruchsversuchs verifizieren. Jedoch ist auch denkbar, dass Nutzer damit nicht nur ihre eigenen Webanwendungen testen, sondern versuchen, Systeme anzugreifen, welche nicht in ihrem Zuständigkeitsbereich liegen. Daher muss dafür gesorgt werden, dass der Nutzer diesen Service auf sein eigenes Netz beschränkt und somit ein Angriff anderer Systeme unterbunden wird.

Da das LRZ Dienstleister für verschiedene Universitäten und Institute ist, soll zudem darauf geachtet werden, dass diese voneinander getrennt werden. Dies hat den Grund, dass den einzelnen Parteien möglicherweise unterschiedliche Tests angeboten werden, die in manchen Fällen gesondert abgerechnet werden müssen.

3.6.9. Leicht anpassbar und erweiterbar

In der IT kommt es sehr oft zu Veränderungen. Fast täglich werden neue Schwachstellen in öffentlichen Projekten entdeckt und für diese Schwachstellen Exploits entwickelt. Auch Tools, die einem Penetrationstester helfen, Schwachstellen aufzudecken und diese auszunutzen, verändern sich über die Zeit. So kann es sein, dass bei einem für diese Arbeit verwendeten IT-Sicherheit-Tool neue Parameter hinzukommen oder die grundlegende Bedienung verändert wird. Damit der Service, welcher im Rahmen dieser Arbeit entwickelt werden soll, auch langfristig einsetzbar ist, muss gewährleistet werden, dass eine spätere Anpassung und Erweiterung möglich ist.

3.6.10. Die Bedienung muss über eine GUI und API möglich sein

Der zu entwickelnde Service sollte für alle Benutzer angenehm zu bedienen sein. Während einige Nutzer es gewöhnt sind, ihre Anwendungen mittels eines Graphical User Interface (GUI) zu bedienen, wollen andere Anwender ihre eigenen Skripte schreiben, um einen maßgeschneiderten Penetrationstest durchzuführen. Damit die Anforderungen aller User abgedeckt werden, soll es neben einer GUI auch eine API geben, damit die Nutzer auch flexibel mit diesem Service arbeiten können.

3.6.11. Schnittstelle zu anderen Sicherheitstools

Das LRZ betreibt bereits das Tool Dr. Portscan, welches in regelmäßigen Abständen Rechner, die am MWN angebunden sind, auf offene Ports untersucht. Hier bietet es sich an, eine Schnittstelle zu Dr. Portscan zu schaffen. Wenn Dr. Portscan zum Beispiel bei einem Server den Port 80 als geöffnet identifiziert, der bislang

noch nicht offen war, könnte ein Penetrationstest auf dem betreffenden Server angestoßen werden, um potentielle Schwachstellen zu identifizieren.

3.6.12. Automatisierung

Hinsichtlich der personellen und finanziellen Begrenzung der einzelnen Lehrstühle und Organisationen soll der Penetrationstest so gut wie möglich automatisiert werden. Da es aufgrund der Natur eines Pentests nicht möglich ist, den kompletten Test zu automatisieren, soll zumindest dafür gesorgt werden, dass die meisten Schritte automatisch ablaufen können und der Benutzer nur dann eingreift, wenn ein eingesetztes Tool an seine Grenzen stößt.

3.6.13. Testen einer Vielzahl von Webservern

Die Kunden des LRZ betreiben teilweise mehrere Webserver, die regelmäßig getestet werden sollen. Damit die Person, die für den Penetrationstest zuständig ist, nicht jeden Server einzeln testen muss, soll es eine Möglichkeit geben, eine Liste von IP-Adressen oder URLs zu importieren, um einen Test auf einer Vielzahl von Webservern durchführen zu können.

3.6.14. Der Penetrationstest einzelner Webserver muss zeitnah beendet werden

Da ein Penetrationstest immer überwacht werden sollte, damit ein ungeplanter Ausfall eines Webservers schnell behoben werden kann, darf ein Test nicht zu lange laufen. Die Umfrage und eine Reihe an Gesprächen haben ergeben, dass das Limit für einen Penetrationstest bei weniger als zwei Stunden liegen sollte.

3.7. Anforderungen zur Informationsbeschaffung

3.7.1. Erkennung geöffneter Ports und dahinter laufender Dienste

Bei einem Penetrationstest ist es wichtig, zu Beginn viel Information über das Zielsystem zu sammeln, um im weiteren Verlauf diese Information zum aktiven Eindringen in das System zu nutzen. Zu dieser Information gehört auch, die geöffneten und geschlossenen Ports des Zielsystems zu kennen. Oft laufen auf einem Zielsystem nicht nur ein Webserver, wie beispielsweise wenn ein Kunde seinen Server mit Plesk betreibt. In diesem Fall läuft Plesk, wenn nicht anders konfiguriert, auf Port 8443. Aus diesem Grund muss der Service in der Phase der Informationsbeschaffung zunächst einen Portscan durchführen. Nachdem die Ports des Zielsystems gescannt wurden, sollen vor allem Ports, die von den Standardports 80 und 443 eines Webservers abweichen, getestet werden, ob sich dahinter ein weiterer Webserver befindet, der angegriffen werden kann. Dies hat den Grund, da bei vielen Webservern Administrationstools wie Plesk betrieben werden, die auf einem Port erreichbar sind, welcher vom Standard HTTP Port abweicht.

3.7.2. Erkennung von eingesetzter Software und deren Versionsstand

Das Updaten von Software wird oft automatisiert durchgeführt. Jedoch kann das Einspielen eines Updates auch fehlerhaft sein, wodurch weiterhin die veraltete Software betrieben wird. Damit dem Nutzer ein guter Überblick über seine Infrastruktur gegeben werden kann, müssen die Server auf die eingesetzten Webanwendungen, Dienste und deren Versionsstand überprüft und dokumentiert werden. Zudem kann diese Information auch bei dem späteren Versuch, aktiv in das System einzudringen, genutzt werden.

3.7.3. Erkennung von bereits erfolgreichen Angriffen

Da nicht ausgeschlossen werden kann, dass ein Webserver bereits erfolgreich angegriffen wurde, soll es dem Benutzer ermöglicht werden, nach Spuren eines erfolgreichen Angriffs zu suchen. Hierbei soll dem Kunden ein Skript angeboten werden, das er auf seiner virtuellen Maschine ausführen kann, um beispielsweise nach Webshells zu suchen. Das Skript soll zudem alle Dateien auf dem betreffenden Server nach merkwürdigen Schlüsselwörtern wie ”hacked by” durchsuchen. Diese sollen durch den Service flexibel konfigurierbar sein.

3.8. Anforderungen zur Informationsbewertung

3.8.1. Flexible Auswahl durchzuführender Tests

Nachdem die Phase der Informationsbeschaffung beendet ist, kann dem Benutzer angezeigt werden, welche Anwendungen und Dienste auf dem zu testenden Server gefunden wurden. Hier soll es ermöglicht werden, flexibel auszuwählen, welche Tests in der nächsten Phase durchgeführt werden sollen.

3.8.2. Priorisieren von Testergebnissen

Bei einem Penetrationstest kommt es immer wieder vor, dass Schwachstellen gefunden werden, die nicht sehr kritisch sind und in naher Zukunft nicht behoben werden. Außerdem können auch False-Positives bei den Scans auftreten. Bei weiteren Tests würden diese Schwachstellen immer wieder im Report auftauchen, was ggf. dazu führen kann, dass der Report unübersichtlich wird und somit neu erkannte Schwachstellen übersehen werden. Um den Report so übersichtlich wie möglich zu halten, muss es eine Funktion geben, mit der ein Benutzer ausgewählte Schwachstellen geringer priorisieren kann. Dies hat dann zur Folge, dass diese Schwachstellen bei zukünftigen Penetrationstests erst am Ende des Reports aufgelistet werden.

3.9. Anforderungen zum aktiven Eindringen

3.9.1. Verwendung von Exploits für bekannte Schwachstellen aller populären CMS

Die meisten Kunden des LRZ verwenden eines der drei bekanntesten Content Management Systeme - Wordpress, Joomla! und Typo3. Dies ist aus der Umfrage und den geführten Gesprächen hervorgegangen. Da diese Content Management Systeme von vielen Kunden verwendet werden, müssen sie bei einem Penetrationstest zusätzlich auf öffentlich bekannte Schwachstellen hin getestet werden.

3.9.2. Generische Tests

Neben den oben erwähnten Content Management Systemen werden am LRZ auch selbst entwickelte Webanwendungen betrieben. Damit ein Penetrationstest auf dem Webserver des Kunden durchgeführt werden kann, muss der Service Schwachstellen selbst erkennen und versuchen, diese auszunutzen.

3.9.3. Penetration verschiedener Betriebssysteme

Das MWN ist ein großes, heterogenes Rechnernetz, in dem Rechner mit verschiedenen Betriebssystemen betrieben werden. Somit wird klar, dass ein Penetrationstest für Webserver mit verschiedenen Betriebssystemen möglich sein muss. Dies wirkt sich vor allem bei der eingesetzten Shell aus, die bei einem Angriff auf den Server eingeschleust wird. Es muss darauf geachtet werden, dass diese Shell auf allen Betriebssystemen reibungslos funktioniert.

3.9.4. Keine Beeinträchtigung des Netzes oder der Server

Damit auch bei einem geplanten Penetrationstest der Betrieb der Server und des Netzes nicht gestört wird, müssen die Tests so konzipiert sein, dass sie sich nicht auf die Erreichbarkeit der Server oder die Performance des Netzwerkes auswirken. Falls es jedoch bestimmte Tests gibt, die sehr aggressiv vorgehen, müssen diese für den Benutzer klar gekennzeichnet werden, dass das Ausführen dieses Tests eine Instabilität der Zielsysteme zur Folge haben könnte.

3.10. Anforderungen zur Abschlussanalyse

3.10.1. Dokumentation der Ergebnisse

Die Dokumentation von gefundenen Schwachstellen ist eine der wichtigsten Funktionen, die ein Service für Penetrationstests besitzen muss, um die gefundenen Schwachstellen nachvollziehen zu können. Der Bericht muss zudem übersichtlich und gut gegliedert sein, damit der Benutzer die gefundenen Schwachstellen leicht erkennen und beheben kann.

3.10.2. Errechnen des CVSS Base-Score

Nachdem der Penetrationstest abgeschlossen ist, soll für den Benutzer ein Report erstellt werden, der alle gefundenen Schwachstellen enthält. Damit die potentiellen Auswirkungen einer gefundenen Schwachstelle besser verdeutlicht werden können, soll für jeden Fund der CVSS-Base-Score berechnet werden. Dieser soll in dem Bericht aufgenommen werden.

3.10.3. CVE-Nummer öffentlich bekannter Schwachstellen

Da der Penetrationstest auch die öffentlich bekannten Schwachstellen in Content Management Systemen und anderer vom Kunden verwendeter Software untersucht, soll zusätzlich zum CVSS Base-Score die CVE-Nummer der entdeckten Schwachstelle in den Bericht aufgenommen werden. Dies ist wichtig, damit der Benutzer genauere Informationen zu dieser Schwachstelle recherchieren kann.

3.10.4. Exportieren von Berichten in andere Formate

Wenn seitens der Hochschule ein Informations-Sicherheits-Management-System (ISMS) eingesetzt wird, wäre es denkbar, dass die Schwachstellen, die durch den Penetrationstest identifiziert wurden, in das ISMS eingetragen werden sollen. Um hinsichtlich personeller Begrenzung dem Benutzer viel Arbeit abzunehmen, bietet es sich an, eine Funktion bereitzustellen, die Berichte in verschiedene Formate exportieren kann.

3.11. Zusammenfassung und Priorisierung der Anforderungen

Da die Anforderungen an einen Service für Penetrationstests nun definiert und beschrieben wurden, sollen sie im Folgenden zur Übersichtlichkeit in tabellarischer Form dargestellt werden. Zudem werden die Anforderungen als Zusatzfeature (+), wichtige Anforderung (++) und Pflichtanforderung (+++) priorisiert. Dabei bedeutet eine Pflichtanforderung, dass sie für einen Penetrationstesting-Service unverzichtbar ist. Eine wichtige Anforderung besagt, dass sie zwar wichtig ist, aber der Service auch ohne diese Anforderung rudimentär nutzbar ist. Ein Zusatzfeature wäre gut zu haben, es ist aber für den Betrieb des Service nicht zwingend nötig. Die Priorisierung wurde durchgeführt, um einen guten Überblick der Wichtigkeit der Anforderungen zu schaffen.

Abschnitt	Anforderung	Priorität
Allgemeine Anforderungen		
3.6.1	Automatischer Start eines Penetrationstests	+
3.6.2	Importieren und Exportieren von CSV-Dateien	++
3.6.3	Benachrichtigung über die Beendigung eines Tests	+
3.6.4	Benutzerverwaltung und Administration durch Superusers	+++
3.6.5	Hierarchische Benutzerrechte innerhalb einer Abteilung	+++
3.6.6	Mehrbenutzer-Betrieb	+++
3.6.7	Sicherheit der Anwendung	+++
3.6.8	Mandantenfähigkeit	+++
3.6.9	Leicht anpassbar und erweiterbar	+++
3.6.10	Die Bedienung muss über eine GUI und API möglich sein	++
3.6.11	Schnittstelle zu anderen Sicherheitstools	++
3.6.12	Automatisierung	+++
3.6.13	Testen einer Vielzahl von Webservern	+
3.6.14	Der Penetrationstest einzelner Webserver muss zeitnah beendet werden	+++
Anforderungen zur Informationsbeschaffung		
3.7.1	Erkennung geöffneter Ports und dahinter laufender Dienst	+++
3.7.2	Erkennung von eingesetzter Software und deren Versionsstand	+++
3.7.3	Erkennung von bereits erfolgreichen Angriffen	++
Anforderungen zur Informationsbewertung		
3.8.1	Flexible Auswahl durchzuführender Tests	+++
3.8.2	Priorisierung von Testergebnissen	+
Anforderungen zum aktiven Eindringen		
3.9.1	Verwendung von Exploits für bekannte Schwachstellen aller populären CMS	++
3.9.2	Generische Tests	++
3.9.3	Penetration verschiedener Betriebssysteme	+++
3.9.4	Keine Beeinträchtigung des Netzes oder der Server	+++
Anforderungen zur Abschlussanalyse		
3.10.1	Dokumentation der Ergebnisse	+++
3.10.2	Errechnen des CVSS Base-Score	+++
3.10.3	CVE-Nummer bekannter Schwachstellen	+++
3.10.4	Exportieren von Berichten in andere Formate	+

Tabelle 3.1.: Zusammenfassung der Anforderungen an einen Service für Penetrationstests

4. Themenverwandte Arbeiten

Nachdem die Anforderungen an einen Service für Penetrationstests beschrieben wurden, sollen in diesem Kapitel themenverwandte Arbeiten vorgestellt werden, die durch eine Recherche gefunden wurden. Am Ende des Kapitels sollen die gerade beschriebenen Anforderungen mit denen der jeweiligen themenverwandten Arbeit verglichen werden, um einen Überblick zu verschaffen, welche Anforderungen nicht abgedeckt wurden.

4.1. Metasploit

Metasploit ist ein sehr verbreitetes Framework, das von vielen Penetrationstestern für den Einbruch in Zielsysteme genutzt wird. Es wurde ursprünglich von HD Moore im Jahr 2003 entwickelt und wird inzwischen von Rapid7 weiterentwickelt. Metasploit-Framework ist ein kostenloses Open Source Tool, das nicht nur von Rapid7 sondern auch von einer großen Community lebt. Metasploit-Framework ist modular aufgebaut und kann somit leicht angepasst und erweitert werden.

Neben dem kostenlosen Metasploit-Framework gibt es auch eine kommerzielle Metasploit Pro Version, die eine Vielzahl zusätzlicher Exploits beinhaltet und durch starke Automatisierung bei professionellen Penetrationstests helfen kann. Da diese Pro-Version jedoch ein kommerzielles Produkt ist und somit hohe Lizenzkosten anfallen, wird in dieser Arbeit nicht weiter auf diese Version eingegangen. Im Folgenden wird die Architektur, die Module und die Benutzerschnittstelle des Metasploit-Frameworks, wie sie in Abbildung 4.1 dargestellt ist, beschrieben.

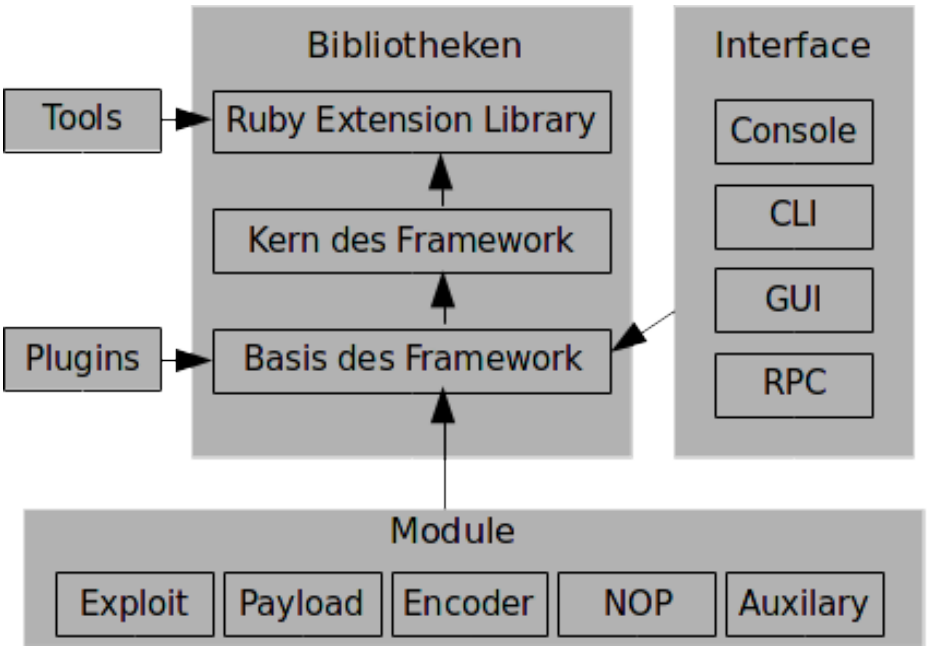


Abbildung 4.1.: Die Architektur von Metasploit-Framework nach [MSF15]

4.1.1. Module

Wie bereits erwähnt, ist Metasploit modular aufgebaut, um eine spätere Erweiterung und Anpassung leichter zu gestalten. So können dem Framework beispielsweise neu entwickelte Exploits hinzugefügt werden, welche dann auf bereits bestehende Payloads zugreifen können.

Exploit

Die Exploits sind Computerprogramme, die eine Schwachstelle in einer Anwendung oder einem System ausnützen, um zum Beispiel einer nicht autorisierten Person Zugriff auf das System zu verschaffen.

Payload

Wenn der Angreifer erst einmal Zugriff auf das System erlangt hat, soll der Zugriff zu einem späteren Zeitpunkt weiterhin möglich sein. Um dies zu realisieren, wird auf dem System Schadcode ausgeführt, um diesen Zugang abzusichern. Dieser Schadcode nennt sich Payload und wird ausgeführt, nachdem der Exploit gegen das Zielsystem erfolgreich war.

Encoder

Um einen Angriff auf Systeme zu erkennen und diese gegebenenfalls abzuwehren, werden oft Intrusion Detection Systems (IDS) oder Virenscanner eingesetzt. Damit ein Angriff durch ein IDS oder einen Virenscanner nicht so leicht entdeckt wird, werden die Payloads so verschleiert, sodass sie von Sicherheitstools nicht als Angriff gewertet werden. Für diesen Zweck verwendet Metasploit Encoder und NOP, um die Erkennungsrate der Payloads durch Sicherheitstools zu minimieren.

Auxiliary

Bei den Auxiliary Modulen (Helfer Module) handelt es sich um Tools, die hauptsächlich zur Informationsbeschaffung über ein Zielsystem benötigt werden. Metasploit bringt eine Vielzahl von diesen Tools mit, wie zum Beispiel Portscanner, Netzwerksniffer oder Scanner für verschiedene Dienste, die auf dem Zielsystem erreichbar sind.

4.1.2. Bibliotheken

Die Bibliotheken von Metasploit sind in drei Hauptkategorien unterteilt - Ruby Extension Library, Framework Kern und Framework Basis. Diese Bibliotheken werden im Folgenden beschrieben.

Ruby Extension Library

Die Ruby Extension Library (REX) bildet die zentrale Komponente des Frameworks. Sie besteht aus verschiedenen Clients, Servern und einer Vielzahl von Klassen, die von anderen Metasploit-Komponenten verwendet werden können. Die REX kann aber auch zur Entwicklung von neuen Exploits dienen.

Framework Kern

Der Framework Kern bietet unterschiedliche Klassen, die unter anderem für Modul- und Session Management zuständig sind. Der Framework Kern kann von den Metasploit Modulen und Plugins als Interface verwendet werden.

Framework Basis

Die Framework Basis soll die Arbeit mit dem Kern erleichtern. Sie fungiert als Benutzerschnittstelle, die unmittelbar auf den Framework Kern zugreift.

4.1.3. User Interface

Metasploit bietet zwei Benutzerschnittstellen an - konsolenbasiert und grafisch. Die meist verwendete Benutzerschnittstelle ist die konsolenbasierte, die an eine Linux- oder Windows Konsole erinnert. Hierbei können die Benutzer Metasploit mit Hilfe von Kommandos steuern. Neben der konsolenbasierten Benutzeroberfläche kann aber auch eine grafische Benutzeroberfläche verwendet werden, welche den Namen Armitage trägt. Jedoch besitzt Metasploit so viele Funktionen, dass bei der grafischen Benutzeroberfläche schnell einige Funktionen untergehen können. [MSF15]

4.2. POTASSIUM: Penetration Testing as a Service

POTASSIUM ist eine wissenschaftliche Arbeit über Penetrationstests als Service für Cloudanbieter. Die Autoren motivieren ihre Arbeit damit, dass bei einem Penetrationstest in einer Cloud-Infrastruktur einige Probleme auftreten können. Ein Beispiel hierfür ist, wenn ein aggressiver Test auf die Ressourcen eines Kunden durchgeführt wird, wie zum Beispiel eine DoS-Attacke. In diesem Fall ist es denkbar, dass nicht nur das angegriffene System eines Kunden ausfällt, sondern auch viele andere Kunden darunter zu leiden haben. Dies liegt an der Natur eines Cloudservices, da sich hier viele Kunden die verfügbaren Resourcen teilen. Wird also das Netzwerk eines Kunden überlastet, so werden auch andere Kunden, die auf den gleichen physischen Resourcen gehostet werden, beeinträchtigt. Aus diesem Grund erlauben Cloud-Provider in der Regel keine Penetrationstests, da sie andere Kunden vom Betrieb ihrer Projekte beeinträchtigen könnten. Da Penetrationstests auch für Cloudanbieter ein wichtiges Instrument sind, um Schwachstellen und ihre Gefährlichkeit aufzudecken, wurde das Tool POTASSIUM an der Universität von Utah entwickelt.

POTASSIUM ist eine Erweiterung von OpenStack, die automatisierte Penetrationstests für Cloudanbieter anbieten soll. Es setzt auf das Metasploit-Framework, kann aber auch durch andere Penetrationstesting-Tools erweitert oder ersetzt werden. Die Grundidee bei POTASSIUM ist, einen Penetrationstest möglich zu machen, ohne dass der Betrieb der Livesysteme beeinträchtigt wird. Dies realisieren die Autoren, indem die zu testenden virtuellen Maschinen geklont werden, um identische Kopien der Livesysteme zu erhalten. Anschließend wird die neu erschaffene Umgebung von den Livemaschinen isoliert und der Penetrationstest mittels des Metasploit-Frameworks gestartet. Der Vorteil die zu testende virtuelle Maschine vor dem Penetrationstest zu klonen liegt darin, dass der Penetrationstest keinen Einfluss auf den Betrieb der eigentlichen Systeme hat und somit zu keiner Zeit die Verfügbarkeit der Livesysteme beeinträchtigt.

4.2.1. Architektur von POTASSIUM

Wie bereits in Abschnitt 4.2 beschrieben, müssen vor Beginn eines Penetrationstests zunächst die zu testenden virtuellen Instanzen geklont werden. Das Vorgehen bei dieser Prozedur ist illustriert in Abbildung 4.2. **Schritt A** zeigt den Ausgangspunkt der Cloud. In **Schritt B** werden zunächst die virtuellen Maschinen, die getestet werden sollen, angelegt. Zu diesem Zeitpunkt haben die VMs die gleiche Struktur wie die originalen VMs. Um nun ein exaktes Speicherabbild zu erhalten, werden in **Schritt C** Snapshots der originalen VMs angelegt und auf die kopierten VMs übertragen. Im **letzten Schritt** werden die kopierten VMs von der Cloudumgebung isoliert und der Penetrationstest durchgeführt.

Die Autoren haben POTASSIUM als Erweiterung zu OpenStack konzipiert, so dass es eine zentrale Komponente beinhaltet, die alle Anfragen für Penetrationstests verwaltet. Die Architektur mit allen dazugehörigen Arbeitsabläufen ist in Abbildung 4.3 dargestellt.

Um dem Leser einen Überblick zu verschaffen, wie die einzelnen Arbeitsschritte von POTASSIUM ablaufen, wird im Folgenden die Architektur aus Abbildung 4.3 erläutert.

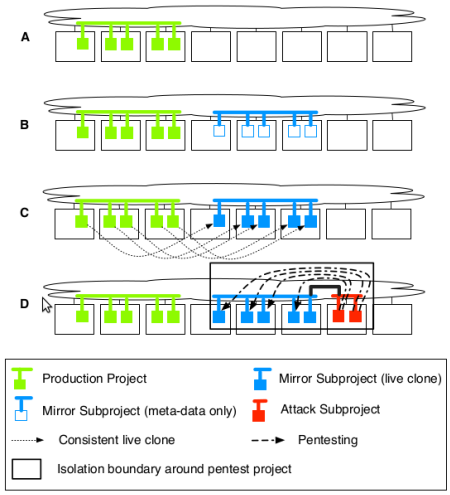


Abbildung 4.2.: POTASSIUM Arbeitsablauf nach [POT15]

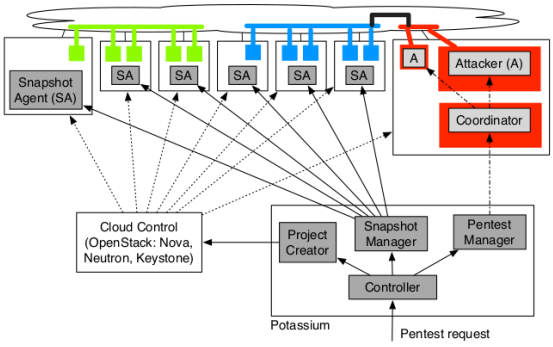


Abbildung 4.3.: POTASSIUM Architektur nach [POT15]

Der **Controller** als zentrale Komponente nimmt die Anfrage für einen Penetrationstest entgegen und kümmert sich um die weiteren Schritte, die nötig sind, um einen Penetrationstest aufzusetzen. Der **Project Creator** ist zuständig für das Isolieren der Umgebung und das Erstellen der VMs, auf denen der Penetrationstest durchgeführt werden soll. Sind die VMs erstellt, kümmert sich der **Snapshot Manager** darum, die Speicherabbilder der originalen VMs zu erstellen und portiert diese auf die neuen VMs. Sobald die Umgebung fertig aufgebaut und isoliert wurde, kommt der Pentest Manager ins Spiel, der den eigentlichen Penetrationstest durchführt.

Bei der Durchführung eines Penetrationstests stellen die Autoren drei Verfahren bereit. Einen internen Penetrationstest, bei dem alle virtuellen Maschinen getestet werden, egal ob sie von einem externen Netzwerk erreichbar sind. Einen externen Test, bei dem nur die Maschinen getestet werden, die auch von außen erreichbar sind und einen Pivot-Verfahren, bei dem getestet wird, wie weit ein Angreifer kommen würde, wenn eine virtuelle Maschine bereits kompromittiert wurde. [POT15]

4.3. OpenVAS/Nessus

OpenVAS ist ein Schwachstellen- und Netzwerkscanner, der aus dem Nessus Projekt hervorgegangen ist und nun von der Greenbone Networks GmbH weiterentwickelt wird. Bis zum Jahr 2005 war Nessus unter einer Open Source Lizenz und wurde dann in eine proprietäre Lizenz umgewandelt. Von diesem Zeitpunkt an wurde die letzte freie Version von Nessus als Fork[1] mit dem Namen OpenVAS ins Leben gerufen. Aufgrund der

[1]Ein Fork ist in der Softwaretechnik eine Abspaltung eines Projekts in ein anderes.

Tatsache, dass OpenVAS ein Fork von Nessus ist, haben diese beiden Tools viel gemeinsam. Jedoch wurden auch einige Komponenten unabhängig von Nessus entwickelt, wie das OpenVAS Management Protokoll.

Das Tool beinhaltet laut der Projektwebseite, mit Stand 2016, über 47.000 Sicherheitstests, wobei laufend neue hinzukommen [VAS16]. OpenVAS setzt wie auch Nessus auf ein Client-Server-Modell, bei dem der OpenVAS-Server eine zentrale Instanz für die Sicherheitstests bildet. Um die Sicherheitstests zu steuern, verbinden sich die Clients mit dem Server und senden diesem die gewünschten Kommandos. OpenVAS unterstützt die Steuerung mittels eines Webbrowsers sowie einer Konsole.

4.3.1. OpenVAS Architektur

Der Aufbau von OpenVAS ist illustriert in Abbildung 4.4. Im Folgenden werden die einzelnen Komponenten von OpenVAS etwas näher betrachtet und erläutert.

