

Kapitel 3: Technische Schwachstellen und Angriffe



1. Grundlegendes zur Angriffsanalyse

- ❑ Notation von Sicherheitsproblemen
- ❑ Angreifermodelle
- ❑ Begriffe und Zusammenhänge

2. Ausgewählte technische Angriffsvarianten

- ❑ Denial of Service (DoS und DDoS)
- ❑ Schadsoftware (Malicious Code - Viren, Würmer, Trojanische Pferde)
- ❑ E-Mail-Security (Hoaxes und Spam)
- ❑ Mobile Code (ActiveX, JavaScript, ...)
- ❑ Systemnahe Angriffe (Buffer Overflows, Backdoors, Rootkits, ...)
- ❑ Web-basierte Angriffe (XSS, ...)
- ❑ Netzbasierte Angriffe (Sniffing, Portscans, ...)

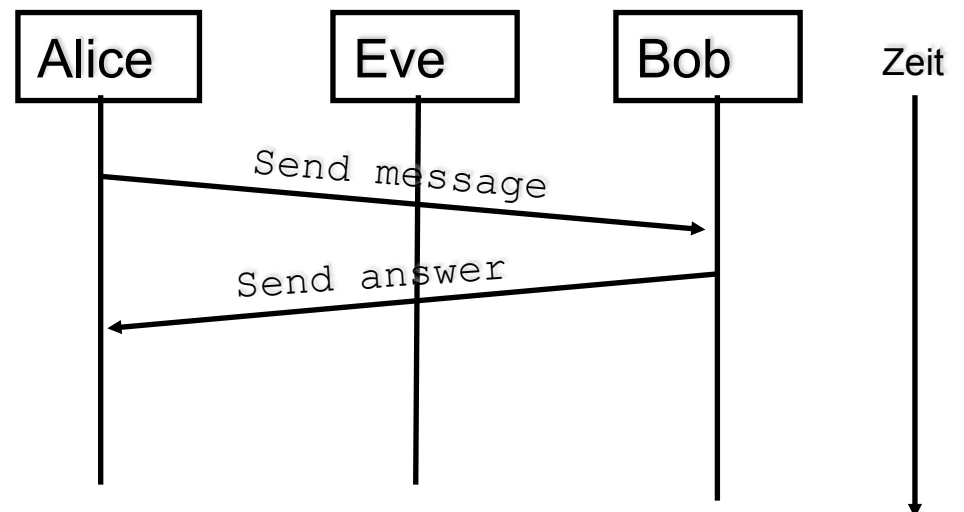
3. Bewertung von Schwachstellen

- ❑ Common Vulnerability Scoring System (CVSS)
- ❑ Untersuchungen zu “Top Security Risks” nach SANS

- Um Sicherheitsprobleme und -protokolle zu erläutern, werden häufig die folgenden Personen verwendet:
- Die „Guten“:
 - **Alice (A)**
Initiator eines Protokolls
 - **Bob (B)**
antwortet auf Anfragen von Alice
 - **Carol (C) und Dave (D)**
sind ggf. weitere gutartige Teilnehmer
 - **Trent (T)**
Vertrauenswürdiger Dritter
(Trusted third party)
 - **Walter (W)**
Wächter (Warden),
bewacht insb. Alice und Bob

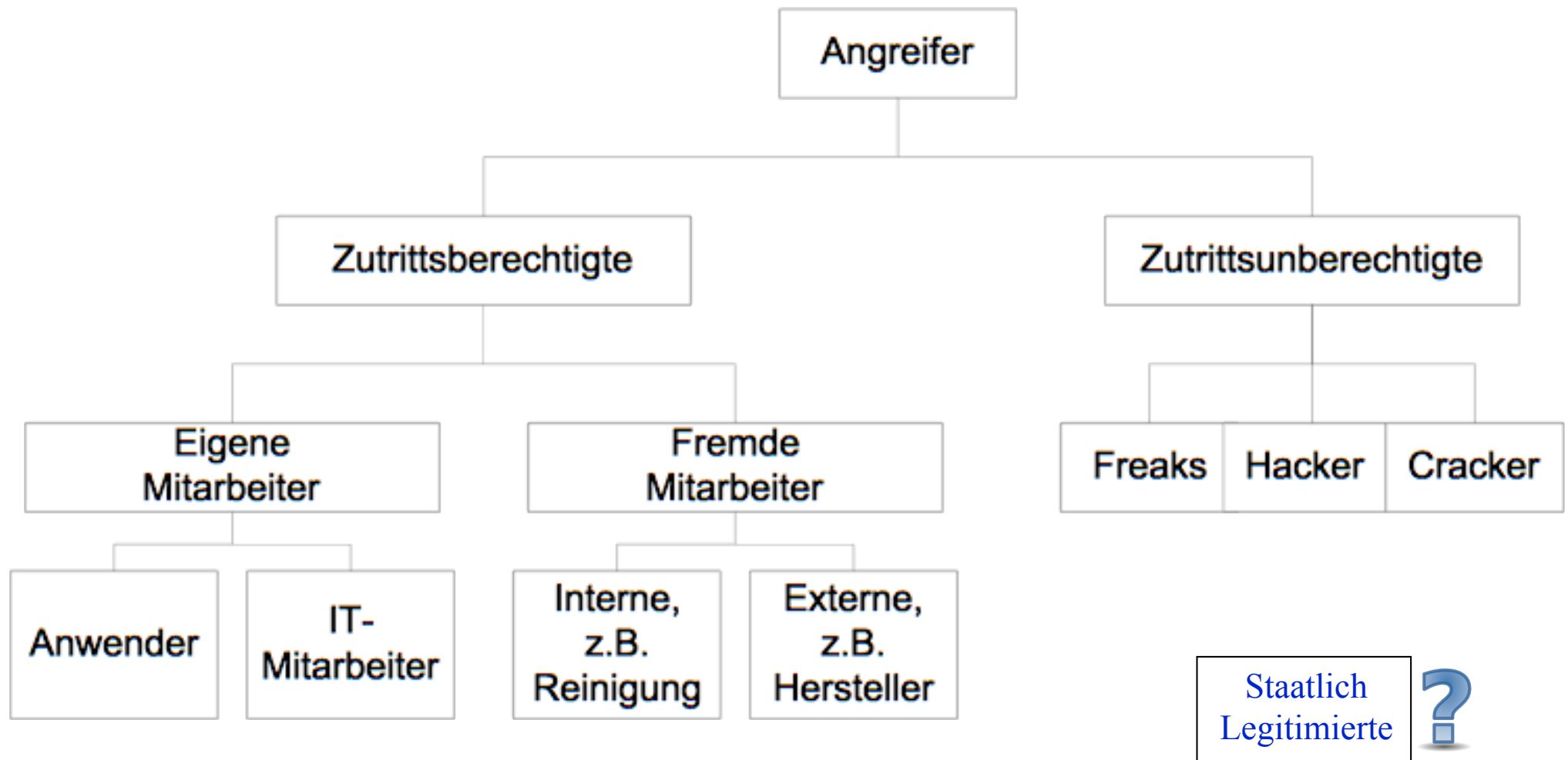
- Die „Bösen“:
 - **Eve (E)**
(Eavesdropper)
Abhörender / passiver Angreifer
 - **Mallory, Mallet (M)**
(Malicious attacker)
Aktiver Angreifer

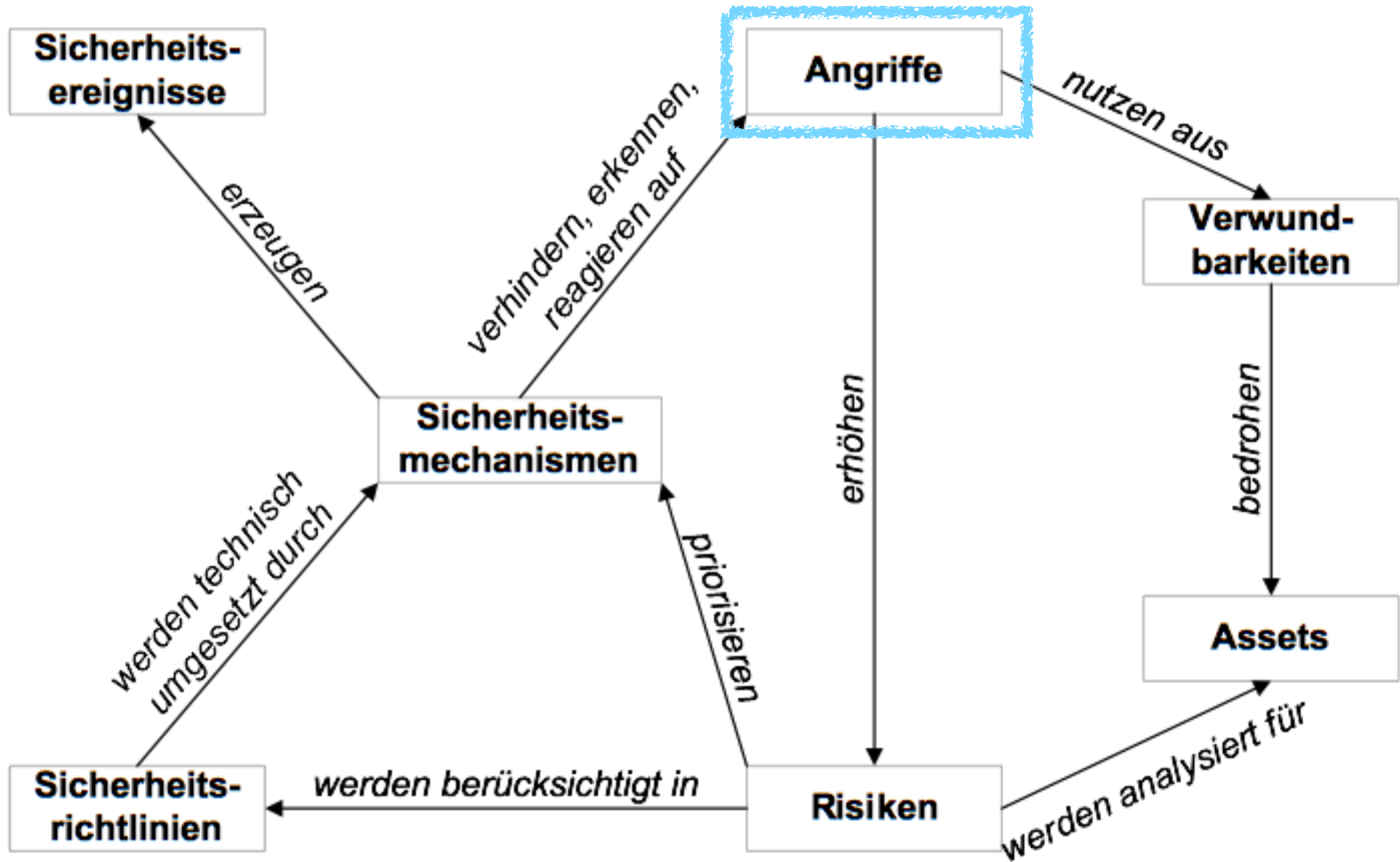
- Bsp.: Abhören der Kommunikation zwischen A und B
(UML Sequence Diagram)



- Antwort auf: Was können/machen Eve, Mallory und Mallet?
- Angreifermodell umfasst insbesondere Angaben zu
 - **Position des Angreifers**
 - Innentäter
 - Besucher, Einbrecher, ...
 - Internet / extern
 - **Fähigkeiten des Angreifers** (= Wissen + finanzielle Möglichkeiten), z.B. bei
 - experimentierfreudigen Schülern und Studierenden :-)
 - Fachleuten mit praktischer Erfahrung
 - erfahrenen Industriespionen / Geheimdiensten
 - **Motivation bzw. Zielsetzung des Angreifers**, z.B.
 - Spieltrieb, Geltungsbedürfnis, Vandalismus
 - Geld
 - Politischer oder religiöser Fanatismus, vermeintlicher Patriotismus
 - Spezifische **Charakteristika durchgeführter Angriffe**, z.B.
 - passives Abhören des Netzverkehrs vs.
 - aktive Eingriffe in die Kommunikation

Tätertypisierung





1. Grundlegendes zur Angriffsanalyse

- ❑ Notation von Sicherheitsproblemen
- ❑ Angreifermodelle
- ❑ Begriffe und Zusammenhänge

2. Ausgewählte technische Angriffsvarianten

- ❑ Denial of Service (DoS und DDoS)
- ❑ Schadsoftware (Malicious Code - Viren, Würmer, Trojanische Pferde)
- ❑ E-Mail-Security (Hoaxes und Spam)
- ❑ Mobile Code (ActiveX, JavaScript, ...)
- ❑ Systemnahe Angriffe (Buffer Overflows, Backdoors, Rootkits, ...)
- ❑ Web-basierte Angriffe (XSS, ...)
- ❑ Netzbasierte Angriffe (Sniffing, Portscans, ...)

3. Bewertung von Schwachstellen

- ❑ Common Vulnerability Scoring System (CVSS)
- ❑ Untersuchungen zu “Top Security Risks” nach SANS

- Erfolgreiche Angriffe haben negative Auswirkungen auf die
 - **Vertraulichkeit** (unberechtigter Zugriff auf Daten) und/oder
 - **Integrität** (Modifikation von Daten) und/oder
 - **Verfügbarkeit** (Löschen von Daten, Stören von Diensten)

- Eigenschaften zur Differenzierung von Angriffen sind z.B.:
 - **Ziel des Angriffs**: C, I und/oder A?
 - **Aktiv oder passiv** (z.B. remote exploit vs. sniffing)
 - **Direkt oder indirekt** (z.B. Manipulation einer Datenbank betrifft WebApp)
 - **Ein- oder mehrstufig** (z.B. kompromittierter Webserver als Sprungbrett)

- Angriffe sind unterschiedlich elegant und schwierig:
 - DDoS-Angriff zum Abschießen eines kleinen Webservers = trivial
 - Aufspüren und Ausnutzen bislang unbekannter Schwachstellen in Anwendungen = aufwendig

1. Grundlegendes zur Angriffsanalyse

- ❑ Notation von Sicherheitsproblemen
- ❑ Angreifermodelle
- ❑ Begriffe und Zusammenhänge

2. Ausgewählte technische Angriffsvarianten

- ❑ Denial of Service (DoS und DDoS)
- ❑ Schadsoftware (Malicious Code - Viren, Würmer, Trojanische Pferde)
- ❑ E-Mail-Security (Hoaxes und Spam)
- ❑ Mobile Code (ActiveX, JavaScript, ...)
- ❑ Systemnahe Angriffe (Buffer Overflows, Backdoors, Rootkits, ...)
- ❑ Web-basierte Angriffe (XSS, ...)
- ❑ Netzbasierte Angriffe (Sniffing, Portscans, ...)

3. Bewertung von Schwachstellen

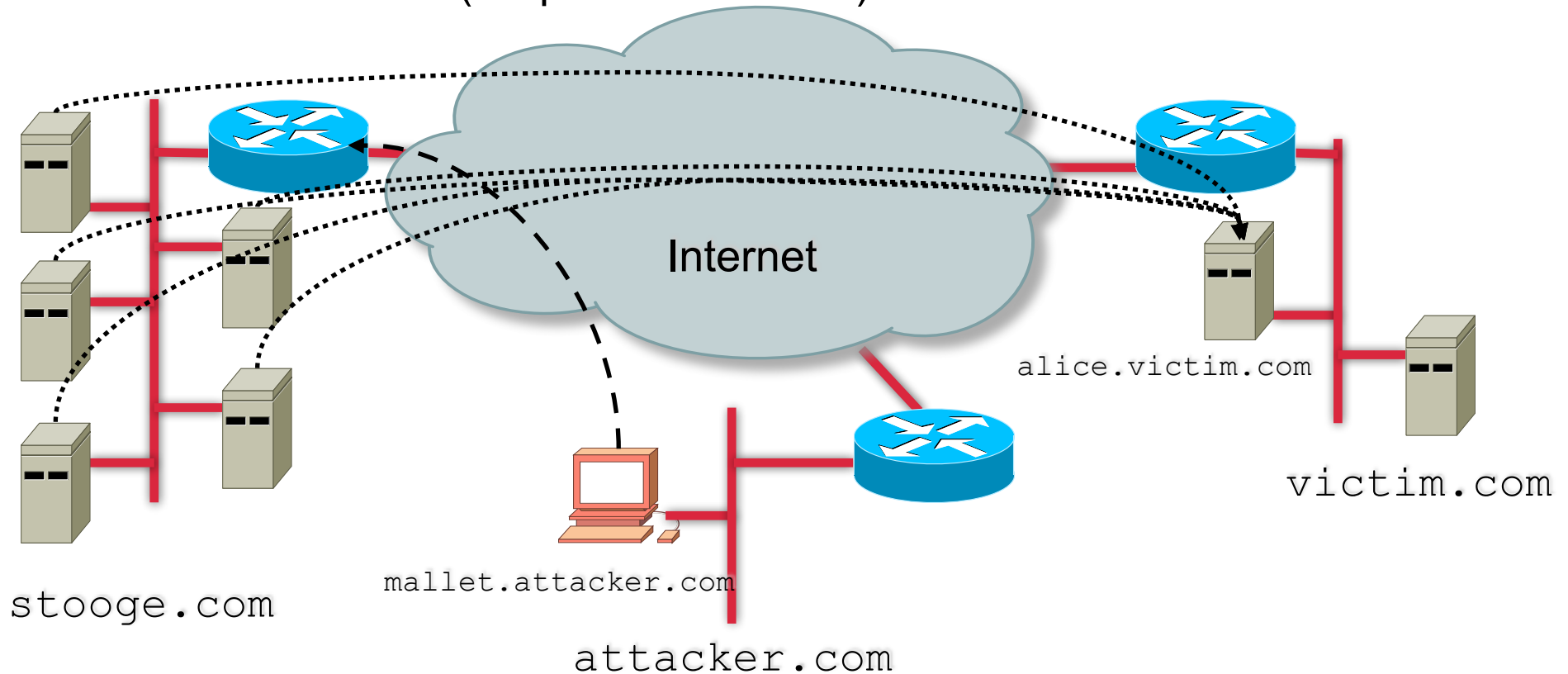
- ❑ Common Vulnerability Scoring System (CVSS)
- ❑ Untersuchungen zu “Top Security Risks” nach SANS