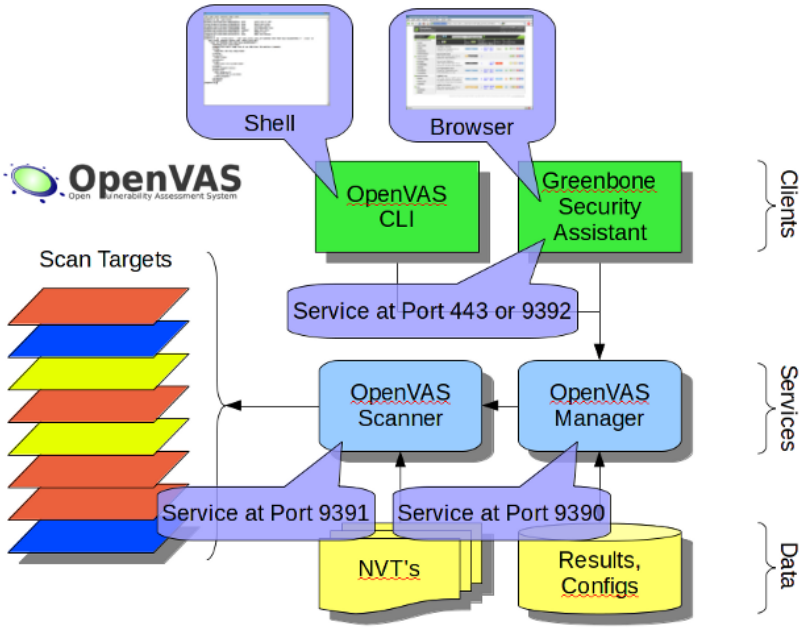


Abbildung 4.4.: OpenVAS Architektur [VAS17]

OpenVAS CLI

Das OpenVAS Command Line Interface (CLI) ermöglicht dem Benutzer das Steuern des Schwachstellenscanners mit Hilfe der Konsole.

Greenbone Security Assistent

Der Greenbone Security Assistent ist das Graphical User Interface von OpenVAS. Es bietet dem Benutzer eine Vielzahl von Funktionen, um beispielsweise den OpenVAS-Server zu konfigurieren oder Einstellungen für einen Schwachstellenscan vorzunehmen.

OpenVAS-Scanner

Den Kern von OpenVAS bildet der OpenVAS-Scanner. Mit ihm lassen sich viele Ziele gleichzeitig auf Schwachstellen untersuchen, wofür er auf eine große Auswahl von Network Vulnerability Tests (NVTs) zugreift. Die NVTs sind vorgefertigte Skripte, die den Ablauf des Schwachstellenscans steuern. Er lässt sich über das

OpenVAS-Transfer-Protokoll (OTP) ansprechen und unterstützt eine SSL-Verschlüsselte Verbindung. Gesteuert wird der Scanner durch die OpenVAS-Manager-Komponente.

OpenVAS Manager

Der OpenVAS Manager ist ein zentraler Dienst, der den Scanner oder auch viele Scanner mittels des OTP steuert. Der Manager nimmt auch die Scan-Resultate entgegen und speichert diese für die spätere Verwendung in einer zentralen SQL-Datenbank. Unter anderem übernimmt der Manager auch die Benutzerverwaltung und die Zugangskontrolle mittels Rollen und Gruppen. Die Clients können mit dem OpenVAS Management Protokoll (OMP) auf den Manager zugreifen, um zum Beispiel Scan-Aufträge abzusetzen. Des Weiteren bietet der Manager auch Filter- und Sortiermöglichkeiten an, womit die Clients immer die gleiche Sicht auf die Scan-Ergebnisse haben. [VAS17]

4.4. Zusammenfassung der vorgestellten Arbeiten

In diesem Kapitel wurden eine Auswahl themenverwandter Arbeiten vorgestellt und beschrieben. Es konnte jedoch keine Arbeit gefunden werden, die alle wichtigen Anforderungen aus Kapitel 3 abdeckt.

Um einen besseren Überblick zu schaffen, welche Anforderungen aus Kapitel 3 im Rahmen der hier vorgestellten Arbeiten erfüllt werden konnten und welche Anforderungen nicht, sollen im Folgenden die Anforderungen für diese Arbeit mit den themenverwandten Arbeiten verglichen werden. Dazu wird die Tabelle der Anforderungen an einen Service für Penetrationstests um drei Spalten für die jeweilige themenverwandte Arbeit erweitert und bewertet. Dabei wird für eine nicht erfüllte Anforderung **o** und für eine Anforderung, die durch die jeweilige Arbeit erfüllt wird, ein **xx** vergeben. Falls eine Anforderung nicht vollständig abgedeckt wird, so soll ein **x** vergeben werden. Aus Platzgründen wurden Abkürzungen für die einzelnen Arbeiten verwendet. Dabei stehen MS, PO und VAS für Metasploit, POTASSIUM und OpenVAS.

Abschnitt	Anforderung	Priorität	MS	PO	VAS
Allgemeine Anforderungen					
3.6.1	Automatischer Start eines Penetrationstests	+	o	o	xx
3.6.2	Importieren und Exportieren von CSV-Dateien	++	xx	xx	xx
3.6.3	Benachrichtigung über die Beendigung eines Tests	+	o	o	xx
3.6.4	Benutzerverwaltung und Administration durch Superusers	+++	o	o	xx
3.6.5	Hierarchische Benutzerrechte innerhalb einer Abteilung	+++	o	o	xx
3.6.6	Mehrbenutzer-Betrieb	+++	o	o	xx
3.6.7	Sicherheit der Anwendung	+++	xx	xx	xx
3.6.8	Mandantenfähigkeit	+++	o	o	xx
3.6.9	Leicht anpassbar und erweiterbar	+++	xx	xx	xx
3.6.10	Die Bedienung muss über eine GUI und API möglich sein	++	xx	o	xx
3.6.11	Schnittstelle zu anderen Sicherheitstools	++	o	o	xx
3.6.12	Automatisierung	+++	xx	xx	x
3.6.13	Testen einer Vielzahl von Webservern	+	xx	xx	xx
3.6.14	Der Penetrationstest einzelner Webserver muss zeitnah beendet werden	+++	xx	xx	xx
Anforderungen zur Informationsbeschaffung					
3.7.1	Erkennung geöffneter Ports und dahinter laufender Dienst	+++	xx	x	xx
3.7.2	Erkennung von eingesetzter Software und deren Versionsstand	+++	xx	xx	xx
3.7.3	Erkennung von bereits erfolgreichen Angriffen	++	o	o	o
Anforderungen zur Informationsbewertung					
3.8.1	Flexible Auswahl durchzuführender Tests	+++	xx	xx	xx
3.8.2	Priorisierung von Testergebnissen	+	o	o	xx
Anforderungen zum aktiven Eindringen					
3.9.1	Verwendung von Exploits für bekannte Schwachstellen aller populären CMS	++	xx	xx	o
3.9.2	Generische Tests	++	o	o	o
3.9.3	Penetration verschiedener Betriebssysteme	+++	xx	xx	o
3.9.4	Keine Beeinträchtigung des Netzes oder der Server	+++	x	x	x
Anforderungen zur Abschlussanalyse					
3.10.1	Dokumentation der Ergebnisse	+++	xx	xx	xx
3.10.2	Errechnen des CVSS Base-Score	+++	xx	xx	xx
3.10.3	CVE-Nummer bekannter Schwachstellen	+++	xx	xx	xx
3.10.4	Exportieren von Berichten in andere Formate	+	o	o	xx

Tabelle 4.1.: Bewertung der Anforderungen an einen Service für Penetrationstests

Wie in Tabelle 4.1 zu sehen ist, eignen sich die soeben vorgestellten Arbeiten nicht vollständig für den in dieser Arbeit zu entwickelnden Service für Penetrationstests. Es wird zum Beispiel die Benutzerverwaltung und die hierarchischen Benutzerrechte nur von OpenVAS unterstützt. OpenVAS hat jedoch den Nachteil, dass es sich dabei um einen Schwachstellenscanner handelt und die entscheidende Komponente der Penetration von Webservern fehlt. Auch die Erkennung von bereits erfolgreichen Angriffen wird durch keine der vorgestellen Arbeiten unterstützt. Zudem ist aufgefallen, dass keine der vorgestellten Arbeiten das aufspüren und ausnnutzen von generischen Schwachstellen unterstützt.

Des Weiteren sind die beiden Tools - Metasploit und OpenVAS - nicht ausreichend als Service konzipiert, sondern mehr als Werkzeug, das bei einem Penetrationstest hilfreich ist. Potassium hingegen wurde als Service Konzipiert, richtet sich aber an Cloudanbieter und bietet somit keine Möglichkeit dedizierte Webserver zu testen.

Eine Erweiterung von OpenVAS wäre unter Umständen möglich, jedoch auch sehr abstrakt. Der Vorteil bei einer neuen Konzipierung des Service für Penetrationstests ist, dass besser auf die Wünsche der Kunden eines IT-Providers eingegangen werden kann. Dies hilft, um den Service auf die Anforderungen der Kunden zu optimieren. Des Weiteren können Vorteile und gute Ansätze, die in den verschiedenen themenverwandten Arbeiten vorgestellt wurden, in das Konzept und die Architektur einfließen. Das Entwickeln eines neuen Konzepts bietet auch einen weiteren Vorteil, dass die Architekur des Service so aufgebaut werden kann, um das Einbinden beliebiger Tools leicht zu gestalten.

5. Konzeptaufbau

Wie bereits beschrieben eignen sich die soeben vorgestellten themenverwandten Arbeiten nicht vollständig für einen Service für Penetrationstests, der am LRZ angeboten werden soll, da sie nicht alle Anforderungen erfüllen. Daher wird im Folgenden ein Konzept für einen Service für Penetrationstest erarbeitet, das die in Kapitel 3 gestellten Anforderungen erfüllt.

5.1. Anwendungsbereich

In diesem Abschnitt wird zunächst der Anwendungsbereich des Service für Penetrationstests beschrieben. Dieser kann je nach Einsatzgebiet, wie beispielsweise ein IT-Provider, flexibel ausgewählt werden. Da in dieser Arbeit ein Service für Penetrationstests am Beispiel des LRZ entwickelt wird, liegt der Fokus jedoch auf dem Einsatz in einem Hochschulrechenzentrum.

In dem hier vorgestellten Konzept soll es ermöglicht werden, dass verschiedene Personengruppen mit unterschiedlichen Rechten einen Penetrationstest so weit wie möglich automatisiert auf deren Webservern durchführen können. Aus diesem Grund werden verschiedene Rollen eingeführt, die sich auch in dem Punkt unterscheiden können, ob es sich um Mitarbeiter des Hochschulrechenzentrums oder um einen Kunden handelt. Beispielsweise bei dem Sicherheitsverantwortlichen gibt es diese Rolle auf der Seite des Rechenzentrums sowie auf der Seite der Kunden.

Weiterhin konzentriert sich dieses Konzept ausschließlich auf Penetrationstests von Webservern und Webanwendungen. Andere Anwendungsgebiete, wie beispielsweise Penetrationstests gegen Router oder Switches, werden in diesem Konzept nicht berücksichtigt. Jedoch soll die Architektur so ausgelegt werden, dass die Tests beispielsweise auch für Netzwerkinfrastrukturen konfiguriert werden können. Eine weitere Begrenzung dieses Konzepts ist die Behebung der gefundenen Schwachstellen, da dies der jeweils zuständigen Personengruppe überlassen wird, ob die Behebung der Schwachstelle als wichtig angesehen wird.

5.2. Rollen

Wie bereits beschrieben, soll der Service von verschiedenen Nutzern mit unterschiedlichen Rechten verwendet werden. Um dies zu realisieren, werden die Benutzer in verschiedene Rollen unterteilt. Die Aufgaben und Rechte, welche die einzelnen Rollen besitzen, werden in den nachfolgenden Abschnitten genauer betrachtet.

5.2.1. Superuser

Bei dem Betrieb eines Service fallen einige administrative Arbeiten an, um ihn auf einen aktuellen Stand zu halten. Diese Arbeiten sollten in regelmäßigen Abständen durchgeführt werden. Für diese administrativen Tätigkeiten wird die Rolle der Superuser eingeführt. Die Aufgaben des Superusers sind nicht nur an eine Person gebunden, sondern können auch von mehreren Mitarbeitern des LRZ übernommen werden. Bei einem sehr großen Kundenstamm könnten somit getrennte Zuständigkeitsbereiche der Superuser geschaffen werden.

Die Aufgabe der Superuser ist, den Betrieb und die Verwaltung des Penetrationstesting Service zu übernehmen. Dazu gehören beispielsweise die Pflege der Benutzerdatenbank und der Scan- und Exploitationwerkzeuge. Für den Aufbau und die Pflege der Benutzerdatenbank kann der Superuser neue Benutzer hinzufügen oder entfernen sowie die Benutzer auf deren zugewiesene Netzsegmente beschränken. Eine weitere Aufgabe des Superusers ist, neu entwickelte Scan- und Exploitationwerkzeuge anzubinden oder alte zu entfernen. Des Weiteren ist

denkbar, dass sich im Laufe der Zeit die Verwendung der Scanwerkzeuge verändert und beispielsweise neue Parameter hinzukommen. In diesem Fall ist es die Aufgabe des Superusers, die verwendeten Tools den neuen Gegebenheiten anzupassen.

Die Mitarbeiter, welche diese Aufgabe übernehmen, müssen selbst keine Penetrationstests durchführen oder Schwachstellen interpretieren. Somit benötigen diese Personen keinerlei Wissen über IT-Sicherheit, jedoch sollten sie gewisse Fähigkeiten im Bereich des Managements mitbringen. Da die Superuser unter anderem Änderungen an der Bedienung der Scanwerkzeuge vornehmen müssen, sollten sie auch eine gewisse Kompetenz in Bezug auf die Bedienung konsolenbasierter Tools aufweisen.

5.2.2. IT-Manager

Der IT-Manager ist für die Ausrichtung und die Aufrechterhaltung der IT-Infrastruktur zuständig und muss einen reibungslosen Betrieb sicherstellen. Des Weiteren fällt die Risikoanalyse und das Entwickeln von Bedrohungsszenarien in seinen Aufgabenbereich, um im Anschluss entsprechende Maßnahmen zum Schutz der Organisation einführen zu können. Daher muss er immer über die Schwachstellen in der IT-Infrastruktur des Unternehmens informiert sein. Der Manager muss dazu nicht jede einzelne Schwachstelle kennen, jedoch muss er einen Gesamtüberblick der Bedrohungen in dem Unternehmen haben. Er muss selbst keine Penetrationstests durchführen und muss auch bei Abschlussgesprächen nicht zwingend teilhaben. Jedoch muss er bei einem Penetrationstest informiert werden und diesen erlauben.

Die Rolle des IT-Managers benötigt keine Funktion zur Durchführung von Penetrationstests und muss daher auch nicht auf IP-Bereiche beschränkt werden. Jedoch soll er Informationen zu dem Ist-Zustand bezüglich der Sicherheit der Webserver im Unternehmen erhalten. Daher ist es nötig, dem Manager ein Dashboard zur Verfügung zu stellen, das eine Zusammenfassung der durch den Service für Penetrationstests gefundenen Schwachstellen bereitstellt.

5.2.3. Sicherheitsverantwortlicher

Der Sicherheitsverantwortliche ist zuständig, für die Sicherheit in einem Unternehmen zu sorgen. Zu seinen Aufgaben zählt, für die Sicherheit laufender Dienste und Anwendungen Sorge zu tragen. Des Weiteren muss er Maßnahmen zur Datensicherheit in einem Unternehmen umsetzen und Notfallpläne bereitstellen. Bei einem Penetrationstest ist er zuständig, diesen zusammen mit dem Penetrationstester zu planen. Nach der Beendigung des Tests muss er an einem Abschlussgespräch teilnehmen. Die Teilnahme an dem Abschlussgespräch ist für ihn wichtig, um geeignete Maßnahmen treffen zu können. Der Sicherheitsverantwortliche muss selbst keine Penetrationstests durchführen, jedoch muss er immer informiert werden, wenn ein Test geplant ist und diesen nach Absprache mit dem IT-Manager gegebenenfalls erlauben.

Ein Sicherheitsverantwortlicher muss aufgrund der Natur seines Jobs gewisse Fähigkeiten mitbringen. So muss er sich im Bereich IT-Sicherheit gut auskennen und in der Lage sein, eine Schwachstelle und ihre potentielle Gefährlichkeit zu erkennen. Des Weiteren muss er in der Lage sein, anhand von Gefährlichkeit und Eintrittswahrscheinlichkeit Risiken abzuschätzen und zu bewerten.

5.2.4. Penetrationstester

Die Aufgabe eines Penetrationstesters ist es, die Webserver und darauf laufenden Webanwendungen zu testen. Dazu muss er zunächst eine Erlaubnis von dem IT-Manager und dem Sicherheitsverantwortlichen einholen. Nach Erhalt dieser Erlaubnis kann er mit der Planung und der Durchführung des Penetrationstests beginnen. Ein Tester benötigt ein fundiertes Wissen über Schwachstellen und deren Behebungsmöglichkeiten, damit er die Information an einen Administrator oder Entwickler weitergeben kann, der diese Schwachstellen im Anschluss behebt. Damit es für die zuständige Person erleichtert wird, die Schwachstelle zu beheben, muss der Penetrationstester eine Empfehlung zur Behebung aussprechen. Die Empfehlung beinhaltet meistens einen Beispiel-Sourcecode wie die Behebung zu realisieren ist.

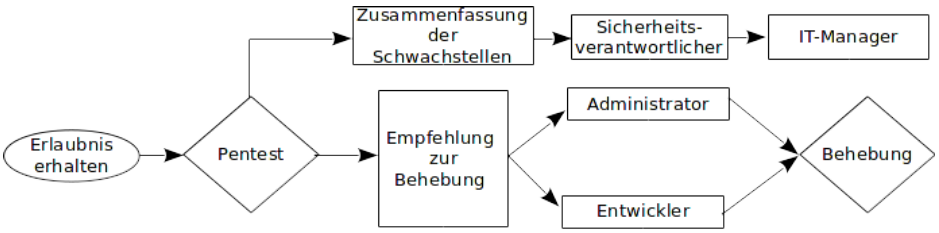


Abbildung 5.1.: Die Arbeitsschritte des Pentesters

Des Weiteren muss der Tester nach Abschluss des Penetrationstests auch den Sicherheitsverantwortlichen über die gefundenen Schwachstellen und deren Gefährlichkeit informieren. Die Arbeitsabläufe des Penetrationstesters sind in Abbildung 5.1 grafisch dargestellt. Ein Tester hat die Möglichkeit, alle betriebenen Webserver innerhalb einer Abteilung, der er angehört, zu testen. Daher beläuft sich die Begrenzung des IP-Bereiches für diese Rolle auf alle der Abteilung zugewiesenen IP-Adressen.

5.2.5. Administrator

Der Administrator ist zuständig, Dienste zur Verfügung zu stellen und diese sicher zu konfigurieren. Damit eine sichere Konfiguration der Dienste möglich ist, muss der Administrator ein Verständnis der Sicherheit von IT-Systemen besitzen. Damit er die IT-Infrastruktur des Unternehmens absichern kann, benötigt er die Information über gefundene Schwachstellen und deren Gefährlichkeit, um angemessen handeln zu können. Er ist auch zuständig, wenn es Störungen in der IT-Infrastruktur gibt und muss diese beheben. Der Administrator muss selbst keine Penetrationstests durchführen, jedoch wäre denkbar, dass er die Möglichkeit hat, einen gezielten Nachtest durchzuführen, um die Wirksamkeit der Änderungen festzustellen. Aus diesem Grund wird die Rolle des Administrators auf die Netzwerksegmente begrenzt, für die er zuständig ist.

5.2.6. Softwareentwickler

Die Aufgabe eines Softwareentwicklers ist das Erarbeiten des Konzepts und die Entwicklung der Software. Außerdem ist er zuständig, bestehende Computerprogramme zu erweitern und an neue Gegebenheiten anzupassen. Es fällt auch in seinen Zuständigkeitsbereich, aufgedeckte Schwachstellen in der von ihm entwickelten Software zu beheben. Der Softwareentwickler sollte ein gutes Verständnis über IT-Sicherheit besitzen, besonders im Bereich der sicheren Softwareentwicklung. Damit er die Möglichkeit hat, die eigene Software auf Sicherheit zu überprüfen, soll auch er Zugriff auf den Service für Penetrationstests erhalten. Jedoch wird die Rolle des Softwareentwicklers auf die IP-Adressen seiner Testserver begrenzt, damit ein Angriff auf andere Webserver, die nicht in seinen Zuständigkeitsbereich fallen, unterbunden wird. Die Ergebnisse des Tests dienen rein dem Softwareentwickler, um die Sicherheit der entwickelten Anwendung zu erkennen. Somit werden die Ergebnisse nicht als Report an andere Rollen wie dem Administrator übergeben.

5.3. Abwägung eines zentral oder verteilt organisierten Service

Eine grundlegende Überlegung ist, wie der Service ausgerichtet werden sollte. Kunden, die diesen Service nutzen, sind Organisationen, die verschiedentlich organisiert sind und dadurch auch einen unterschiedlichen Fokus auf die Sicherheitstests haben. Während eine Organisation auf eine bestimmte Webapplikation setzt, wird von einer anderen Organisation unter Umständen ein anderes Produkt bevorzugt. Daher sollte auch der Fokus der zur Verfügung stehenden Sicherheitstest auf der eingesetzten Software liegen, damit der Katalog der Tests übersichtlich gehalten werden kann. Hierbei könnte ein zentraler Ansatz gewählt werden, bei dem ein einzelner Service vom LRZ betrieben wird, den sich die verschiedenen Organisationen teilen. Bei diesem Ansatz muss darauf geachtet werden, dass die Architektur so ausgerichtet wird, dass die einzelnen Rollen der

Organisation zugeordnet werden können, der sie angehören. Wenn die Benutzer einer Organisation zugeordnet werden können, bietet sich die Möglichkeit nur Sicherheitstests anzuzeigen, die auch benötigt werden.

Eine weitere Möglichkeit ist ein verteilter Ansatz, bei dem für jede Organisation ein eigener Service bereitgestellt wird. Der entscheidende Vorteil bei dieser Variante ist, dass der Kunde administrativen Zugang zu diesem Service erhalten könnte und somit auch in der Lage wäre, selbst entwickelte Sicherheitstests flexibel einzubinden. Des Weiteren könnte der Kunde auch Änderungen am Service vornehmen, um ihn zugeschnitten an die Bedürfnisse der Organisation anzupassen. Wenn jedoch neue Sicherheitstests, die durch das LRZ entwickelt werden, hinzukommen, ist der administrative Mehraufwand nicht zu unterschätzen, da sie erst an die unterschiedlichen Services verteilt und eingepflegt werden müssen.

5.4. Vorbereitungsphase

Eine wichtige Phase, bevor mit einem Penetrationstest begonnen wird, ist die Vorbereitungsphase. Hier müssen zunächst alle Schritte geklärt werden, die in den nachfolgenden Phasen durchgeführt werden. Zu Beginn muss der Tester von dem Sicherheitsverantwortlichen und dem IT-Manager eine Genehmigung für den Penetrationstest einholen. Die Genehmigung ist ein wichtiger Schritt, da ein Penetrationstest einerseits gewisse Risiken bezüglich des reibungslosen Betriebs mit sich bringt und andererseits auch rechtliche Konsequenzen haben kann, wenn dies nicht genehmigt wurde. Des Weiteren muss festgehalten werden, falls es zu Störungen kommen sollte, wer die Verantwortung hat diese zu beheben. Eine Anforderung an den Service ist, dass der reguläre Betrieb nicht beeinträchtigt wird. Jedoch können bei einem Penetrationstest, auch wenn die Tests sorgfältig darauf geprüft wurden keine Beeinträchtigung hervorzurufen, Störungen auftreten. In diesem Fall muss protokolliert werden, welcher Test diese Beeinträchtigung hervorgerufen hat, damit der Grund gefunden und der Test angepasst werden kann.

Nachdem die Genehmigung des Vorgesetzten eingeholt und auch der Verantwortliche bei Störfällen festgelegt wurde, muss der Ablauf sorgfältig geplant werden. Für die Planung sollte es ein Kickoffmeeting mit dem Tester, dem Vorgesetzten und gegebenenfalls weiteren Beteiligten, wie dem Sicherheitsverantwortlichen und dem Administrator geben, damit alle Randbedingungen für den Test geklärt werden können und alle Beteiligten das gleiche Verständnis bezüglich des Testablaufs haben. Bei diesem Kickoffmeeting muss besprochen werden, welche Webserver getestet werden sollen. In einigen Fällen gibt es kritische Systeme, deren potenzielle Beeinträchtigung der Organisation einen großen Schaden zufügt und daher von einem Penetrationstest ausgeschlossen werden. Nachdem festgelegt wurde, welche Webserver einem Test unterzogen werden, muss geklärt werden, auf welche Dienste ein Angriff ausgeführt wird. Hier stellt sich die Frage, ob nur der Webserver und die darauf laufenden Webanwendungen angegriffen werden sollen oder auch weitere Dienste, wie beispielsweise der FTP- oder Datenbankserver. Des Weiteren muss auch besprochen werden, um welche Tests es sich handeln soll. Dabei kann entschieden werden, welche Techniken zum Einsatz kommen. Wird ein kompletter Penetrationstest durchgeführt, der beispielsweise auch eine Brute-Force-Attacke auf den Login der gewählten Dienste beinhaltet oder wird nur versucht, Schwachstellen aufzudecken und diese auszunützen?

Wenn alle Bedingungen geklärt wurden, wie der Test ablaufen soll, beginnt die Überlegung der Platzierung des Penetrationstests. Penetrationstests sollten nach Möglichkeit innerhalb des eigenen Netzes und außerhalb durchgeführt werden. Dies hat den Grund, da in einem Rechenzentrum oft Sicherheitsmaßnahmen getroffen werden, um das eigene Netz von außen, also dem Internet abzuschotten. Oft werden auch Tools wie Intrusion Detection Systeme eingesetzt, die den Netzwerkverkehr analysieren und gegebenenfalls einen Alarm auslösen. Damit ein besserer Überblick geschaffen wird, welche Schwachstellen von einem Innentäter oder einem externen Angreifer entdeckt und ausgenutzt werden können, bietet es sich an, den Penetrationstest von innen und von außen durchzuführen.

Ein weiterer wichtiger Aspekt, der besprochen werden muss, ist, zu welcher Zeit der Penetrationstest gestartet werden soll und wie aggressiv die Tests vorgehen dürfen. Soll ein Webserver getestet werden, der unter Tags von vielen Personen verwendet wird, wäre denkbar, dass der Penetrationstest nicht in dieser Zeit durchgeführt wird, damit die Ressourcen des Servers nicht zu sehr beansprucht werden. Diese Problematik könnte auch dadurch gelöst werden, indem ein Schattensystem verwendet wird. Bei einem Schattensystem wird ein exaktes Abbild des zu testenden Webservers erstellt, damit nicht auf dem Livesystem getestet werden muss. Dieser Ansatz erfordert jedoch mehr Aufwand und wird nicht immer als Alternative gewählt. Für den Fall, dass ein

Kunde diesen Mehraufwand nicht eingehen will, soll der Service die Möglichkeit bieten, die Geschwindigkeit des Tests anzupassen. Hierbei könnte entweder ein aggressiver Ansatz gewählt werden, damit ein Test schnell beendet wird oder auch ein schonender Test, der den Scan langsam durchführt, damit der Server keiner zu hohen Last ausgesetzt ist.

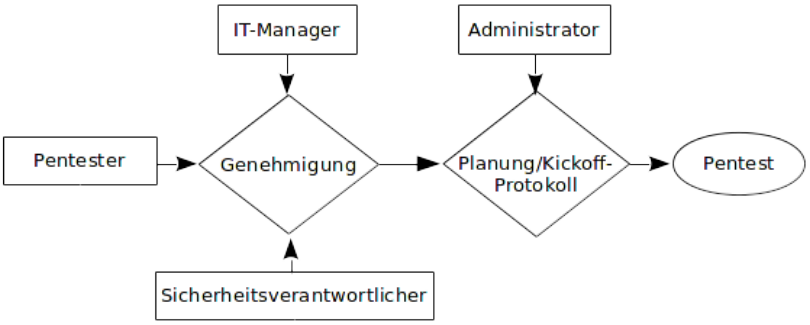


Abbildung 5.2.: Eine Darstellung des Workflows in der Planungsphase

Wurden alle diese Punkte geklärt, muss ein Kickoff-Protokoll erstellt werden, in dem die Rahmenbedingungen des Penetrationstests festgehalten werden. Dieses Kickoff-Protokoll dient einerseits dem Tester als Leitfaden für den Test, andererseits können mit einem festgeschriebenen Dokument auch spätere Vorwürfe, wie ein unerlaubtes Testen einer Anwendung, aus dem Weg geräumt werden. Des Weiteren dient das Kickoff-Protokoll auch der rechtlichen Absicherung von Tester und Vorgesetztem, da dem Tester zu einem späteren Zeitpunkt kein Vorwurf gemacht werden kann, aber auch der Vorgesetzte oder Sicherheitsverantwortliche den Tester zur Rechenschaft ziehen kann, wenn sich dieser nicht an die Vereinbarung hält. Die einzelnen Schritte, die in der Vorbereitungsphase nötig sind, werden in Abbildung 5.2 illustriert.

Damit der Kunde, welcher den Penetrationstest durchführen will, auch in der Vorbereitungsphase unterstützt wird, soll dieser Service eine Vorlage für ein Kickoff-Protokoll bereitstellen, das dann von den beteiligten Personen im Kickoffmeeting ausgefüllt wird. Die Vorlage soll so aufgebaut sein, dass zu jedem Punkt eine kleine Beschreibung vorhanden ist, damit dem Benutzer die Vorbereitung zu einem Penetrationstest erleichtert wird.

5.5. Informationsbeschaffung

Wenn die Vorbereitungsphase abgeschlossen ist und alle Beteiligten über die Rahmenbedingenen informiert sind, kann mit dem nächsten Schritt des Penetrationstests begonnen werden. In dieser Phase kann der Tester den Service nutzen, um detaillierte Informationen und einen guten Überblick über das Zielsystem zu erhalten. Die Informationsbeschaffung besteht aus verschiedenen Arten von Scans. Zuerst wird ein Asset-Discovery-Scan durchgeführt, um im Anschluss den Webserver und darauf laufende Dienste und Webanwendungen gezielt nach Schwachstellen abzusuchen. Damit ein Test nicht gestört wird, sollten auf dem Zielsystem keine Intrusion Prevention Systeme, wie beispielsweise Fail2Ban, aktiviert sein, da diese Tools einen Schwachstellenscan blockieren würden. Bei einem echten Angreifer lassen sich diese Tools jedoch durch die Verwendung verschiedener IP-Adressen umgehen und bieten daher nur bei automatisierten Scans einen besseren Schutz.

Des Weiteren soll der Tester dabei unterstützt werden, das Zielsystem nach bereits erfolgreichen Angriffen abzusuchen, indem nach existierenden Webshells und bestimmten Schlüsselwörtern im Programmcode der Webanwendung gesucht wird. Dazu kann er selbst bestimmte Schlüsselwörter konfigurieren und sich im Anschluss ein Skript erzeugen lassen, das er auf dem Webserver ausführen kann.

5.5.1. Asset-Discovery-Scan

Zu Beginn des Asset-Discovery-Scan wird mittels eines Portscanning-Werkzeugs das Zielsystem nach geöffneten Ports abgesucht, um die ersten Informationen über laufende Dienste und deren Versionsstände zu

erhalten. Dies könnte mit *nmap* realisiert werden, da er ein stabiles Tool für Portscans ist und sehr viele verschiedene Scan-Typen mitbringt. Ein Vorteil bei der Verwendung von *nmap* sind die zahlreichen Skripte der Nmap-Scripting-Engine (NSE), die als Zusatzmodule mitgeliefert werden. Dazu gehört beispielsweise *http-wordpress-enum*, welches die Versionsnummer, Plugins und Themes von WordPress Installationen erkennt. Ein weiteres hilfreiches NSE-Skript ist *http-generator*, das den Inhalt des Generator-Metafelds ausgibt. In diesem Metafeld ist bei vielen CMS der Hersteller und die Versionsnummer hinterlegt. Außerdem können mit *nmap* auch die Ergebnisse in verschiedenen Formaten ausgegeben werden. Hierzu zählt beispielsweise die Ausgabe der Ergebnisse im XML-Format, das sich viel leichter parsen lässt als eine Ausgabe in eine reine Textdatei.

Nmap hat sehr viele Vorteile, jedoch gibt es auch eine Vielzahl anderer Portscan-Werkzeuge, die frei erhältlich sind. Welches Tool für den Portscan und die Versionserkennung verwendet wird ist letztendlich implementierungsabhänging.

5.5.2. Schwachstellenscan

Nachdem die geöffneten Ports und die Versionsstände der verwendeten Dienste und Webanwendungen des Zielsystems ermittelt wurden, können sie auf Schwachstellen überprüft werden. An dieser Stelle werden drei Klassen von Tests durchgeführt. **(1)** Wurde im Asset-Discovery-Scan eine CMS-Installation erkannt, müssen, wie schon in den Anforderungen beschrieben, öffentlich bekannte Schwachstellen in veralteten CMS-Installationen erkannt werden. Gegebenenfalls kann das CMS auch noch mit einem unabhängigen Schwachstellenscan untersucht werden. **(2)** Falls durch den Asset-Discovery-Scan kein CMS erkannt wurde, kann es sich entweder um ein unerkanntes CMS oder um eine selbst entwickelte Webanwendung handeln. In diesem Fall kann der Benutzer entweder einen gezielten Scan manuell auswählen oder es wird ein unabhängiger Schwachstellenscanner ausgeführt, der die Webanwendung nach den typischen Schwachstellen wie SQL-Injection, RFI/LFI etc. absucht. **(3)** Eine weitere Klasse an Tests geht gegen erkannte Dienste wie FTP oder Datenbankserver, die auf der Zielmaschine in Betrieb sind. Bei diesen Tests wird überprüft, ob die Dienste sicher konfiguriert wurden und auch keine Standardpasswörter in Verwendung sind.

Auch hier ist es wichtig, geeignete Scan-Werkzeuge zu finden. So gibt es für die populärsten CMS wie WordPress und Joomla! Schwachstellenscanner, die auf öffentlich bekannte Schwachstellen sowie auf Konfigurationsfehler prüfen. Auch für selbst entwickelte Webanwendungen gibt es eine zahlreiche Auswahl von Scan-Werkzeugen, die eingesetzt werden können, um die Anwendung nach Schwachstellen abzusuchen. Ein sehr leistungsfähiges Tool für diese Zwecke ist *w3af*, welches frei erhältlich ist. Es lässt sich mit Hilfe von Konfigurationsdateien komplett automatisieren und bietet wie auch *nmap* die Ausgabe der Ergebnisse im XML-Format an. Der modulare Aufbau dieses Tools macht es möglich, flexibel verschiedene Schwachstellenscans je nach anzugreifender Anwendung anzusteuern. Somit wäre denkbar, für verschiedene Angriffsszenarien zugeschnittene Konfigurationsdateien bereitzustellen, die von dem Service verwendet werden können.

Da auch laufende Dienste, wie beispielsweise der FTP-Server, auf Schwachstellen überprüft werden sollen, muss zunächst der Hersteller und die eingesetzte Version erkannt werden. Dafür gibt es je nach Dienst verschiedene Techniken, wie bei einem FTP-Server das Überprüfen der Banner. Nachdem der Hersteller und die Version feststehen, kann diese Information mit einer Schwachstellendatenbank abgeglichen werden, um herauszufinden, ob für den betreffenden Dienst eine bekannte Schwachstelle existiert. Nachdem dieser Test abgeschlossen ist, kann je nach Dienst eine Wörterbuchattacke mit schwachen Passwörtern auf das Login des Dienstes durchgeführt werden.

5.5.3. Aggregation der Ergebnisse

Nachdem die Dienste, installierten Webanwendungen und deren Schwachstellen erkannt wurden, müssen die Ergebnisse zusammengefasst und gespeichert werden. Eine Schwierigkeit bei dieser Aggregation stellt die Verwendung verschiedener Scan-Werkzeuge dar. Da die verwendeten Tools auch unterschiedliche Ergebnisse produzieren, müssen die Daten auf eine gemeinsame Struktur gebracht werden, damit sie in der nächsten Phase dem Benutzer in einer übersichtlichen Form angezeigt werden können. Dazu werden zunächst die gesammelten Informationen in vier Kategorien unterteilt und analysiert, welche Informationen für den Tester, aber auch

für einen aktiven Einbruchsversuch wichtig sind. Die Informationen, die von den verschiedenen Scan-Typen benötigt werden, um sie in der nächsten Phase bewerten zu können, sind in Tabelle 5.1 zusammengefasst.

Eine weitere Schwierigkeit ist, dass der Service bei späterer Verwendung leicht erweiterbar sein soll. Um dies zu realisieren, wird für jedes verwendete Scan-Werkzeug ein eigenes Reporting-Modul entwickelt, das für die Aggregation der Ergebnisse in einer einheitlichen Form zuständig ist. Diese Module können dann flexibel an das Scan-Modul angebunden, erweitert oder ersetzt werden, wenn sich die Ausgabe der verwendeten Tools in späteren Versionen verändern sollten.

Wichtige Informationen eines Asset-Discovery-Scan
Eine Liste geöffneter Ports
Eine Liste laufender Dienste sowie der Port hinter dem sie laufen
Herstellerinformation der Dienste
Versionsstand der Dienste
Installierte Webapplikationen
Versionsstand der installierten Webapplikationen
Wichtige Informationen bei bekannten Schwachstellen in CMS
CMS Titel
Schwachstellenbezeichnung
Exploit-ID (interne Kennung des Exploit)
Wichtige Informationen bei anderen Schwachstellen
Schwachstellenbezeichnung
Schwachstellen-ID (interne Kennung der Schwachstelle)
Methode (GET/POST)
Verwundbare Parameter
Angreifbare URL
URLs in denen die Schwachstelle gefunden wurde
Sensitive Daten (z.B gefundene Backups wie config.php.bak)
Wichtige Informationen bei bekannten Schwachstellen in laufenden Diensten
Dienstname
Port hinter dem der Dienst läuft
Schwachstellenbezeichnung
Exploit-ID (interne Kennung des Exploit)
Sensitive Daten (bspw. gefundene Passwörter)

Tabelle 5.1.: Eine Liste wichtiger Informationen der verschiedenen Scan-Typen

Da die Ergebnisse dem Benutzer zu einem späteren Zeitpunkt übersichtlich angezeigt werden sollen, ist eine Speicherung der Ergebnisse in einer reinen Textdatei ungeeignet. Viel mehr müssen sie in einer strukturierten Form gespeichert werden, sodass sie leicht ausgelesen werden können. Um dies zu realisieren bieten sich Formate wie XML oder JSON an, da die gesammelten Informationen aufgrund der Struktur der Formate leicht geparst und weiterverarbeitet werden können.

5.6. Informationsbewertung

Dem Tester liegen nach der Phase der Informationsbeschaffung detaillierte Ergebnisse vor, welche Dienste und Anwendungen auf dem Zielsystem installiert sind und welche Schwachstellen auf dem Zielsystem existieren. Diese Information muss in dieser Phase dem Benutzer angezeigt und durch den Service nach verschiedenen Kriterien bewertet werden. Wie schon durch die OWASP top 10 beschrieben, gibt es eine Vielzahl von Schwachstellen, wovon sich aber nicht alle für eine Penetration des Zielsystems eignen. Daher müssen die gefundenen Schwachstellen nach Gefährlichkeit und Ausnutzbarkeit bewertet werden. Diese Bewertung fällt bei öffentlich bekannten Schwachstellen leicht, da hier meist ein Exploit existiert, der bei einem aktiven Einbruchsversuch

lediglich ausgeführt werden muss. Wenn es sich aber um eine nicht bekannte Schwachstelle wie einer SQL-Injection in einer selbst entwickelten Webanwendung handelt, fällt die Bewertung schon schwerer. In diesem Fall benötigt der Service Zugriff auf eine Datenbank oder eine Datei auf der Festplatte, in der Schwachstellen vorab nach Gefährlichkeit bewertet wurden. Eine Datenbank, die bewertete Schwachstellen beinhaltet, würde sich jedoch besser anbieten, da eine Suche und Filterung der Datensätze leichter zu realisieren sind.

Dem Tester sollen alle gesammelten Informationen zur Verfügung gestellt werden, auch wenn es sich um eine nicht direkt ausnutzbare Schwachstelle wie beispielsweise XSS handelt. Nicht direkt ausnutzbar in diesem Kontext bedeutet, dass die Schwachstelle für ein aktives Eindringen in das Zielsystem nicht geeignet ist, jedoch in Kombination mit anderen Schwachstellen oder gutgläubigen Benutzern einen Angriffspunkt bieten kann. Da es nicht möglich ist, diese Schwachstellen automatisiert auszunutzen, müssen sie dem Tester dennoch angezeigt werden, damit er sie manuell überprüfen kann.

Damit es dem Benutzer erleichtert wird, die erlangten Informationen auch über diesen Service hinaus bewerten zu können, sollen an dieser Stelle der CVSS Base Score berechnet werden, um dem Nutzer diesen anzuzeigen. Mit Hilfe des CVSS Base Score kann der Tester schon vor dem aktiven Eindringen in das Zielsystem die Gefährlichkeit einer Schwachstelle abschätzen. Falls es sich um eine öffentlich bekannte Schwachstelle handelt, soll zusätzlich die CVE-Nummer bereitgestellt werden, damit der Tester eine genauere Recherche zu der gefundenen Schwachstelle durchführen kann. Die CVE-Nummern bekannter Schwachstellen werden üblicherweise in Datenbanken eingetragen, aus denen man sie zu einem späteren Zeitpunkt auslesen kann. Jedoch geben viele Schwachstellenscanner wie *WPScan* diese Nummern bereits beim Auffinden der Schwachstelle zurück, wodurch das Beziehen dieser Information leichter fällt.

5.7. Aktive Eindringversuche

Nachdem die Informationen über die auf dem Zielsystem laufenden Dienste, Anwendungen und deren Schwachstellen gesammelt und bewertet wurden, kann versucht werden, automatisiert in das Zielsystem einzudringen. Hierzu soll der Tester flexibel aus einem Katalog der gefundenen Informationen auswählen können, welche Komponente auf dem Zielsystem anzugreifen ist. Der Tester kann in dieser Phase einstellen, wie weit der Penetrationstest gehen darf. Dabei wäre beispielsweise möglich, bei einer gefundenen SQL-Injection nur bestimmte Teile der Datenbank auszulesen, um zu erkennen, ob der Injection-Angriff Wirkung zeigt. Der Tester könnte aber auch einstellen, dass mit Hilfe der SQL-Injection eine Webshell auf die Festplatte des Zielsystems geschrieben wird, falls die Schreibrechte auf das Dateisystem des Zielrechners nicht ordungsgemäß konfiguriert wurden.

Bei den meisten Angriffsszenarien will ein Angreifer ein Programm in das Zielsystem einschleusen, womit er Systemkommandos ausführen kann, damit er im weiteren Schritt versuchen kann, die Rechte eines unprivilegierten Benutzers zu erweitern. Das Erweitern der Privilegien ist jedoch nur in Ausnahmefällen automatisierbar, wenn beispielsweise auf dem Zielsystem ein veralteter Kernel im Einsatz ist und für diese Schwachstelle ein Exploit existiert. Daher soll der Service für Penetrationstests den Tester bis zum aktiven Eindringen in das Zielsystem unterstützen. Eine weitere Untersuchung des jeweiligen Betriebssystems, ob eine Privilegienerweiterung möglich ist, soll vom Tester manuell durchgeführt werden.

Wenn von dem Service ein aktiver Einbruchsversuch ausgeführt werden soll, stellt sich die Frage, wie eine erfolgreiche Penetration des Zielsystems überprüft werden kann. Eine relativ einfache Lösung ist eine Datei mit einer zufälligen Zeichenfolge einzuschleusen und zu überprüfen, ob diese auch korrekt auf die Festplatte des Zielsystems geschrieben wurde. Dies könnte jedoch zu einem Problem werden, wenn der Pfad zu dem Root-Verzeichnis des Webservers nicht bekannt ist oder die Datei von der Webanwendung in einen unbekannten Ordner geschrieben wird. Damit in diesem Fall überprüft werden kann, ob die Datei geschrieben wurde, muss der Service die Webseite nach oft verwendeten Ordnerstrukturen durchsuchen, um somit die Datei zu finden. Eine weitere Möglichkeit ist, eine Backconnect Shell auf das Dateisystem des Zielsystems zu schreiben und auszuführen. Diese Lösung erfordert aber, dass der Tester auf seinem Computer beispielsweise *netcat* installiert hat, das auf eingehende Verbindungen hört. Bei dieser Lösung kann aber vonseiten des Service nicht erkannt werden, ob die Penetration erfolgreich war. Eine dritte und vermutlich letzte Möglichkeit ist im Payload des Exploit ein Programm auszuliefern, das eine Anfrage zurück an den Service sendet. Da es aber nicht durch jede Schwachstelle möglich ist, Systemkommandos auszuführen, muss je nach auszunutzender

Schwachstelle ein passendes Verfahren verwendet werden. Wichtig hierbei ist, dass geschriebene Dateien nach dem Penetrationstest gelöscht werden, damit der Webserver keiner weiteren Gefahr ausgesetzt ist.

Eine weitere Überlegung beim aktiven Eindringen in ein System ist, welche Schwachstellen sich besonders gut dafür eignen. Aus dieser Überlegung können dann gezielt Tools ausgewählt werden, die eine Penetration des Zielsystems unterstützen. Der einfachste Fall ist eine öffentlich bekannte Schwachstelle, für die ein Exploit existiert. In diesem Fall werden keine weiteren Tools benötigt, da ein Exploit in der Regel alles beinhaltet, damit in ein System eingedrungen werden kann. Da der Service für Penetrationstests jedoch auch Schwachstellen erkennt, für die kein Exploit im Umlauf ist, werden spezielle Tools benötigt. Welche Schwachstellen sich ganz besonders für eine automatisierte Penetration eignen, zeigt die folgende Liste.

- SQL-Injection
- RFI/LFI
- Unrestricted File Upload
- Schwache Passwörter für Dienste wie FTP oder Datenbankserver

Für eine SQL-Injection kann beispielsweise das unter Penetrationstestern sehr bekannte Tool *SQLMap* eingesetzt werden, um Datenbanken auszulesen und zu versuchen, Dateien auf die Festplatte des Zielsystems zu schreiben. Eine RFI/LFI Schwachstelle lässt sich gewöhnlicherweise sehr leicht ausnutzen, wenn sie einmal gefunden wurde. Für diesen Schwachstellentyp kann *Fimap* oder auch ein selbst entwickeltes Tool zum Einsatz kommen. Falls Schwachstellen an einem Webserver erkannt wurden, die von dem Service nicht automatisch ausgenutzt werden konnten, muss durch den Penetrationstester zusätzlich ein manueller Nachtest durchgeführt werden, um die Schwere der Schwachstelle besser einschätzen zu können.

5.8. Abschlussanalyse

In der Phase der Abschlussanalyse muss der Benutzer über die Beendigung des Tests benachrichtigt und die Ergebnisse aus den vorherigen Phasen zu einem Report zusammengefasst werden. Dabei müssen alle gefundenen Schwachstellen sowie erfolgreich ausgenutzte Angriffsvektoren in den Abschlussbericht aufgenommen werden. Es sollen zudem für jede Schwachstelle eine Beschreibung, CVSS Base Score und wenn vorhanden die CVE-Nummer mit aufgenommen werden. Des Weiteren sollen auch Lösungsmöglichkeiten in dem Report enthalten sein, damit festgehalten wird, welche Schritte für eine Absicherung des Webservers nötig sind. Die Struktur und Inhalte des Abschlussberichts wurden aus dem Penetration Testing Execution Standard (PTES) entnommen [PTS12]. Nach diesem Standard soll sich der Bericht in zwei Hauptabschnitte zusammensetzen - der allgemeinen Zusammenfassung und den technischen Details. Die Struktur ist wie folgt aufgebaut:

Allgemeine Zusammenfassung:

- **Hintergründe:** Warum wurde der Penetrationstest gemacht und was verspricht sich die Organisation davon? Welche Tests wurden durchgeführt?
- **Risiko Ranking:** Welche Risikoklassen gibt es? Der Standard schlägt extrem, hoch, mittel, moderat und gering vor.
- **Allgemeine Funde:** In diesem Abschnitt werden grob die gefundenen Schwachstellen aufgelistet.
- **Empfehlungen:** Hier werden der Organisation Empfehlungen zum Absichern ihrer Webserver ausgesprochen.

Technische Details

- **Einleitung:** Die Einleitung beinhaltet, wer involviert war, die Kontaktinformation der Beteiligten was getestet wurde und wie getestet wurde.
- **Informationsbeschaffung:** Wie wurden Informationen beschafft und welche Tools wurden dazu verwendet?

- **Schwachstellenbewertung:** Was wurde gefunden und wie schwer werden diese Schwachstellen eingeschätzt? In diesem Abschnitt kann der CVSS Base Score der Schwachstellen aufgelistet werden.
- **Exploitation:** Welche Attacken wurden durchgeführt und welche Tools wurden dazu verwendet? Waren die Attacken erfolgreich?
- **Risikobewertung:** In diesem Abschnitt wird bewertet wie hoch das Gesamtrisiko für die Organisation ist.

Die Berichte sollen jeweils für die verschiedenen Rollen generiert werden. Dabei erhält der Tester den detailreichsten Bericht, den er mit dem Sicherheitsverantwortlichen in einem Abschlussgespräch besprechen muss. Auch für den Entwickler und den Administrator werden Berichte generiert. Dabei enthält der Entwicklerbericht Informationen zu gefundenen und ausgenutzten Schwachstellen. Der Bericht für den Administrator enthält zudem noch Informationen zu gefundenen Konfigurationsfehlern, wie zum Beispiel wenn bei einem Webserver ungewollt das Directory Listing aktiviert ist.

Nachdem die Berichte generiert wurden, soll dem Nutzer die Möglichkeit geboten werden, flexibel Berichte in verschiedenen Formaten exportieren zu können. Während manche Benutzer einen übersichtlichen und gut leserlichen Bericht in beispielsweise PDF-Format bevorzugen, will ein anderer Nutzer den Bericht im Anschluss weiterverarbeiten und wünscht sich daher einen durch ein Programm auslesbaren Bericht in XML- oder JSON-Format. Welche Formate von den jeweiligen Benutzern gewünscht sind, soll flexibel konfigurierbar sein.

5.9. Architektur

Da die einzelnen Schritte eines Penetrationstests, die von dem Service angeboten werden, erläutert wurden, kann in diesem Abschnitt auf die Architektur eingegangen werden. Da die spätere Erweiterbarkeit als wichtige Anforderung an den Service gestellt wurde, soll ein modularer Ansatz, angelehnt an die des OpenVAS Schwachstellenscanners, gewählt werden. Dieser hat den Vorteil, dass zu einem späteren Zeitpunkt flexibel neu entwickelte Scan- und Exploit-Werkzeuge hinzugefügt oder entfernt werden können, ohne Änderungen an dem Server vornehmen zu müssen. Eine weitere Anforderung an den Service ist der Mehrbenutzerbetrieb, da es denkbar ist, dass der Service von vielen Benutzern zur gleichen Zeit verwendet wird. Damit dies möglich ist und die Benutzer nicht warten müssen bis ein anderer Penetrationstest beendet ist, wurde die Architektur so aufgebaut, dass die Scan- und Exploitkomponenten redundant aufgebaut werden können und der Service somit auch mit vielen gleichzeitigen Penetrationstests gut skaliert. Eine abstrakte Sicht auf den Aufbau der Architektur wird in Abbildung 5.3 grafisch dargestellt. Im Folgenden wird nun auf die einzelnen Komponenten näher eingegangen.

5.9.1. Server

Der Server ist die zentrale Komponente in dieser Architektur und kümmert sich um die komplette Steuerung des Penetrationstests. Zur Speicherung der Daten ist der Server an eine Datenbank angebunden, die alle wichtigen Informationen zu Kunden, laufenden Tests und den Speicherpfad der Ergebnisse enthält. Eine detaillierte Beschreibung des Datenmodells kann in Abschnitt 6.6 nachgelesen werden. Der Server setzt sich aus mehreren Komponenten zusammen, die in diesem Abschnitt beschrieben werden.

Benutzeroberfläche

Die Benutzer können sich mittels des Webinterfaces mit ihren Zugangsdaten direkt an den Server anmelden und ihre Penetrationstests planen und steuern. Es wird zusätzlich die Möglichkeit geboten, den Server über eine API anzusteuern. Dies bietet dem Benutzer eine gewisse Flexibilität, einen Penetrationstest bequem mittels einer grafischen Benutzeroberfläche durchzuführen, aber auch eigene Skripte zu erstellen, um einen Test automatisiert anzustoßen. Bei der Verwendung der API muss sich der Benutzer gegenüber dem Server authentifizieren und die nötigen Informationen für einen Penetrationstest übermitteln. Der Server prüft daraufhin die Gültigkeit des

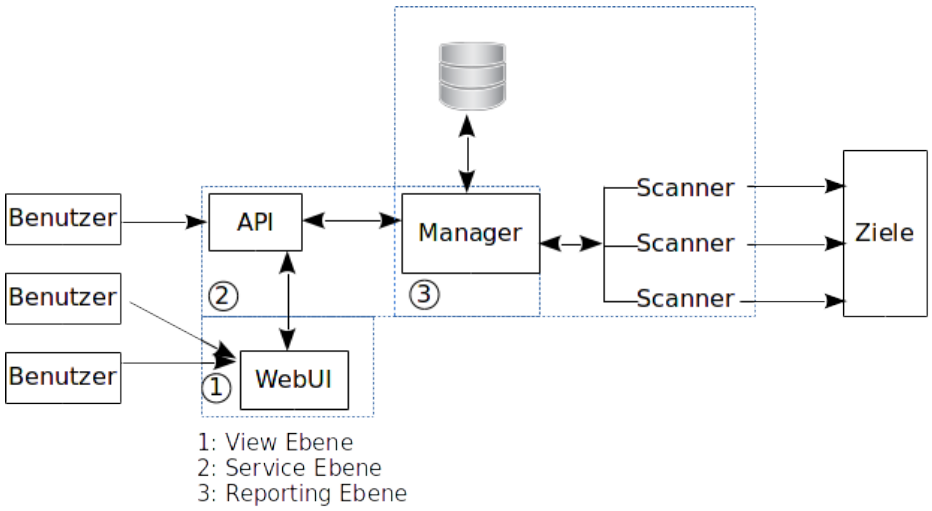


Abbildung 5.3.: Eine grafische Darstellung der Architektur des Service

Benutzerkontos und dessen Rechte und führt bei Übereinstimmung den Penetrationstest durch. Als Rückgabe erhält er je nach Konfiguration einen Report in XML oder JSON Format. Die Kommunikation zwischen Benutzer und Server soll stets über eine gesicherte SSL-Verbindung ablaufen. Die benötigten Informationen die der Nutzer an den Server übermitteln muss sind:

- Benutzername/Passwort
- URL oder IP-Adresse des Zielsystems
- Aggressivität des Tests
- Login Information
- Art des Test (Schwachstellenscan oder Eindringversuche)
- Platzierung des Tests (von innen oder von außen)
- Scan-Typen
- Ausgabeformat der Ergebnisse (XML/JSON)

Die API kann auch dazu verwendet werden, andere Sicherheitstools wie beispielsweise Dr. Portscan an den Service anzubinden, um somit in der Lage zu sein, automatisierte Schwachstellenscans auf Servern durchzuführen, die einen für Webserver typischen Port geöffnet haben. Da von anderen Sicherheitstools keine automatischen Eindringversuche ausgeführt werden sollten, wird durch den Service lediglich ein Schwachstellenscan angeboten. Dies hat den Grund, dass ein voller Penetrationstest stehts überwacht werden sollte und dies bei einem automatischen Anstoßen eines Tests nicht gegeben ist. Daher wird für die Anbindung anderer Tools ein eigener Endpunkt angeboten, der folgende Informationen benötigt:

- Benutzername/Passwort (z.B. eines zentralen Administrators)
- URL oder IP-Adresse des Zielsystems
- Login Information
- Platzierung des Tests (von innen oder von außen)
- Scan-Typen, wie Portscan und Schwachstellenscan

Wenn bei einem Scan, der beispielsweise durch Dr. Portscan angestoßen wurde, Schwachstellen erkannt werden, soll der zentrale Administrator eine Benachrichtigung per Email bekommen, die ihn über die Existenz neuer Ergebnisse informiert. Dieser kann sich im Anschluss an der Benutzerschnittstelle anmelden und den Bericht einsehen.

Manager

Bei einer großen Anzahl von Benutzern, die diesen Service verwenden, müssen die Scans verschiedener Zielsysteme verwaltet werden. Diese Verwaltung wird von dem Manager übernommen, der die Aufträge der Benutzer auf die Scanner verteilt. An dieser Stelle wird auch überprüft, ob der Benutzer die Rechte besitzt, das betreffende Zielsystem anzugreifen. Hierzu wird die IP des anzugreifenden Webservers mit den in der Datenbank hinterlegten IP-Bereichen abgeglichen.

Damit die Kommunikation zwischen den beiden Komponenten nicht mitgelesen werden kann, soll eine SSL-Verschlüsselte Verbindung aufgebaut werden. Für die Kommunikation zwischen Manager und Scanner gibt es grundlegend zwei Kommunikationsmodelle - den synchronen- oder asynchronen Ansatz. Bei einer synchronen Kommunikation bleibt die Verbindung zwischen den Komponenten bestehen bis der Test durchgelaufen ist und blockiert somit den anfragenden Server. Bei einer asynchronen Variante wartet der Server nicht auf eine Antwort und kann sich um andere Anfragen der Nutzer kümmern. Wenn der Scanner die Arbeit verrichtet hat, kann der Server die Ergebnisse abholen und dem Benutzer anzeigen. Dies zeigt, dass bei diesem Service, der von verschiedenen Benutzern verwendet wird, ein asynchroner Ansatz zur Kommunikation besser geeignet ist.

Die Kommunikation zwischen Manager und Scanner beginnt, wenn ein Scanauftrag bereit steht. In diesem Fall sucht der Manager in der Datenbank nach einem Scanner und fragt dessen Status ab. Wenn der Scanner zu diesem Zeitpunkt neue Aufträge annimmt, wird eine Verbindung aufgebaut. Nachdem die Verbindung erfolgreich aufgebaut wurde, übermittelt der Manager alle wichtigen Informationen, die für die jeweilige Aufgabe nötig sind. Nach der Übermittlung wird in einem festgelegten Intervall angefragt, ob die Aufgabe fertiggestellt wurde. Falls dies der Fall ist, kann der Manager die Resultate abholen und auf der Festplatte speichern. Für den Fall, dass ein Scanner wegen eines Fehlers ausfällt und eine Verbindung nicht möglich ist, wird dies in die Datenbank eingetragen und ein anderer Scanner gewählt. Falls ein Scanner ausgefallen ist, wird diese Information per Email an einen der Superuser übermittelt, damit er prüfen kann, aus welchem Grund der Scanner nicht online ist und gegebenenfalls weiter Schritte zur Behebung des Fehlers vornehmen kann. Nachdem der Fehler behoben wurde, muss der Superuser den Scanner in der Datenbank wieder als online markieren.