- Angriff versucht, das Zielsystem oder Netzwerk für berechnigte Anwender unbenutzbar zu machen, z.B. durch:
 - Überlastung
 - Herbeiführen einer Fehlersituation
 - Ausnutzung von Programmierfehlern oder Protokollschwächen, die z.B. zum Absturz führen
- Häufige Arten von DoS-Angriffen
 - Anforderung bzw. Nutzung beschränkter oder unteilbarer Ressourcen des OS (z.B. CPU-Zeit, Plattenplatz, Bandbreite,....)
 - Zerstörung oder Veränderung der Konfiguration
 - Physische Zerstörung oder Beschädigung
- Beispiel:
 - Angestellter konfiguriert “out of office” Mail mit CC: an interne Mailingliste. Außerdem konfiguriert er automatische Bestätigung durch Empfänger.
⇒ **Mailstorm**

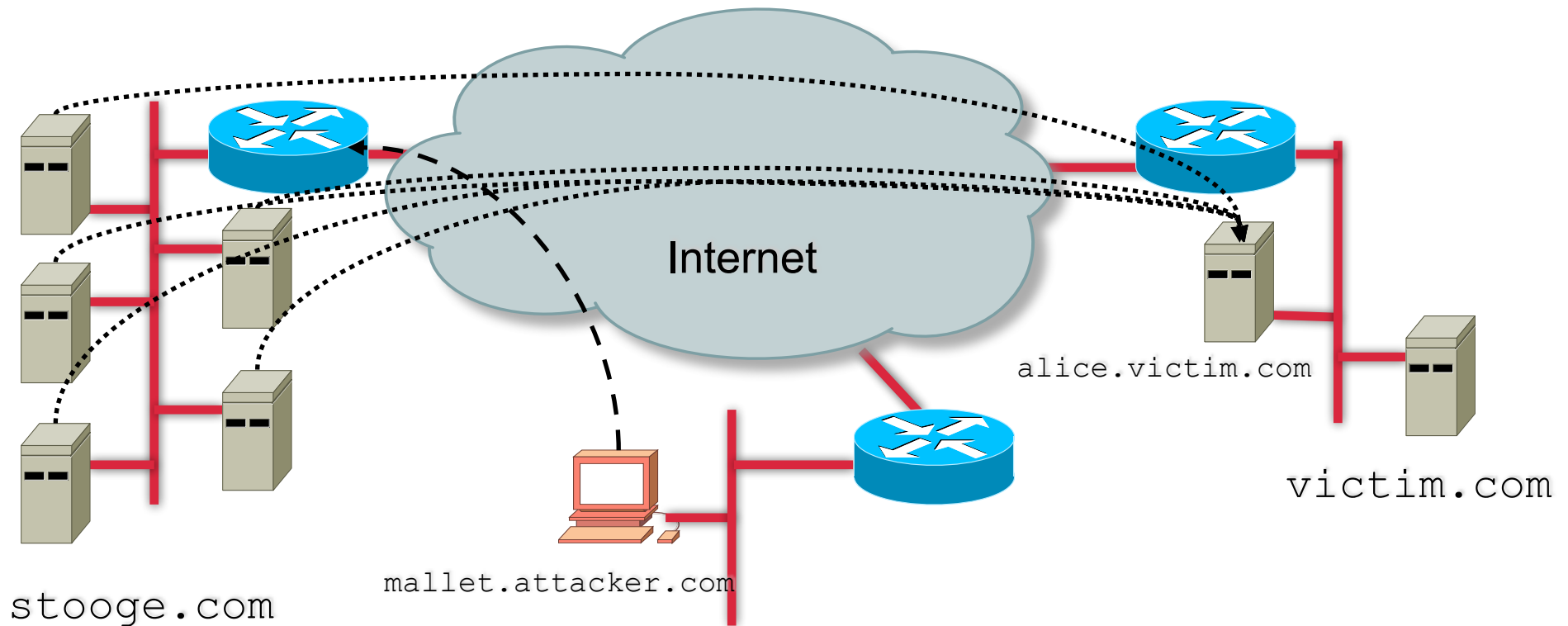
- **E-Mail Bombing:**
Überflutung der Inbox mit Mails
- **E-Mail Subscription Bombing:**
Opfer wird auf hunderten Mailinglisten registriert
- **Buffer Overflows; am Bsp. von Ping of Death**
 - IP-Paket größer als die max. erlaubten 2^{16} Bytes
 - Übertragen in mehreren Fragmenten; andernfalls würden die Router das Paket verwerfen.
 - Reassemblieren der Fragmente im Zielsystem führt zu Überlauf des internen Puffers im IP-Stack
 - Evtl. Absturz des Betriebssystems
 - Betraf u.a. Win95, WinNT, Linux, Solaris (bis 2007)
- **Ausnutzung von Programmfehlern**
 - **Land:** gefälschtes IP-Paket mit *IP Source Adr. = IP Destination Adr.* und *Source Port = Dest. Port*
⇒ 100 % CPU Last bei best. Implementierungen (1997)
 - **Teardrop:** Fragmentierte Pakete enthalten Feld `Fragment Offset` Hier Manipulation, so dass sich Fragmente „überlappen“
⇒ u.U. Absturz des Systems (Win95, WinNT, Linux 2.0)
- **Aufbrauchen von Bandbreite bzw. Betriebssystem-Ressourcen**
 - Fluten des Netzwerkes des Opfers (z.B. SMURF)
 - SYN-Flooding
 - Low Orbit Ion Cannon (LOIC)

Klassische DoS-Techniken: SMURF

- Angreifer sendet Strom von ping Paketen (ICMP) mit gefälschter Absender-Adresse (`alice.victim.com`) (Adressfälschung wird auch als IP-Spoofing bezeichnet) an IP-Broadcast Adresse von `stooge.com`
- Alle Rechner aus dem Netz von `stooge.com` antworten an `alice.victim.com` (Amplification attack)



SMURF: Gegenmaßnahmen?



- **Überkompensation:**
ICMP oder IP-Broadcast am Router komplett deaktivieren
- **Besser:**
 - ❑ Server so konfigurieren, dass sie nicht auf Broadcast-Pings antworten
 - ❑ Router so konfigurieren, dass sie von außen an die Broadcast-Adresse gerichtete Pakete nicht weiterleiten

■ Begriffsbildung:

- Domain Name System (Zuordnung von Namen zu IP-Adressen)
- Kleines Paket des Angreifers führt zu großen Paket an Opfersystem

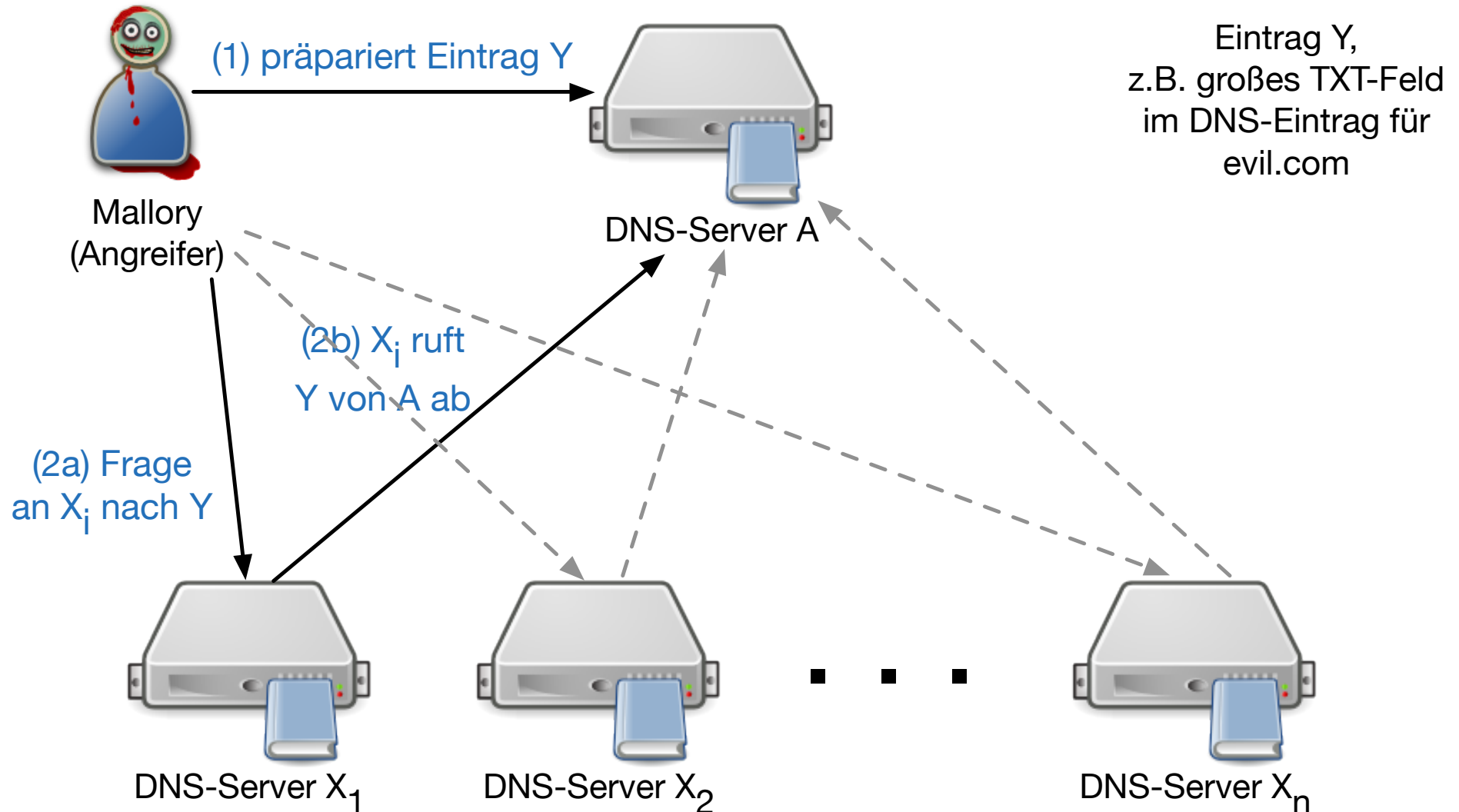
■ Grundprinzip:

- Sehr **kleines UDP-Paket zur Abfrage** des DNS-Servers (ca. 60 Byte)
- Gefälschte Absenderadresse (i.A. die des DoS-Opfers)
- **Antwort kann sehr groß werden** (bis theor. 3000 Byte)
- Verstärkungsfaktor 50
- Schmalbandiger Uplink reicht aus, um Multi-Gigabit Traffic zu erzeugen

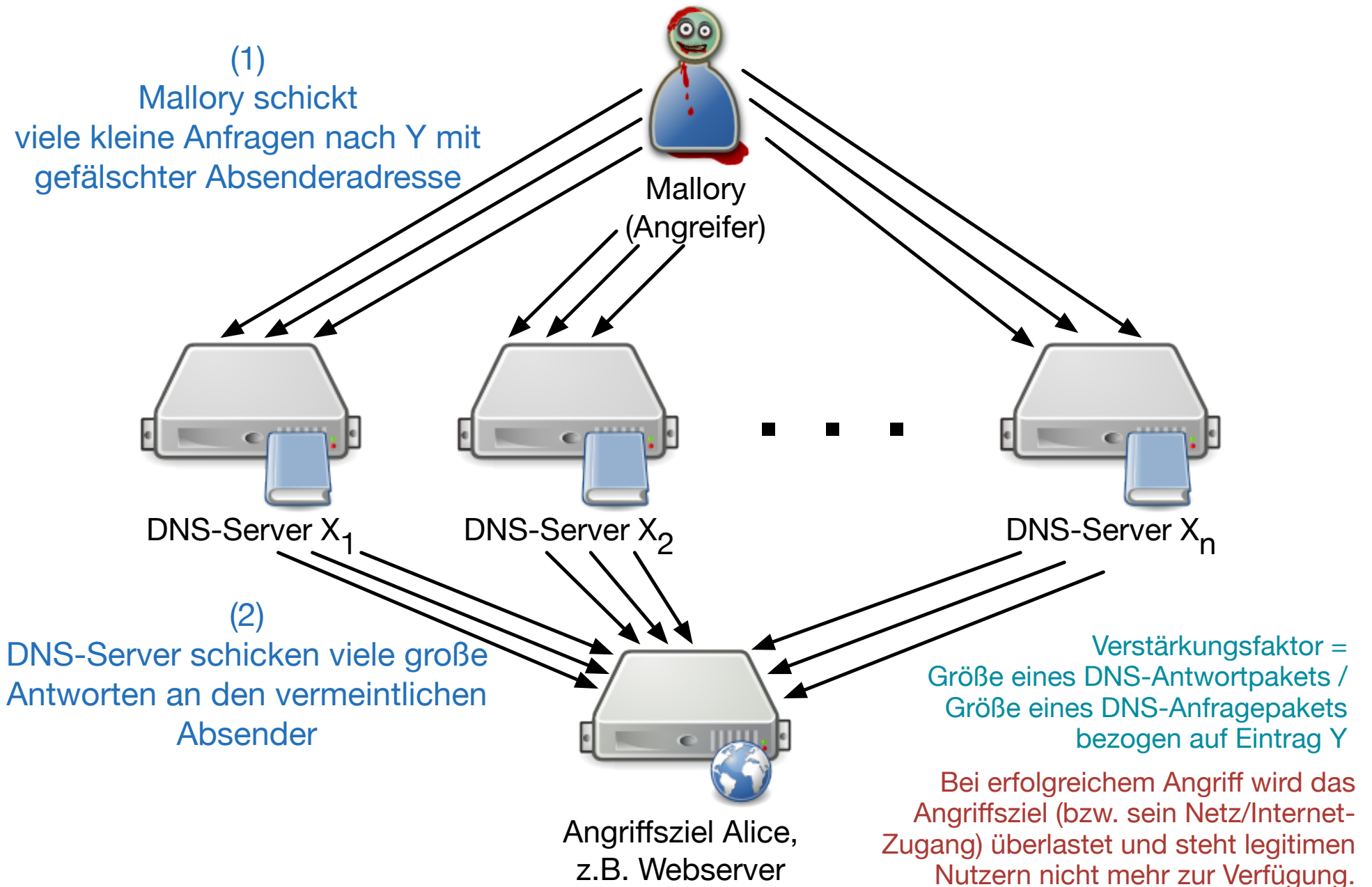
■ Historie:

- Angriffe auf DNS-Root-Nameserver 2006
- Seit Frühjahr 2012 häufige Scans nach DNS-Servern, wachsende Anzahl an Vorfällen; inzwischen größtenteils behoben, aber gallische Dörfer bleiben.