5.9.2. Scanner

Die Scanner-Komponente hat Zugriff auf alle verfügbaren Scan-Module, die sie nach Bedarf einbinden kann, um einen Scan durchzuführen. Die zur Verfügung stehenden Module sind in der Datenbank hinterlegt und werden dem Scanner von dem Manager übermittelt. Dieser modulare Aufbau ist hilfreich, wenn der Service zu einem späteren Zeitpunkt mit neuen Scan-Modulen erweitert werden soll, da hierfür nur die neuen Module entwickelt werden müssen, aber keine Änderungen an der Scanner-Komponente nötig ist.

Bekommt der Scanner von dem Manager einen neuen Auftrag übermittelt, wird ein neuer Thread gestartet, in dem die durch den Benutzer gewählten Scan-Typen nach der Reihe ausgeführt werden. Wie viele Threads ein Scanner zur gleichen Zeit ausführen kann, wird von den Superusern je nach zur Verfügung stehender Ressourcen des betreffenden Servers konfiguriert. Damit der Service auch mit einer großen Anzahl von Benutzern skaliert, wurden die Scanner nach einem verteilten Ansatz konzipiert, sodass sie auch auf mehreren Servern platziert werden können. Dies hat auch noch weitere Vorteile, da die Scanner so auch außerhalb des Netzes platziert werden können, um ein Scan der Zielsysteme von außen zu realisieren. Die Systemarchitektur, wie die Scanner platziert werden können, ist grafisch in Abbildung 5.4 dargestellt. Ein großer Vorteil dieser Architektur ist, dass auch Penetrationstests in Organisationen durchgeführt werden können, die von außen komplett abgeschottet sind. Hierzu kann der Penetesting-Server inklusive der Scannerkomponenten innerhalb der Organisation aufgebaut werden. Es ist also nicht zwingend nötig den Pentesting-Server alleine am LRZ anzubieten, sondern kann auch in verschiedenen Organisationen betrieben werden.

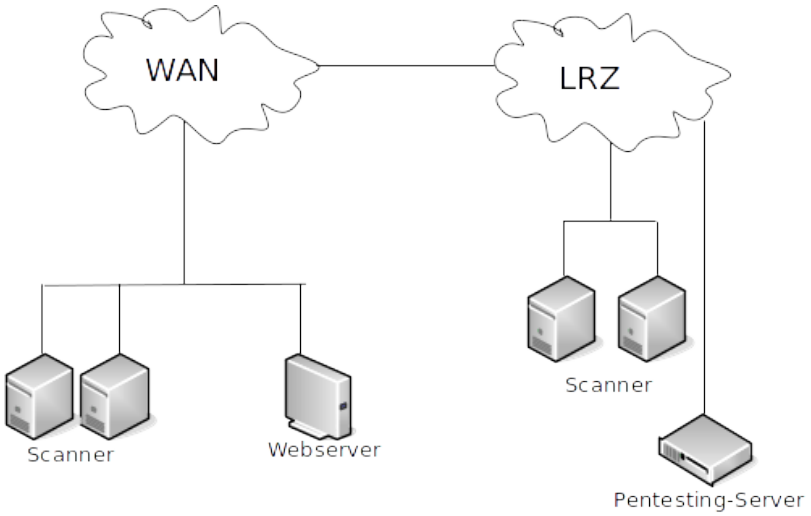


Abbildung 5.4.: Eine grafische Darstellung der Systemarchitektur

Wie bereits beschrieben, kommuniziert der Manager über eine sichere SSL-Verschlüsselte Verbindung mit dem Scanner, um das Mitlesen der Datenübertragung zu unterbinden. Damit der Scanner nicht von beliebigen Personen angesprochen werden kann und somit übernommen wird, muss sich der Server mit einem SSL-Zertifikat gegenüber dem Scanner authentifizieren.

Nachdem alle nötigen Informationen übermittelt wurden, arbeitet der Scanner selbstständig alle Scans ab und gibt die Resultate auf Anfrage zurück an den Manager, der diese für die weitere Verwendung auf der Festplatte speichert. Der Pfad zu den Resultaten wird dann in der Datenbank hinterlegt, damit sie zu einem späteren Zeitpunkt wieder gefunden werden. Nach Abschluss des Scans und Übermittlung der Ergebnisse an den Manager werden alle gesammelten Informationen von der Festplatte des Scanners gelöscht und auf neue Aufträge gewartet.

6. Implementierung

Da das Konzept aus Kapitel 5 erläutert wurde, kann im Folgenden auf die Implementierung näher eingegangen werden. Dabei wurden die wichtigsten Teile eines Service für Penetrationstests implementiert. In diesem Kapitel werden zunächst die Implementierungsgrundlagen beschrieben und im weiteren Verlauf genauer auf die Implementierung der einzelnen Phasen eines Penetrationstests und die Datenhaltung eingegangen.

6.1. Implementierungsgrundlagen

Für die Implementierung des Service für Penetrationstests wurde die Entscheidung getroffen, eine GUI auf Basis einer Webanwendung mit einer zusätzlich verfügbaren API zu implementieren. Diese Entscheidung stützt sich auf die Ergebnisse der Umfrage, da hierbei von den meisten Teilnehmern eine Benutzeroberfläche im Browser gewünscht war. Dies hat auch den Vorteil, dass die Benutzer des Service keine zusätzliche Anwendung auf ihrem Computer installieren müssen, sondern ihre Penetrationstests direkt über den Browser steuern können. Es wurde trotzdem entschieden, für den Service auch eine API anzubieten, die es ermöglichen soll, weitere Benutzeroberflächen beispielsweise als Desktopanwendung zu implementieren und an den Service anzubinden. Somit entsteht eine bessere Flexibilität, die Steuerung des Service weiter auszubauen. Des Weiteren kann die API auch dazu verwendet werden, dass andere Sicherheitstools mit dem Service verbunden werden können.

Als Programmiersprache wurde auf Python gesetzt, da diese Sprache leicht erlernt werden kann und auch unter Entwicklern weit verbreitet ist. Dies zeigt auch eine Umfrage, die von *Stackoverflow*, einem unter Entwicklern sehr bekannten Internetforum, durchgeführt wurde [STA17]. Da der Service bei einem späteren Einsatz im LRZ oft mit neuen Scan-Werkzeugen erweitert werden soll, wurde darauf geachtet, dass die gewählte Programmiersprache von vielen Entwicklern beherrscht wird und sich somit leicht Personen finden lassen, die den Service erweitern können. Ein Vorteil von Python ist zudem, dass der Programmcode gut leserlich ist und eine Anbindung anderer Programmiersprachen wie C++ leicht möglich ist. Des Weiteren gibt es bei Python zahlreiche externe Klassen, die als Module nachinstalliert werden können. Die große Auswahl der Module erleichtert die Implementierung eines Projekts, da dem Entwickler somit viel Programmierarbeit abgenommen wird.

Zur Implementierung des Service wurden verschiedene Python-Module ausgewählt, wie beispielsweise ein Web-Framework oder ein Template-Modul zur Generierung des Kickoff-Protokolls und der Berichte. Die Auswahl der Python-Module und anderer benötigter Software wird im Folgenden kurz beschrieben.

6.1.1. Django

Django ist ein in Python geschriebenes Web-Framework, das unter einer OpenSource Lizenz steht. Es folgt dem in der Softwareentwicklung bekannten Model-View-Controller-Konzept, welches das Modell und die Anzeige strikt auseinander hält. Dies realisiert Django mit der View-Komponente, die den Python Programmcode enthält und das dynamische Verhalten der Webseite steuert. Für die Anzeige besitzt Django eine HTML-Engine, welche HTML-Templates verwendet, um die Inhalte der Webseite dem Benutzer anzuzeigen. Diese HTML-Templates enthalten neben den HTML-Tags Platzhalter, die mit dynamischen Inhalten gefüllt werden können. Zudem können in den HTML-Templates auch bedingte Anweisungen und Schleifen verwendet werden, um die Anzeige der Inhalte dynamisch steuern zu können.

6.1.2. Django REST Framework

Bei dem Django REST Framework handelt es sich um ein Toolkit, das die Implementierung einer Web-API unterstützt. Representative State Transfer (REST) ist ein Architektur Modell, das beschreibt, wie Webstandards für Webservices eingesetzt werden können.

Da der Service auch eine API anbieten soll, wurde für die Implementierung das Django REST Framework verwendet. Dies bot sich an, da auch schon Django als Web-Framework für diesen Service gewählt wurde. Somit konnte die GUI im Browser und die API komfortabel in einem Pythonskript entwickelt werden.

6.1.3. Py3o.template

Das Python-Modul Py3o.template, das für die Implementierung des Service verwendet wurde, ist ein Modul zum Erstellen von Berichten mittels eines vordefinierten OpenOffice Dokuments. Der Vorteil dieses Moduls ist, dass zum Erstellen von Berichten OpenOffice nicht installiert sein muss und somit keine weiteren Softwarepakete benötigt werden. Zudem ist es plattformunabhängig einsetzbar.

Das Projekt sagt von sich selbst, dass es von dem Python-Modul *relatorio* inspiriert wurde, sich jedoch mehr auf die OpenOffice Formate OpenDocument Textformat (ODT) und OpenDocument Spreadsheet (ODS) konzentriert. Damit die Informationen, die in dem Bericht aufgenommen werden sollen, richtig platziert werden können, muss zunächst eine Vorlage eines OpenOffice Dokuments erstellt werden, das die Struktur des Reports, also die einzelnen Abschnitte des später zu erstellenden Berichts, widerspiegelt. In dieser Vorlage können Platzhalter eingefügt werden, die zu einem späteren Zeitpunkt dynamisch mit Inhalten befüllt werden. Dies eignet sich gut für einen Service für Penetrationstests, da hier die Struktur des Berichts vorab klar ist und somit leicht eine Vorlage bereitgestellt werden kann. Es eignet sich auch gut für die Erstellung der Kickoff-Protokolle und kann somit für alle Reports und Protokolle verwendet werden, die für einen Penetrationstest nötig sind.

Mit Py3o.template lassen sich aus den OpenOffice Dokumenten auch andere Formate wie PDF erstellen. Jedoch muss in diesem Fall ein Server mit einer OpenOffice oder LibreOffice Installation aufgesetzt werden, der den Py3o.renderserver installiert hat. Für eine leichte Bereitstellung eines Renderservers bietet das Projekt auf deren Homepage ein Docker-Image zum Herunterladen an. Dieses Image enthält einen fertig installierten und eingerichteten Renderserver, der nur noch in Docker importiert werden muss.

6.1.4. Python-Nmap

Python-Nmap ist ein Modul für Python das verwendet werden kann, um mit wenig Aufwand Portscans mit dem Tool *Nmap* durchzuführen. Es kann flexibel anhand bestimmter Standardparameter genutzt werden. Außerdem bietet es auch die Möglichkeit, einen Befehl mit selbst definierten Parametern auszuführen.

Python-Nmap kümmert sich vollständig um das Parsen der Ergebnisse und gibt dem Benutzer so leicht Zugriff auf die Funde. Ein weiterer Vorteil dieses Moduls ist, dass es ermöglicht wird asynchrone Scans durchzuführen, indem eine Callback-Funktion bereitgestellt wird, welche die Ergebnisse von Nmap empfängt und weiterverarbeitet. Damit wird ermöglicht, eine Vielzahl von Systemen asynchron zu scannen. Die Ergebnisse werden dann etappenweise an die definierte Callback-Funktion zurückgeliefert.

6.1.5. Web Application Audit and Attack Framework

Für die Schwachstellenscans wurde das Tool Web Application Audit and Attack Framework (Wa3f) verwendet. Dieses Tool ist in Python geschrieben und besitzt einen modularen Aufbau hinsichtlich der Angriffsvektoren. Dabei wird für jeden Schwachstellentyp ein eigenes Modul angesteuert, das den Scan daraufhin ausführt. Dieser Aufbau ermöglicht es, leicht weitere Scantypen hinzuzufügen.

Die Entscheidung, dieses Tool für den Service einzusetzen, hat einerseits den Grund, dass das Tool auch in Python geschrieben ist und daher keine weiteren Softwarepakete installiert werden müssen und andererseits,

dass es eine Vielzahl verschiedener Scantypen in einem Tool vereint. Ein weiterer Vorteil dieses Werkzeugs ist, dass der genaue Scanablauf mit Konfigurationsdateien gesteuert wird und somit keine weitere Interaktion mit dem Tool nötig ist. Die dafür benötigten Konfigurationsdateien können dynamisch erstellt werden, welche den Scan so durchführten, wie es vom Benutzer gewählt wurde.

6.1.6. PostgreSQL Datenbank

Zur Haltung der anfallenden Daten, wie beispielsweise Benutzerprofile und zur Verfügung stehender Scan- und Exploitwerkzeuge, wird eine zentrale relationale Datenbank verwendet. Hier ist die Entscheidung auf eine PostgreSQL Datenbank gefallen, da sie auch mit einer großen Menge an Daten gut skaliert. Die Verwendung einer SQL Datenbank bietet auch den Vorteil, dass die Abfragesprache SQL sehr verbreitet ist und somit viele wissen, wie man mit dieser Datenbank umgeht.

Damit man aus einem in Python geschriebenen Projekt auf eine PostgreSQL Datenbank zugreifen kann, gibt es verschiedene Python-Module, die nachinstalliert werden können. Dazu zählen beispielsweise Psycopg2, py-postgresql oder bpgsql. Eine weitere Möglichkeit ist die Verwendung von SQLAlchemy, das ein Datenbank-Toolkit für Python ist. Mit SQLAlchemy können auch eine Vielzahl anderer relationaler Datenbanken angebunden werden. Die Wahl des zu verwendenden Python-Moduls ist jedoch auf Psycopg2 gefallen, da es das bekannteste Modul für das Arbeiten mit PostgreSQL Datenbanken ist und somit viele Personen mit dem Umgang dieses Moduls vertraut sind. Eine detaillierte Beschreibung des Datenmodells erfolgt in Abschnitt 6.6.

6.2. Vorbereitungsphase

Wie bereits in Abschnitt 5.4 beschrieben, ist die Vorbereitung zu einem Penetrationstest eine wichtige Phase um die Rahmenbedingungen des Tests zu klären. Damit der Benutzer in dieser Phase von dem Service unterstützt wird, wurde ein Formular implementiert, das die vom Benutzer eingegebenen Daten nach dem Absenden automatisch zu einem Kickoff-Protokoll zusammenfasst.

Für die Generierung des Kickoff-Protokolls wurde das zuvor beschriebene Python-Modul Py3o.template verwendet. Dazu wurde eine Vorlage basierend auf einer OpenOffice/LibreOffice ODT Datei angefertigt. Damit es möglich ist, das Kickoff-Protokoll mit den vom Benutzer eingegebenen Daten automatisch zu befüllen, müssen Platzhalter in das Dokument eingefügt werden. Bei LibreOffice können diese Platzhalter in der Menüleiste unter dem Punkt *Felder* definiert werden. Dazu müssen verschiedene Namen für die zu befüllenden Felder festgelegt werden, die dann durch das Python Programm ausgefüllt werden. Falls ein Feld mit mehreren Daten befüllt werden soll, so können in der Vorlage auch Schleifen (beispielsweise For oder While) verwendet werden. Auf die Verwendung von Schleifen in Py3o.template soll hier nicht weiter eingegangen werden. Näheres zur Verwendung von Py3o.template kann aber in der Dokumentation[1] nachgelesen werden.

Die durch den Service generierten Kickoff-Protokolle werden auf dem Dateisystem des Servers abgespeichert und für einen späteren Zugriff archiviert. Der Benutzer hat die Möglichkeit, sich dieses Archiv anzeigen zu lassen und die Protokolle herunterzuladen.

6.2.1. Ansicht der Vorbereitungsphase

Das Formular zur Erstellung des Kickoff-Protokolls ist dargestellt in Abbildung 6.1. Dieses Formular soll während des Kickoff-Meetings ausgefüllt werden. Es besteht aus zwei Teilen, den allgemeinen Informationen und den Informationen zum Testobjekt.

In den allgemeinen Informationen können Angaben zur Organisation, Ort und Datum an dem das Kickoff-Meeting stattgefunden hat, angegeben werden. Des Weiteren können die Namen und die zugehörige Organisation der einzelnen Teilnehmer ausgefüllt werden. Die Felder zu den Teilnehmern sind jedoch nicht verpflichtend. Somit können auch einzelne Felder ausgelassen werden, wenn beispielsweise einer der Teilnehmer bei dem Kickoff-Meeting nicht anwesend und auch für den Penetrationstest nicht zuständig ist.

[1] http://py3otemplate.readthedocs.io/en/latest/templating.html

Im Abschnitt der Informationen zum Penetrationstest kann das Zielsystem des Penetrationstests und auch der genaue Ablauf des Tests definiert werden. Dazu gibt es ein Feld, in dem die URL des Zielsystems eingetragen wird. Diese URL wird nach Absenden des Formulars gespeichert, damit in den nächsten Phasen des Tests ein Ziel ausgewählt werden kann, ohne die URL erneut einzugeben. Als Nächstes müssen die Details zum Penetrationstest ausgefüllt werden. Hier steht eine Auswahl zur Verfügung, welche Tests der Penetrationstester durchführen darf und welche Dienste angegriffen werden sollen. Zum Schluss muss noch angegeben werden, wie aggressiv die Tests vorgehen dürfen, welche Art von Tests gewünscht sind - also ob ein BlackBox- oder WhiteBox-Test durchgeführt werden soll - und ob es sich um einen von innen oder von außen ausgeführten Penetrationstest handelt.

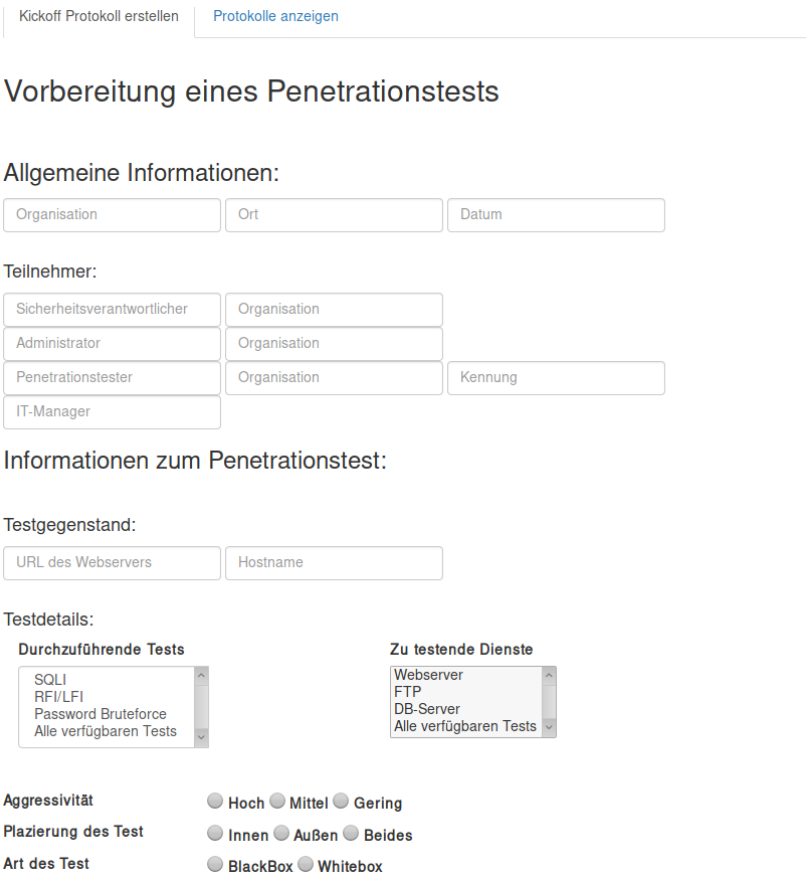


Abbildung 6.1.: Die Ansicht zur Erstellung des Kickoff-Protokolls

Nachdem das Formular für das Kickoff-Protokoll ausgefüllt und abgesendet wurde, können die generierten Kickoff-Protokolle über den Reiter ”Protokolle Anzeigen” angezeigt werden. Hier bietet sich die Möglichkeit, die Protkolle herunterzuladen und gegebenenfalls für jeden Beteiligten eine Kopie anzufertigen. Die Anzeige der generierten Kickoff-Protokolle ist in Abbildung 6.2 dargestellt.

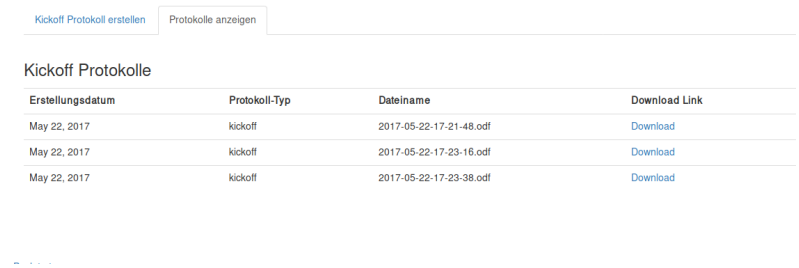


Abbildung 6.2.: Die Ansicht zur Archivierung der Kickoff-Protokolle

6.3. Informationsbeschaffung

Nachdem das Kickoff-Protokoll erstellt und alle Rahmenbedingungen für den anstehenden Penetrationstest geklärt wurden, kann der Tester mit seiner Arbeit beginnen. Dazu meldet er sich an den Service an und bekommt eine Ansicht der zur Verfügung stehenden Ziele. Die Ansicht zur Informationsbeschaffung wird in Abbildung 6.3 dargestellt. Sie bietet dem Penetrationstester die Möglichkeit, das Ziel und die verfügbaren Tests auszuwählen. Damit der Tester flexibler mit dem Service umgehen kann, wird hier noch die Möglichkeit gegeben, die Art des Tests sowie die Platzierung einzustellen. Wird dies vom Penetrationstester nicht angegeben, so wird der Test so durchgeführt, wie er im Kickoff-Protokoll definiert wurde.

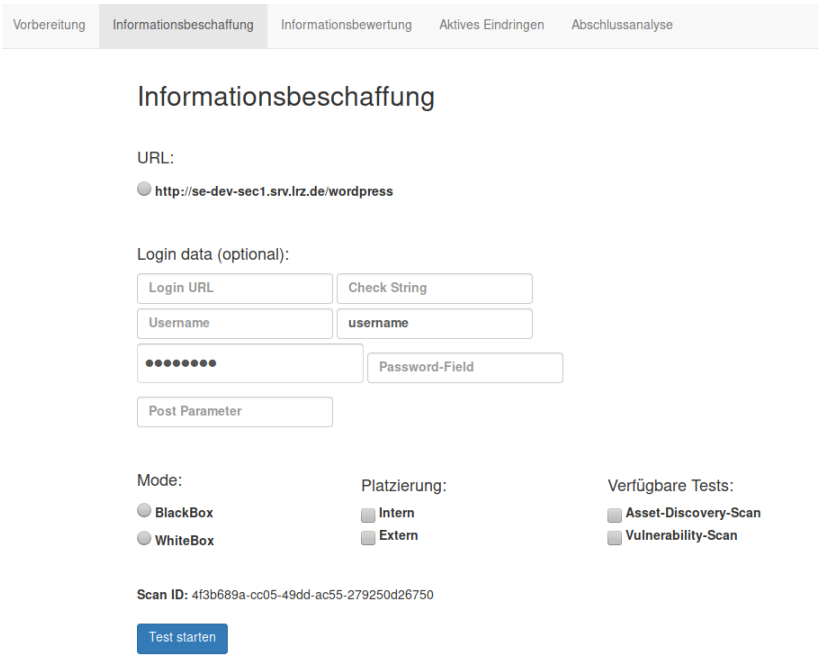


Abbildung 6.3.: Die Ansicht zur Informationsbeschaffung

Nachdem der Penetrationstester den Auftrag abgesendet hat, werden die weiteren Arbeitsabläufe zur Informationsbeschaffung selbstständig durch den Service ausgeführt. Im Folgenden wird auf den Arbeitsablauf, die Kommunikation zwischen Server und Scanner und die dazu verwendeten Tools näher eingegangen.

6.3.1. Arbeitsablauf von Server und Scanner

Wie in Abschnitt 5.9 beschrieben, ist die Architektur des Service so aufgebaut, dass es sich bei dem Server und der Scanner um getrennte Komponenten handelt, die über das Netzwerk kommunizieren. Die Arbeitsabläufe der einzelnen Komponenten sind in Abbildung 6.4 grafisch dargestellt. Im Folgenden werden die einzelnen Schritte dieser Grafik erläutert.

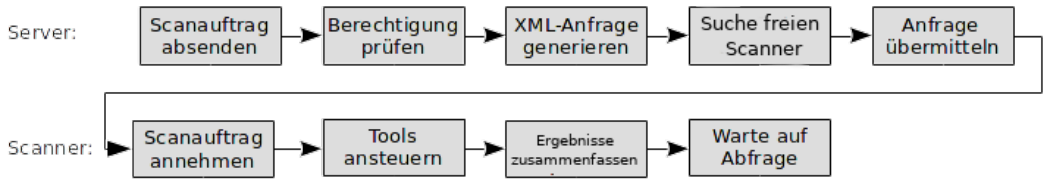


Abbildung 6.4.: Der Workflow bei der Informationsbeschaffung

Zu Beginn wird von dem Benutzer der Scanauftrag an den Server übermittelt. Dieser prüft daraufhin die Berechtigung des Benutzers anhand der hinterlegten IP-Adressen, die er scannen darf. Für den Fall, dass der Benutzer die Berechtigung besitzt, den angegebenen Webserver zu scannen, erstellt der Server eine Anfrage im XML-Format, die alle nötigen Informationen zu den verschiedenen Scan-Typen sowie die User- und Scan-ID beinhaltet. Optional kann die Anfrage auch Informationen für einen Login am Zielsystem beinhalten. Die User-ID wird benötigt, um einen Scan einem Benutzer zuzuordnen und die Scan-ID sorgt dafür, dass zwischen verschiedener Scanaufträge eines Benutzers unterschieden werden kann. Wie eine solche XML-Anfrage aussieht, wird in Abschnitt 6.3.2 gezeigt.

Nachdem diese Anfrage erstellt wurde, wählt der Server zur Verfügung stehende Scanner aus und sendet eine Anfrage an einen der Scanner, ob dieser noch genügend Kapazität hat, neue Scanaufträge anzunehmen. Damit der Scanner genügend ausgelastet wird, können verschiedene Aufträge parallel ausgefürt werden. Die Anzahl der parallel laufenden Scanaufträge kann dabei flexibel für jeden einzelnen Scanner je nach zur Verfügung stehender Ressourcen konfiguriert werden. Falls der Scanner schon mit zu vielen Aufträgen beschäftigt ist, wählt der Server den nächsten Scanner und stellt auch hier eine Anfrage. Für den Fall, dass alle Scanner beschäftigt sind, wartet der Server eine in der Konfiguration festgelegte Zeit, bevor er erneut die Scanner befragt. Wenn der Scanner jedoch zu diesem Zeitpunkt frei ist, wird die Anfrage im XML-Format übermittelt. Zu diesem Zeitpunkt ist der Server wieder frei und kann weitere Aufträge bearbeiten und übermitteln.

Sobald der Scanner den Auftrag des Servers erhält, extrahiert er die Informationen aus der XML-Anfrage und startet einen neuen Thread, der die übergebenen Scans nach der Reihe ausführt. Damit es ermöglicht wird, den Service zu einem späteren Zeitpunkt mit neuen Scan-Typen zu erweitern, wurde eine Funktion bereitgestellt, die verschiedene Scan-Typen dynamisch nachladen kann. Dafür wird der Funktion ein Array übergeben, das die Modul- und Klassennamen der Scanmodule beinhaltet. Das Nachladen dieser Scanmodule wird mit der Python Bibliothek *importlib* realisiert. Listing 6.1 zeigt die Funktion, die als neuer Thread ausgeführt wird.

Listing 6.1: Laden und Ausführen der gewählten Scan-Module

```
def handle_scans(scanTypes, scanData):
    data = []
    results = ''
    module_dir = "scanner_modules."

    for scanType in scanTypes:
        mod = scanType.split(c.TYPE_DELIMITER)
        module_name = mod[0]
        class_name = mod[1]
        load_module = getattr(importlib.import_module(module_dir+module_name), class_name)
        instance = load_module(scanData)
        data.append(instance.run())
        if isinstance(data, dict):
            data = [data]
    for result in data:
        results += result
    return "<scanResult>"+results+"</scanResult>"
```

Diese Funktion nimmt die folgenden Parameter:

- **scanTypes**: Hierbei handelt es sich um ein Array, welches den Modulnamen und Klassennamen der anzusteuernden Scanmodule enthält.
- **scanData**: Dies ist ein Objekt, das alle wichtigen Informationen, wie die Scan-ID, User-ID und die Adresse des Zielsystems enthält.

6.3.2. Kommunikation

Für die Kommunikation zwischen dem Server und den Scannern wurde eine SSL-Verschlüsselte Verbindung mittels Sockets implementiert. Mit einer SSL-Verschlüsselten Verbindung wird dafür gesorgt, dass die Daten

sicher über das Netzwerk übertragen werden. Zudem muss sich der Server bei den Scannern mit einem SSL-Zertifikat authentifizieren. Eine Authentifizierung mittels SSL-Zertifikat wurde gewählt, damit ein potentieller Angreifer, der die IP-Adresse eines Scanners kennt, diesen nicht steuern kann, so lange er kein passendes Zertifikat dafür besitzt.

Um die Authentifizierung zwischen Server und Scanner zu realisieren, wurde ein selbstsigniertes Zertifikat erstellt. Dieses Zertifikat kann beispielsweise unter Linux mit dem Softwarepaket *OpenSSL* erstellt werden. Die dazu nötigen Kommandos werden in Listing 6.2 gezeigt.