■ Bsp: <http://blog.cloudflare.com/65gbps-ddos-no-problem>

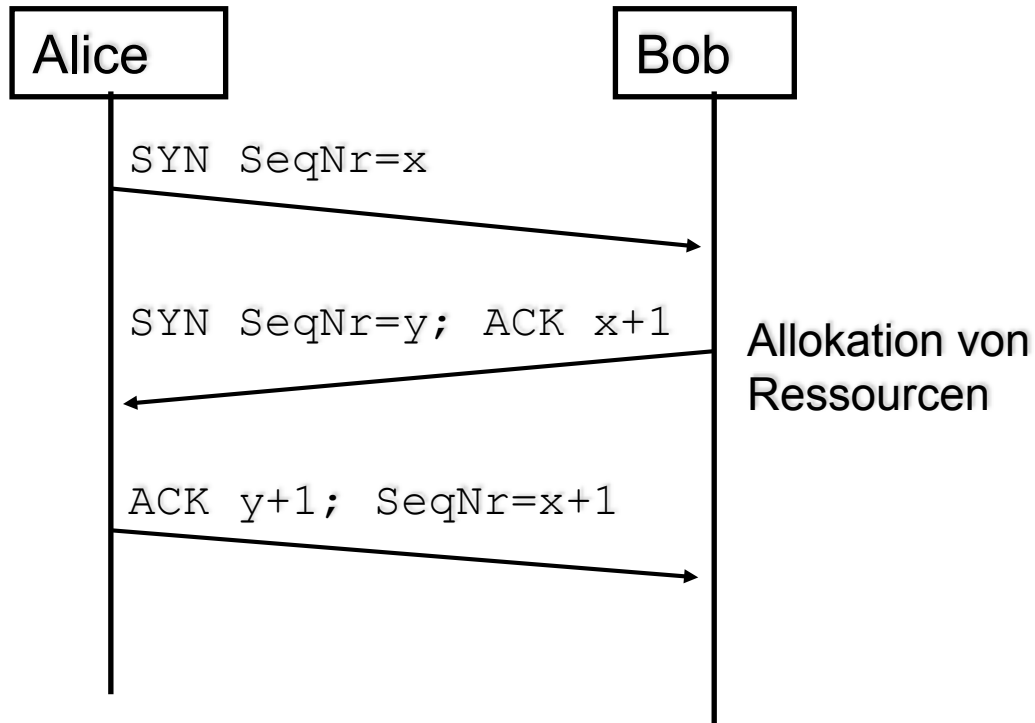


Ergebnis: DNS-Server X_i haben Eintrag Y in ihrem Cache und liefern ihn auf Anfrage aus

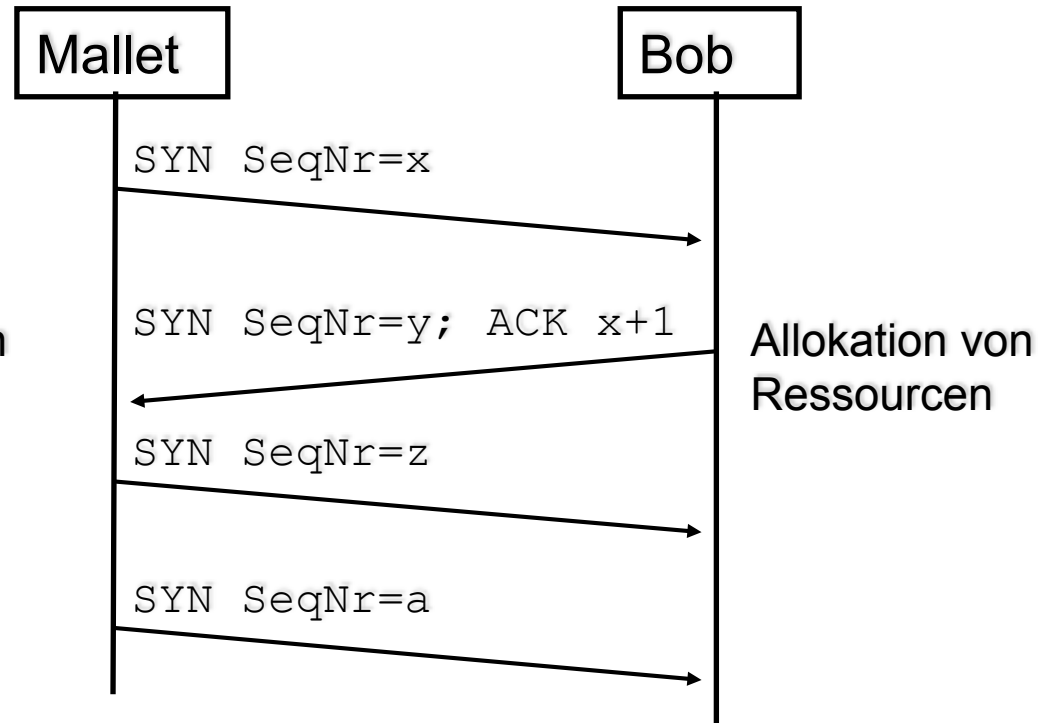


- DNS Server X_n beantworten rekursive Anfragen aus dem Internet
- Ablauf (vgl. vorherige Folien):
 - Angreifer sucht oder präpariert DNS-Server A mit langen Feldern (z.B. TXT-Feld oder DNSSEC-Key-Feld) eines Eintrages Y
 - Anfrage nach Eintrag auf Server A an Server X_i
 - X_i fragt A und schreibt Ergebnis Y in seinen Cache
 - Danach viele Anfragen nach Y an die Server X_n mit gefälschter Absenderadresse von Alice
 - Folge: Alice wird mit DNS-Antworten überflutet
- Gegenmaßnahme:
 - Keine rekursiven Anfragen von extern beantworten
 - [Schwellenwerte für identische Anfragen desselben vermeintlichen Clients]
- MWN im September 2012:
 - 58 weltweit erreichbare DNS-Server
 - 26 beantworten Anfragen rekursiv

■ TCP 3-Way-Handshake zum Verbindungsaufbau



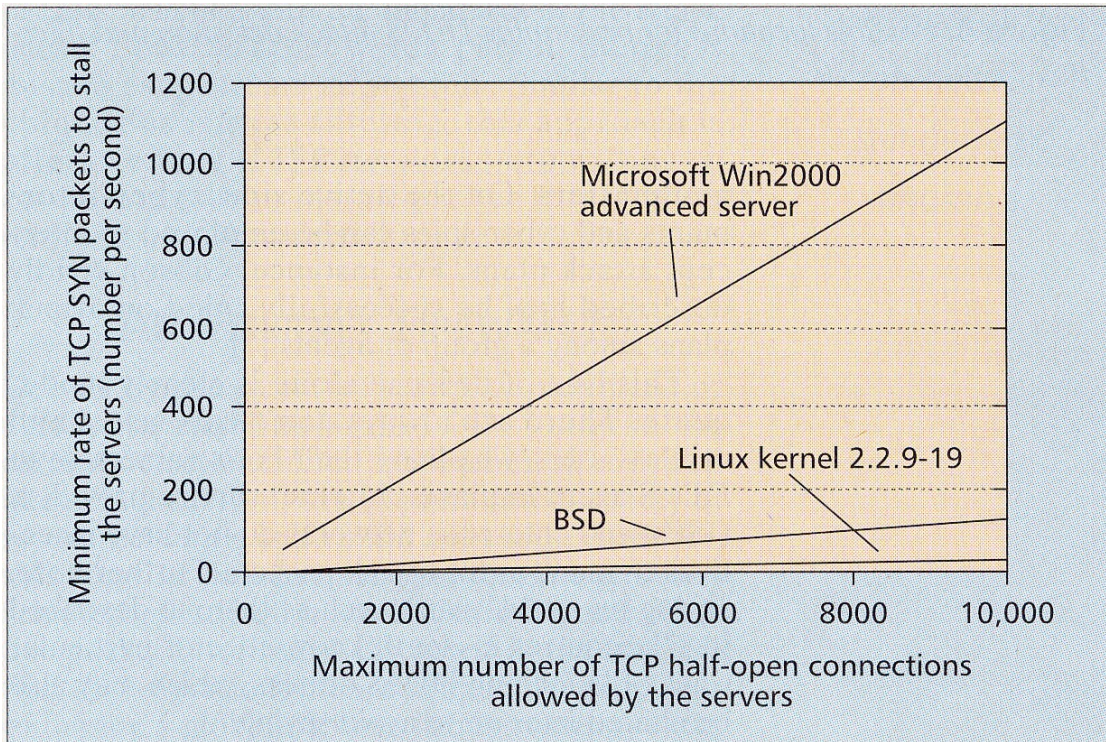
■ SYN Flooding



- „Halboffene“ TCP-Verbindungen so lange aufbauen, bis Ressourcen von Bob erschöpft sind.
- Bob kann dann keine weiteren Netzverbindungen mehr aufbauen.

SYN-Flood: Reaktion der Betriebssysteme

- Minimale Anzahl von SYN-Paketen für erfolgreichen DoS
Quelle: [Chang 02]

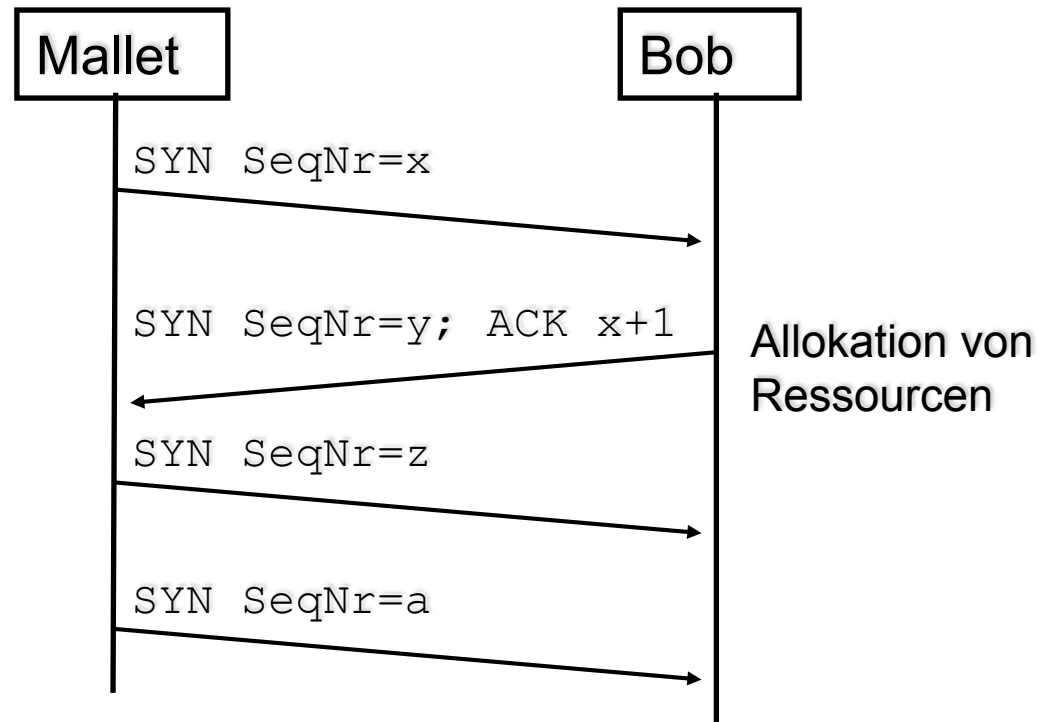


- Wiederholung von „verlorenen“ SYN-Paketen:

- Exponential Backoff zur Berechnung der Wartezeit
 - Linux und W2K (3s, 6s, 12s, 24s,....)
 - BSD (6s, 24s, 48s,)
- Abbruch des Retransmit
 - W2K nach 2 Versuchen (d.h. nach 9 Sekunden)
 - Linux nach 7 Versuchen (d.h. nach 381 Sekunden)
 - BSD nach 75 Sekunden

SYN Flooding: Gegenmaßnahmen?

- Timer definieren:
Falls ACK nicht innerhalb dieser Zeitspanne erfolgt, Ressourcen wieder freigeben.
 - ✓ Nutzt nur bedingt
- Falls alle Ressourcen belegt:
Zufällig eine halboffene Verbindung schliessen
 - ✓ Nutzt nur bedingt
- Maximale Anzahl gleichzeitig halboffener Verbindungen pro Quell-Adresse festlegen
 - ✓ Immer noch Problem bei DDoS
- SYN Cookies (Bernstein 1996):
Seq.Nr. y von Bob „kodiert“ Adressinfo von Mallet. Ressourcen werden erst reserviert, wenn tatsächliches ACK $y+1$ von Mallet eingeht.
 - ✓ Legitime Verbindung kommt nicht zustande, wenn das ACK-Paket von Alice verloren geht und Alice im Protokollablauf zunächst Daten von Bob erwartet.



■ Historie:

- Trinoo erstmals im Juli 99 aufgetaucht; Aug. 99: 227 Clients greifen eine Maschine der Uni Minnesota an (2 Tage Down-Zeit)
- 7. Feb. 2000: Yahoo 3 Stunden Downzeit (Schaden ~ 500.000 \$)
- 8. Feb. 2000: Buy.com, CNN, eBay, Zdnet.com, Schwab.com, E*Trade.com und Amazon.
(Bei Amazon 10 Stunden Downzeit und ~ 600.000 \$ Schaden)

■ Idee:

DoS-Angriffswerkzeuge werden auf mehrere Maschinen verteilt und führen auf Befehl eines Masters Angriff durch.

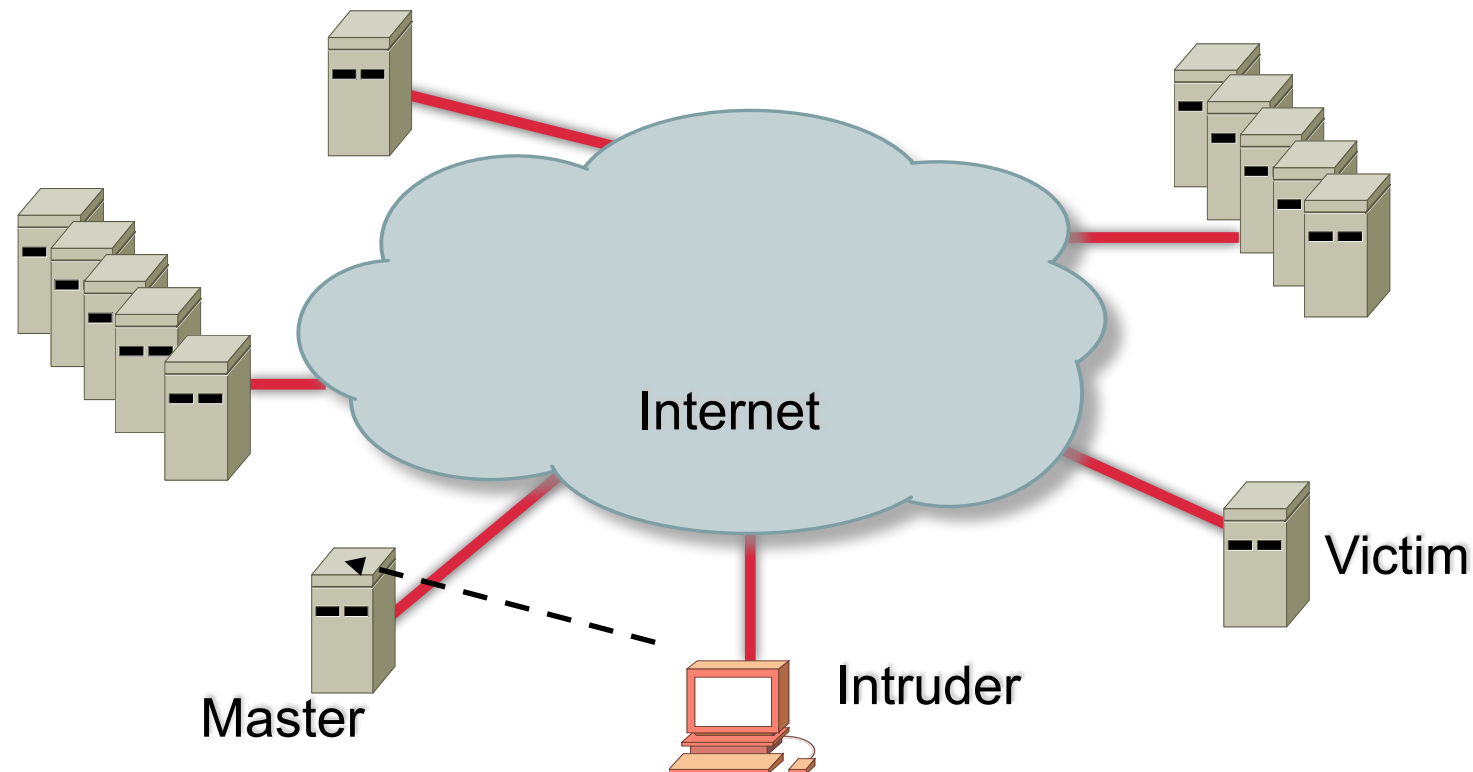
■ Terminologie

- Intruder oder Attacker: Angreifer (Person)
- Master oder Handler: Koordinator des Angriffs (Software)
- Daemon, Agent, Client, Zombie, Bot oder bcast-Programm: Einzelkomponente, die Teil des DDoS durchführt (Software)
- Victim oder Target: Ziel des Angriffs

■ Beispiele:

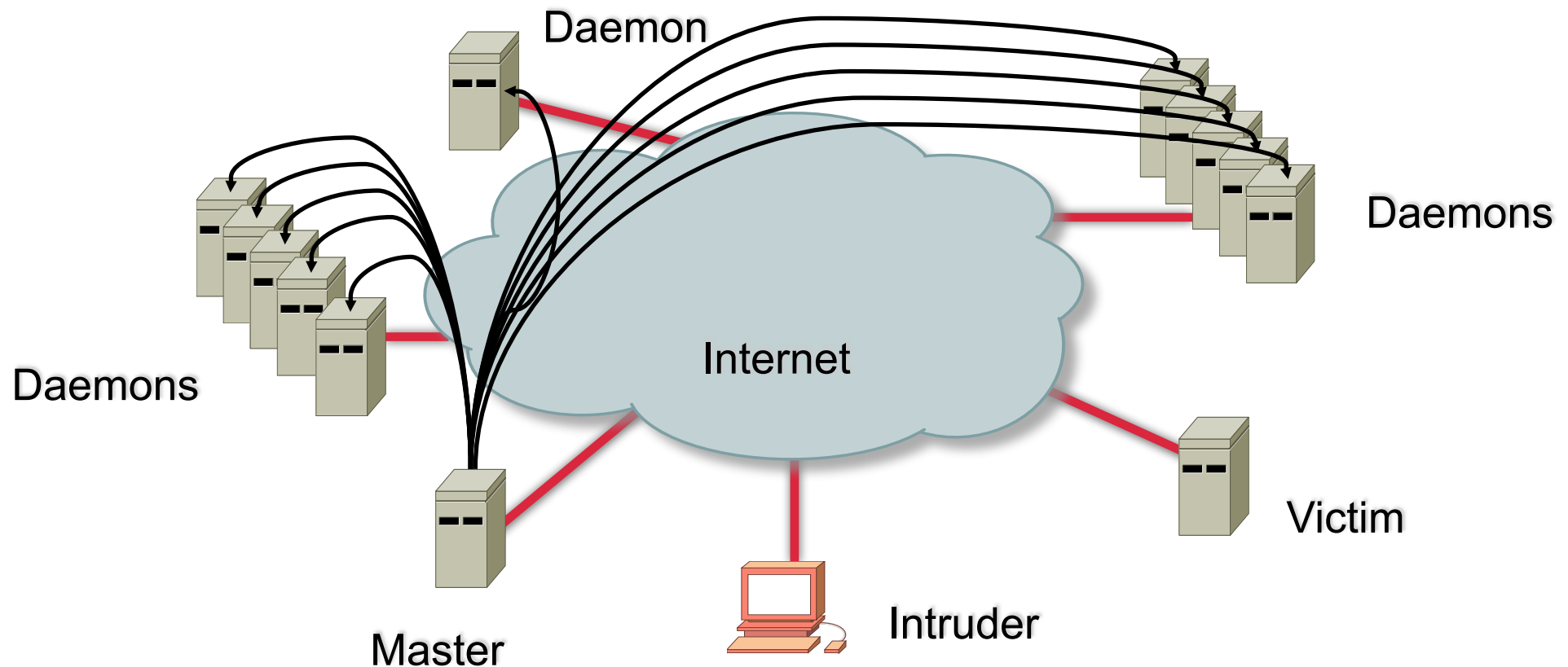
- Trinoo (Trin00)
- Tribe Flood Network (TFN) und TFN2K
- Stacheldraht
- Low Orbit Ion Cannon (LOIC)

- Dreistufiges Verfahren:
 1. Intruder findet Maschine(n), die kompromittiert werden können;
Hacking-Werkzeuge, Scanner, Rootkits, DoS/DDoS-Tools werden installiert; \Rightarrow Maschine wird Master



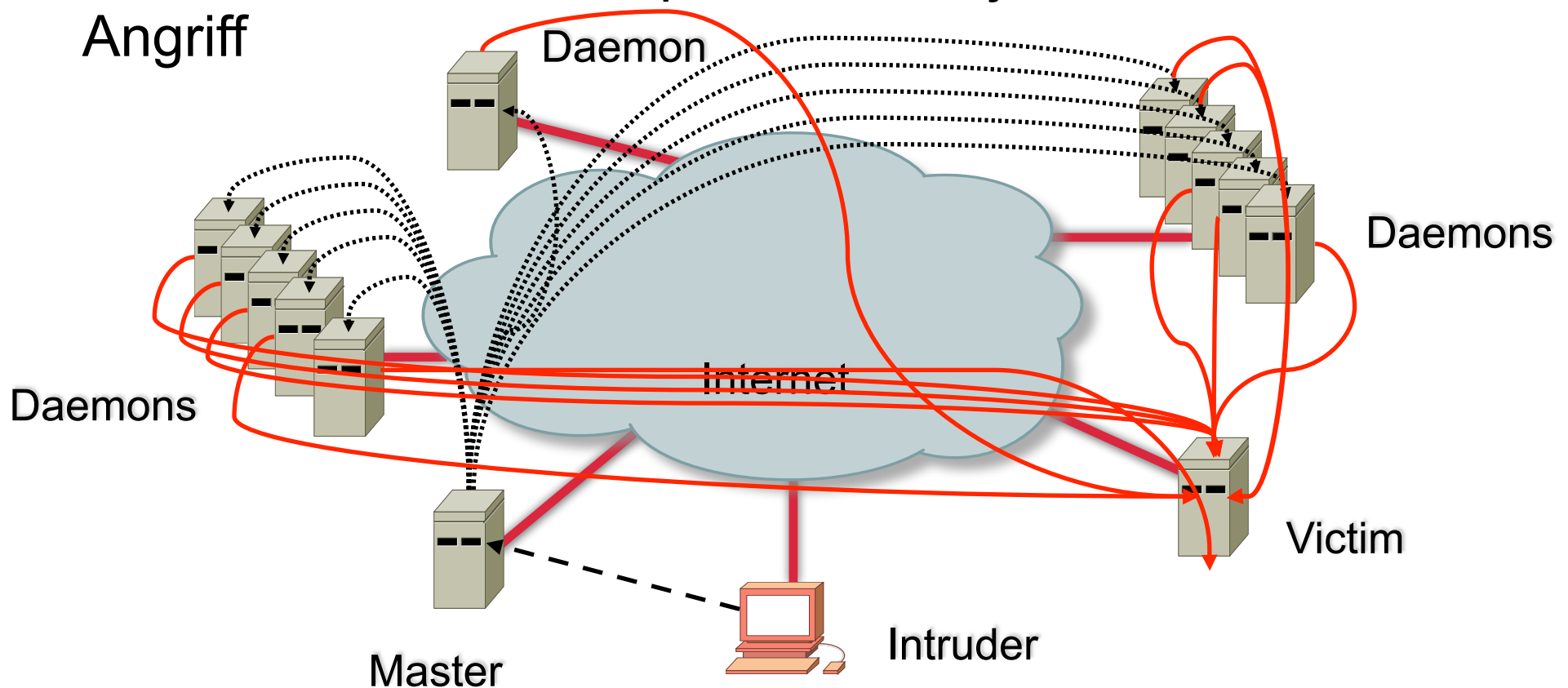
DDoS-Ablauf (Fortsetzung)

2. Master versucht automatisiert, weitere Maschinen zu kompromittieren, um DDoS-Software (Daemon) zu installieren, bzw. schiebt anderen Nutzern Malware unter.



DDoS-Ablauf (Fortsetzung)

- Intruder startet Programm auf Master, das allen Daemonen mitteilt, wann und gegen wen der Angriff zu starten ist.
Zum vereinbartem Zeitpunkt startet jeder Daemon DoS-Angriff

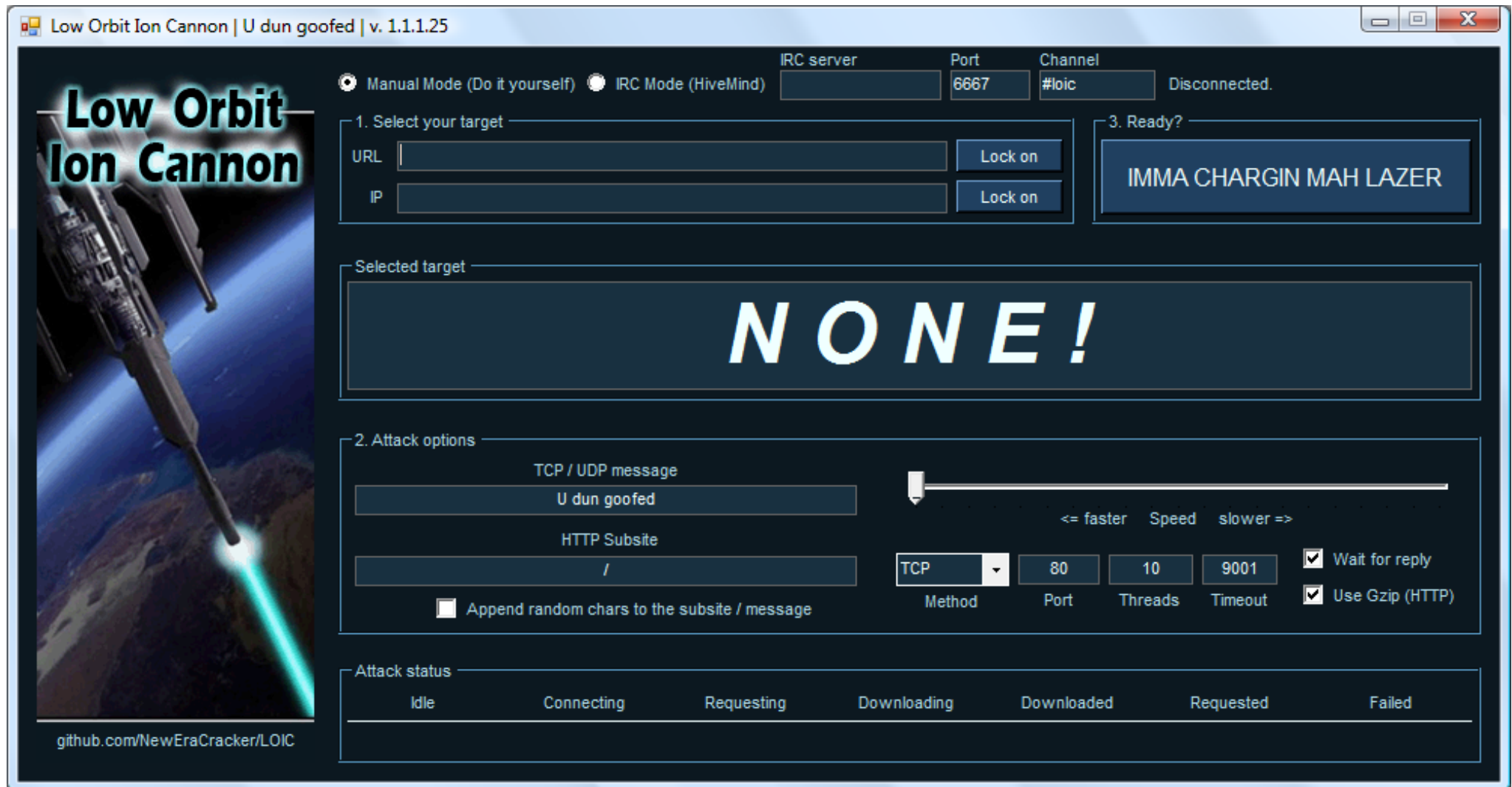


- Trinoo oder Trin00 (1999)
 - Verteilter UDP Flooding Angriff
- Kommunikation:
 - Intruder → Master: Master hört auf TCP-Port 27665; Passwort „betaalmostdone“
 - Master → Daemon: Daemon auf UDP-Port 27444, Passwort „144adsl“
 - Daemon → Master: Master auf UDP-Port 31335
Beim Start *HELLO* Nachricht des Daemon per UDP an Master
 - Keep-Alive-Kommunikation:
Master → Daemon: png
Daemon → Master: PONG
- Tribe Flood Network (TFN)
 - Master kompromittiert UNIX- Systeme über RPC-Buffer-Overflow
 - SYN-, ICMP-, UDP-Flooding
 - SMURF-Angriff
- Kommunikation:
 - wird vollständig in ICMP ECHO und ICMP REPLY Nachrichten „versteckt“: Kommando wird im Identifier Feld des ICMP Paketes kodiert; z.B.
345 -> SYN-Flooding;
890 -> UDP-Flooding;
 - Kein Passwort-Schutz
- Stacheldraht = Trinoo + TFN + verschlüsselte Kommunikation + Auto-Update des Agenten
- Inzwischen hunderte Derivate und Weiterentwicklungen... mit GUI/ Weboberfläche/Chat steuerbar.

Details unter

<http://packetstorm.linuxsecurity.com/distributed/trinoo.analysis.txt>

- Open Source „Network Stress Testing Application“
- Flooding mit TCP- oder UDP-Paketen
- Weltweit in den Massenmedien bekannt geworden Ende 2010 im Rahmen der „Operation Payback“:
 - DDoS-„Racheakt“ an VISA, Mastercard, PayPal und Amazon wegen Stop der Dienstleistung für WikiLeaks.
 - Tausende Internet-Nutzer beteiligten sich „freiwillig“ durch Installation der Software bzw. Nutzung einer JavaScript-Variante per Web-Browser.
- Beteiligung an DDoS-Angriffen ist vielerorts illegal:
 - Victim protokolliert Quell-IP-Adressen der LOIC-Angreifer
 - Internet-Provider kennen die entsprechenden Benutzer
 - „Operation Payback“: Festnahmen in England, Spanien und Türkei
 - Gesetzgebung:
 - Deutschland: Computersabotage nach §303b StGB (Freiheitsstrafe + zivilrechtliche Ansprüche)
 - Holland: bis zu sechs Jahre Haftstrafe



■ Generell:

- ❑ Pauschaler Schutz gegen (D)DoS-Angriffe ist praktisch fast unmöglich
- ❑ Aber:
 - Spezifika einzelner Angriffe erlauben oft gute Schutzmaßnahmen
 - Ggf. temporäres Overprovisioning,
vgl. Spamhaus & DDoS protection provider Cloudflare

■ Schutz gegen DoS-Angriffe auf einzelne Vulnerabilities:

- ❑ Software-Updates und Konfigurationsanpassungen

■ Schutz gegen Brute-Force-(D)DoS-Angriffe:

- ❑ Firewall-Regeln, ggf. basierend auf Deep-Packet-Inspection
- ❑ Aussperren von Angreifern möglichst schon beim Uplink
- ❑ Zusammenarbeit mit den Internet-Providern der Angriffsquellen

■ Allgemeine Ansätze:

- ❑ Anzahl Verbindungen und Datenvolumen überwachen (Anomalieerkennung)
- ❑ Bug- und Sicherheitswarnungen (z.B. CERT) verfolgen

Betreff: DDOS www.zhs-muenchen.de
Datum: Mon, 5 Sep 2011 02:50:02 -0600
Von: <amiliaivgspopek@yahoo.com>
An: <hostmaster@lrz.de>

Your site www.zhs-muenchen.de will be subjected to DDoS attacks 100 Gbit/s.

Pay 100 btc(bitcoin) on the account 17RaBqjGLisGzLRaAUVqdA2YHgspdkD1rJ

Do not reply to this email

- Erpressungsversuche richten sich gegen zahlreiche Firmen und auch mehrere bayerische Hochschuleinrichtungen.
- Bei ausbleibender Zahlung finden tatsächlich DDoS-Angriffe statt; DDoS-Botnet besteht aus ca. 40.000 Maschinen.
- DDoS-Bots senden die folgende Anfrage:
- Filter-Kriterien:
 - Accept-Language *ru* (bei dt./eng. Website)
 - „Host“-Header nicht an erster Stelle

```
GET / HTTP/1.1
Accept: */*
Accept-Language: ru
User-Agent: [useragent string]
Accept-Encoding: gzip, deflate
Host: [target domain]
Connection: Keep-Alive
```

■ Fidor Bank München

- ❑ DDoS-Angriff am Freitag 24.10.2014 ab 18:30 Uhr
- ❑ Erpresserschreiben veröffentlicht: <https://www.facebook.com/fidorbank/posts/10152859627718417>
- ❑ Lt. Erpresserschreiben war es ein SYN-Flood-Angriff
- ❑ Bank erstattet Anzeige, schaltet Webseite temporär ab, Zahlungskarte kann nicht mehr genutzt werden

■ sipgate

- ❑ Test-Angriff am 23.10.2014 ab 3:35 Uhr
- ❑ Erpresserschreiben am Vormittag, Lösegeldforderung in Bitcoins
- ❑ Drei Angriffswellen über mehrere Tage
- ❑ sipgate-Kunden können während der Angriffe nicht mehr telefonieren
 - Z.T. sehr hohe Schäden bei Firmenkunden vermutet
- ❑ sipgate-Hotline wird überrannt, diverse Presseberichte
- ❑ Beschreibung des Ablaufs von sipgate: <https://medium.com/@sipgate/ddos-attacke-auf-sipgate-a7d18bf08c03>

Quelle: <http://www.heise.de/newsticker/meldung/Sipgate-und-Fidor-Bank-sollten-mit-DDoS-Angriffen-erpresst-werden-2435043.html>

1. Grundlegendes zur Angriffsanalyse

- ❑ Notation von Sicherheitsproblemen
- ❑ Angreifermodelle
- ❑ Begriffe und Zusammenhänge

2. Ausgewählte technische Angriffsvarianten

- ❑ Denial of Service (DoS und DDoS)
- ❑ Schadsoftware (Malicious Code - Viren, Würmer, Trojanische Pferde)
- ❑ E-Mail-Security (Hoaxes und Spam)
- ❑ Mobile Code (ActiveX, JavaScript, ...)
- ❑ Systemnahe Angriffe (Buffer Overflows, Backdoors, Rootkits, ...)
- ❑ Web-basierte Angriffe (XSS, ...)
- ❑ Netzbasierte Angriffe (Sniffing, Portscans, ...)