Listing 6.2: Konsolenkommandos zum Erstellen selbstsignierter SSL-Zertifikate

```
openssl genrsa -des3 -out server.org.key 2048
openssl rsa -in server.org.key -out server.key
openssl req -new -key server.key -out server.csr
openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt
```

Für den Fall, dass die Scannerkomponente dem Kunden zur Verfügung gestellt wird, um auch interne Scans im eigenen Netzsegment durchzuführen, kann für jeden Kunden ein eigenes Zertifikat bereitgestellt und in der Datenbank eingetragen werden. Der Server kann somit für jeden Kunden das passende Zertifikat für die Kommunikation auswählen. Dies bedeutet, der Server wählt bei einem Scan innerhalb des Kundennetzwerkes das Zertifikat des Kunden aus und bei einem externen Scanner, auf den der Kunde selbst keinen Zugriff hat, ein gesondertes Zertifikat. Somit kann der Kunde, der in Besitz des Zertifikats des internen Scanners ist, den externen Scanner nicht ohne den Server übernehmen. Jedoch könnte der Kunde den Scanner in seinem eigenen Netz auch ohne den Server betreiben, da er in Besitz des Zertifikates ist. Es wird aber unterbunden, dass ein externer Angreifer Attacken mit dem Scanner des Kunden durchführt.

Damit der Server und der Scanner über das Netzwerk kommunizieren können, wurde ein eigenes Kommunikationsprotokoll entwickelt. Für die Übergabe der nötigen Informationen eines Scanauftrags wurde eine Kommunikation im XML-Format implementiert, da die Informationen, die von den beteiligten Komponenten benötigt werden, leicht geparst und ausgewertet werden können. Ein XML-Schema, wie eine Scan-Anfrage aufgebaut ist, zeigt Listing 6.3. Die Felder *login_username* bis *check_string* können optional ausgefüllt werden. Sie können verwendet werden, wenn eine Webapplikation einen Benutzerlogin voraussetzt. Des Weiteren steht noch das Feld *mode* zur Verfügung, das verwendet werden kann, um die Aggressivität des Scans festzulegen.

Listing 6.3: XML-Anfrage des Servers an den Scanner zur Informationsbeschaffung

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
<xs:element name="jobdata">
        <xs:complexType>
          <xs:sequence>
                <xs:element type="xs:integer" name="userID" minOccurs="1"/>
                <xs:element type="xs:string" name="scanID" minOccurs="1"/>
                <xs:element type="xs:anyURI" name="URL" minOccurs="1"/>
                <xs:element name="modes">
                <xs:element type="xs:string" name="login_username" minOccurs="0"/>
                <xs:element type="xs:string" name="login_password" minOccurs="0"/>
                <xs:element type="xs:string" name="user_field" minOccurs="0"/>
                <xs:element type="xs:string" name="password_field" minOccurs="0"/>
                <xs:element type="xs:string" name="login_url" minOccurs="0"/>
                <xs:element type="xs:string" name="check_string" minOccurs="0"/>
                  <xs:complexType>
                    <xs:sequence>
                      <xs:element type="xs:integer" name="mode"/>
                    </xs:sequence>
                  </xs:complexType>
                </xs:element>
                <xs:element name="scanTypes">
                  <xs:complexType>
                    <xs:sequence>
                      <xs:element type="xs:string" name="scanType"/>
                      <xs:element type="xs:string" name="parameters"/>
                    </xs:sequence>
                  </xs:complexType>
                </xs:element>
          </xs:sequence>
        </xs:complexType>
        </xs:element>
</xs:element>
```

6.3.3. Verwendete Tools

Damit zu einem späteren Zeitpunkt die implementierten Scanfunktionen nachvollzogen werden können, sollen im Folgenden die verwendeten Hilfstools kurz beschrieben werden. Zudem sollen in diesem Abschnitt auch weitere Tools für die spätere Weiterentwicklung vorgeschlagen werden.

Für den Asset-Discovery-Scan wurde Nmap verwendet, damit der Zielserver auf geöffnete Ports untersucht werden kann. Der Portscan wird mit dem Parameter (-sV) für eine zusätzliche Versionserkennung ausgeführt. Damit auch bei Webapplikationen der Hersteller und die verwendete Version erkannt werden kann, wurde das NSE-Skript *http-generator* zum Auslesen des Generator-Tags eingesetzt und für die Erkennung von Wordpress Plugins und Themes das NSE-Skript *http-wordpress-enum*.

Eine Möglichkeit zur späteren Erweiterung, die eine Erkennung der Versionen und Schwachstellen verschiedener CMS-Hersteller zulässt, sind Tools wie *WPScan*, *JoomlaScan*, *CMSMAP* oder WebSorrow. Dies ist nur eine kleine Auswahl an verwendbaren Tools. Welche dieser Tools bei der Weiterentwicklung gewählt werden, hängt aber von dem Einsatzgebiet und den gewünschten Funktionen ab, da sich einige der genannten Tools im Funktionsumfang überschneiden.

Für den Schwachstellenscan wurde das schon zuvor beschriebene Tool w3af verwendet, das mit sehr vielen Plugins ausgestattet ist.

6.4. Informationsbewertung

Nachdem das Zielsystem mit dem Asset-Discovery-Scan und dem Schwachstellenscan untersucht wurde, zeigt der Service dem Benutzer geöffnete Ports, dahinter laufende Dienste und deren Versionsnummern an. Zudem werden auch gefundene Schwachstellen detailliert aufgelistet.

Wenn die Versionsnummern der eingesetzten Anwendungen erfolgreich erkannt wurden, können diese anhand einer Liste verwundbarer Versionen abgeglichen werden. Falls es öffentlich bekannte Exploits zu den eingesetzten Anwendungen gibt, können auch diese dem Benutzer vorgeschlagen werden. Der Hersteller der Anwendung und die verwundbaren Versionsnummern können durch den Superuser in der Datenbank eingetragen und somit automatisch mit den gefundenen Versionsnummern des Asset-Discovery-Scans abgeglichen werden.

In dieser Implementierung wurden aufgrund der begrenzten Zeit nur ein Exploit für das WordPress CMS implementiert und in die Datenbank eingepflegt. Bei einem Produktiveinsatz des Service können jedoch eine Vielzahl von Exploits aufgenommen werden, um somit die Wahrscheinlichkeit für das Aufspüren von gravierenden Sicherheitslücken zu erhöhen.

6.4.1. Ansicht zur Informationsbewertung

Die Ansicht der Informationsbewertung wurde so gestaltet, dass der Benutzer einen guten Überblick der gefundenen Informationen bekommt. Um eine übersichtliche Form zu wahren, wurden die Informationen in die Kategorien offene Ports, gefundene Schwachstellen und verfügbare Exploits unterteilt. Abbildung 6.5 zeigt, wie die Ansicht gestaltet wurde. Im Folgenden wird auf die einzelnen Kategorien genauer eingegangen.

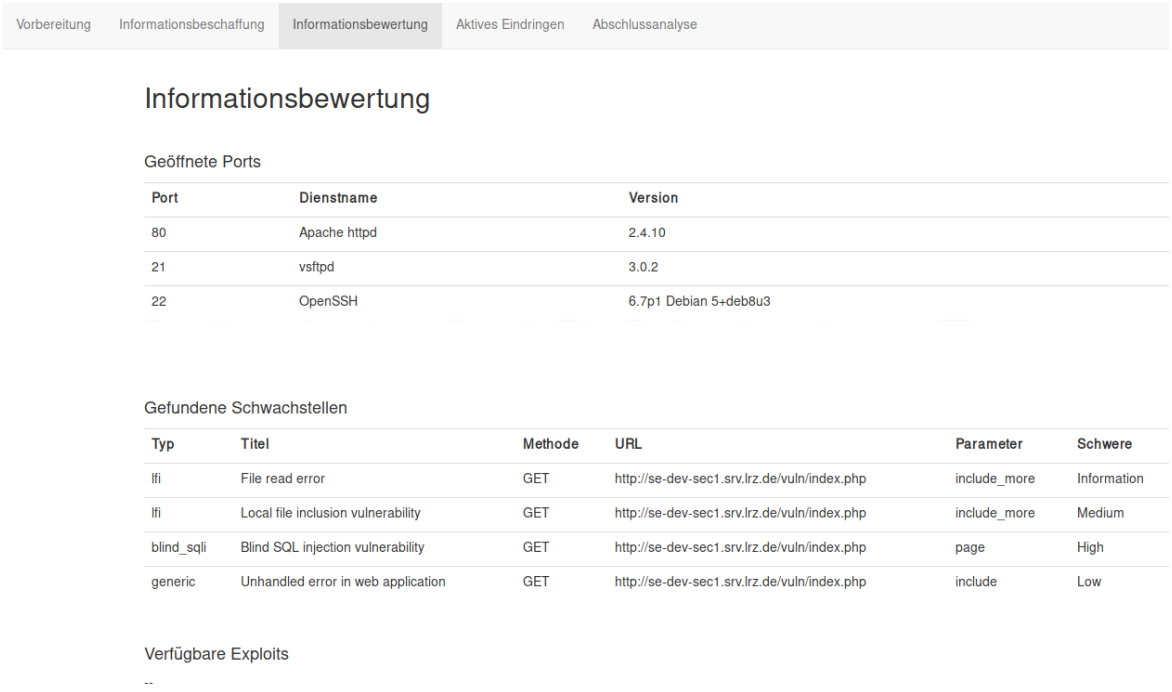


Abbildung 6.5.: Die Ansicht für die Informationsbewertung

In der ersten Kategorie wird neben den geöffneten Ports der Name des dahinter laufenden Dienstes und die Versionsnummer angezeigt. Dies kann den Benutzer dabei unterstützen, auch selbst Recherchen zu potentiellen Verwundbarkeiten des betreffenden Dienstes durchzuführen. Zudem bekommt der Benutzer einen guten Überblick, welche Dienste mit welchen Versionsnummern auf dem Zielserver laufen.

Die nächste Kategorie gibt dem Benutzer einen Überblick zu den Schwachstellen, die durch den Service identifiziert wurden. Des Weiteren werden alle Details wie die betreffende URL und der verwundbare Parameter angezeigt, damit der Benutzer die Schwachstelle auch manuell überprüfen kann. Dies ist hilfreich, wenn ein Einbruchsversuch durch den Service nicht gelungen ist und der Penetrationstester die Schwachstellen manuell überprüfen muss.

Die letzte Kategorie zeigt Exploitvorschläge an, für die das Zielsystem möglicherweise anfällig ist. Der Benutzer hat hier die Möglichkeit, einen oder mehrere Exploits auszuwählen, mit deren Hilfe das Ziel angegriffen wird.

6.5. Aktive Eindringversuche

Nachdem die Informationsbewertung abgeschlossen ist, kann der Penetrationstester zur nächsten Phase, dem aktiven Eindringen in das Zielsystem, übergehen. Damit dem Benutzer vorgeschlagen werden kann, welche Einbruchsmöglichkeiten zur Verfügung stehen, verwendet der Service die gesammelten Informationen aus der vorherigen Phase und gleicht diese mit den verfügbaren Einbruchswerkzeugen ab. Alle Schwachstellen, für die ein Einbruchswerkzeug existiert, können so erkannt und dem Benutzer angezeigt werden. Der Penetrationstester kann nun flexibel auswählen, welche Tests er durchführen will. Es besteht auch die Möglichkeit, für den Test Exploits zu wählen, die der Service nicht als potentiell wirksam erkannt hat, der Penetrationstester diese aber trotzdem ausführen will. Nachdem der Benutzer den Auftrag abgesendet hat, prüft der Server die Berechtigung des Benutzers und kommuniziert im Anschluss mit einem freien Scanner, um den Auftrag zu übermitteln. Die Interaktion zwischen dem Server und Scanner läuft ähnlich der in den Abschnitten 6.3.1 und 6.3.2 beschriebenen Abläufen ab. Der Unterschied ist lediglich der Aufbau der XML-Anfrage und der Information, welche Art von Scan gewünscht wird. Das XML-Schema dieser Anfrage wird in Listing 6.4 gezeigt.

Listing 6.4: XML-Anfrage des Servers an den Scanner für die Penetrationsphase

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="jobdata">
    <xs:complexType>
      <xs:sequence>
                <xs:element type="xs:byte" name="userID"/>
                <xs:element type="xs:string" name="scanID"/>
                <xs:element name="scanTypes">
                <xs:element type="xs:string" name="login_username" minOccurs="0"/>
                <xs:element type="xs:string" name="login_password" minOccurs="0"/>
                <xs:element type="xs:string" name="user_field" minOccurs="0"/>
                <xs:element type="xs:string" name="password_field" minOccurs="0"/>
                <xs:element type="xs:string" name="login_url" minOccurs="0"/>
                <xs:element type="xs:string" name="check_string" minOccurs="0"/
          <xs:complexType>
            <xs:sequence>
              <xs:element name="option">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element type="xs:anyURI" name="url"/>
                    <xs:element type="xs:string" name="variable"/>
                    <xs:element type="xs:string" name="method"/>
                    <xs:element type="xs:string" name="mode"/>
                    <xs:element type="xs:string" name="shell"/>
                    <xs:element type="xs:string" name="scanType"/>
                    <xs:element type="xs:string" name="password"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Eine kurze Erklärung zu den Optionsfeldern wird in folgender Liste gegeben:

- **url:** Die komplette URL, ohne GET-Parameter, in der die Schwachstelle gefunden wurde.
- **variable:** Der GET/POST Parameter, in dem sich die Schwachstelle befindet.
- **method:** Die HTTP-Methode, die für den betreffenden Parameter angewandt wird (POST/GET).

- **mode:** Hier gibt es die Möglichkeit, eine volle Penetration oder nur ein Verifizieren der Ausnutzbarkeit einer Schwachstelle auszuführen.
- **shell:** Dies ist ein optionales Feld zur Auswahl der gewünschten Shell, die hochgeladen werden soll.
- **scanType:** Eine Array der auszuführenden Exploit-Module.
- **password:** Dies ist ein optionales Feld, um das Passwort zum Zugangsschutz der Shell zu setzen.

6.5.1. Ansicht zum aktiven Eindringen

Die Benutzeroberfläche bietet dem Penetrationstester eine Reihe an Einstellungsmöglichkeiten, wie es in Abbildung 6.6 dargestellt ist. Er kann flexibel auswählen, in welchem Umfang ein Penetrationstest durchgeführt werden soll. Dazu kann er einen vollen Einbruchsversuch wählen, bei dem durch den Service versucht wird, die Schwachstelle so weit auszunutzen, um eine Shell auf das Zielsystem einzuschleusen. Will der Pentester das nicht, kann die Schwachstelle nur auf ihre Ausnutzbarkeit geprüft werden. Hierzu wird durch den Service versucht, die Schwachstelle zu verwenden, um sensitive Daten, wie beispielsweise den Inhalt der Datenbank oder der Dateien auf der Festplatte des Zielsystems zu lesen. Wenn dies erfolgreich war, werden dem Benutzer diese Informationen angezeigt.

Für den Fall, dass der Penetrationstester eine volle Penetration wünscht, kann er im nächsten Feld die gewünschte Shell auswählen. In dieser Implementierung steht entweder Weevely oder eine Webshell zur Verfügung. Damit das System durch das Hochladen der Shell nicht noch verwundbarer wird, muss der Benutzer ein Passwort vergeben, das beim Aufruf der Shell abgefragt wird. Als zusätzliche Sicherheitsmaßnahme generiert der Service eine zufällige Buchstabenkombination aus sechs Zeichen, die als Dateiname für die Shell verwendet wird.

Bevor der Penetrationstest gestartet wird, kann der Benutzer auswählen, welche der gefundenen Schwachstellen ausgenützt werden sollen und gegebenenfalls auch verfügbare Exploits wählen, welche in dem Test mit einbezogen werden. Nachdem alle Einstellungen getroffen wurden, kann der Test gestartet werden, der dann völlig selbstständig durchläuft.

Wenn der Penetrationstest beendet ist, gelangt der Benutzer über den Reiter ”Ergebnisse anzeigen” zu den Resultaten des Tests, wie in Abbildung 6.7 zu sehen ist. Hier werden die getesteten Schwachstellen aufgelistet und danach bewertet, ob die Penetration erfolgreich war.

Wenn es möglich war, eine Shell auf dem Zielsystem zu platzieren, wird in der Spalte ”Shell” die URL, mit der die Shell aufzurufen ist, angezeigt. Das Feld ”Data” beinhaltet Daten, welche von der Festplatte des Zielsystems gelesen wurden. Dieses Feld wird nur dann verwendet, wenn der Benutzer in der vorherigen Ansicht keine volle Penetration gewählt hat oder wenn das Einschleusen einer Shell nicht möglich war. Die letzte Spalte beinhaltet Informationen, welche Aktionen bei dem Versuch, aktiv einzudringen, durchgeführt wurden. Dies soll dem Penetrationstester helfen, einerseits den Angriff nachzuvollziehen und andererseits das System nach dem Angriff zu bereinigen.

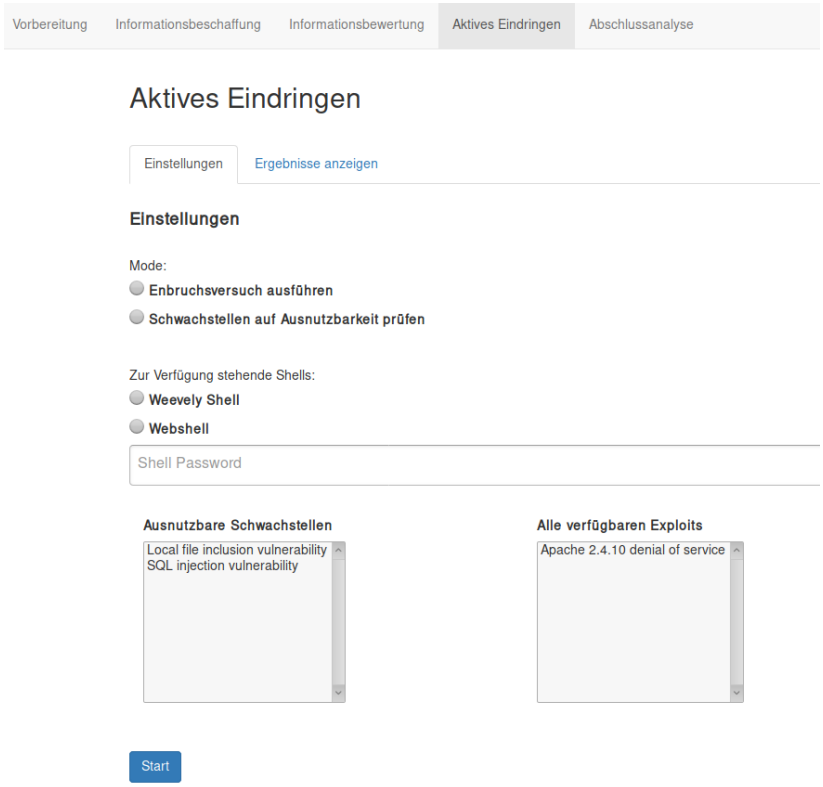


Abbildung 6.6.: Die Ansicht zu den Einstellungen für die aktiven Eindringversuche

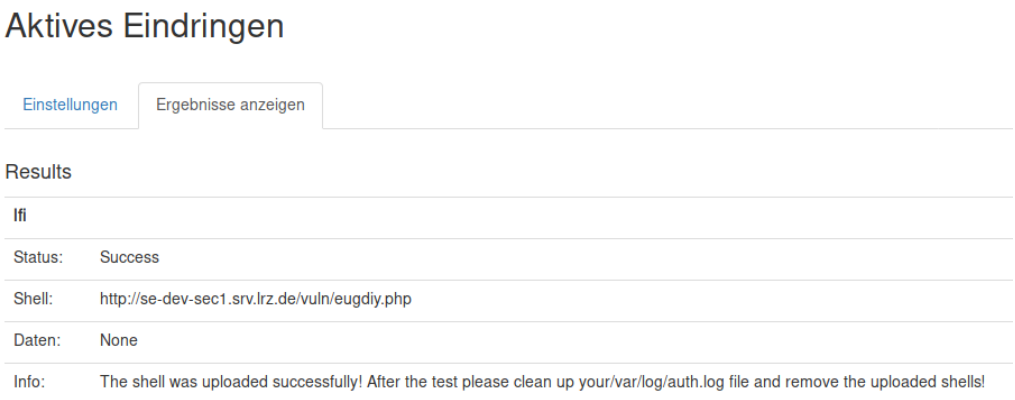


Abbildung 6.7.: Die Ansicht der Ergebnisse nach einem Eindringversuch

6.6. Datenhaltung

Für die Speicherung der anfallenden Daten hat der Server Zugriff auf eine zentrale Datenbank. Wie bereits beschrieben, wurde in dieser Implementierung die relationale Datenbank PostgreSQL verwendet, die auch bei einer großen Menge an Daten gut skaliert. Die Wahl, eine SQL-Datenbank zu verwenden, wurde getroffen, da die Abfragesprache SQL sehr weit verbreitet ist und viele den Umgang damit beherrschen. Außerdem ist ein Vorteil bei der Verwendung einer Datenbank, dass sich die Daten effizienter auslesen und filtern lassen als beispielsweise das Speichern der Daten im XML-Format. Bei einem späteren Produktiveinsatz können auch Maßnahmen zur Hochverfügbarkeit der Datenbank getroffen werden und Replikate der Datensätze erstellt

werden. Eine grafische Darstellung des verwendeten Datenmodells zeigt Abbildung 6.8.

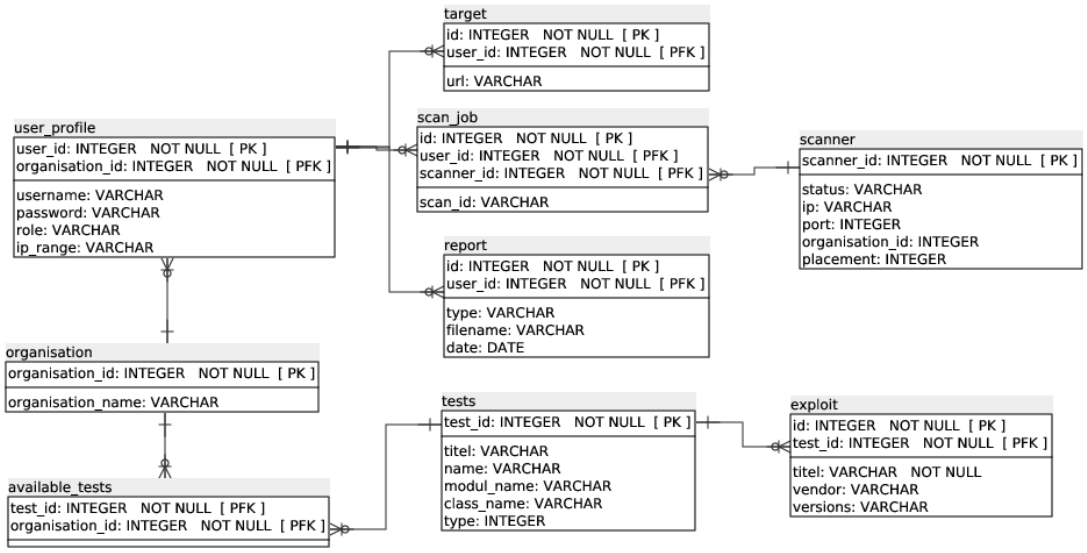


Abbildung 6.8.: Grafische Darstellung des Datenbank-Modells

Das Datenmodell besteht aus einer Reihe von Tabellen, die jeweils Relationen zueinander besitzen. Es wurde versucht, die Anzahl der Tabellen so gering wie möglich zu halten, damit der Aufbau übersichtlich bleibt. Bei einer späteren Weiterentwicklung des Service kann das Datenmodell bei Bedarf jedoch leicht erweitert werden. Im Folgenden werden die Tabellen und deren Spalten beschrieben.

Die Tabelle *user_profile* beinhaltet alle nötigen Informationen der Benutzer:

Name	Typ	Beschreibung
user_id	Integer (PK)	Dies ist eine eindeutige Identifikation für jeden Eintrag.
username	String	In dieser Spalte wird der Name des Benutzers gespeichert. Dieser sollte so gewählt werden, dass er eindeutig ist und nicht mehrmals vorkommt.
password	String	Diese Spalte beinhaltet den Passwort-Hash des Benutzers. In dieser Implementierung wurde dafür PBKDF2 verwendet.
role	String	Diese Spalte beinhaltet die Rolle des Benutzers.
organisation_id	Integer (FPK)	Der Fremdschlüssel zur Identifikation der Organisation der ein User angehört.
ip_range	Integer	Hier wird die IP-Range eingetragen, die ein Benutzer scannen darf. Der Eintrag kann entweder in CDIR-Format oder kommasepariert sein.

Die Tabelle *organisation* wird verwendet, damit ein Benutzer eindeutig einer Organisation zugeordnet werden kann. Somit ist es möglich, bestimmten Organisationen Zugriff auf ausgewählte Tests zu gewährleisten.

Name	Typ	Beschreibung
organisation_id	Integer (PK)	Eine eindeutige Identifikation für jede Organisation.
organisation	String	Jeder Benutzer gehört einer bestimmten Organisation an, die hier eingetragen wird.

Die Tabelle *target* wird verwendet, um die Zielsysteme, die durch die Benutzer bereitgestellt werden, speichern zu können.

Name	Typ	Beschreibung
id	Integer (PK)	Eine eindeutige Identifikation für jeden Datensatz.
user_id	Integer (FPK)	Der Fremdschlüssel des Benutzers zu dem dieses Ziel gehört.
url	String	Diese Spalte enthält die URL des Zielsystems.

In der Tabelle *scan_job* werden die Aufträge, welche von den Benutzern abgesendet werden, eingetragen. Diese Tabelle dient dem Server, um die Aufträge von der Scannerkomponente abzuholen und eindeutig einem Benutzer zuordnen zu können.

Name	Typ	Beschreibung
id	Integer (PK)	Eine eindeutige Identifikation für jeden Eintrag.
user_id	Integer (FPK)	Diese Spalte enthält den Fremdschlüssel des Benutzers, der den Auftrag gesendet hat.
scanner_id	Integer (FPK)	Diese Spalte enthält den Fremdschlüssel des Scanners, dem dieser Auftrag übermittelt wurde. Anhand der Information dieser Spalte kann der Server erkennen, ob es sich um einen internen oder externen Scan handelt und in welcher Organisation der Scanner betrieben wird.
scan_id	String	Diese Spalte beinhaltet für jeden Scan eine eindeutige Identifikation, um verschiedene Aufträge eines Benutzers zuordnen zu können. In dieser Implementierung wird die Scan-ID mittels dem Python-Modul *uuid* generiert.

Die Tabelle *report* beinhaltet die wichtigsten Informationen zum Auffinden des Reports auf dem Dateisystem des Servers.

Name	Typ	Beschreibung
id	Integer (PK)	Eine eindeutige Identifikation für jeden Eintrag.
user_id	Integer (FPK)	Diese Spalte enthält den Fremdschlüssel des Benutzers, für den dieser Report generiert wurde.
type	String	Diese Spalte enthält die Art des Reports. Hier könnte beispielsweise Kickoff-Report, Penetrationstester-Report, Administrator-Report etc. eingetragen werden.
filename	String	In dieser Spalte wird der Dateiname des Berichts eingetragen, damit er auf dem Dateisystem des Servers wieder gefunden werden kann.
date	Date	Diese Spalte enthält das Datum und die Uhrzeit an dem der Bericht erstellt wurde.

Die Tabelle *scanner* enthält alle Informationen zu den verfügbaren Scannern.

Name	Typ	Beschreibung
scanner_id	Integer (PK)	Eine eindeutige Identifikation für jeden Datensatz.
status	Integer	Diese Spalte enthält den Status des Scanners. Der Status kann entweder online oder offline sein. Die Spalte kann von dem Server verändert werden, sobald er den betreffenden Scanner nicht erreichen kann. In diesem Fall wird der Status auf offline gestellt und der Superuser per Email benachrichtigt.
ip	String	Die IP des Scanners, damit der Server eine Verbindung aufbauen kann.
port	Integer	Der Port auf dem der Scanner für neue Verbindungen lauscht.
organisation_id	Integer	Diese Spalte enthält bei internen Scannern die ID der Organisation, damit diese zugeordnet werden kann.
placement	Integer	Diese Spalte enthält die Information, ob es sich um einen internen oder einen externen Scanner handelt. Somit kann der Server je nach Wunsch des Benutzers den richtigen Scanner auswählen.

Die Tabelle *tests* beinhaltet alle zur Verfügung stehenden Scan- und Exploitmodule.

Name	Typ	Beschreibung
test_id	Integer (PK)	Eine eindeutige Identifikation für jeden Eintrag.
titel	String	Der Titel des Scan- oder Exploitwerkzeugs.
name	String	Der Name des Scan- oder Exploitwerkzeugs. Diese Spalte wird vom Server verwendet, um dem Benutzer bei den Ergebnissen den Namen des Moduls anzuzeigen.
modul_name	String	Diese Spalte beinhaltet den Dateinamen des Moduls. Dies ist wichtig, um dynamisch Module nachladen zu können.
class_name	String	In dieser Spalte wird der Klassenname des Moduls eingetragen. Auch hier wird dies vom Scanner verwendet, um dynamisch Module nachzuladen.
type	Integer	Diese Spalte gibt an, ob es sich bei dem jeweiligen Modul um einen Scanner oder Exploiter handelt.

Die Tabelle *exploit* beinhaltet alle verfügbaren Exploits. Diese Tabelle wird vom Server verwendet, um erkannte Applikationen und deren Versionsstände mit öffentlich bekannten Exploits abzugleichen und gegebenenfalls dem Benutzer mögliche Exploits vorzuschlagen.

Name	Typ	Beschreibung
id	Integer (PK)	Eine eindeutige Identifikation für jeden Eintrag.
test_id	Integer (FPK)	Der Fremdschlüssel des zu verwendenden Exploit-Moduls.
titel	String	Der Titel des Exploits.
vendor	String	Der Hersteller der Applikation.
versions	String	Die verwundbaren Versionen entweder kommasepariert oder im Format von-bis.

6.7. Erweiterung des Service

Da es bei der Sicherheit von IT-Systemen immer neue Angriffstechniken und neue Tools zum Scannen und Ausnutzen von Schwachstellen gibt, muss der Service für Penetrationstests auch zu einem späteren Zeitpunkt leicht erweiterbar sein. Die Möglichkeit der Erweiterung wurde bei der Implementierung mit bedacht und so konzipiert, dass es für den Superuser, der den Service verwaltet und administriert, leicht fällt, neue Komponenten anzubinden.

Für den Fall, dass eine neue Scankomponente hinzukommt, sind zwei Schritte nötig. Als erstes muss eine Klasse implementiert werden, die diesen Scan ausführt und gegebenenfalls Hilfstools ansteuert. Nachdem die Klasse implementiert wurde, kann sie leicht an den Service angebunden werden, indem der Superuser den Dateinamen der Klasse, den Klassennamen und den Scantypen in die Datenbank einträgt. Beim Scantypen handelt es sich um die Unterscheidung, ob die Klasse zur Informationsbeschaffung oder zum aktiven Eindringen bestimmt ist. Nachdem diese zwei Schritte durchgeführt wurden, erkennt der Service die neue Komponente und stellt sie dem Benutzer zur Verfügung.

Falls der Service so abgeändert werden soll, dass das im LRZ betriebene Tool Dr. Portscan für den Scan offener Ports verwendet werden soll, kann auch hier leicht eine neue Klasse implementiert werden, die Dr. Portscan ansteuert und die Ergebnisse entgegennimmt. Nachdem die nötigen Informationen in der Datenbank eingetragen wurden, steht der neue Scantyp sofort zur Verfügung.

Die Scantypen, die von dem Service angesteuert werden, können somit beliebige Aktionen ausführen. Die einzige Limitierung dabei ist, dass sie zur erfolgreichen Kommunikation zwischen Scanner und Server das in Abschnitt 6.3.2 und 6.5 erläuterte XML-Schema verwenden müssen. Falls bestimmte Felder des XML-Schemas nicht benötigt werden, können diese leer gelassen werden, dürfen jedoch nicht fehlen.

7. Evaluation

Da das Konzept für den Service vorgestellt und ein Prototyp implementiert wurde, befasst sich das folgende Kapitel mit der Evaluation der prototypischen Implementierung des Penetrationstesting Service. Damit ein aussagekräftiger Test der Software möglich ist, wird der Service anhand verschiedener Webapplikationen getestet und die Resultate bewertet. Die Webanwendungen, an denen der Prototyp getestet wird, sind eine selbst entwickelte Webseite die einige Schwachstellen enthält, eine Wordpress-Installation mit einem verwundbaren Plugin, Damn Vulnerable Web App (DVWA) und eine Webanwendung, die von Studenten am LRZ entwickelt wurde. Zudem wird am Ende dieses Kapitels ein Penetrationstest verschiedener Webangebote der Bayerischen Staatsbibliothek beschrieben. Diese Auswahl wurde getroffen, da somit viele der im Produktiveinsatz möglichen Szenarien abgedeckt werden und ein guter Überblick der Stärken sowie der Grenzen des Service für Penetrationstests geschaffen wird.

7.1. Beschreibung der Testobjekte

Da möglichst alle Funktionen des Prototyps getestet werden sollen, wurden verschiedene Webapplikationen auf einem virtuellen Server, der durch das LRZ bereitgestellt wurde, installiert. Dieser Abschnitt gibt einen Überblick, welche Software für die Evaluation herangezogen wurde.

7.1.1. Verwundbare Webapplikation aus eigener Entwicklung

Für einen ersten Test wurde eine Webapplikation mittels der Skriptsprache PHP entwickelt, die verschiedene Schwachstellen enthält. Bei den eingebauten Schwachstellen handelt es sich um eine SQL-Injection, eine Blind-SQL-Injection und eine Local File Inclusion. Außerdem wurde eine Login-Funktion hinzugefügt, damit der Service auch mit einer Authentifizierung getestet werden kann.

7.1.2. Verwundbare WordPress Instanz

Da viele Kunden auf ihren Webservern Content Management Systeme betreiben, wurde auf dem Testserver ein WordPress in der aktuellen Version 4.8 installiert. Da es sich um eine aktuelle WordPress Applikation handelt, ist davon auszugehen, dass keine Schwachstelle durch einen automatisierten Scan gefunden werden kann. Daher wurde ein verwundbares Plugin installiert. Bei diesem Plugin handelt es sich um WP-Vault in der Version 0.8.6.6. Dieses Plugin und der dazugehörige Exploit kann auf der Webseite exploit-db.com und vielen anderen Webseiten im Internet gefunden werden.

7.1.3. Damn Vulnerable Web Application

Damit ein ausgiebiger Test des Service möglich ist, wurde die Entscheidung getroffen, eine sehr verwundbare Webapplikation auf dem Testserver zu installieren. Nach etwas Recherche im Internet, welche Software sich dazu gut eignet, ist der Fokus auf Damn Vulnerable Web Application (DVWA)[1] gefallen. Diese Software wurde in PHP mit einer Anbindung an eine MySQL Datenbank entwickelt. Sie beinhaltet eine Vielzahl von Schwachstellen und wurde hauptsächlich für Personen entwickelt, die ihre Fähigkeiten in Bezug auf IT-Sicherheit testen wollen. Da DVWA mit vielen ausnutzbaren Schwachstellen ausgestattet ist, eignet sich diese Webapplikation sehr gut für einen Test auf die Effektivität des Penetrationstesting Service.

[1] http://dvwa.co.uk

7.1.4. Security Document Management System des LRZ

Bei dieser Testinstanz handelt es sich um ein Security Document Management System, das vom LRZ betrieben wird. Diese Webapplikation ist ein Entwicklungs- und Testsystem zur Erstellung von Sicherheitskonzepten. Diese Anwendung wurde im LRZ von Studenten entwickelt und eignet sich gut für einen Test des Service, da es eine Webapplikation ist, die oft von Mitarbeitern des LRZ verwendet wird und einem realistischen Szenario sehr nahe kommt.

7.1.5. Projekte der Bayerischen Staatsbibliothek

Ein weiterer Test des Service wurde bei der Bayerischen Staatsbibliothek (BSB) als Großkunde des LRZ durchgeführt. Die Mitarbeiter der Abteilung für die Onlinepräsenz haben dafür vier neue Projekte für einen Penetrationstest zur Verfügung gestellt. Bei dem ersten Projekt handelt es sich um eine Webapplikation, die den Benutzern Zugang zu elektronischen Medien wie CD, DVD oder Bänder zur Verfügung stellt. Das zweite Projekt ist eine Webseite, die Personenlexika zur Personenrecherche anbietet. Dabei wurden die Personenlexika komplett gescannt und alle Inhalte erfasst. Bei dem dritten Projekt handelt es sich um ein Projekt, bei dem die Bestände der BSB digitalisiert und ins Internet gebracht wurden. Die letzte Webpräsenz, die für einen Penetrationstest angeboten wurde, ist ein Universallexikon aller Wissenschaften und Künste.

7.2. Testdurchführung

Da die Testobjekte beschrieben wurden, können in diesem Abschnitt die Experimente mit dem Service für Penetrationstests ausgeführt werden. Die Experimente wurden immer nach dem gleichen Muster durchgeführt. Zuerst wird mit der Phase der Informationsbeschaffung begonnen, damit festgestellt werden kann, welche Dienste auf dem Testserver betrieben werden. Danach werden die Informationen bewertet und gezeigt, welche Schwachstellen durch den Service identifiziert werden konnten. Nachdem diese Phasen abgeschlossen sind, wird ein aktiver Einbruchsversuch durchgeführt, um zu erkennen, wie weit in das System vorgedrungen werden kann und wo die Grenzen des Service liegen.

7.2.1. Verwundbare Webapplikation aus eigener Entwicklung

Für den ersten Test des prototypisch implementierten Service wurde die Webapplikation aus eigener Entwicklung herangezogen. Zunächst wurde die Login-Funktion deaktiviert, um den Service ohne Authentifizierung zu testen. Dazu wurden die drei Phasen - Informationsbeschaffung, Informationsbewertung und das aktive Eindringen in das Zielsystem - durchgeführt, das im Folgenden näher beschrieben wird.

Informationsbeschaffung

Für die Informationsbeschaffung wurde der Scanner im internen Netz verwendet und die Untersuchung der laufenden Dienste sowie potenzieller Schwachstellen aktiviert. Da die Webseite aus eigener Entwicklung nur eine geringe Zahl verfügbarer Links bereitstellt, konnte der Scanner diese Phase sehr schnell beenden und die Resultate zurück an den Server senden.

Informationsbewertung

Nachdem der Service die Phase der Informationsbeschaffung beendet hat, kann mit der Informationsbewertung begonnen werden. Hierzu kann der Penetrationstester in der Menüleiste die betreffende Phase auswählen und bekommt eine Liste der gefundenen Dienste, die auf dem Zielsystem betrieben werden. Des Weiteren werden die Schwachstellen, die durch den Scan identifiziert wurden, aufgelistet und bewertet. In diesem Szenario konnte der Service alle Schwachstellen, die in der Webseite vorhanden sind, finden. Die Ansicht der identifizierten Schwachstellen wird in Abbildung 7.1 gezeigt.

Informationsbewertung

Geöffnete Ports

Port	Dienstname	Version
80	Apache httpd	2.4.10
21	vsftpd	3.0.2
22	OpenSSH	6.7p1 Debian 5+deb8u3
111		2-4

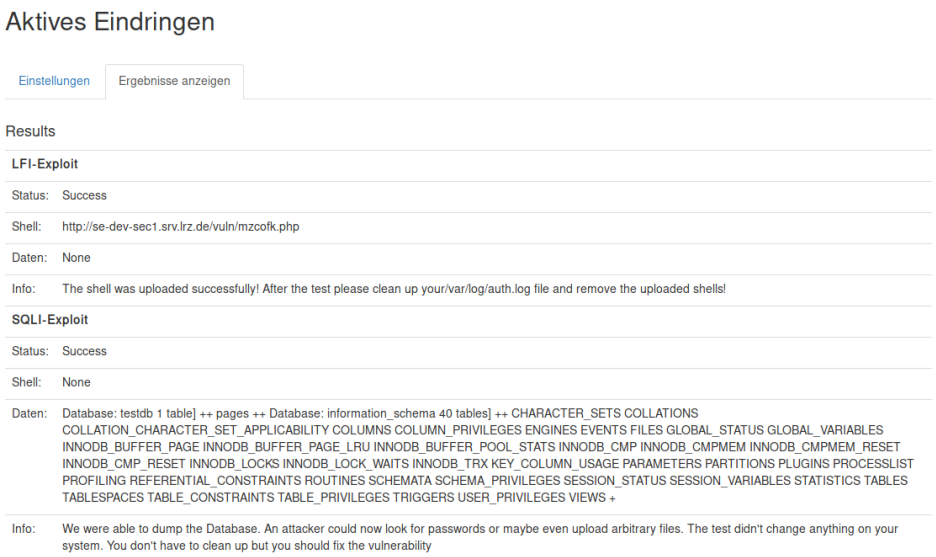
Gefundene Schwachstellen

Typ	Titel	Methode	URL	Parameter	Schwere
lfi	File read error	GET	http://se-dev-sec1.srv.lrz.de/vuln/index.php?include_more=%2Findex.php	include_more	Information
lfi	Local file inclusion vulnerability	GET	http://se-dev-sec1.srv.lrz.de/vuln/index.php?include_more=%2Fetc%2Fpasswd	include_more	Medium
blind_sqli	Blind SQL injection vulnerability	GET	http://se-dev-sec1.srv.lrz.de/vuln/index.php?page=46%22%20OR%20%2246%22%3D%2246%22%20OR%20%2246%22%3D%2246	page	High

Abbildung 7.1.: Die Ergebnisse der Informationsbeschaffung

Aktive Eindringversuche

In der Nächsten Phase des Penetrationstests wird das aktive Eindringen in das Zielsystem getestet. Hierfür wurde versucht, anhand der zuvor identifizierten Schwachstellen eine Weevely Shell in das System einzuschleusen. Wie in Abbildung 7.2 zu sehen ist, war der Einbruchsversuch mittels Local File Inclusion möglich. Der Einbruchsversuch durch die SQL-Injection war nicht möglich, jedoch konnte die Datenbank des verwundbaren Systems ausgelesen werden und wird dem Penetrationstester in der Liste der Ergebnisse angezeigt.



Aktives Eindringen

Einstellungen | Ergebnisse anzeigen

Results

LFI-Exploit	
Status:	Success
Shell:	http://se-dev-sec1.srv.lrz.de/vuln/mzcofk.php
Daten:	None
Info:	The shell was uploaded successfully! After the test please clean up your/var/log/auth.log file and remove the uploaded shells!
SQLI-Exploit	
Status:	Success
Shell:	None
Daten:	Database: testdb 1 table] ++ pages ++ Database: information_schema 40 tables] ++ CHARACTER_SETS COLLATIONS COLLATION_CHARACTER_SET_APPLICABILITY COLUMNS COLUMN_PRIVILEGES ENGINES EVENTS FILES GLOBAL_STATUS GLOBAL_VARIABLES INNODB_BUFFER_PAGE INNODB_BUFFER_PAGE_LRU INNODB_BUFFER_POOL_STATS INNODB_CMP INNODB_CMPMEM INNODB_CMPMEM_RESET INNODB_CMP_RESET INNODB_LOCKS INNODB_LOCK_WAITS INNODB_TRX KEY_COLUMN_USAGE PARAMETERS PARTITIONS PLUGINS PROCESSLIST PROFILING REFERENTIAL_CONSTRAINTS ROUTINES SCHEMATA SCHEMA_PRIVILEGES SESSION_STATUS SESSION_VARIABLES STATISTICS TABLES TABLESPACES TABLE_CONSTRAINTS TABLE_PRIVILEGES TRIGGERS USER_PRIVILEGES VIEWS +
Info:	We were able to dump the Database. An attacker could now look for passwords or maybe even upload arbitrary files. The test didn't change anything on your system. You don't have to clean up but you should fix the vulnerability