3. Bewertung von Schwachstellen

- ❑ Common Vulnerability Scoring System (CVSS)
- ❑ Untersuchungen zu “Top Security Risks” nach SANS

■ Definition:

- ❑ Befehlsfolge; benötigt Wirtsprogramm zur Ausführung
- ❑ Kein selbstständig ablauffähiges Programm
- ❑ Selbstreplikation (Infektion weiterer Wirte (Programme))

■ Allgemeiner Aufbau:

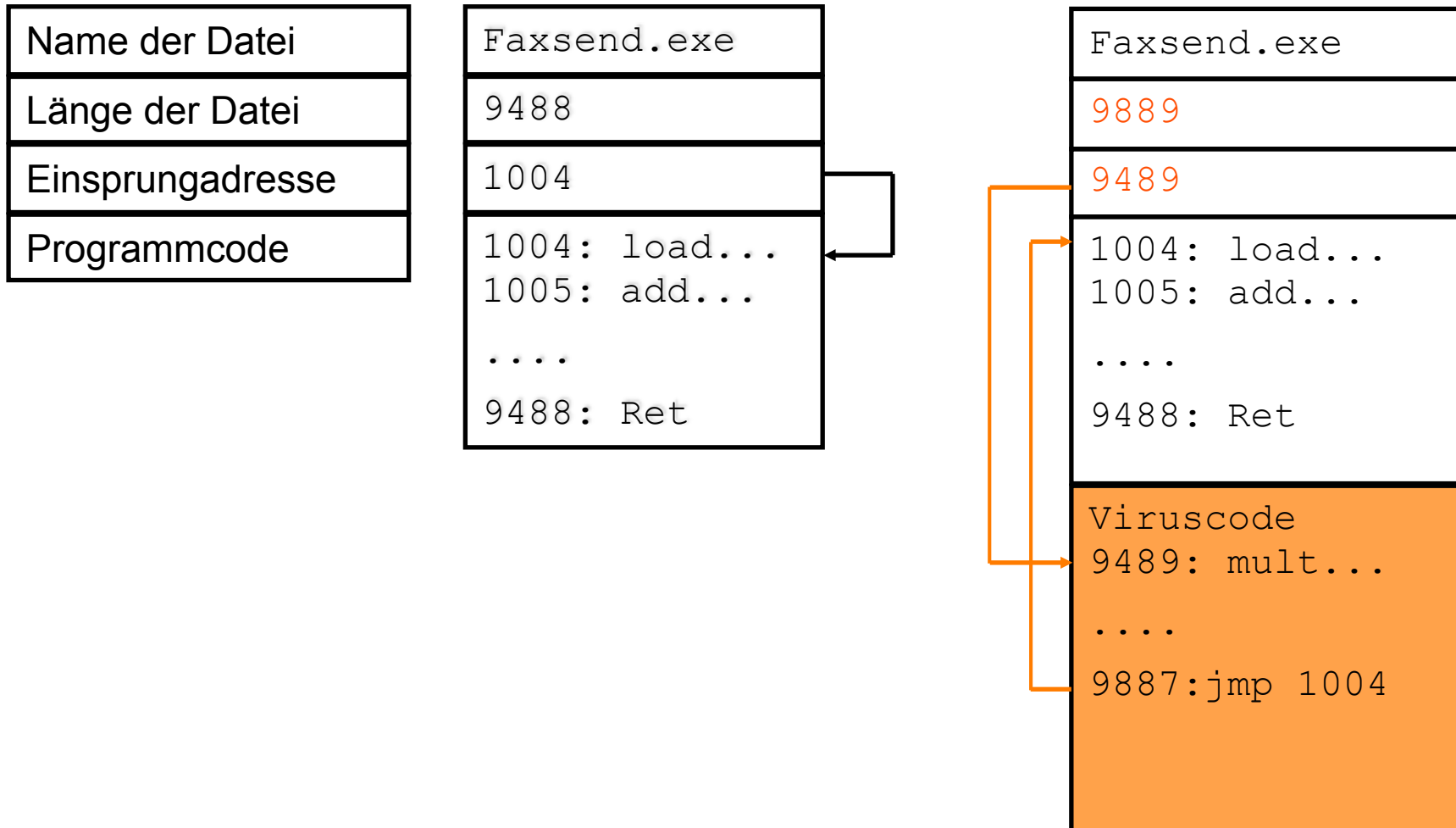
Viruserkennung	<pre>void function virus { signature</pre>
Infektionsteil	<pre> suche Programm p ohne signature kopiere Virus in p</pre>
Schadensteil ggf. mit Bedingung	<pre> if (wochentag == Freitag && tag == 13) { lösche alle Dateien } }</pre>
Sprung	<pre> springe an den Anfang des Wirtsprogramm }</pre>

- ❑ Daneben ggf. Tarnungsteil (selbstentschlüsselnder Code, Padding, ...)

Programm-Viren: Infektion

- Dateiformat vor der Infektion
(vereinfachtes Beispiel)

- Datei nach der Infektion



■ Klassifikation nach Infektionsziel:

- **Programm-Virus (Link-Virus)**
Infiziert ausführbare Dateien (MS-DOS/
Windows: .exe, .com, .sys)
- **Bootsektor-Virus**
Infiziert den Bootsektor von Festplatten oder
Disketten
- **Makro-, Daten-Virus**
Infiziert „Daten-Dateien“ mit eingebetteten
Makro-Sprachen (z.B. Visual Basic in MS
Office, Postscript, PDF, Flash, ...)

■ Unterklassen:

- **Multipartiter bzw. hybrider Virus**
Infiziert mehr als ein Ziel, z.B. Bootsektor +
Programme
- **Polymorpher Virus**
Verschlüsselt den Viruscode; damit für Anti-
Viren-Software (AV) schwerer zu finden

□ **Retro-Virus**

Greift aktiv AV an; versucht Scanner
so zu verändern, dass er unentdeckt
bleibt.

□ **Stealth-Virus**

Virus versucht, sich vor AV zu
verstecken. Sobald AV Dateien scannt,
entfernt der Virus seinen Code aus
den infizierten Dateien
(Wiederherstellung des
Originalzustandes)

□ **Tunneling-Virus**

AV versucht, Systemaufrufe zum
Schreiben u.a. in den Bootsektor zu
überwachen. Virus ermittelt jedoch die
direkte Speicheradresse des
entsprechenden Systemaufrufs und
umgeht dadurch das AV-Monitoring.

26.10.2011 13:36

 « Vorige | Nächste »

Avira verdächtigt sich selbst

 vorlesen / MP3-Download

Ein reguläres Update der Antiviren-Software von Avira lieferte eine Signatur, die auf eine der eigenen Dateien ansprang. Die Bibliothek AESCRIPT.DLL wurde dann plötzlich als "TR/Spy.463227" erkannt und gemeldet.

Ein Avira-Mitarbeiter bestätigte in den [Support-Foren](#) das Problem und erklärte, dass deshalb die Auslieferung des Updates bereits gestoppt wurde. Eine weitere Aktualisierung soll das Problem auch wieder beheben. ([ju](#))



Quelle: <http://www.heise.de/newsticker/meldung/Avira-verdaechtigt-sich-selbst-1367031.html>

Sophos Endpoint Security and Control

Quarantäne-Manager

Anzeigen: Alle Objekte

Typ	Name	Details	Verfügbare Maßnah...
Virus/Spyware	Shh/Updater-B	C:\Programme\Sophos\AutoUpdate\SingleGUIPlugin.dll	Verschieben, Löschen
Virus/Spyware	Shh/Updater-B	C:\Programme\Sophos\AutoUpdate\inetconn.dll	Verschieben, Löschen
Virus/Spyware	Shh/Updater-B	C:\Programme\Sophos\AutoUpdate\ALsvc.exe	Verschieben, Löschen

Alles markieren Aufheben Entfernen Maßnahme durchführen

3 Objekte (0 gewählt)


ildquelle: <http://www.nickles.de/forum/viren-spyware-datenschutz/2012/sophos-virenschicht-sich-selbst-in-quarantaene-538944296.html>

- 20.09.2012: Sophos verschiebt sich selbst in Quarantäne, lässt keine Updates mehr zu

- Zwei Haupt-Angriffsvektoren:
 - Angreifer bringen bekannte Viren-Signaturen in harmlosen Dateien unter und lassen diese über Online-Virens Scanner testen
=> Im Worst Case werden z.B. die entsprechenden Files auf eine Blacklist gesetzt und von den Anwendersystemen gelöscht.
 - Antivirus-Softwarehersteller erstellt Fake-Signaturen, die von der Konkurrenz ungetestet übernommen werden.

Schwere Vorwürfe gegen Firmenchef Eugene Kaspersky

 heise online 15.08.2015 14:38 Uhr – Dorothee Wiegand

 vorlesen

Zwei Ex-Mitarbeiter des Antiviren-Herstellers Kaspersky beschuldigen ihren ehemaligen Chef, er habe sie damit beauftragt, Konkurrenzprodukte zu sabotieren.

Zwei ehemalige Mitarbeiter des Antiviren-Herstellers Kaspersky beschuldigen den Firmenchef persönlich. In einem Bericht der amerikanischen Nachrichtenagentur Reuters werden die beiden namentlich nicht genannten Personen zitiert. Demnach habe Kaspersky einige Mitarbeiter damit beauftragt, Konkurrenzprodukte zu sabotieren. Konkret hätten sie den Auftrag bekommen, indirekt Produkte anderer AV-Hersteller so zu manipulieren, dass sie bei harmlosen Dateien Probleme melden, also Fehlalarme hervorrufen – die sogenannten False-Positive-Fälle. Aktionen dieser Art soll es über 10 Jahre gegeben haben.

Die beiden Ex-Kaspersky-Mitarbeiter sagten gegenüber der Nachrichtenagentur, dass sie einem kleinen Kreis von Kollegen angehört hätten, der immer wieder solche Sabotage-Aufträge erhielt; die restlichen Mitarbeiter seien nicht eingeweiht gewesen. Ihr Chef habe die Manipulationen verlangt, da er verärgert über die Arbeitsweise der Konkurrenz gewesen sei. Statt eigene Verfahren zu entwickeln, würden die Mitbewerber nur Kaspersky-Produkte kopieren - so soll sich der Firmengründer geäußert haben. Mit den Manipulationen sollten laut Reuters-Bericht vor allem Microsoft, AVG und Avast geschädigt werden, aber auch weitere Antiviren-Hersteller.

<http://www.heise.de/newsticker/meldung/Schwere-Vorwuerfe-gegen-Firmenchef-Eugene-Kaspersky-2779946.html>

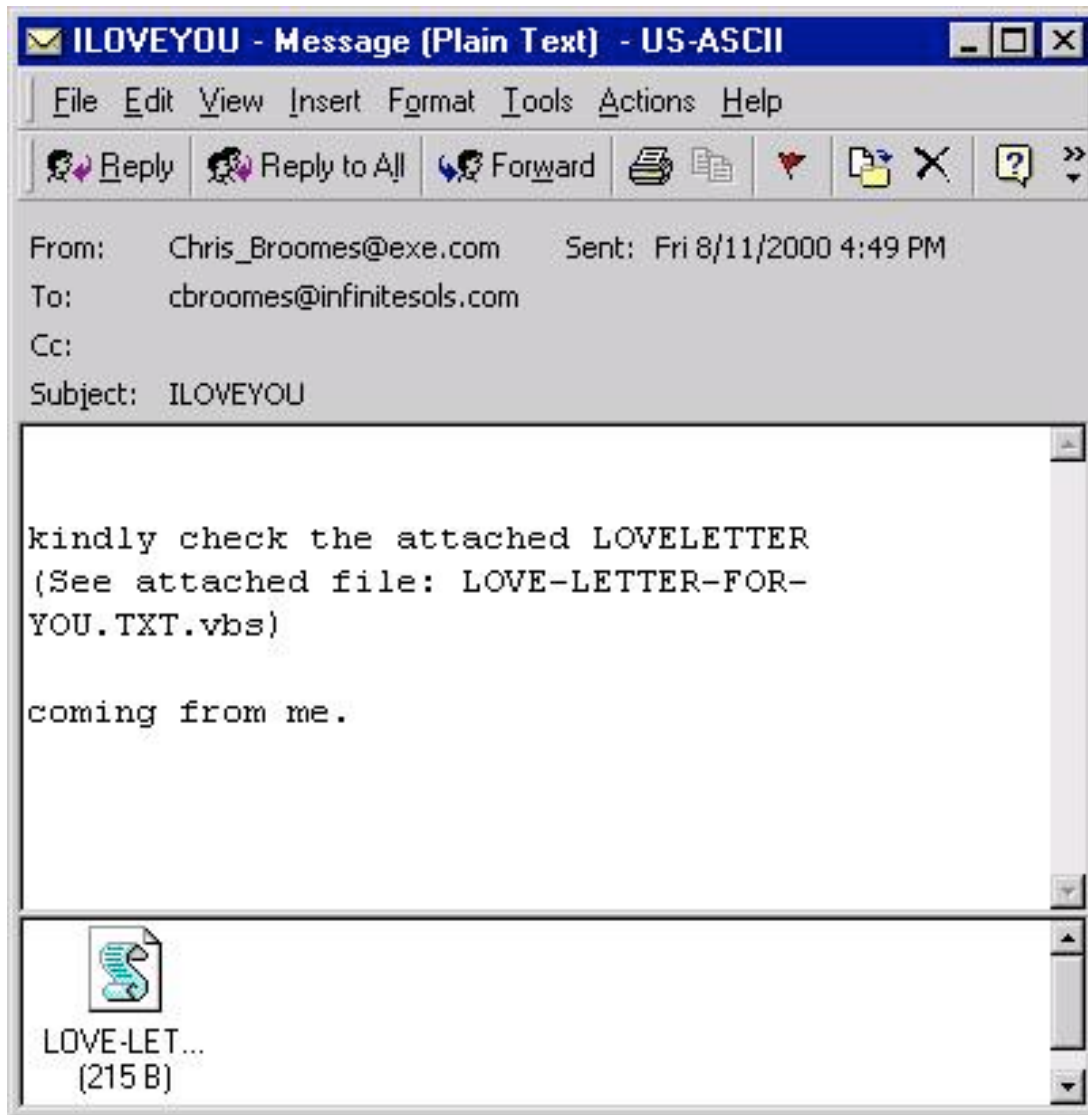
■ Definition

- ❑ Eigenständig lauffähiges Programm - benötigt keinen Wirt!
- ❑ Selbstreplikation (z.B. über Netz oder USB-Sticks (mit „Autorun“))
- ❑ Einzelne infizierte Maschinen werden als Wurm-Segmente bezeichnet

■ Beispiele:

- ❑ Internet-Wurm (1988, vgl. Kap. 1)
- ❑ ILOVEYOU (Mai 2000; ausführbares E-Mail-Attachment, verschickt sich an alle im Adressbuch eingetragenen E-Mail-Adressen)
- ❑ Code Red (Juli 2001; Defacement von Microsoft IIS Webservern)
- ❑ SQL Slammer (2003, vgl. Kap. 1)
- ❑ Conficker (November 2008; Windows-Exploits + Wörterbuch-Angriff; infizierte Maschinen formen Botnet, weltweit > 15 Mio. infizierte Rechner)
- ❑ Stuxnet (Juni 2010, vgl. Kap. 1)
- ❑ Morto (Sommer 2011; Wörterbuch-Angriff via Remote Desktop Protocol)
- ❑ NGRBot (Sept. 2012; tarnt sich per Rootkit, späht Daten aus, blockt Updates)
- ❑

Beispiel: Würmer



Bildquelle: <http://imps.mcmaster.ca/courses/SE-4C03-07/wiki/zagorars/iloveyou.jpg>



Bildquelle: https://lh3.ggpht.com/-hyoPp-zVETc/UALnW5vAcBI/AAAAAAAAAE0/L7H3nUI2Adw/s1600/code_red_thumb.jpg



Bildquelle: <http://inforsecurity.wordpress.com/2010/01/07/virus-conficker-em-65-milhoes-de-maquinas-no-mundo-todo-17-de-dezembro-de-2009/>

■ Definition:

- Ein Programm, dessen Ist-Funktionalität nicht mit der angegebenen Soll-Funktionalität übereinstimmt:
 - Sinnvolle oder attraktive „Nutzfunktionalität“
 - Versteckte (Schad-) Funktionalität
 - Keine selbständige Vervielfältigung

■ Beispiel: Unix Shell Script Trojan [Stoll 89]:

```
echo "WELCOME TO THE LBL UNIX-4 COMPUTER"
echo "LOGIN:"
read account_name
echo "PASSWORD:"
(stty -echo;\
 read password;\
 stty echo; echo "");\
 echo $account_name $password >> /tmp/.pub)
echo "SORRY, TRY AGAIN."
```


- Rundung bei der Zinsberechnung
 - Nutzfunktion: Zinsberechnung mit drei Stellen Genauigkeit
 - Versteckte Funktionalität: Abgerundete Beträge ab der 4. Stelle aufsummieren und auf definiertes Konto buchen.
- T-Online Power Tools (1998)
 - Nutzfunktion: Unterstützende Werkzeuge für den T-Online Decoder
 - Versteckte Funktionalität: Bei der Registrierung (Shareware) werden T-Online-Zugangsdaten übermittelt
- FBI's Magic Lantern / D.I.R.T (Data Interception by Remote Transmission) (2001)
 - Integrierbar in (Nutzfunktion):
 - Word, Excel, Powerpoint
 - RTF (Rich Text Format)
 - Word Perfect
 - Autorun.bat auf CDs
 -
 - Versteckte Funktionalität:
 - Keyboard-Logger
 - Auslesen entfernter Daten
 - Passphrase-Logging (z.B. PGP Private Key Passphrase)
 - Übertragung des entfernten Bildschirminhalts
 - Übertragung v. entferntem Audio (falls Mikro vorhanden)
- „Staatstrojaner“

- **Veröffentlichte Analyse (08.10.2011)**
<http://www.ccc.de/system/uploads/76/original/staatstrojaner-report23.pdf>
- **Chaos Computer Club (CCC) analysiert zugespilte DLL: mfc42ul.dll**
 - Wird per Registry-Eintrag geladen
 - Klinkt sich bei der Initialisierung in explorer.exe ein
- **Funktionen:**
 - Screenshots
 - Abhören von Skype- und VoIP-Gesprächen
 - Nachladen weiterer Module
 - Kommunikation mit Command and Control (C&C) Server



Bundestrojaner als Plastik des CCC
Photo: mellowbox/Flickr

■ Kommunikation:

- Einseitig verschlüsselt zwischen Malware und C&C-Server
- Mit AES-ECB (Electronic Code Book Mode)
 - Jeder Block wird mit dem identischen Schlüssel verschlüsselt, d.h. gleiche Klartextblöcke ergeben identische Chiffre-Blöcke
 - Schlüssel in allen Varianten identisch
- „Authentisierung“ über konstanten Banner-String „C3PO-r2d2-POE“
 - Angreifer kann sich als C&C ausgeben
- Kommando-Kanal (C&C → Malware) unverschlüsselt; keine Authentisierung
 - Malware somit durch Dritte steuerbar
 - Durch Nachladefunktion der Malware kann komplettes System durch Dritten übernommen werden
 - Zielperson kann durch gefälschte Beweise belastet werden
- Fest kodierte Adresse des C&C Servers: 207.158.22.134
 - Adresse gehört Hosting Provider Web Intellects in Ohio, USA