Abbildung 7.2.: Die Ergebnisse des aktiven Eindringversuchs

7.2.2. Verwundbare WordPress Instanz

Für den nächsten Test wurde eine WordPress Instanz auf dem Testserver installiert. Damit es für den Service einen Angriffspunkt gibt, wurde das verwundbare Plugin WP-Vault nachinstalliert. Im Folgenden werden wieder die einzelnen Phasen des Penetrationstests beschrieben.

Informationsbeschaffung

Bei der Phase der Informationsbeschaffung wurde, wie bei dem vorangegangenen Test, ein Asset-Discovery-Scan und ein Schwachstellenscan gewählt. Der Scan der WordPress Instanz hat aufgrund der vielen Unterseiten etwas länger gedauert als der Scan der Webapplikation aus eigener Entwicklung. Nach Abschluss der beiden Scans wurden die Ergebnisse für die Ansicht zurück an den Server gesendet.

Informationsbewertung

Die Ansicht der Informationsbewertung (vgl. Abbildung 7.3) zeigt, dass der Service das verwundbare WordPress-Plugin erfolgreich identifiziert hat. Das Aufspüren möglicher anderer Verwundbarkeiten ist dem Service nicht gelungen. Die Einschätzung, dass weitere Schwachstellen durch den Service nicht gefunden werden, war jedoch schon vor dem Scan klar, da es sich um eine aktuelle WordPress Installation handelt, die zu diesem Zeitpunkt keine bekannten Schwachstellen beinhaltet. Der Asset-Discovery-Scan konnte wie bei dem vorherigen Experiment alle Dienste, welche auf dem Zielsystem betrieben werden, erfolgreich erkennen.


Informationsbewertung

Geöffnete Ports

Port	Dienstname	Version

Gefundene Schwachstellen

Typ	Titel	Methode	URL	Parameter	Schwere

Verfügbare Exploits

Typ	Titel	Vendor	Version	URL	Parameter
lfi	WP-Vault Plugin	WP-Vault	0.6.8.8	http://se-dev-sec1.srv.lrz.de/wordpress/?wpv-js=xml-request	wpv-js

Abbildung 7.3.: Die Ergebnisse der Informationsbeschaffung

Aktive Eindringversuche

Nachdem der Service das verwundbare WordPress-Plugin erfolgreich identifiziert hat, kann im nächsten Schritt versucht werden, in das Zielsystem einzudringen. Hierfür wurde das Modul, das den Exploit gegen das WordPress-Plugin beinhaltet, gewählt. Damit der Penetrationstester Systemkommandos auf dem Zielsystem ausführen kann, wurde das Hochladen einer Weevely Shell gewählt. Wie in Abbildung 7.4 zu sehen ist, konnte der Service den Exploit erfolgreich ausführen und die Shell auf das Zielsystem einschleusen.

7.2.3. Damn Vulnerable Web Application

In einem weiteren Experiment wurde die Webseite Damn Vulnerable Web Application auf dem Testserver installiert. Dieses Experiment soll zeigen, wie viele Schwachstellen der Service erkennen und ausnutzen kann. Ein weiterer Vorteil dieses Experiments ist, dass eine Authentifizierung nötig ist, damit die Schwachstellen gefunden werden können. Somit zeigt dieser Test, wie der Service mit einem Penetrationstest einer durch Login geschützten Webseite umgeht.

Informationsbeschaffung

Wie auch in den letzten Experimenten wird mit der Informationsbeschaffung begonnen. Hierfür wurde wieder ein Asset-Discovery-Scan und ein Schwachstellenscan gewählt. Aufgrund der großen Anzahl der Unterseiten,

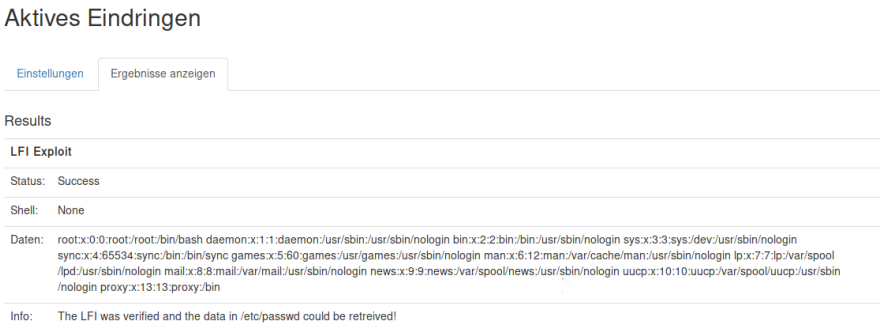


Abbildung 7.4.: Die Ergebnisse des aktiven Eindringversuchs

die DVWA beinhaltet, hat der Scan zur Informationsbeschaffung lange gedauert, bis die Ergebnisse zurück an den Server gesendet wurden. Jedoch hat keiner der Scans länger als 15 Minuten in Anspruch genommen. Die Dauer des Scans zur Informationsbeschaffung ist daher auch für einen realen Einsatz des Service hinnehmbar.

Informationsbewertung

Nachdem die Ergebnisse zur Informationsbeschaffung zurück an den Server gesendet wurden, können im nächsten Schritt die erlangten Informationen bewertet werden. Wie in Abbildung 7.5 zu sehen ist, konnte der Service eine Vielzahl an Schwachstellen identifizieren. Die meisten gefundenen Schwachstellen sind SQL-Injections. Zudem konnte auch eine XSS- und Local File Inclusion Schwachstelle gefunden werden. Da es sich bei der XSS Schwachstelle um eine Schwäche handelt, die clientseitig ausgeführt wird, eignet sich diese nicht für das aktive Eindringen in das Zielsystem. Die anderen beiden identifizierten Schwachstellen können jedoch für die nächste Phase des Penetrationstests näher betrachtet werden.

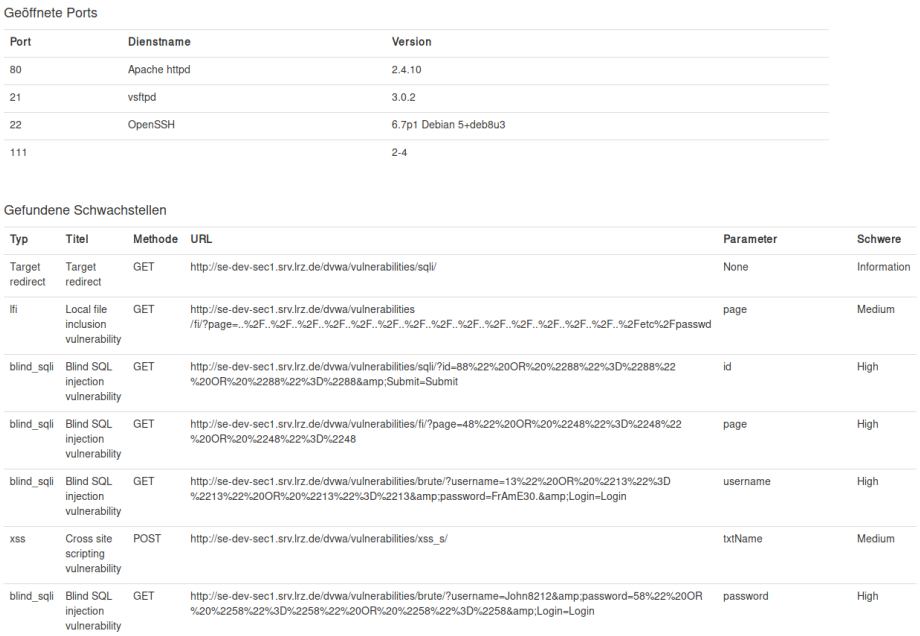


Abbildung 7.5.: Die Ergebnisse der Informationsbeschaffung

Aktive Eindringversuche

In dieser Phase wird wieder versucht, die identifizierten Schwachstellen auszunutzen. Dazu werden die Local File Inclusion und einige der SQL-Injections für den Einbruchsversuch gewählt. Wie in den vorherigen Experimenten wird als Shell wieder auf Weevely gesetzt. Nachdem dieser Auftrag abgesendet und die Ergebnisse zurück an den Server gegeben wurden, ist zu erkennen, dass die Local File Inclusion wieder erfolgreich ausgenutzt und somit eine Shell in das Zielsystem eingeschleust wurde. Bei einer der SQL-Injections konnte die Datenbank ausgelesen werden. Die SQL-Injection einer anderen Unterseite konnte vom Service nicht automatisiert ausgenutzt werden. Damit auch diese Schwachstelle überprüft werden kann, muss der Penetrationstester anhand der Information, die er vom Service erhalten hat, einen manuellen Penetrationstest durchführen. Die Ergebnisse des aktiven Eindringversuchs sind in Abbildung 7.6 dargestellt.

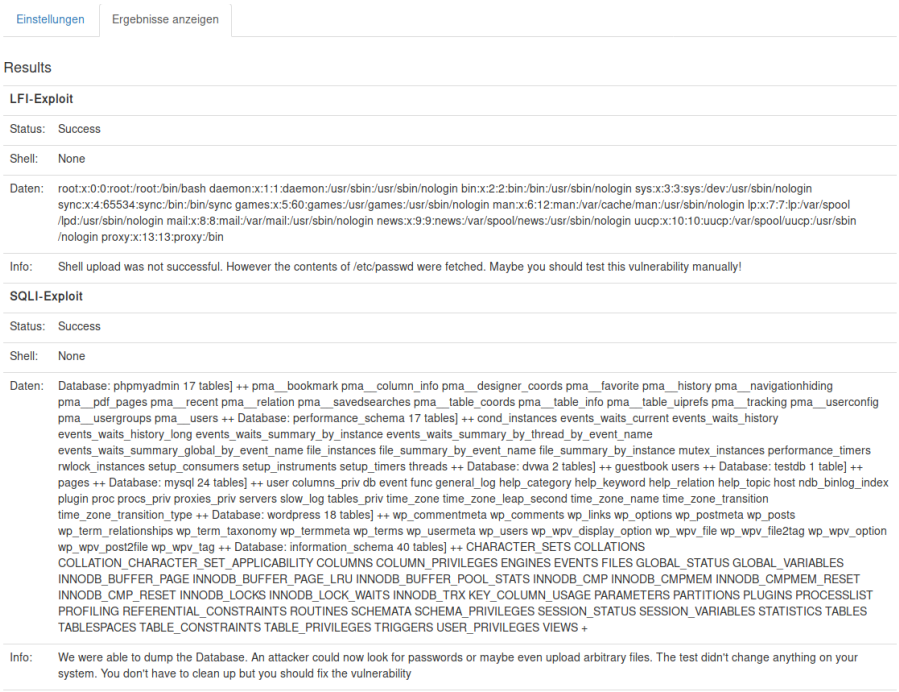


Abbildung 7.6.: Die Ergebnisse des aktiven Eindringversuchs

7.2.4. Security Document Management System des LRZ

Ein weiterer Test des Service wurde an einer Webapplikation durchgeführt, die am LRZ betrieben wird. Dieser Test soll zeigen, wie sich der Service bei einem Livesystem verhält. Abbildung 7.7 zeigt, wie die Webapplikation aussieht.

Informationsbeschaffung

Wie bei den zuvor erläuterten Experimenten wurde zur Informationsbeschaffung ein Asset-Discovery-Scan und ein Schwachstellenscan gewählt. Das Zielsystem hatte nur wenige Unterseiten, das dazu führte, dass der Test zügig beendet wurde.

Informationsbewertung

Nachdem der Test beendet wurde, konnte mit der Informationsbewertung begonnen werden. Wie Abbildung 7.8 zeigt, konnten leider auf dem Zielsystem keine Schwachstellen identifiziert werden. Lediglich der Asset-

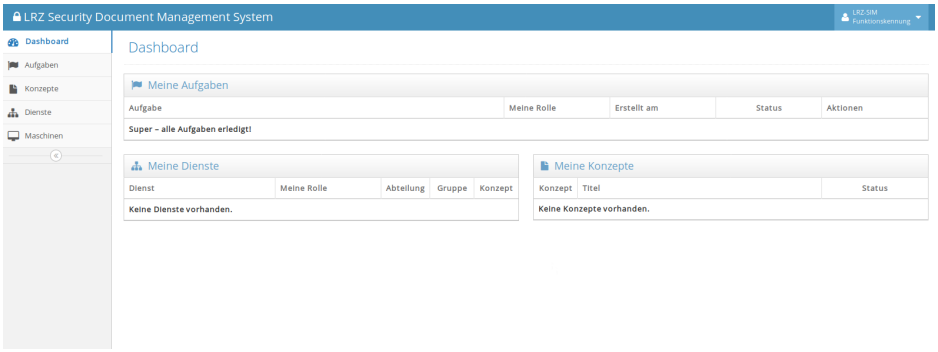


Abbildung 7.7.: Die Ansicht des Security Document Management System

Discovery-Scan erkannte, dass die beiden Ports 80 und 443 geöffnet sind. Da das Security Document Management System nur sehr wenige Unterseiten hat und auch nur wenige potentiell angreifbare Formularfelder und URL-Parameter aufweist, bietet diese Seite nur wenig Angriffsfläche.

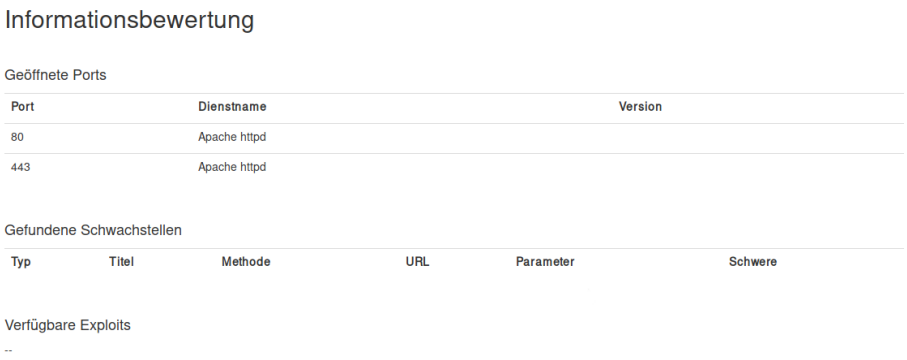


Abbildung 7.8.: Die Ergebnisse der Informationsbeschaffung

7.2.5. Projekte der Bayerischen Staatsbibliothek

Wegen vertraulicher Inhalte wurde dieser Abschnitt auf Anfrage der Bayerischen Staatsbibliothek entfernt.

7.3. Resümee

Nachdem der Service bei einer Vielzahl an Webanwendungen getestet wurde, ist zu sehen, dass die prototypische Implementierung gut funktioniert hat. Bei der selbst entwickelten Webanwendung konnte der Service alle eingebauten Schwachstellen identifizieren und auch ausnutzen.

Die WordPress Instanz mit dem verwundbaren Plugin konnte auch identifiziert und ausgenutzt werden. Für einen Produktiveinsatz müssen jedoch noch weitere bekannte Schwachstellen und Exploits für in Frage kommende Software in den Service eingepflegt werden. Wird an dieser Stelle eine große Auswahl an Exploits bereitgestellt, ist die Wahrscheinlichkeit höher, gravierende Sicherheitslücken in Webanwendungen oder betriebener Dienste zu finden.

Bei den bisherigen Experimenten waren die eingebauten Schwachstellen schon vorab bekannt. Daher wurde das Experiment mit DVWA durchgeführt, um zu sehen, wie der Service mit unbekannten Schwachstellen umgeht. Dies hat gezeigt, dass durch den Service auch Schwachstellen identifiziert werden konnten, die dem

Penetrationstester vorher noch nicht bekannt waren. Auch der aktive Einbruchsversuch konnte bei DVWA erfolgreich durchgeführt werden.

Damit auch getestet werden kann, wie der Service für Penetrationstests mit Webanwendungen umgeht, die sich im Produktiveinsatz befinden, wurde der Penetrationstest am LRZ und bei einem Großkunden des LRZ durchgeführt. Bei der Testdurchführung an einer vom LRZ angebotenen Webanwendung ist der Service gescheitert, da die Webseite wenig Angriffspotenzial geboten hat und zudem gut abgesichert war. Bei dem Test an Webanwendungen der Bayerischen Staatsbibliothek konnten jedoch Schwachstellen gefunden werden. Diese waren aber nicht kritisch und somit blieb ein aktives Eindringen ohne Erfolg. Jedoch zeigt dieses Experiment, dass der Service auch bei Webanwendungen der Kunden, die bald weltweit erreichbar sein werden, Schwachstellen finden kann. Des Weiteren hat dieses Experiment auch gezeigt, dass es unter den identifizierten Schwachstellen auch False-Positives geben kann. Daher sollten die durch den Service gefundenen Schwachstellen immer auch manuell überprüft werden, um fälschlicherweise erkannte Schwächen zu identifizieren.

8. Zusammenfassung und Ausblick

Da der Prototyp implementiert und an mehreren Beispielen evaluiert wurde, kann in diesem Kapitel die Erfüllung der Anforderungen an den Service mit dem Konzept und der Implementierung verglichen werden. Im Anschluss wird die Arbeit zusammengefasst und ein Ausblick zur Verbesserung und weiteren Entwicklung gegeben.

8.1. Erfüllung der Anforderungen

In diesem Abschnitt werden die in Kapitel 3 beschriebenen Anforderungen mit dem Konzept und der Implementierung verglichen. Dabei wird ein **o** vergeben, wenn die Anforderung erfüllt wurde. Falls eine Anforderung nicht erfüllt werden konnte, wird dies mit einem **x** gekennzeichnet.

Abschnitt	Anforderung	Konzept	Implementierung
Allgemeine Anforderungen			
3.6.1	Automatischer Start eines Penetrationstests	x	x
3.6.2	Importieren und Exportieren von CSV-Dateien	x	x
3.6.3	Benachrichtigung über die Beendigung eines Tests	o	x
3.6.4	Benutzerverwaltung und Administration durch Superusers	o	o
3.6.5	Hierarchische Benutzerrechte innerhalb einer Abteilung	o	o
3.6.6	Mehrbenutzer-Betrieb	o	o
3.6.7	Sicherheit der Anwendung	o	o
3.6.8	Mandantenfähigkeit	o	o
3.6.9	Leicht anpassbar und erweiterbar	o	o
3.6.10	Die Bedienung muss über eine GUI und API möglich sein	o	o
3.6.11	Schnittstelle zu anderen Sicherheitstools	o	o
3.6.12	Automatisierung	o	o
3.6.13	Testen einer Vielzahl von Webservern	o	o
3.6.14	Der Penetrationstest einzelner Webserver muss zeitnah beendet werden	o	o
Anforderungen zur Informationsbeschaffung			
3.7.1	Erkennung geöffneter Ports und dahinter laufender Dienst	o	o
3.7.2	Erkennung von eingesetzter Software und deren Versionsstand	o	o
3.7.3	Erkennung von bereits erfolgreichen Angriffen	o	x
Anforderungen zur Informationsbewertung			
3.8.1	Flexible Auswahl durchzuführender Tests	o	o
3.8.2	Priorisierung von Testergebnissen	o	x
Anforderungen zum aktiven Eindringen			
3.9.1	Verwendung von Exploits für bekannte Schwachstellen aller populären CMS	o	o
3.9.2	Generische Tests	o	o
3.9.3	Penetration verschiedener Betriebssysteme	o	x
3.9.4	Keine Beeinträchtigung des Netzes oder der Server	o	o
Anforderungen zur Abschlussanalyse			
3.10.1	Dokumentation der Ergebnisse	o	x
3.10.2	Errechnen des CVSS Base-Score	o	x
3.10.3	CVE-Nummer bekannter Schwachstellen	o	x
3.10.4	Exportieren von Berichten in andere Formate	o	x

Tabelle 8.1.: Vergleich der Anforderungen

Wie in Tabelle 8.1 zu sehen ist, wurden im Konzept fast alle Anforderungen erfüllt. Lediglich der automati-

sche Start eines Penetrationstests sowie der Import und Export von CSV Dateien wurde ausgelassen, da diese Funktionen Teil der Implementierung sein sollten. Diese können bei der Weiterentwicklung des Service aufgenommen werden. Aufgrund des engen Zeitrahmens für diese Arbeit konnten bei der Implementierung nicht alle Anforderungen umgesetzt werden. Auch dies kann bei der Weiterentwicklung des Service aufgenommen werden.

8.2. Zusammenfassung der Arbeit

Für den in dieser Arbeit vorgestellten Service für Penetrationstests wurden in Kapitel 3 die Anforderungen analysiert. Zunächst wurde die Ausgangssituation am LRZ erläutert und eine Abgrenzung dieses Service von einem klassischen Schwachstellenscan gezogen. Um es zu ermöglichen, die Anforderungen an den Service abzuleiten, wurde eine Umfrage erstellt, damit die Kunden und Mitarbeiter des LRZ ihre Meinung mit einfließen lassen können. Diese Umfrage hat mit 67 Teilnehmern ein aussagekräftiges Ergebnis gebracht, um daraus einen Anforderungskatalog zu erstellen. Im Anschluss wurden die Anforderungen nach Wichtigkeit analysiert und gewichtet.

Nachdem die Anforderungen an den Service erarbeitet wurden, konnten bereits existierende Arbeiten begutachtet und abgeglichen werden. Dabei wurden drei Arbeiten - Metasploit, OpenVAS und Potassium - näher erläutert und mit den Anforderungen, die im vorherigen Kapitel erarbeitet wurden, verglichen. Dabei wurde festgestellt, dass keiner der betrachteten Arbeiten alle Anforderungen erfüllt. Jedoch konnten einige Konzepte, wie ein modularer und dezentraler Aufbau in der Konzeptentwicklung übernommen werden.

Im Anschluss wurde das Konzept als der zentrale Teil dieser Arbeit ausgearbeitet, sodass die vorab definierten Anforderungen erfüllt werden. Dabei stand im Fokus, die verschiedenen Rollen der Beteiligten bei einem Penetrationstest vorzustellen und die einzelnen Phasen und Aktivitäten des Service für Penetrationstests zu erläutern. Diese Phasen wurden - angelehnt an den vom Leitfaden für Penetrationstests des BSI definierten Phasen - gegliedert. Dabei behandelt die erste Phase die Vorbereitung zu einem Penetrationstest, bei dem alle durchzuführenden Tests und die Verantwortlichen in einem Protokoll festgehalten werden. In der zweiten Phase wird so viel Information wie möglich über das Zielsystem eingeholt, um in der dritten Phase diese Information zu bewerten. In der vierten Phase wird versucht, anhand der gefundenen Informationen aktiv in das Zielsystem einzudringen. Die letzte Phase des Tests befasst sich mit der Zusammenfassung aller Ergebnisse zu einem Abschlussbericht.

Da die fünf Phasen, die der Service für Penetrationstests durchläuft, beschrieben wurden, konnte auf die Architektur des Service näher eingegangen werden. Der in dieser Arbeit beschriebene Ansatz war ein dezentraler, damit die einzelnen Komponenten flexibel im Netzwerk platziert werden können. Des Weiteren wurde auch die Möglichkeit der Verwendung einer API beschrieben, damit einerseits möglich ist, verschiedene GUIs bereitzustellen, aber auch externe Sicherheitstools wie Dr. Portscan an den Service angebunden werden können.

Nachdem das Konzept ausgearbeitet wurde, konnte der Service prototypisch implementiert werden. Dabei wurde auf die Programmiersprache Python gesetzt. Die GUI und die API wurden dabei mit dem Web-Framework Django und Django-REST-Framework umgesetzt. Bei der Entwicklung des Prototyps wurden die einzelnen Phasen, wie im Konzept beschrieben, implementiert. Des Weiteren wurde das Protokoll zur Kommunikation zwischen den einzelnen Komponenten und die Arbeitsabläufe des Service beschrieben. Eine Herausforderung bei der Implementierung war, die Komponenten so zu gestalten, dass sie auch mit einer großen Anzahl an Benutzern gut skaliert. Dies konnte gelöst werden, indem vor allem die Scannerkomponente redundant betrieben werden kann, um somit die Last auf verschiedene Scanner zu verteilen. Am Ende des Implementierungskapitels wurde ein mögliches Datenmodell vorgestellt, das beim Betrieb des Service zur Verwendung kommen kann.

Abschließend wurde der Prototyp verwendet, um die Implementierung an einigen Testobjekten zu evaluieren. Dabei wurden eine selbst entwickelte Webanwendung mit vorsätzlich eingebauten Schwachstellen, eine WordPress Instanz und eine Anwendung mit sehr vielen Schwachstellen (DVWA) für den Test herangezogen. Damit der Service auch realitätsnah evaluiert werden konnte, wurde zudem ein Penetrationstest an einer Webanwendung am LRZ durchgeführt. Des Weiteren hat sich ein Großkunde des LRZ - die Bayerische Staatsbibliothek - einverstanden erklärt, den Service an dort neu entwickelten Webanwendungen zu evaluieren.

8.3. Ausblick

Im Folgenden wird ein Ausblick gegeben, welche Komponenten noch implementiert werden müssen, damit ein Produktiveinsatz des Service am LRZ realisiert werden kann. Des Weiteren soll aufgezeigt werden, wie der Service für das Auffinden und das Verifizieren der Gefährlichkeit von Schwachstellen optimiert werden kann.

Der erste Test an verschiedenen Webanwendungen hat gezeigt, dass der Prototyp bereits gut funktioniert. Jedoch gibt es noch einige Verbesserungsmöglichkeiten, die in Betracht gezogen werden sollten. Da es möglich ist, dass der Service False-Positives zurückliefert, wäre es gut, wenn der Benutzer die Möglichkeit hat bestimmte Ergebnisse auszufiltern, die bei einem erneuten Scan nicht mehr angezeigt werden. Des Weiteren wäre gut, wenn der Penetrationstester die Freiheit besitzt, Ergebnisse beispielsweise nach der Gefährlichkeit zu sortieren.

Da aus Zeitgründen die Berichterstellung nicht mehr implementiert werden konnte, gehört dies auch zur späteren Weiterentwicklung des Service für Penetrationstests. Derzeit werden alle Ergebnisse im XML-Format auf der Festplatte des Servers gespeichert. Das Modul zur Erstellung der Abschlussberichte muss diese XML-Dateien auslesen und daraus einen einheitlichen Bericht im PDF Format für die verschiedenen Zielgruppen erstellen. Diese können dem Benutzer dann zum Download angeboten werden. Es wäre auch noch denkbar, dem Benutzer ein Dashboard zur Verfügung zu stellen, das eine Zusammenfassung aller durchgefürten Penetrationstests anzeigt. Somit kann ein guter Überblick der Sicherheitslage in der Organisation geschaffen werden.

Der Schwachstellenscan ist in dem Prototyp derzeit für eine breite Masse an Webanwendungen ausgelegt. Damit auch detailreiche Scans von verschiedenen CMS ermöglicht werden, sollten noch weitere Module entwickelt werden, die auf CMS spezialisiert sind. Hierbei können Tools wie WPScan oder JoomlaScan verwendet werden, die bereits in dieser Arbeit vorgestellt wurden. Auf diesem Weg können angreifbare Plugins und Themes der CMS besser untersucht werden.

Bevor der Service im LRZ zum Einsatz kommt, sollten weitere Einbruchswerkzeuge und Exploits hinzugefügt werden, damit die Penetration gefundener Schwachstellen zuverlässiger wird.

Bei dieser Arbeit lag der Fokus auf Schwachstellen in Webanwendungen. Jedoch wäre auch denkbar, dies zu erweitern, damit der Service auch andere Netzwerkkomponenten wie Router, Switches oder auch Firewalls auf Verwundbarkeiten und Fehlkonfigurationen untersuchen kann. Diese Erweiterung lässt sich mit Hilfe des in Kapitel 5 beschriebenen Konzepts gut realisieren. Ob für diese Erweiterung Änderungen am Konzept vorgenommen werden müssen, bedarf jedoch weiterer Analyse.

A. Anhang

A.1. Code Documentation

Damit eine spätere Weiterentwicklung erleichtert wird, ist im Folgenden die Dokumentation des Programmcodes aufgelistet.

fapts

Code Documentation

June 19, 2017

Contents

1 Module fapts.constants

This is FAPTS constants file

1.1 Variables

Name	Description
SCANNER_PATH	**Value:** `'scanner_modules/'`
EXPLOITER_PATH	**Value:** `'exploit_modules/'`
MAX_THREADS	**Value:** `10`
DELIMITER	**Value:** `'|'`
TYPE_DELIMITER	**Value:** `'#'`
SCANNER_FREE	**Value:** `'1'`
SCANNER_WORKING	**Value:** `'0'`
IS_FREE	**Value:** `'0'`
SCAN_JOB	**Value:** `'1'`
EXPLOIT_JOB	**Value:** `'2'`
STATUS	**Value:** `'4'`
FINISHED_SCAN	**Value:** `'6'`
ERROR	**Value:** `'99'`
SCAN_STARTED	**Value:** `'3'`
WORKING	**Value:** `'5'`
INSIDE_SCAN	**Value:** `0`
OUTSIDE_SCAN	**Value:** `1`
__package__	**Value:** `None`

2 Module fapts.scanner.daemon

2.1 Functions

handle_scans(*scanTypes*, *scanData*)

This function handles scans for information gathering. It is executed in a new thread. The function takes the following parameters:

Parameters

`scanTypes:` This parameter is an Array which holds a list of scan modules to be executed

`scanData:` This is the ScanData object. It holds the values for userID, scanID, target etc.

Return Value

The function returns the results of all executed scans in XML-Format

handle_exploits(*scanData*, *exploit_info*)

This function handles exploit jobs for the penetration phase. It is executed in a new thread. The function takes the following parameters:

Parameters

`scanData:` This is the ScanData object. It holds all relevant Information like the scanID, userID etc.

`exploit_info:` This parameter is an Array which holds a list of exploit modules to be executed

Return Value

The function returns the results of all executed exploit jobs in XML-Format

run()

This function is the main loop which waits for new connections from the server and handles those connections

2.2 Variables

Name	Description
HOST	**Value:** `''`
PORT	**Value:** `9009`
MAX_CONNECTIONS	**Value:** `5`
__package__	**Value:** `'fapts.scanner'`
__warningregistry__	**Value:** {`('Not importing directory \'/home/bruzzzla/PycharmProjec...`

3 Module fapts.scanner.ip

3.1 Variables

Name	Description
__package__	**Value:** `'fapts.scanner'`

3.2 Class CheckIP

This class checks the validity of an URL or IP and does some useful things like extract the hostname, protocol path of a given resource.

3.2.1 Methods

__init__(*self*, *address*)

get_ip(*self*)

This function checks if the input is an IP

Return Value
The IP, hostname or None if invalid input

get_hostname(*self*)

This function returns the hostname of a given URL

Return Value
The hostname

get_proto(*self*)

This function returns the used protocol.

Return Value
The protocol

get_path(*self*)

This function returns the path of an URL

Return Value
The path of the url

is_url(*self*)

This function checks if the input is an URL

Return Value
True or False

is_ip(*self*)

This function checks if the input is an IP

Return Value
True or False

4 Module fapts.scanner.scandata

4.1 Variables

Name	Description
__package__	**Value:** `'fapts.scanner'`

4.2 Class Scandata

This class manipulates and extracts informations from the scan data

4.2.1 Methods

__init__(*self*, *xml*)

Constuctor method for scan data

Parameters

`xml`: This parameter holds the xml string

getUserID(*self*)

This function extracts the user ID

Return Value

User ID

getScanID(*self*)

This function extracts the scan ID

Return Value

Scan ID

getURL(*self*)

This funciton extracts the URL

Return Value

Target URL

getUsername(*self*)

This funciton extracts the username for authenticated scans

Return Value

username

getPassword(*self*)

This funciton extracts the password for authenticated scans

Return Value
password

getUserField(*self*)

This funciton extracts the user field for authenticated scans

Return Value
user field

getPassField(*self*)

This funciton extracts the password field for authenticated scans

Return Value
password field

getCheckStr(*self*)

This funciton extracts a check string for authenticated scans. This is important to check if the session is still active

Return Value
check string

getLoginUrl(*self*)

This funciton extracts the login url for authenticated scans

Return Value
login url

buildAuth(*self*)

getScanTypes(*self*)

This function extracts the scan types

Return Value
Scan types

getXMLFile(*self*)

This function returns the entire XML string

Return Value
XML string

addScanTypes(*self*, *scanTypes*)

This function takes the scan types and adds new scan types

Parameters

`scanTypes:` An array with scan types

Return Value

Returns the newly built xml string including the new scan types

removeScanType(*self*, *scanType*)

This function removes scan types form the XML string

Parameters

`scanType:` An array with scantypes

Return Value

Returns the newly built xml string without the defined scan types.

getExploitTypes(*self*)

This function extracts the exploit types

Return Value

Returns the exploit types

is_auth(*self*)

5 Module fapts.server.daemon

5.1 Functions

run()

This function is the main loop for the server side daemon. It checks for new jobs in the database and sends them to the scanner.

5.2 Variables

Name	Description
cur_path	**Value:** `'/home/bruzzzla/PycharmProjects/fapts/fapts/server'`
config	**Value:** `configparser.ConfigParser()`
SLEEP	**Value:** `u'60'`
result_path	**Value:** `u'server/storage/results/'`
install_path	**Value:** `u'/home/bruzzzla/PycharmProjects/fapts/'`
path_to_result	This funciton collects the results from the scanners **Value:** `u'/home/bruzzzla/PycharmProjects/fapts/fapts/server/stora...`
__package__	**Value:** `'fapts.server'`
__warningregistry__	**Value:** `{(u'You passed a bytestring as 'filenames'. This will not...`

6 Module fapts.server.db_handler

6.1 Functions

open_db_connection()

This function opens a connection to the database

Return Value

It returns the database object

6.2 Variables

Name	Description
config	**Value:** `configparser.ConfigParser()`
DB_USER	**Value:** `u'ps_user'`
DB_PASS	**Value:** `u'securepasswd'`
DB_NAME	**Value:** `u'ps_db'`
DB_HOST	**Value:** `u'127.0.0.1'`
__package__	**Value:** `'fapts.server'`
__warningregistry__	**Value:** {`(u'You passed a bytestring as 'filenames'. This will not...`

7 Module fapts.server.manager.manager

7.1 Functions

get_scanner_connection(*placement*)

This function tries to connect to one of the scanners. If it can not connect to one of the scanners, it will write to the database that the specific scanner is offline and sends a mail to the superuser.

Parameters

`placement:` This parameter tells this function if the scanner should be inside oder outside the local network. Values could be 0 or 1

send_scan_job(*placement*, *xmlRequest*, *mode*)

This method creates a XML request and sends it to the scanner.

Parameters

`placement:` This parameter tells if the scan should be executed inside or outside of the local network.

`xmlRequest:` This parameter holds the XML request that should be sent.

`mode:` This parameter defines if it is a scan job or an exploit job.

Return Value

The return value is True if the job was sent successfully and False if not.

7.2 Variables

Name	Description
config	**Value:** `configparser.ConfigParser()`
base_path	**Value:** `u'/home/bruzzzla/PycharmProjects/fapts/'`
__package__	**Value:** `'fapts.server.manager'`
__warningregistry__	**Value:** {`('Not importing directory \'/home/bruzzzla/PycharmProjec...`

8 Module fapts.server.models

8.1 Class Profile

django.db.models.Model
└ **fapts.server.models.Profile**

Database model for user profile

8.1.1 Class Variables

Name	Description
user	**Value:** `models.OneToOneField(User, on_delete= models.CASCADE)`
role	**Value:** `models.CharField(max_length= 30)`
organisation	**Value:** `models.CharField(max_length= 50)`
organisation_id	**Value:** `models.CharField(max_length= 50, primary_key= True)`
ipRange	**Value:** `models.GenericIPAddressField()`

8.2 Class Report

django.db.models.Model
└ **fapts.server.models.Report**

Database model for the reports

8.2.1 Class Variables

Name	Description
organisation_id	**Value:** `models.CharField(max_length= 100)`
type	**Value:** `models.CharField(max_length= 30)`
filename	**Value:** `models.CharField(max_length= 100)`
generation_date	**Value:** `models.DateField()`

8.3 Class Scanner

django.db.models.Model
└ **fapts.server.models.Scanner**

Database model for the scanners

8.3.1 Class Variables

Name	Description
scanner_id	**Value:** `models.IntegerField(primary_key= True)`
status	**Value:** `models.IntegerField()`
scanner_ip	**Value:** `models.GenericIPAddressField()`
scanner_port	**Value:** `models.IntegerField()`
placement	**Value:** `models.IntegerField()`

8.4 Class ScanJob

django.db.models.Model

fapts.server.models.ScanJob

Database model for the scan job

8.4.1 Class Variables

Name	Description
user_id	**Value:** `models.IntegerField()`
scan_id	**Value:** `models.CharField(max_length= 40)`
status	**Value:** `models.CharField(max_length= 10)`
scanner_id	**Value:** `models.IntegerField()`

8.5 Class Targets

django.db.models.Model

fapts.server.models.Targets

Database model for the targets

8.5.1 Class Variables

Name	Description
user_id	**Value:** `models.IntegerField()`
url	**Value:** `models.CharField(max_length= 500)`
scan_id	**Value:** `models.CharField(max_length= 40)`

8.6 Class Tests

django.db.models.Model
└ **fapts.server.models.Tests**

Database model for scan types

8.6.1 Class Variables

Name	Description
titel	**Value:** `models.CharField(max_length= 500)`
name	**Value:** `models.CharField(max_length= 100)`
modul_name	**Value:** `models.CharField(max_length= 100)`
class_name	**Value:** `models.CharField(max_length= 100)`
type	**Value:** `models.IntegerField()`

8.7 Class Exploits

django.db.models.Model
└ **fapts.server.models.Exploits**

Database model for exploits

8.7.1 Class Variables

Name	Description
titel	**Value:** `models.CharField(max_length= 500)`
name	**Value:** `models.CharField(max_length= 100)`
vendor	**Value:** `models.CharField(max_length= 100)`
version	**Value:** `models.CharField(max_length= 20)`
modul_name	**Value:** `models.CharField(max_length= 100)`
class_name	**Value:** `models.CharField(max_length= 100)`

8.8 Class Exploitable

django.db.models.Model
└ **fapts.server.models.Exploitable**

Database model for vulnerable applications

8.8.1 Class Variables

Name	Description
title	**Value:** `models.CharField(max_length= 200)`
extension_name	**Value:** `models.CharField(max_length= 100)`
version	**Value:** `models.CharField(max_length= 20)`
test_name	**Value:** `models.CharField(max_length= 100)`

9 Module fapts.server.reporters.kickoff

9.1 Functions

getGeneralInfo(*data*, *item*)

This function aggregates the general information for the kickoff-protocol

Parameters

`data:` This parameter holds the data as dict

`item:` This parameter holds the placeholders for the odf template

getParicipants(*data*, *item*)

This function aggregates the participants of the kickoff-meeting

Parameters

`data:` This parameter holds the data as dict

`item:` This parameter holds the placeholders for the odf template

getTestInfo(*data*, *item*)

This function aggregates the information for the test object

Parameters

`data:` This parameter holds the data as dict

`item:` This parameter holds the placeholders for the odf template

getAttacks(*data*, *testItems*)

This function aggregates the attacks that the user has chosen

Parameters

`data:` This parameter holds the data as array

`testItems:` This parameter holds the placeholders for the odf template

getServices(*data*, *testServices*)

This function aggregates the services to be tested

Parameters

`data:` This parameter holds the data as array

`testServices:` This parameter holds the placeholders for the odf template

buildReport(*djangoRequest*, *output*)

This function finally builds the report

Parameters

`djangoRequest:` The request form Django containing GET/POST fields

`output:` This parameter holds the filename for the report

Return Value

Returns true or false

9.2 Variables

Name	Description
config	**Value:** `configparser.ConfigParser()`
KICKOFF_TEMPLATE_LOCATION	**Value:** `u'server/storage/docs/Vorlage_Kickoffprotokoll.odt'`
__package__	**Value:** `'fapts.server.reporters'`
__warningregistry__	**Value:** `{(u'You passed a bytestring as 'filenames'. This will not...`

9.3 Class Item

object
└
fapts.server.reporters.kickoff.Item

This is a helper class to work with OpenOffice placeholders

9.3.1 Methods

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __init__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

9.3.2 Properties

Name	Description
Inherited from object	
__class__	

10 Module fapts.server.views

10.1 Functions

loginUser(*request*)

This function handles the user login

Parameters

`request:` The Django request object. It holds GET/POST values

Return Value

Render the login page

userLogout(*request*)

This function handles logouts

Parameters

`request:` The Django request object. It holds GET/POST values

Return Value

Redirects to the welcome page

index(*request*)

This function displays the welcome page after login

Parameters

`request:` The Django request object. It holds GET/POST values

Return Value

Render index page after login

preparation(*request*)

This function sets up everything for the preparation phase. This page is only accessible for logged in users.

Parameters

`request:` The Django request object. It holds GET/POST values

Return Value

Render the view for preparation phase

gathering(*request*)

This method creates the XML request to be sent to the scanner. It also renders the views for the information gathering phase

Parameters

`request`: The Django request object. It holds GET/POST values

Return Value

Shows the views

review(*request*)

This method displays the view for reviewing the information collected in the information gathering phase.

Parameters

`request`: The Django request object. It holds GET/POST values

Return Value

Renders the view

penetration(*request*)

This method prepares the exploitation of a target system and sends it to the scanner. It also displays the view for the penetration phase.

Parameters

`request`: The Django request object. It holds GET/POST values

Return Value

Renders the view for penetration phase

report(*request*)

This function creates a report from the given information. This function needs to be implemented.

Parameters

`request`: The Django request object. It holds GET/POST values

Return Value

Render the view for generated reports.

10.2 Variables

Name	Description
cur_path	**Value:** `os.path.dirname(os.path.realpath(__file__))`

Index

A.2. API Documentation

Im Folgenden wird die Dokumentation der integrierten API aufgelistet.

fapts

API Documentation

June 19, 2017

Contents

1 Module fapts.api

Routes for API start here!

1.1 Functions

get_tests(*request*)

This route returns the scanner and exploiter which are available in the database. The values for the type are as follows: 1: for scanners 2: for exploiters

api_gathering(*request*)

This route starts the information gathering by creating the XML-Requests and send it to the server. It takes only POST parameters.

Parameters

`username:` The username

`password:` The users password

`url:` The target url

`modes:` This specifies if the test should be started from inside or outside the local network. The values are 0 for inside and 1 for outside. If you want both you can send modes comma separated.

`tests:` This is a comma separated list of the scan names to invoke (eg. asset, vuln). You got that from the get_tests route.

Return Value

The scan ID

api_review(*request*)

This route returns the the results for the scan- and exploit jobs. It takes only POST parameters

Parameters

`username:` The username for authentication

`password:` The password for the user

`scan_id:` The current scan ID

Return Value

The results for the scans in JSON format

api_penetration(*request*)

This route starts a penetration job. It takes only POST parameters.

Parameters

`shell:`	The shell which should be used (weevely or webshell) This parameter is optional as long as mode is not full penetration
`mode:`	The mode to execute the exploit job (full or verify)
`exploit:`	This is a list of exploits to execute
`scan_id:`	This is the current scan ID
`username:`	The username for authentication
`password:`	The password for the given user
`vulnerabilities:`	This is a comma separated list of found vulnerabilities

api_get_penetration_results(*request*)

This route returns the result for the penetration job. It takes only POST parameters

Parameters

`username:`	The username for authentication
`password:`	The password for the given user
`scan_id:`	The scan ID for the current session

Return Value

The results for the penetration in JSON format

Index

A.3. Kickoff Protokoll

In diesem Abschnitt wird das Kickoff-Protokoll gezeigt, welches bei einem Penetrationstest an der Bayerischen Staatsbibliothek entstanden ist.

Kickoff Protokoll

Allgemeine Informationen

Allgemeines

Organisation	Bayerische Staatsbibliothek
Ort	München
Datum	18.07.2017

Teilnehmer

Name	Abteilung	Rolle
Fedor Bochow	Bayerische Staatsbibliothek	Sicherheitsverantwortlicher
Beatrice Popa, Elisabeth Eder	Bayerische Staatsbibliothek	Administrator
Manuel Kauschinger	LMU	Penetrationtester

Informationen zum Testablauf

Testgegenstand

IP der Webserver	https://mdz-typo3.wa1cms-stg.digitale-sammlungen.de
Hostname	--

Details zum Test

Durchzuführende Tests	Alle verfügbaren Tests
Anzugreifende Dienste	Webserver
Aggressivität	mittel
Plazierung des Test	Innen
Art des Test	Blackbox

Informationen zum Tester

Name	Manuel Kauschinger
Zugehörigkeit	LMU
Kennung	-

Erlaubnis zur Testdurchführung

Name	Rolle	Abteilung
-	IT-Manager	
Fedor Bochow	Sicherheitsverantwortlicher	

Unterschriften zur Genehmigung und Durchführung

_____ _____

Sicherheitsverantwortlicher **IT-Manager**

_____

Penetrationtester

A.4. Umfrage

Im Folgenden werden alle Ergebnisse der Umfrage aufgelistet.

Results

Survey 761196

Number of records in this query:	67
Total records in survey:	67
Percentage of total:	100.00%

Field summary for A1

Werden Content Management Systeme (CMS) eingesetzt oder werden Webseiten selbst entwickelt?

Answer	Count	Percentage
Es werden CMS eingesetzt (A1)	28	41.79%
Webanwendungen werden selbst entwickelt (A2)	7	10.45%
Es wird beides eingesetzt (A3)	26	38.81%
No answer	6	8.96%
Not displayed	0	0.00%

Field summary for A15

Falls CMS eingesetzt werden, welche Produkte verwenden Sie?

Answer	Count	Percentage
WordPress (SQ001)	8	11.94%
Joomla! (SQ002)	8	11.94%
Typo3 (SQ003)	24	35.82%
Andere (SQ004)	30	44.78%

Field summary for A5

In welchem Zyklus wird nach Updates gesucht und diese eingespielt?

Answer	Count	Percentage
Täglich (A1)	7	10.45%
Wöchentlich (A2)	12	17.91%
Monatlich (A3)	12	17.91%
Halbjährlich (A4)	9	13.43%
Nie (A5)	4	5.97%
No answer	23	34.33%
Not displayed	0	0.00%

Field summary for A7

Welches Betriebssystem wird für die Webserver verwendet?

Answer	Count	Percentage
Microsoft Windows (A1)	8	11.94%
Linux (A2)	45	67.16%
BSD (A3)	1	1.49%
Gemischt (A4)	5	7.46%
No answer	8	11.94%
Not displayed	0	0.00%

Field summary for A2

Werden Webanwendungen basierend auf Ajax eingesetzt?

Answer	Count	Percentage
Ja (A1)	26	38.81%
Nein (A2)	24	35.82%
No answer	17	25.37%
Not displayed	0	0.00%

Field summary for A6

Werden SOAP/REST Web Services eingesetzt?

Answer	Count	Percentage
Ja (A1)	20	29.85%
Nein (A2)	18	26.87%
Weiß ich nicht (A3)	17	25.37%
No answer	12	17.91%
Not displayed	0	0.00%

Field summary for A8

Werden Web Application Firewalls eingesetzt?

Answer	Count	Percentage
Ja (A1)	7	10.45%
Nein (A2)	25	37.31%
Teilweise (A3)	5	7.46%
Weiß ich nicht (A4)	24	35.82%
No answer	6	8.96%
Not displayed	0	0.00%

Field summary for A9

Werden Intrusion Detection Systeme eingesetzt?

Answer	Count	Percentage
Ja (A1)	7	10.45%
Nein (A2)	31	46.27%
Weiß ich nicht (A3)	22	32.84%
No answer	7	10.45%
Not displayed	0	0.00%

Field summary for F7

Wie viele Webserver werden auch mit IPv6 betrieben?

Answer	Count	Percentage
Viele (A1)	9	13.43%
Nur vereinzelt (A2)	13	19.40%
Keine (A3)	24	35.82%
No answer	21	31.34%
Not displayed	0	0.00%

Field summary for A11

Welche Schwachstellen sind bei Ihren Webservern in den letzten Jahren am häufigsten vorgekommen?

Answer	Count	Percentage
Answer	30	44.78%
No answer	37	55.22%
Not displayed	0	0.00%

ID	Response
5	Automatisierte Hacks durch veraltete Software.
16	Nicht wirklich bekannt. Anscheinend gab es keine DoS-Angriffe. Schwachstellen könnten in den relativen alten Apache- und OpenSSL-Versionen schlummern, die aus der eingesetzten SLES 11.4-Distribution stammen.
26	Default-Paßwörter von Datenbanken wurden nicht geändert (oder erst gar nicht gesetzt).
42	XSS weakness
45	Austausch von Dateien gegen solche mit anzüglichem Inhalt
47	CSRF XSS Öffentliche Server ohne SSH-Passwort
49	1. daten die sind öffentlich (und solten nicht so) zB: AWStats (jetzt dieses Problem ist gelöst) 2. ungesicherte Ordner in die Dateien Verzeichniss
50	Kein bekannten
60	PHP, Apache Module, SSL/SSH, Allgemeine Schwachstellen in der Software der Server
64	Keine bekannt.
67	XSS -> Typo3
73	keine bekannt
75	Veraltete Software (Apache, Tomcat usw.), veraltete Protokolle
77	Apache-Überlastung wegen DDos Angriffen.
80	Noch keine - neue Webpräsentation geht in Kürze online.
91	XSS, SQL Injection, Information Disclosure, Remote Code Execution
97	Angriffe auf das Joomla! CMS über php
100	keine bekannt
105	Übernahme eines Joomla 1.x-Systems über ein Editor-plugin. Reparatur war leicht, da es sich um virtuelles System handelte.
107	PHP-basierte Schwachstellen diverser Art in Drupal
110	Insecure Direct Object Reference Insecure Communications (kommt inzwischen bei uns kaum noch vor)
112	Wir benutzen Fiona. Direkt sind uns keine Schwachstellen aufgefallen, das muss aber nichts heißen. Die genauen Informationen (siehe auch oben), sind sicher von den Fiona-Verantwortlichen an der LMU verfügbar
118	Bisher noch keine
119	keine bekannt
121	php
126	Cross-Site-Scripting Preisgabe von Systemdaten (OS- und Webserver-Versionen)
127	Schwachstellen des LAMP-Servers (zB Heartbleed, PHP) SQL injection Cross site scripting
134	Meldungen für TYPO3: XSS
137	Fehler im php-code Alles was bei apache und php anfiel.
138	- CMS-Sicherheitslücken (Drupal, OCS)

Field summary for A10

Müssen geplante Penetrationstests genehmigt werden? Wer muss informiert werden?

Answer	Count	Percentage
Answer	31	46.27%
No answer	36	53.73%
Not displayed	0	0.00%

ID	Response
5	"Müssen" in dem Sinne, dass es dafür einen klar definierten Prozess gibt, nicht.
11	Ja, um den Produktivbetrieb nicht zu gefährden. Penetrationstests sollten zunächst auf "baugleichen" Test-Umgebung durchgeführt werden.
16	Keine Genehmigung, aber bitte um Information an sim@lrz.de
26	Natürlich müssen solche Tests genehmigt oder zumindest abgesprochen werden. Informiert werden müssen wir (also die Admins). Wir sprechen das dann mit der BVB-Verbundzentrale ab, und die wiederum muß das mit den Bibliotheken koordinieren.
37	Projektleiter IT-Gruppe Geisteswissenschaften (als Dienstleister)
42	LRZ hostmaster
47	Ja Sysadmins, Projektmanager, Gruppenleiter betroffener Systeme
49	Ja. Wenn möglich, möchten wir eine neue Seite prüfen, vor GoLive, in die nächste 2 Wochen. Zu kontaktieren: joachim.mattner@mri.tum.de . Danke
50	it@ifkw.lmu.de
60	Ja, Dienststellenleiter und IT Leiter müssen vorab informiert werden.
64	Ja. Das Team.
67	ja, RBG der Fakultät für Physik
73	nein
75	Ja, von Kunden der gehosteten Systeme
77	Ja. BVB.
80	M.E. keine Genehmigung nötig. Info an: Stefan.Schmidt@zsm.mwn.de
91	Ja, Rechnerbetriebsgruppe
97	Genehmigung nicht nötig. Info bitte an Ludwig.Hoegner@tum.de
100	nein
105	Ist o.K. Niemand muss informiert werden, eine Mitteilung über die Ergebnisse wäre natürlich sehr hilfreich.
107	Bitte einfach testen.
109	Technischer Kontakt der Arbeitsgruppe
110	- Dienststellenleiter/in - und der bzw. die das Projekt betreuende Techniker/in
111	Administrator und Entwickler
112	Ich vermute ja. s.o. Fiona Verantwortliche an der LMU.
117	Nein
119	meines Wissens nicht
121	sysadmin, webmaster
127	ja, Info an den Betreiber bzw. Admin des Webservers.
138	Webserver-Administratoren
145	nein

Field summary for B5

Soll es Berichte für verschiedene Zielgruppen geben? (Administrator, Netzverantwortlicher ect.)

Answer	Count	Percentage
Ja (A1)	43	64.18%
Nein (A2)	15	22.39%
No answer	9	13.43%
Not displayed	0	0.00%

Field summary for B2

Sollen gefundene Information, wie Versionsnummern der eingesetzten Anwendungen und Diensten in dem Bericht aufgelistet werden?

Answer	Count	Percentage
Ja (A1)	54	80.60%
Nein (A2)	4	5.97%
No answer	9	13.43%
Not displayed	0	0.00%

Field summary for B6

Soll im Bericht der CVSS-Base-Score der gefundenen Schwachstellen angezeigt werden?

Answer	Count	Percentage
Ja (A1)	49	73.13%
Nein (A2)	2	2.99%
No answer	16	23.88%
Not displayed	0	0.00%

Field summary for B3

Sollen im Bericht die CVE-Information von bekannten Schwachstellen angezeigt werden?

Answer	Count	Percentage
Ja (A1)	50	74.63%
Nein (A2)	2	2.99%
No answer	15	22.39%
Not displayed	0	0.00%

Field summary for B4

Eigene Anregungen?

Answer	Count	Percentage
Answer	5	7.46%
No answer	62	92.54%
Not displayed	0	0.00%

ID	Response
5	Ich gebe Feedback zur Umfrage via dieser Box: Die Fragen sind unklar formuliert - was bedeutet denn "sollen"? Es gibt bei uns (Betreiber Webhosting am LRZ mit ca. 700 Kunden) derzeit keine Pläne, so etwas umzusetzen. Es klingt sehr interessant, Kunden in Bezug auf die Sicherheit ihrer Software besser beraten zu können - aufgrund der dünnen Personaldecke hier im Team ist dies momentan aber sehr unrealistisch.
11	Es wäre schön, wenn neben der Liste der Schwachstelle auch Hinweise gegeben werden, wie eine Schwachstelle "behoben" werden kann.
60	Sehr gute Idee mit dem Scannen von Webanwendungen!
112	Ich glaube, dass dringend auch ein Szenario der verschiedenen Angriffsflächen für Administratoren (ich bin Master User) geben sollte. Die Kompromittierung der Master-User-Kennung kann m.E. nicht zu gering eingeschätzt werden. Ich bin mir bewusst, dass Ihre Initiative in eine andere Richtung zielt, wäre aber gern bereit, bei Pen-Tests gegen Administratoren mitzuwirken.
118	Bitte das Dokument so erstellen, dass auch unerfahrene Nutzer mit wenigen Fachkenntnissen nicht über Fachbegriffe stolpern bzw. das Erklärungen aufgeführt werden.

Field summary for F5

Soll die Möglichkeit geschaffen werden bestimmte Ergebnisse aus dem Abschlussbericht auszufiltern?

Answer	Count	Percentage
Ja (A1)	24	35.82%
Nicht nötig (A2)	32	47.76%
No answer	11	16.42%
Not displayed	0	0.00%

Field summary for F2

Soll es möglich sein, Penetrationstests zu einem geplanten Zeitpunkt automatisch auszuführen?

Answer	Count	Percentage
Ja (A1)	42	62.69%
Nicht nötig (A2)	14	20.90%
No answer	11	16.42%
Not displayed	0	0.00%

Field summary for F8

Soll es die Möglichkeit geben Berichte in andere Formate, wie PDF,ODP,HTML zu exportieren?

Answer	Count	Percentage
Ja (A1)	41	61.19%
Nicht nötig (A2)	18	26.87%
No answer	8	11.94%
Not displayed	0	0.00%

Field summary for F10

Soll es ermöglicht werden die Webanwendungen auf bereits erfolgreiche Angriffe zu untersuchen? (Suche nach Webshells, Hacked by ect.)

Answer	Count	Percentage
Ja (A1)	52	77.61%
Nicht nötig (A2)	3	4.48%
No answer	12	17.91%
Not displayed	0	0.00%

Field summary for F9

Soll es Schnittstellen zu anderen Anwendungen geben?

Answer	Count	Percentage
Answer	3	4.48%
No answer	64	95.52%
Not displayed	0	0.00%

ID	Response
5	Feedback zu den Fragen: "Soll" habe ich jetzt interpretiert als "wäre wünscheswert, falls so ein Tool für unser Webhosting entwickelt würde". On-Demand-Penetrationstests sehe ich kritisch, da wir das gerne selbst steuern würden - und sonst ggf. einen legitimen Penetrationstest als einen Angriffsversuch werten und ggf. sperren.
91	Nicht nötig
119	nein

Field summary for F1

Welche Art von Graphical User Interface ist gewünscht?

Answer	Count	Percentage
Im Webbrowser (A1)	51	76.12%
Lokal auf dem PC als Desktopanwendung (A2)	3	4.48%
Kommandokonsole (A3)	1	1.49%
No answer	12	17.91%
Not displayed	0	0.00%

Field summary for F6

Wie lange darf ein umfassender Test maximal laufen? Gibt es Begrenzungen?

Answer	Count	Percentage
< 15 Minuten (A1)	5	7.46%
< 30 Minuten (A2)	8	11.94%
< 1 Stunde (A3)	5	7.46%
< 2 Stunden (A4)	9	13.43%
Beliebig (A5)	29	43.28%
No answer	11	16.42%
Not displayed	0	0.00%

Field summary for F11

Haben Sie eigene Anregungen die mit einfließen sollen?

Answer	Count	Percentage
Answer	8	11.94%
No answer	59	88.06%
Not displayed	0	0.00%

ID	Response
11	Manuelles starten der Tests wäre wünschenswert. Anzeige der Änderung zwischen mehreren Testläufen als Kurzbericht, wenn möglich, um zu prüfen ob Konfigurationsänderungen einen positiven effekt gebracht haben.
45	solange der Test die Erreichbarkeit nicht wesentlich einschränkt spilet die Testdauer keine Rolle
47	Auch Bedienung über REST-API wäre toll.
50	Die mögliche Dauer des Tests hängt von der Uhrzeit ab.
112	Ich finde diese Initiative sehr gut und werde bzw. würde sie auch weiterhin unterstützen. In letzer Zeit gibt es immer wieder Meldungen, dass Rechner gezielt angegriffen werden, vor allem wenn sie sich gerade nicht hinter dem MWN "verbergen" können. Ich würde mir stark eine Art "Bringen Sie uns unseren Rechner und wir stellen Ihnen alle möglichen Fallen und sehen, ob der Rechner ausreichend gesichert ist und Sie über ein ausreichendes Sicherheitsbewusstsein verfügen"-Tests wünschen. Das ist sehr umfangreich, könnte aber helfen die wissenschaftliche Community und deren IT-Infrastruktur weiter zu stärken.
119	keine
126	Teile der Webseite ausklammern (da z.B. durch Whitelisting geschützt). Möglichkeit, Anmeldedaten anzugeben, damit auch interne Bereiche (mit Anmeldung) getestet werden können.
137	Auswahlmöglichkeit, ob potentiell destruktive Tests ausgeschlossen werden sollen. Nicht jeder will gleich sein Produktionssystem abschiessen.

Field summary for P1

Wie wichtig würden Sie es einstufen, dass anhand von gefundenen Versionsnummern bekannte Exploits auf ihre Wirksamkeit getestet werden?

Answer	Count	Percentage	Sum
1 (1)	4	5.97%	11.94%
2 (2)	4	5.97%	
3 (3)	7	10.45%	10.45%
4 (4)	15	22.39%	
5 (5)	29	43.28%	65.67%
No answer	0	0.00%	
Not displayed	8	11.94%	
Arithmetic mean	0		
Standard deviation	0		
Sum (Answers)	59	100.00%	100.00%
Number of cases		0%	

Field summary for P2

Wie wichtig ist Ihnen die Überprüfung von Fehlkonfigurationen des Webservers?

Answer	Count	Percentage	Sum
1 (1)	2	2.99%	2.99%
2 (2)	0	0.00%	
3 (3)	2	2.99%	2.99%
4 (4)	9	13.43%	
5 (5)	47	70.15%	83.58%
No answer	0	0.00%	
Not displayed	7	10.45%	
Arithmetic mean	0		
Standard deviation	0		
Sum (Answers)	60	100.00%	100.00%
Number of cases		0%	

Field summary for P4

Wie wichtig würden Sie einen Test auf schwache Passwörter einstufen?

Answer	Count	Percentage	Sum
1 (1)	6	8.96%	17.91%
2 (2)	6	8.96%	
3 (3)	8	11.94%	11.94%
4 (4)	18	26.87%	
5 (5)	21	31.34%	58.21%
No answer	0	0.00%	
Not displayed	8	11.94%	
Arithmetic mean	0		
Standard deviation	0		
Sum (Answers)	59	100.00%	100.00%
Number of cases		0%	

Field summary for P3

Wie wichtig ist Ihnen die Möglichkeit z.B Listen von IP-Adressen/Domains anhand von CVS oder XML Dateien zu importieren/exportieren?

Answer	Count	Percentage	Sum
1 (1)	10	14.93%	37.31%
2 (2)	15	22.39%	
3 (3)	16	23.88%	23.88%
4 (4)	7	10.45%	
5 (5)	3	4.48%	14.93%
No answer	0	0.00%	
Not displayed	16	23.88%	
Arithmetic mean	0		
Standard deviation	0		
Sum (Answers)	51	100.00%	100.00%
Number of cases		0%	

Field summary for P6

Wie wichtig ist es Ihnen über die Beendigung eines Tests per Email zu erfahren?

Answer	Count	Percentage	Sum
1 (1)	2	2.99%	7.46%
2 (2)	3	4.48%	
3 (3)	12	17.91%	17.91%
4 (4)	16	23.88%	
5 (5)	26	38.81%	62.69%
No answer	0	0.00%	
Not displayed	8	11.94%	
Arithmetic mean	0		
Standard deviation	0		
Sum (Answers)	59	100.00%	100.00%
Number of cases		0%	

Field summary for P9

Wie wichtig ist es Ihnen eigene, personalisierte Spezialtools durch einen Admin einbinden zu lassen?

Answer	Count	Percentage	Sum
1 (1)	11	16.42%	32.84%
2 (2)	11	16.42%	
3 (3)	19	28.36%	28.36%
4 (4)	7	10.45%	
5 (5)	6	8.96%	19.40%
No answer	0	0.00%	
Not displayed	13	19.40%	
Arithmetic mean	0		
Standard deviation	0		
Sum (Answers)	54	100.00%	100.00%
Number of cases		0%	

Field summary for P10

Wie wichtig ist es Ihnen einen Penetrationtest mittels einer API zu steuern?

Answer	Count	Percentage	Sum
1 (1)	12	17.91%	29.85%
2 (2)	8	11.94%	
3 (3)	14	20.90%	20.90%
4 (4)	13	19.40%	
5 (5)	5	7.46%	26.87%
No answer	0	0.00%	
Not displayed	15	22.39%	
Arithmetic mean	0		
Standard deviation	0		
Sum (Answers)	52	100.00%	100.00%
Number of cases		0%	

Field summary for P11

Wie wichtig ist es Ihnen Tests selbst auswählen zu können um Ihr System gezielt auf Schwachstellen zu überprüfen?

Answer	Count	Percentage	Sum
1 (1)	3	4.48%	10.45%
2 (2)	4	5.97%	
3 (3)	16	23.88%	23.88%
4 (4)	18	26.87%	
5 (5)	17	25.37%	52.24%
No answer	0	0.00%	
Not displayed	9	13.43%	
Arithmetic mean	0		
Standard deviation	0		
Sum (Answers)	58	100.00%	100.00%
Number of cases		0%	

Field summary for P12

Wie wichtig ist es Ihnen die Behebung von Schwachstellen nachzuverfolgen?

Answer	Count	Percentage	Sum
1 (1)	2	2.99%	5.97%
2 (2)	2	2.99%	
3 (3)	9	13.43%	13.43%
4 (4)	18	26.87%	
5 (5)	27	40.30%	67.16%
No answer	0	0.00%	
Not displayed	9	13.43%	
Arithmetic mean	0		
Standard deviation	0		
Sum (Answers)	58	100.00%	100.00%
Number of cases		0%	

A.5. Inhalte der CD

Die beiliegende CD beinhaltet die Ausarbeitung (eine volle Version und eine Version ohne die Ergebnisse der Tests bei der Bayerischen Staatsbibliothek), die Folien der Vorträge und die prototypische Implementierung des Service für Penetrationstests. **Bei der Online-Puplikation dieser Arbeit soll auf Bitte der Bayerischen Staatsbibliothek die Version verwendet werden, bei der die Ergebnisse entfernt wurden.**

A.5.1. Installation des Service

Damit der Service betrieben werden kann, müssen einige Abhängigkeiten Installiert werden. Bei einem Debian basierten Betriebssystem kann durch den Paketmanager folgende Abhängigkeiten nachinstalliert werden. *apt-get install python2.7 python-pip postrgresql* . Nachdem dieses Kommando ausgeführt wurde müssen einige Python-Abhängigkeiten installiert werden. Dies kann mit Hilfe der *requirements.txt* erledigt werden, die sich im Root-Verzeichnis des Programmcodes befindet. Hierzu muss folgendes Kommando ausgeführt werden: *pip2 -r requirements.txt*

Da der Service einige Hilfstools verwendet könnte dafür eine weitere Installation und Konfiguration nötig sein. Nachfolgend werden die URLs zu den Dokumentationen der Hilfstools aufgelistet:

- http://docs.w3af.org/en/latest/
- https://github.com/sqlmapproject/sqlmap/wiki
- https://docs.djangoproject.com/en/1.11/

Nachdem alle Abhängigkeiten installiert wurden, muss die Datenbank angelegt werden. Diese kann mit Hilfe der Django *migrate* Funktion erstellt werden. Dazu muss in das Verzeichnis *fapts* navigiert und folgendes Kommando ausgeführt werden: **python2 manage.py migrate**. Anschließend kann Django mit *python2 manage.py startserver* gestartet werden. Nachdem Django gestartet wurde, ist der Service unter http://localhost:8000 erreichbar.

Für den Produktiveinsatz empfiehlt sich, einen Webserver wie beispielsweise Apache2 aufzusetzen. Genaueres über diese Konfiguration kann in der Dokumentation nachgelesen werden.

Literaturverzeichnis

[AOD10] GmbH art of defence: *Benutzerhandbuch hyperguard*, 2010. http://mirko.dziadzka.de/vorlesung/is2010/grundwissen_webappsec.pdf Abruf: 18.06.17.

[BSI03] BSI: *Studie: Durchführungskonzept für Penetrationstests*, 2003. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/Penetrationstest/penetrationstest_pdf.pdf?__blob=publicationFile&v=2 Abruf: 18.06.17.

[BSI13a] BSI: *Glossar des Bundesamtes für Sicherheit in der Informationstechnik*, 2013. https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/Glossar/glossar_node.html Abruf: 18.06.17.

[BSI13b] BSI: *Leitfaden zur Entwicklung sicherer Webanwendungen*, 2013. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/Webanwendungen/Webanw_Auftragnehmer.pdf;jsessionid=86DF41B5B43807C069900AB85B94D5A4.1_cid091?__blob=publicationFile&v=1 Abruf: 18.06.17.

[CLK13] Neef, Sebastian: *Was ist Clickjacking?*, 2013. https://technik.blogbasis.net/was-ist-clickjacking-06-04-2013 Abruf: 20.07.17.

[CVE17] Corporation, The MITRE: *About CVE*, 2017. https://cve.mitre.org/about/index.html Abruf: 03.08.17.

[CVS15] FIRST: *CVSS Specification Document*, 2015. https://www.first.org/cvss/specification-document Abruf: 18.06.17.

[DRP16] *68 Millionen Nutzer betroffen: Beliebter Online-Dienst gehackt*, 2016. http://www.chip.de/news/68-Millionen-Nutzer-betroffen-Beliebter-Online-Dienst-gehackt_99252985.html Abruf: 18.06.17.

[HEI08] Bachfeld, Daniel: *„Hacker-Paragraf ": iX-Chefredakteur zeigt sich selbst an*, 2008. https://www.heise.de/security/meldung/Hacker-Paragraf-iX-Chefredakteur-zeigt-sich-selbst-an-191403.html Abruf: 18.06.17.

[HEI09] Heidrich, Jörg: *„Hacker-Paragraf ": Verfahren gegen iX-Chefredakteur eingestellt*, 2009. https://www.heise.de/ix/meldung/Hacker-Paragraf-Verfahren-gegen-iX-Chefredakteur-eingestellt-205502.html Abruf: 18.06.17.

[HEI13] Haar, Tobias: *Zur Strafe*, 2013. https://www.heise.de/ix/artikel/Zur-Strafe-1892408.html Abruf: 18.06.17.

[HYD17] *THC Hydra*. http://sectools.org/tool/hydra/ Abruf: 20.07.17.

[ILS16] *Internet Statistik*, 2016. http://www.internetlivestats.com/total-number-of-websites/ Abruf: 18.06.17.

[LFB16] Franceschi-Bicchierai, Lorenzo: *Hackers Stole 65 Million Passwords From Tumblr, New Analysis Reveals*, 2016. https://motherboard.vice.com/read/hackers-stole-68-million-passwords-from-tumblr-new-analysis-reveals Abruf: 18.06.17.

[LRZ] LRZ: *LRZ Image Broschüre*. https://www.lrz.de/wir/lrz-flyer/lrz_Image_broschuere.pdf Abruf: 18.06.17.

[LRZ12] Hommel, Reiser: *Das Münchner Wissenschaftsnetz Konzepte, Dienste, Infrastrukturen, Management*, 2012. https://www.lrz.de/services/netz/mwn-netzkonzept/mwn-netzkonzept.pdf Abruf: 18.06.17.

[LRZ15] LRZ: *LRZ Jahresbericht 2015*, 2015. https://www.lrz.de/wir/berichte/JB/JBer2015.pdf Abruf: 18.06.17.

[LRZ16a] LRZ: *Webhosting-Angebote für Institutionen*, 2016. https://www.lrz.de/services/netzdienste/webhosting/webhosting-institutionen/ Abruf: 18.06.17.

[LRZ17a] LRZ: *Serverbetrieb*, 2017. https://www.lrz.de/services/serverbetrieb/ Abruf: 18.06.17.

[MLO16] Mlot, Stephanie: *More Celeb Twitter Accounts Hacked After LinkedIn Data Dump*, 2016. http://uk.pcmag.com/software/82004/news/more-celeb-twitter-accounts-hacked-after-linkedin-data-dump Abruf: 18.06.17.

[MSF15] Messner, Michael: *Hacking mit Metasploit - Das umfassende Handbuch zu Penetration Testing und Metasploit*, Band 2. Dpunkt, 2015.

[OWA13] Foundation, OWASP: *OWASP Top 10*, 2013. https://www.owasp.org/images/f/f8/OWASP_Top_10_-_2013.pdf Abruf: 03.08.17.

[POH01] Pohl, Hartmut: *Taxonomie und Modellbildung in der Informationssicherheit*. Datenschutz und Datensicherheit, 11, 2004.

[POT15] Li, Richard, Dallin Abendroth, Xing Lin, Yuankai Guo, Hyun-wook Baek, Eric Eide, Robert Ricci und Jacobus Van der Merwe: *POTASSIUM: penetration testing as a service*. Seiten 30–42, 2015.

[PSC13] vonEye, Hommel, Metzger: *Dr. Portscan: Ein Werkzeug für die automatisierte Portscan-Auswertung in komplexen Netzinfrastrukturen*. 2013.

[PTS12] standard.org pentest: *Penetration Testing Execution Standard*, 2012. http://www.pentest-standard.org/index.php/PTES_Technical_Guidelines Abruf: 18.06.17.

[SMD16] Schmidt, Jürgen: *LinkedIn-Passwort-Leck hat desaströse Ausmaße*, 2016. https://www.heise.de/security/meldung/LinkedIn-Passwort-Leck-hat-desastroese-Ausmasse-3210793.html Abruf: 18.06.17.

[SOP12] Sophos: *Sicherheitsrisiko Webserver: So schließen Sie die Hintertüren*, 2012. http://www.unique-projects.com/downloads/Sophos/sophosclosingbackdoornetworkapplicationvulnerabilitieswpna.pdf Abruf: 18.06.17.

[SQL17] *SQLMAP: Automatic SQL injection and database takeover tool*. http://sqlmap.org/ Abruf: 20.07.17.

[STA17] Stackoverflow: *Eine Umfrage zur Verbreitung von Programmiersprachen*, 2017. https://insights.stackoverflow.com/survey/2017#technology-programming-languages Abruf: 18.06.17.

[STG71] Justiz, Bundesministerium der: *Strafgesetzbuch*, 1998. http://www.gesetze-im-internet.de/bundesrecht/stgb/gesamt.pdf Abruf: 18.06.17.

[SWA06] Borrmann, Micha und Sebastian Schreiber: *Sicherheit von Web-Applikationen aus Sicht eines Angreifers*. Datenschutz und Datensicherheit-DuD, 30(10):599–603, 2006.

[VAS16] GmbH, Greenbone Networks: *About OpenVAS Software*, 2016. http://www.openvas.org/about.html Abruf: 13.07.17.

[VAS17] GmbH, Greenbone Networks: *About OpenVAS Software*, 2016. http://www.openvas.org/software.html Abruf: 18.06.17.

[WEE17] *Weevely*. https://github.com/epinna/weevely3 Abruf: 20.07.17.

[WPS17] Wordpress.org: *About Wordpress*. https://wordpress.org/about/ Abruf: 20.07.17.

[XBL16] Blanco, Xiomara: *MySpace hacking might be the biggest data breach yet*, 2016. https://www.cnet.com/news/myspace-hacking-might-be-the-biggest-data-breach-ever/ Abruf: 18.06.17.

[ZEI16] *Yahoo: Daten von mindestens 500 Millionen Nutzern gestohlen*, 2016. http://www.zeit.de/news/2016-09/22/internet-yahoo-daten-von-mindestens-500-millionen-nutzern-gestohlen-22212402 Abruf: 18.06.17.

[ZEN17] *OWASP Zend Attack Proxy Project*. https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project Abruf: 20.07.17.