- Nicht alle Kommandos konnten identifiziert werden
- 18 Befehle: „--“ Kommando wird von Dispatcher nicht behandelt
 - cmd 1, cmd 10, cmd 11, cmd 15: --
 - cmd 2: Client verbindet sich neu und versucht, Daten abzusetzen (ähnlich cmd 13)
 - cmd 3: Screenshot geringer Qualität
 - cmd 4: Registrieren eines Kernelmode-Treibers
 - cmd 5: Installation aller malwarespezifischen Dateien im Dateisystem; Quelle noch nicht geklärt
 - cmd 6: Löschen der Malware aus dem Dateisystem und Reboot
 - cmd 7: Entladen der Malware
 - cmd 8: Liste aller Softwarekomponenten
 - cmd 9: wie cmd 3, nur mit drei Argumenten
 - cmd 12: Setzen irgendwelcher Werte
 - cmd 13: Screenshot von Webbrowser und Skype
 - cmd 14: Nachladen eines Programms und unmittelbare Ausführung

- **Auf allen Systemen (Desktop + Server):**
 - Anti-Viren-Software installieren und aktuell halten
 - Keine Software zweifelhafter Herkunft installieren
 - Getrennt gelagerte, regelmäßig erstellte Daten-Backups

- **Auf Desktop-Systemen:**
 - Funktionen wie automatische Makro-Ausführung, Autorun etc. deaktivieren
 - Ggf. virtuelle Maschinen zum „Surfen“ und Ausprobieren von Software verwenden (Isolation, Sandboxing)

- **(Primär) auf Server-Systemen:**
 - Integrity-Checker einsetzen (→ Host Intrusion Detection Systeme)
 - Schreibrechte sehr restriktiv vergeben (Need-to-know-Prinzip)
 - *(Bei Verwundbarkeiten ohne andere Lösung: Impfen, d.h. in die Programme wird bewusst die Signatur des Virus eingetragen.)*

- Diverse “Apps” für Smartphones und Desktops
 - Vordergründig oft kostenlose, interessante Anwendung
 - Im Hintergrund:
 - Übermitteln des gesamten Adressbuchs an Hersteller
 - Übermitteln der eindeutigen Geräteerkennung an Werbenetzwerke
 - Umleiten des Internet-Traffic über Server des Herstellers
 - Mining von Bitcoins o.ähnl.
 - Versand von Premium-SMS o.ähnl.
 - Ohne Analyseumgebung (z.B. Simulator, Netzmonitoring) für Anwender nicht erkennbar

- Hardware-basierte/-nahe Trojanische Pferde
 - Manipulierte Hardware / Firmware, z.B. NSA Supply-Chain Interdiction
 - BadUSB: Z.B. Manipulierte USB Memory-Sticks mit Tastaturemulation zum Absetzen von beliebigen Befehlen

Die NSA fängt Postsendungen ab

Bild 1 von 3

(TS//SI//NF) Such operations involving **supply-chain interdiction** are some of the most productive operations in TAO, because they pre-position access points into hard target networks around the world.



(TS//SI//NF) Left: Intercepted packages are opened carefully; Right: A “load station” implants a beacon

Blick hinter die Kulissen

So werden Pakete offenbar geöffnet (links) und die enthaltene Technik manipuliert (rechts).

Bild: Glenn Greewald, "Die totale Überwachung"

1. Grundlegendes zur Angriffsanalyse

- ❑ Notation von Sicherheitsproblemen
- ❑ Angreifermodelle
- ❑ Begriffe und Zusammenhänge

2. Ausgewählte technische Angriffsvarianten

- ❑ Denial of Service (DoS und DDoS)
- ❑ Schadsoftware (Malicious Code - Viren, Würmer, Trojanische Pferde)
- ❑ E-Mail-Security (Hoaxes und Spam)
- ❑ Mobile Code (ActiveX, JavaScript, ...)
- ❑ Systemnahe Angriffe (Buffer Overflows, Backdoors, Rootkits, ...)
- ❑ Web-basierte Angriffe (XSS, ...)
- ❑ Netzbasierte Angriffe (Sniffing, Portscans, ...)

3. Bewertung von Schwachstellen

- ❑ Common Vulnerability Scoring System (CVSS)
- ❑ Untersuchungen zu “Top Security Risks” nach SANS

■ GEZ-Gebührenerstattung:

Die öffentlich-rechtlichen Rundfunkanstalten ARD und ZDF haben im Frühjahr einen Gewinn von über 1 Mrd. DM erwirtschaftet. Dieses ist gemäß Bundesverfassungsgericht unzulässig. Das OLG Augsburg hat am 10.01.1998 entschieden, daß an diesem Gewinn der Gebührenzahler zu beteiligen ist. Es müssen nach Urteil jedem Antragsteller rückwirkend für die Jahre 1997, 1998 und 1999 je Quartal ein Betrag von DM 9,59 (insgesamt 115,08 DM) erstattet werden.

ACHTUNG! Dieses Urteil wurde vom BGH am 08.04.98 bestätigt.[....] Bitte möglichst viele Kopien an Verwandte, Freunde und Bekannte weiterleiten, damit die Gebühren auch ihnen erstattet werden.

■ AIDS-Infektion im Kino:

Vor einigen Wochen hat sich in einem Kino eine Person auf etwas Spitzes gesetzt, das sich auf einem der Sitze befand. Als sie sich wieder aufgerichtet hat, um zu sehen, um was es sich handelte, da hat sie eine Nadel gefunden, die in den Sitz mit einer befestigten Notiz gestochen war: "Sie wurden soeben durch das HIV infiziert". Das Kontrollzentrum der Krankheiten berichtet über mehrere ähnliche Ereignisse, kürzlich vorgekommen in mehreren anderen Städten.

Alle getesteten Nadeln SIND HIV positiv. Das Zentrum berichtet, dass man auch Nadeln in den Geldrückgabe-Aussparungen von öffentlichen Automaten (Billette, Parking, etc.) gefunden hat. Sie bitten jeden, extrem vorsichtig zu sein in solchen Situationen. Alle öffentlichen Stühle müssen mit Wachsamkeit und Vorsicht vor Gebrauch untersucht werden. Eine peinlich genaue sichtliche Inspektion sollte ausreichen. Außerdem fordern sie jeden auf, allen Mitgliedern Ihrer Familie und Ihrer Freunde diese Nachricht zu übermitteln.

Dies ist sehr wichtig!!! Denk, dass Du ein Leben retten kannst, indem Du diese Nachricht weiter verteilst.

Frank Richert
Polizeidirektion Hannover

Autobahnpolizei Garbsen

- Warnung vor „extrem gefährlichem Virus“
- “Keine AV-Software kann diesen Virus erkennen”
- “Warnen Sie alle Bekannten und Freunde”
- Nicht plausible Bedrohung
(z.B. physische Zerstörung des Rechners)
- Verweis auf namhafte Unternehmen oder
Forschungseinrichtungen
- Kettenbriefe im klassischen Sinn:
 - Gewinnspiele oder Glücksbriefe
 - „Geld zurück“
 - E-Petitionen
 - Pyramidensysteme
 - „Tränendrüsenbriefe“
- Schutzmaßnahmen: Hoax-Mail löschen und NICHT
verbreiten
- Beispiele: <http://hoax-info.tubit.tu-berlin.de/list.shtml>

- Unerwünschte Werbemails (unsolicited commercial e-mail, UCE)
- Begriff SPAM
 - SPAM eingetragenes Warenzeichen von Hormel Food
 - „Spam“-Sketch aus Monty Python's Flying Circus
- E-Mail-Spam-Aufkommen
 - Am Beispiel LRZ, ein Tag im Oktober 2008
 - Zustellversuche für 14.556.000 Mails
 - Spam und Viren-Mails: 14.436.000 (~99,18 %)
 - Abgelehnte Mails: 14.400.000 (~99 %)
 - Als Spam markiert: 35.000 (~0,24 %)
 - Viren-Mails: 1.000 (~0,01 %)
 - Gewünschte Mails („Ham“): 120.000 (~0,82 %)
- Probleme:
 - Eingangs-Mailbox wird mit Spam überflutet
 - Extrem störend, oft „gefährlicher“ Inhalt
 - Zusätzlicher Aufwand (Speicherplatz, Arbeitszeit)
 - Zusätzliche Kosten (Infrastruktur, Übertragung, Personal,....)



Zielgruppenorientierter Spam: Beispiel

Subject: UNIVERSITY DIPLOMAS

Date: Tue, 08 Aug 1996 18:47:06 -0400 (EDT)

Obtain a prosperous future and secure the admiration of all for as little as \$125.

Diplomas from prestigious non-accredited universities based on your life experience.

No tests, no classes, no interviews.
All diplomas available including bachelors, masters, and doctorates (PhD's).

No one is turned down.

Your diploma puts a University Job Placement Counselor at your disposal.

Confidentiality assured.

CALL NOW to receive your diploma within days!!!

1-603-623-0033, Extension 307

Open Every Day Including Sundays and Holidays

Information Regarding Your account:

Dear PayPal Member!

Attention! Your PayPal account has been violated!

Someone with ip address 86.34.211.83 tried to access your personal account!

Please **click the link below** and enter your account information to confirm that you are not currently away. You have 3 days to confirm account information or your account will be locked.

[Click here to activate your account](#)

You can also confirm your email address by logging into your PayPal account at <http://www.paypal.com/> Click on the "Confirm email" link in the Activate Account box and then enter this confirmation number: 1099-81971-4441-9833-3990

Thank you for using PayPal!
The PayPal Team

Please do not reply to this e-mail. Mail sent to this address cannot be answered. For assistance,



PayPal Email ID PP391

Protect Your Account Info

Make sure you never provide your password to fraudulent websites.

To safely and securely access the PayPal website or your account, open a new web browser (e.g. Internet Explorer or Netscape) and type in the PayPal login page (<http://paypal.com/>) to be sure you are on the real PayPal site.

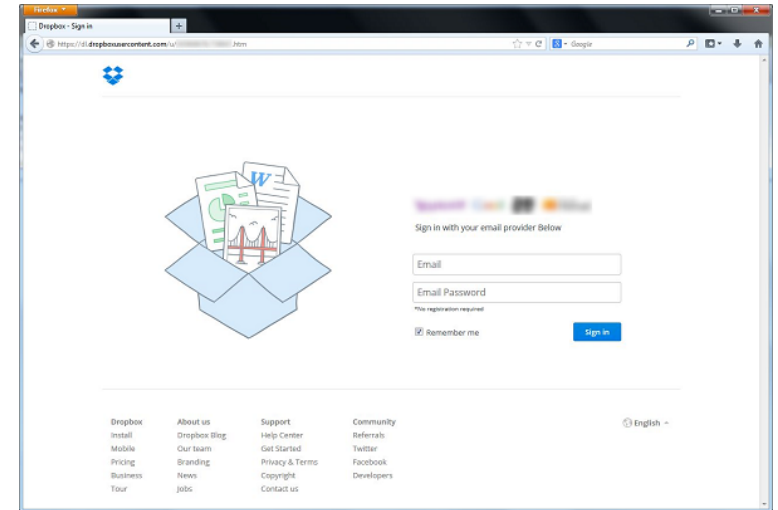
PayPal will never ask you to enter your password in an email.

For more information on protecting yourself from fraud, please review our Security Tips at <https://www.paypal.com/us/securitytips>

Protect Your Password

You should never give your PayPal password to anyone.

- Phishing-Mail mit Dropbox als vermeintlichem Absender
- Angreifer betreibt Phishing-Website über offizielle Dropbox-Domain `dropboxusercontent.com`
- Zugriff auf Phishing-Website über HTTPS somit mit offiziellem Dropbox-Serverzertifikat
- Diverse Logos von E-Mail-Providern motivieren zur Eingabe weiterer Accounts und Passwörter
- Ähnlicher Angriff im März 2014 über Google Docs



Bildquelle: Symantec

Beispiel: Gefälschte Abmahn-Mails fordern Bitcoins (10/2014)

- Verbraucherzentrale Rheinland-Pfalz warnt vor gefälschten Abmahnschreiben
- Als Absender sind reale Anwaltskanzleien angegeben
- Empfänger wird beschuldigt, urheberrechtlich geschütztes Videomaterial abgerufen zu haben
- E-Mail enthält Links auf vermutlich Malware-verseuchte Webseiten
- Forderung nach Entschädigungszahlung in Bitcoins

Quelle: <https://www.verbraucherzentrale-rlp.de/porno-phishing-mails>

- Software, die eingehende Mails nach Spam durchsucht
- Arten von Spam-Filtern:
 1. Blacklist / Whitelist Ansatz:
Aussperren von Mail-Servern und Mail-Domänen, die üblicherweise von Spammer benutzt werden.
 2. Regelbasiert:
Nachricht wird inhaltlich nach Spam-Merkmalen durchsucht; sowohl im Header als auch im Body der Mail.
 3. Filtersoftware lernt aus Beispielen:
Neuronale Netze oder Bayes-Filter bewerten Mailinhalte.
- Vor- u. Nachteile dieser Spam-Filter:
 1. Effizient zu implementieren; aber grobgranular, keine inhaltliche Prüfung.
 2. Sehr hohe Erkennungsraten; aber E-Mail muss vollständig entgegen genommen werden, kontinuierlicher Aufwand für Konfigurationspflege.
 3. Gut in Mail-Clients zu integrieren; aber Erkennungsrate abhängig von Training (NN) bzw. Modellierung (Bayes).

■ Fehlerarten bei der Erkennung

- Filter, die „automatisch“ Entscheidungen treffen, machen zwei Arten von (systematischen) Fehlern:
- **Falsch positiv:** Mail wird als Spam erkannt, obwohl sie Ham ist
- **Falsch negativ:** Mail wird als Ham bewertet, obwohl sie Spam ist

■ Welche Fehlerart ist problematischer?

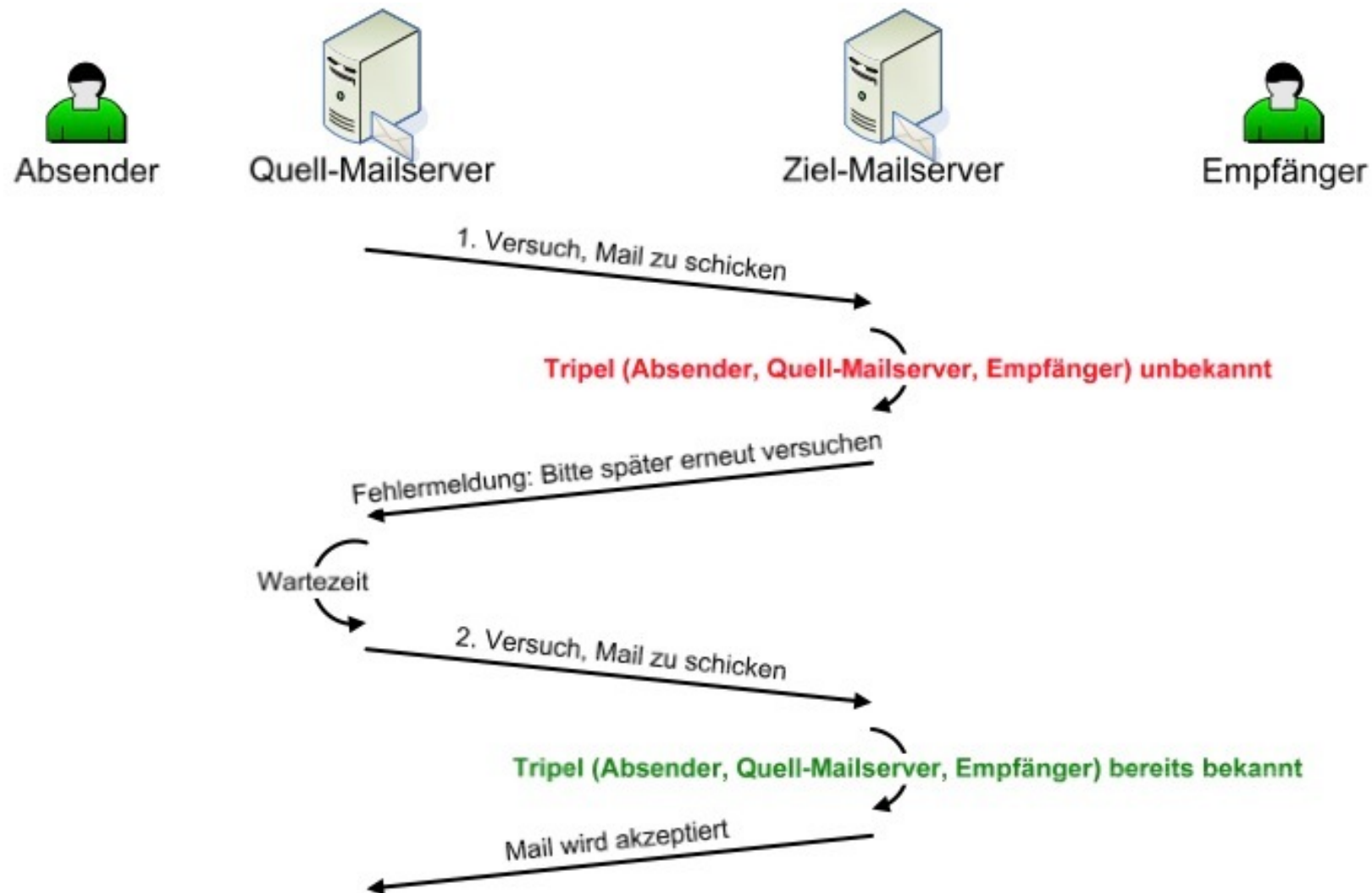
■ Policy für Spambehandlung:

- Spam-Mail löschen und Empfänger ggf. benachrichtigen
- Spam-Mail markieren und dann ausliefern
- Welche Variante bevorzugen (unter Beachtung der Fehlerarten)?
- Vgl. auch Urteil Landgericht Bonn, 15 O 189/13

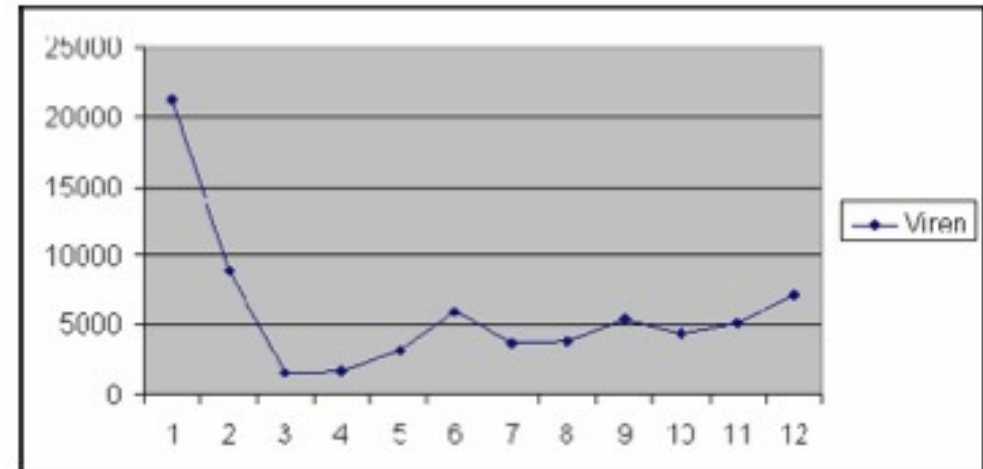
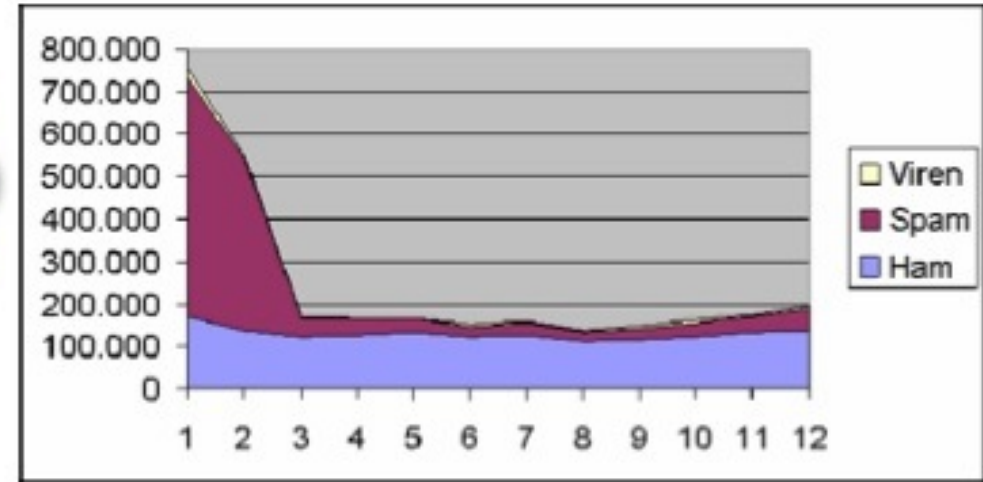
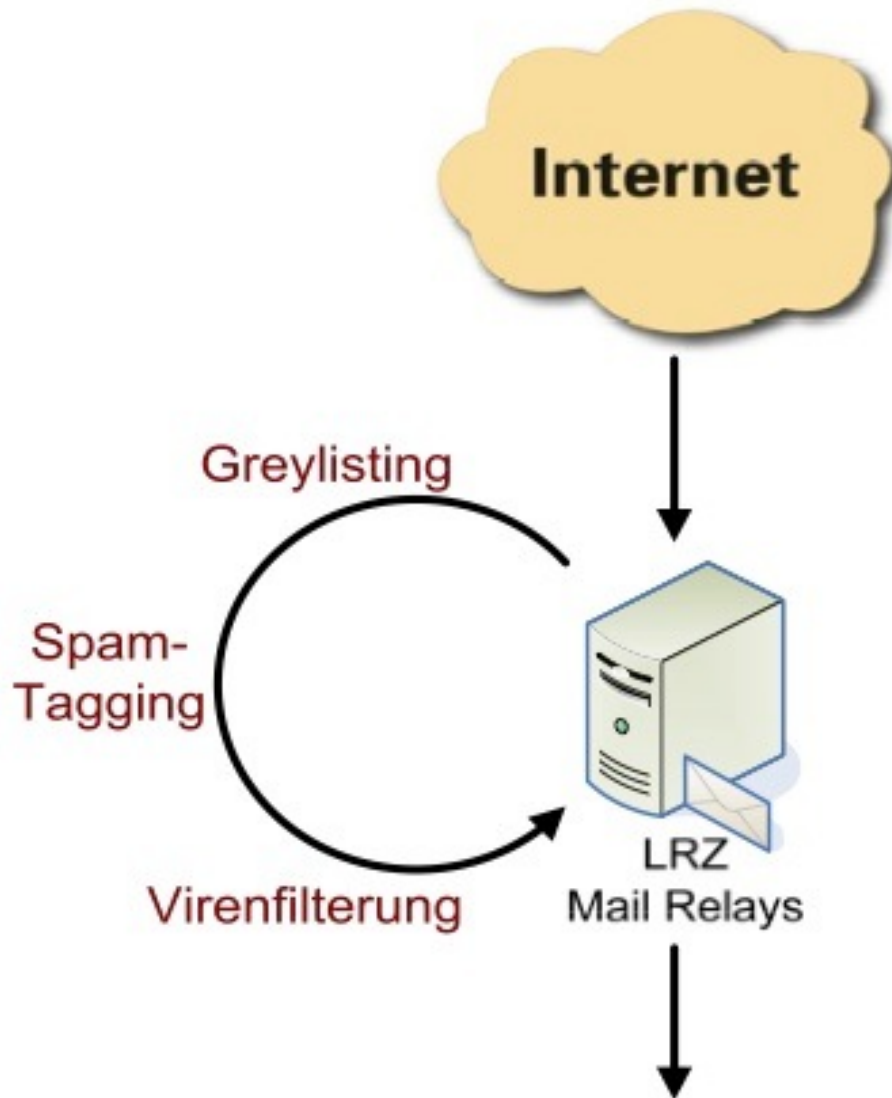
■ Beispiele:

- SpamAssassin (<http://spamassassin.apache.org/>)
 - Implementiert alle Filterarten (Blacklist, Regelbasis, Bayes-Filter)
 - Zentral und dezentral einsetzbar, fein-granular konfigurierbar
- Spamfilter als Cloud-Dienst: Mail-Gateway mit Spamfilter bei externem Dienstleister - kein eigener Konfigurationsaufwand, aber “Mitleser”...

Greylisting gegen Spam (1/2)



Greylisting gegen Spam (2/2)



1. Grundlegendes zur Angriffsanalyse

- ❑ Notation von Sicherheitsproblemen
- ❑ Angreifermodelle
- ❑ Begriffe und Zusammenhänge

2. Ausgewählte technische Angriffsvarianten

- ❑ Denial of Service (DoS und DDoS)
- ❑ Schadsoftware (Malicious Code - Viren, Würmer, Trojanische Pferde)
- ❑ E-Mail-Security (Hoaxes und Spam)
- ❑ Mobile Code (ActiveX, JavaScript, ...)
- ❑ Systemnahe Angriffe (Buffer Overflows, Backdoors, Rootkits, ...)
- ❑ Web-basierte Angriffe (XSS, ...)
- ❑ Netzbasierte Angriffe (Sniffing, Portscans, ...)

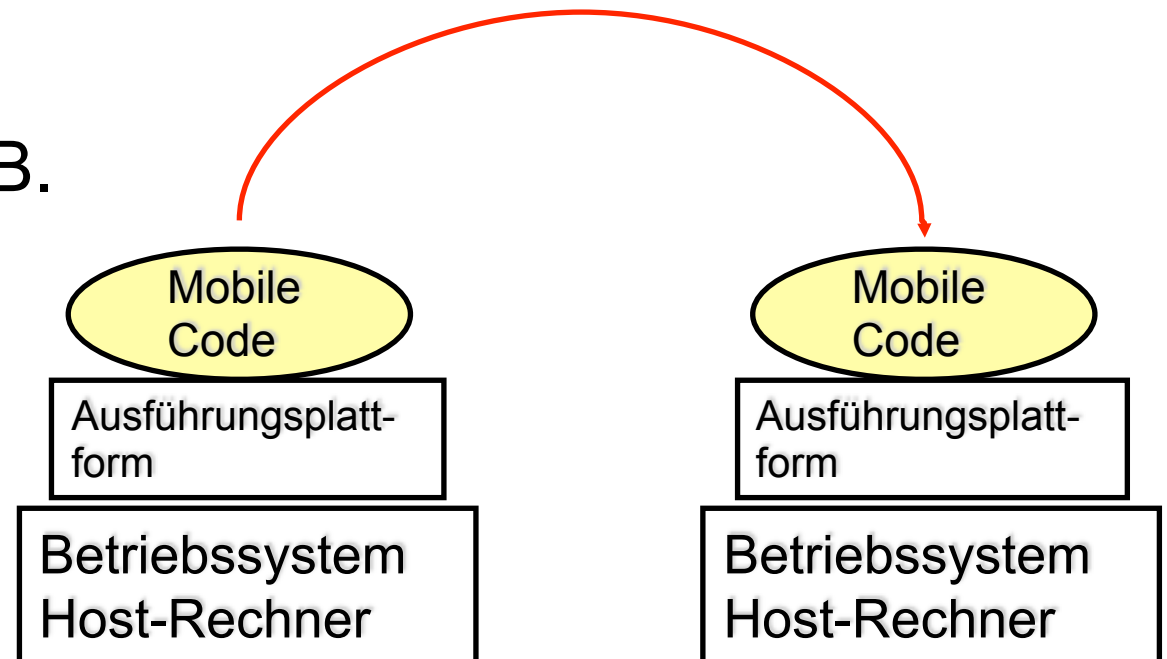
3. Bewertung von Schwachstellen

- ❑ Common Vulnerability Scoring System (CVSS)
- ❑ Untersuchungen zu “Top Security Risks” nach SANS

- Abgrenzung zu Viren, Würmern und Trojanischen Pferden fließend
- Hier - Mobile Code (aktiver Inhalt):
 - Code wird auf entferntem Rechner generiert,
 - typischerweise in Webseiten eingebettet und
 - auf lokalem Client-Rechner ausgeführt.
 - I.d.R. Ausführungsplattform oder Interpreter zur Ausführung erforderlich

- **Verwendete Sprachen z.B.**

- ActiveX
- JavaScript
- Java
- ActionScript (Flash)
- Silverlight
- HTML 5



- Von Microsoft entwickelte Erweiterung von OLE (Object Linking and Embedding)
- ActiveX Control:
 - Wiederverwendbare Komponente
 - Binärformat
 - Standardisierte Schnittstelle
 - Beliebige Programmiersprache zur Entwicklung (z.B. C, Basic, C#,...)
 - Wird innerhalb des Browsers ausgeführt
- Probleme bei der Einführung:
 - Keine Ausführungsbeschränkung
 - Voller Betriebssystemzugriff
 - Selbe Rechte wie ausführender Benutzerprozess
- Beispiele für ActiveX Malware:
 - Internet Explorer (1996):
“Signed” ActiveX Control, das bei der Ausführung den Rechner herunterfährt.
 - Chaos Computer Club (CCC) Demonstrator (27.01.1997)
 - Control sucht nach Quicken
 - Erstellt Überweisung und trägt diese in die Liste offener Überweisungen in Quicken ein.
 - Quicken konnte mit einer PIN/TAN-Kombination mehrere Überweisungen übertragen, d.h. unvorsichtiger User wird „gefälschte“ Überweisung mit übertragen
 - `www.iks-jena.de/mitarb/lutz/security/activex.html`

- Entwickelt von Netscape
 - Skriptsprache; syntaktisch angelehnt an C, C++ u. Java
 - Einbettung aktiver Inhalte in Webseiten
 - Wird innerhalb des Browsers ausgeführt.
- JavaScript Skript:
 - Kein Zugriff auf das Dateisystem (*außer* auf Cookies)
 - Keine Netzverbindungen (*außer* Download von URLs)
- Probleme
 - Kein explizites Sicherheitsmodell
 - Entwicklungsgrundsatz: „Identify (security holes) and patch approach“
- Umfangreiche Liste von Schwachstellen und Implementierungsfehlern
- Netscape 2.x
 - Auslesen der History
 - Lesender und schreibender Zugriff auf das Dateisystem
- Netscape 3.x
 - Versenden von Mail
- Netscape 4.x
 - Hidden frame mit eingebetteter Post Methode + Attachment sendet Files an böswilligen Web-Server
 - JavaScript eingebettet in Cookie; damit z.B. Lesen der Bookmarks oder HTML-Dateien im Cache
www.peacefire.org/security/jscookies/

- CVE-Datenbank (Common Vulnerabilities and Exposures) führt dreistellige Anzahl von JavaScript-bezogenen Sicherheitsproblemen (<http://cve.mitre.org/index.html>)

The screenshot shows a ZDNet news article. The main headline is "RIM: Blackberry-Nutzer sollen JavaScript deaktivieren". The article is dated 16. März 2011, 09:52 Uhr and is written by Ryan Naraine and Stefan Beiersmann. The article text discusses a security issue where RIM users are advised to deactivate JavaScript in their browsers to protect against remote code execution. A photo of a BlackBerry Torch 9800 is included. To the right, there are social media recommendations and a Facebook plugin showing 2,027 likes for the article.

ZDNet Suche Newsletter

DE-Edition News IT-Business Tests & Technik Mobile Business Security Developer ITpapers Downloads Populär Specials

ZDNet / News / Security

RIM: Blackberry-Nutzer sollen JavaScript deaktivieren

von Ryan Naraine und Stefan Beiersmann, 16. März 2011, 09:52 Uhr

Themenseiten

Security, Smartphone, Browser, Research In Motion

Mehr zum Thema

- Google stopft WebKit-Lücke in Chrome 10
- Pwn2Own 2011: Hacker demonstrieren Lücken in iOS und Blackberry OS
- Bericht: Blackberry OS 7 erscheint Ende 2011
- Bericht: RIM bringt Blackberry Messenger auf Android und iOS

Research In Motion hat eine Behelfslösung für die in der vergangenen Woche bekannt gewordene Anfälligkeit im Blackberry-Browser veröffentlicht. Das Unternehmen rät betroffenen Kunden, JavaScript im Browser zu deaktivieren. Damit soll verhindert werden, dass Angreifer die Sicherheitslücke ausnutzen können.



Die Schwachstelle hatten die Sicherheitsforscher Vincenzo Iozzo, Ralf Philipp Weinmann und Willem Pinckaers auf der Hackerkonferenz Pwn2Own demonstriert. Sie nutzten sie, um die Kontrolle über ein Blackberry Torch 9800 mit der Firmwareversion 6.0 oder höher zu erlangen. Unter anderem speicherten sie eine Datei auf dem Handy und zeigten damit, dass Remote-Codeausführung möglich ist.

RIM hat seinen Nutzern empfohlen, JavaScript zu deaktivieren, um sich vor einer Schwachstelle im Blackberry-Browser zu schützen (Bild: RIM).

Empfehlungen auf Facebook

- Registrieren** Erstelle ein Konto oder **melde dich an**, um zu sehen, was deine Freunde machen.
- Facebook wegen Bespitzelung in mehreren US-Staaten verklagt**
54 Personen empfehlen das.
- Google stellt Android 4.0 Ice Cream Sandwich offiziell vor**
11 Personen empfehlen das.
- Erneute Blackberry-Ausfälle in Europa, Nahost und Afrika**
60 Personen empfehlen das.
- EU erwägt Regulierung von E-Book-Formaten**
22 Personen empfehlen das.

Soziales Plug-in von Facebook

ZDNet.de auf Facebook
Gefällt mir 2,027

Nachrichten des Tages

- Firma RSA Security stellt u.a. weltweit stark verbreitete Token zur Authentifizierung her (RSA SecurID)
- Spear-Phishing Angriff auf RSA-Mitarbeiter: Excel-Attachment „2011 Recruitment Plan.xls“, vermutlich mit Excel 2007 geöffnet.
- Eingebettetes SWF-File nutzt Adobe-Flash-Player-Lücke aus.
- Schadcode (Abwandlung von „poison ivy“) späht Mitarbeiterrechner aus und überträgt u.a. Passwörter an den Angreifer.

- Folgen:
 - SecurID-Quellen und -Seeds werden ausgespäht
 - US-Rüstungsunternehmen Lockheed Martin wird mit „nachgebauten“ SecurID-Token gehackt; zahlreiche weitere Unternehmen betroffen
 - Rund 40 Millionen SecurID-Token werden ausgetauscht

12.08.2015 12:23

125

Adobe und das Sieb: 35 Flash-Lücken gestopft

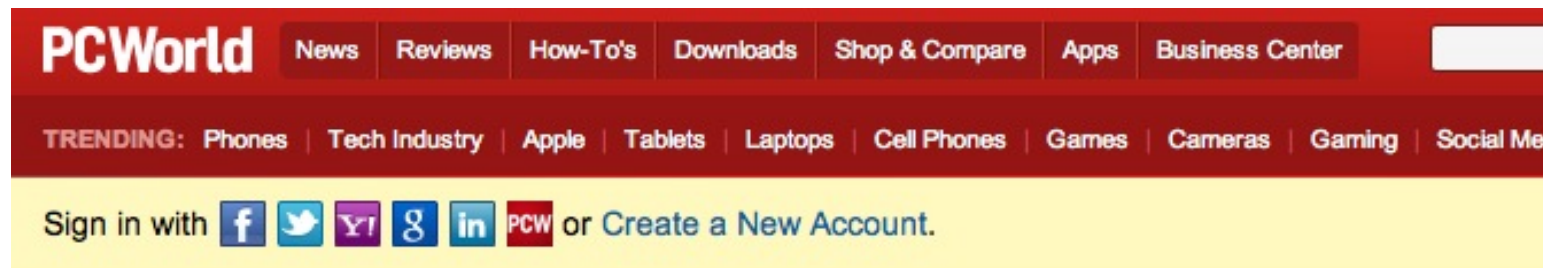


Am heutigen Patchday liefert Adobe neue Flash-Versionen aus, die insgesamt 35 Sicherheitslücken stopfen sollen – die meisten davon sind kritisch.

Es nimmt kein Ende: Erneut stellt Adobe Flash-Updates für alle Versionen bereit. Erneut stopfen sie so viele Lücken, dass man sie gar nicht mehr einzeln aufzählen mag. Ganze 35 Lücken mit eigenem Common Vulnerability Identifier (CVE) listet Adobe in seinem [Security Bulletin APSB15-19](#) auf.







34 der Lücken sind als hoch kritisch eingestuft, sprich: Angreifer, die sie ausnutzen, können damit ein System übers Netz mit Schad-Software infizieren. Betroffen sind vor allem die Versionen für Windows und Mac vor 18.0.0.232 und dabei auch die bei Internet Explorer und Chrome verwendeten Flash-Erweiterungen, die [Microsoft](#) beziehungsweise [Google](#) direkt stopfen. Aber auch Flash für Linux und Adobe AIR sind anfällig.

<http://m.heise.de/security/meldung/Adobe-und-das-Sieb-35-Flash-Luecken-gestopft-2777079.html>



PCWorld News Reviews How-To's Downloads Shop & Compare Apps Business Center

TRENDING: Phones | Tech Industry | Apple | Tablets | Laptops | Cell Phones | Games | Cameras | Gaming | Social Me

Sign in with       or Create a New Account.

PCWorld » Software » Multimedia

Recommend:   0  +1 0  0  Email 1 Comment Print

Silverlight Declared Secure


Microsoft's upcoming Web media tool is safe from most common exploits, security tests indicate.

By Eric Lai, Computerworld May 2, 2007 6:00 pm

Silverlight, Microsoft Corp.'s upcoming Web media software, may be several months from its official release, but experts have already reached a consensus -- albeit a weak one -- about how secure it will prove to be.

That consensus favors Microsoft's argument that the software won't be easily exploitable by hackers. Microsoft says that Silverlight, a browser plug-in that works with Internet Explorer, Firefox and Safari, has key attributes that should prevent Silverlight from such exploits.




Quelle: http://www.pcworld.com/article/131472/silverlight_declared_secure.html

Worldwide Log

Products & Services
Support
How to Buy
Training & Events
Partners

HOME > SECURITY INTELLIGENCE OPERATIONS > Latest Threat Information > Multivendor Security Alerts

Vulnerability Alert

Microsoft Silverlight Pointer Handling Memory Corruption Vulnerability

<p>Threat Type: Unintended Weakness: Arbitrary Code Execution</p> <p>IntelliShield ID: 21034</p> <p>Version: 1</p> <p>First Published: August 10, 2010 02:54 PM EDT</p> <p>Last Published: August 10, 2010 02:54 PM EDT</p> <p>Vector: Network</p> <p>Authentication: None</p> <p>Exploit: Unproven</p> <p>Port: Not Available</p> <p>CVE: CVE-2010-0019</p>	<p>Urgency: Unlikely Use </p> <p>Credibility: Confirmed </p> <p>Severity: Moderate Damage </p> <p>CVSS Base: 9.3 CVSS Calculator</p> <p>CVSS Temporal: 6.9 CVSS Version 2</p>	<p>Powered by:</p> <p>Related Links:</p> <p>Solutions</p> <ul style="list-style-type: none"> Security Solutions E-mail Security Threat Control Threat Control <p>Products & Services:</p> <ul style="list-style-type: none"> Cisco Security Manager Security Products Security Services
---	--	--

Version Summary: Microsoft Silverlight contains a vulnerability that could allow an unauthenticated, remote attacker to execute arbitrary code with the privileges of a targeted user. Updates are available.

Quelle: <http://tools.cisco.com/security/center/viewAlert.x?alertId=21034>

- Browser werden mehr und mehr zum vollwertigen “Betriebssystem”

- Neue Funktionen ..., z.B.:
 - Web Storage API
 - WebSockets API
 - Cross-Origin Resource Sharing

- ... bergen neue Risiken, z.B.:
 - Benutzer stellen Rechenleistung und Speicherplatz zur Verfügung
 - Clients bauen (beliebige) Netzverbindungen auf

- Beispiel: distPaste (Jan-Ole Malchow, FU Berlin)
 - <http://www.dfn-cert.de/dokumente/workshop/2013/FolienMalchow.pdf>
 - Speichert Dateien ggf. verteilt auf mehrere Clients (2,5 MB pro Node)
 - Wer ist verantwortlich für die Inhalte?

1. Grundlegendes zur Angriffsanalyse

- ❑ Notation von Sicherheitsproblemen
- ❑ Angreifermodelle
- ❑ Begriffe und Zusammenhänge

2. Ausgewählte technische Angriffsvarianten

- ❑ Denial of Service (DoS und DDoS)
- ❑ Schadsoftware (Malicious Code - Viren, Würmer, Trojanische Pferde)
- ❑ E-Mail-Security (Hoaxes und Spam)
- ❑ Mobile Code (ActiveX, JavaScript, ...)
- ❑ Systemnahe Angriffe (Buffer Overflows, Backdoors, Rootkits, ...)
- ❑ Web-basierte Angriffe (XSS, ...)
- ❑ Netzbasierte Angriffe (Sniffing, Portscans, ...)

3. Bewertung von Schwachstellen

- ❑ Common Vulnerability Scoring System (CVSS)
- ❑ Untersuchungen zu “Top Security Risks” nach SANS

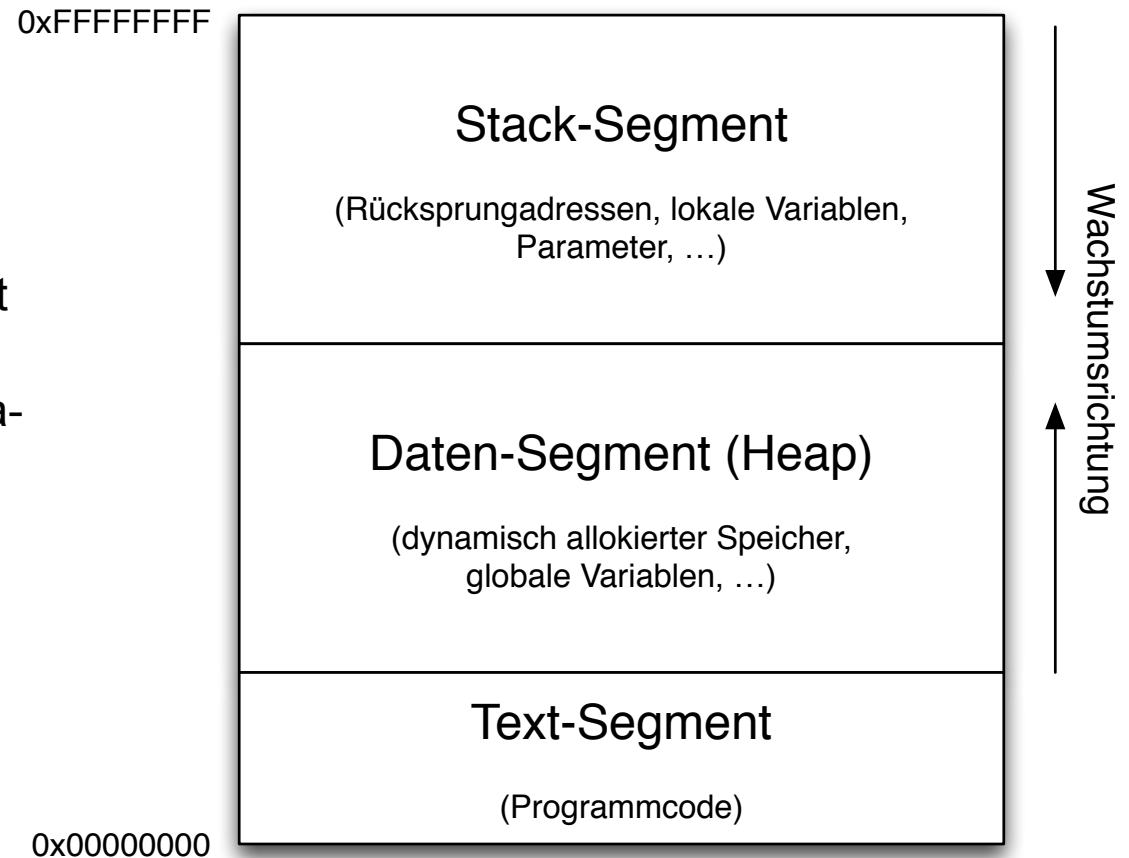
Exploits: Buffer Overflow (hier: stack smashing)

- Ziel: Ausführen von Code auf fremdem Rechner unter fremden Rechten (z.B. *root*)

- Vorgehen:

- Auswahl des Ziels:
 - Lokal: Programm, das z.B. mit SUID (Set User ID)-Bit, d.h. mit Rechten des Eigentümers (meist *root*), läuft.
 - Remote: Netzdienst, z.B. Samba-Fileserver
- Überschreiben interner Programmpuffer, z.B. durch überlange Eingabe
- Dabei Manipulation z.B. der Rücksprungadresse, dadurch Ausführen von bestimmter Programmsequenz des Angreifers; z.B. Code zum Starten einer Shell

- Speicherabbild eines Programms (am Bsp. Unix)



```
1 #include <string.h>
2
3 void kopiere_eingabe (char *eingabe)
4 {
5     char kopie_der_eingabe[128];
6     strcpy(kopie_der_eingabe, eingabe);
7 }
8
9 int main (int argc, char **argv)
10 {
11     kopiere_eingabe(argv[1]);
12 }
```

Hinweis:

Betrifft nicht nur Kommandozeilenparameter, sondern z.B. auch interaktive Eingaben, Datenpakete über Netz, Parsen von Dateien, ...

- Kommandozeilenparameter (`argv[1]`) wird vom Angreifer gesteuert.
- Programmierer hat Eingabe < 128 Zeichen angenommen.
- Wenn `strlen(argv[1]) > 127`, dann reicht der reservierte Speicherplatz für die Kopie des Strings nicht aus („buffer overflow“).
- Folge: Andere Stack-Elemente werden überschrieben („stack smashing“).

Ausnutzen von Buffer Overflows in Stack-Segmenten

- Ziel: Stack gezielt überschreiben, so dass
 - Rücksprungadresse auf Angreifer-Code umgebogen wird
 - Angreifer-Code das System kompromittiert (z.B. Starten einer interaktiven Shell oder Nachladen beliebiger Schadprogramme)

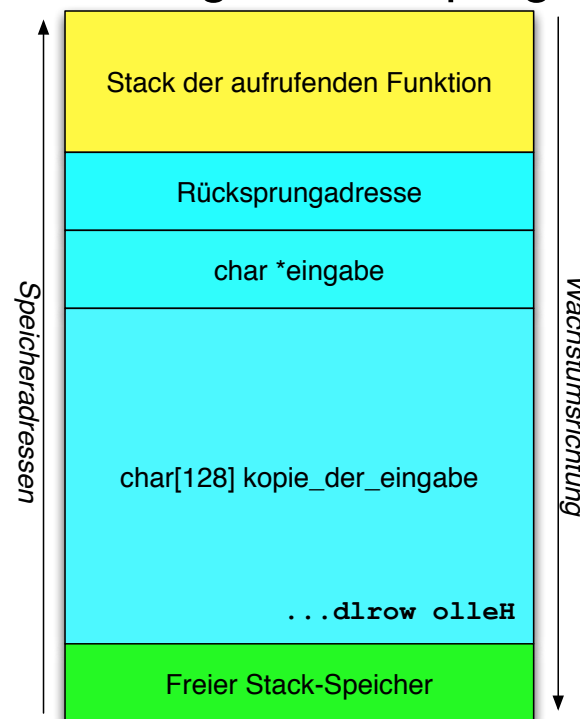
```

1 #include <string.h>
2
3 void kopiere_eingabe (char *eingabe)
4 {
5     char kopie_der_eingabe[128];
6     strcpy(kopie_der_eingabe, eingabe);
7 }
8
9 int main (int argc, char **argv)
10 {
11     kopiere_eingabe(argv[1]);
12 }

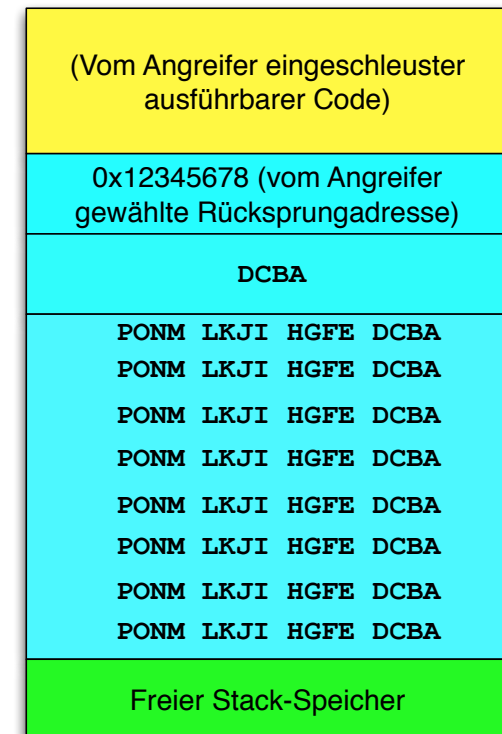
```

Quelltext

Anmerkung: Darstellung des Stack-Aufbaus vereinfacht!



Stack bei regulärer Eingabe



Stack bei Buffer Overflow

- Rücksprungadresse ist absolut (nicht relativ) anzugeben.
- Lösung: NOPs vor eigentlichem Schadcode:

```
Rücksprung erfolgt  
„irgendwo“ hierhin:  NOP  
                      NOP  
                      → NOP  
                      NOP  
Schadcode beginnt  
ab hier:             NOP  
                      NOP  
                      → mov AH, 1  
                      int 21  
                      ...
```

- Das Stack-Segment bietet nur wenig Speicherplatz für eingeschleusten Code.
- Lösungen: Shellcode kompakt in Assembler programmieren; dynamisches Nachladen von Schadcode.
- Quellcode von proprietärer Software nicht verfügbar.
- Lösung: Fuzzing

■ Ziele:

- Nachbildung des Funktionsaufrufs `system("/bin/sh");`
- Shellcode darf keine Nullbytes (0x00) enthalten, damit u.a. strcpy nicht abbricht.

■ Beispiel (Quelle: www.shell-storm.org; Autor: kernel_panic)

```
char code[] = "\x31\xc9\xf7\xe1\x51\x68\x2f\x2f"  
              "\x73\x68\x68\x2f\x62\x69\x6e\x89"  
              "\xe3\xb0\x0b\xcd\x80";
```

■ Größe: 21 Bytes, Plattform: Linux/x86

- Alternative zum Ausführen eigenen Codes: *return-to-libc*, d.h. Einsprung in Standard-Funktionsbibliothek mit eigenen Parametern (z.B. wiederum Aufruf von `system()`).

- Am Besten: Sicheres Programmieren, z.B. `strncpy` statt `strcpy`
 - Unterstützung durch Code-Analyse-Tools, z.B. Splint
- Stack-Guarding:
 - Beim Aufruf einer Unterfunktion wird hinter der Rücksprungadresse ein Kontrollzeichen („Canary“) abgelegt.
 - Vor dem Rücksprung wird geprüft, ob das Kontrollzeichen noch intakt ist.
 - Variante: Mehrere Kopien der Rücksprungadresse.
- Nicht-ausführbare Stacks (non-executable stack)
 - Code auf dem Stack wird vom Betriebssystem generell nicht ausgeführt, damit auch kein eingeschleuster Shellcode.
 - Inzwischen von vielen Prozessoren hardware-unterstützt („NX bit“)
 - Schützt aber weder vor Shellcode auf dem Heap noch vor *return-to-libc*
- Address space layout randomization (ASLR)
 - Speicherbereiche u.a. für Stack werden zufällig gewählt.
 - Angreifer hat es schwerer, die richtige Rücksprungadresse anzugeben.

■ Heap Corruption

- Überschreiben von programminternen Datenstrukturen mit vom Angreifer vorgegebenen Werten

■ Problematisch sind nicht nur String-Operationen

- int-Überlauf
- Schleifen mit Abbruchkriterien, die von der Angreifer-Eingabe nicht erfüllt werden
- Multi-byte character encodings (Unicode)

■ Format String Attacks

- `printf(buffer)` statt `printf("%s", buffer)` bei Benutzereingaben wie `"%x"`
- Überschreiben interner Datenstrukturen bei Anwendung z.B. auf `sprintf()`

■ Literatur:

- Buffer Overflow Attacks. Detect, Exploit, Prevent; Syngress Media 2005

- Buffer Overflows sind kein auf Server oder Multi-User-Systeme begrenztes Problem!
- CVE-2010-0364: VLC Media Player - .ogg Files
- CVE-2013-1954: VLC Media Player - .asf Files
- Modifizierte .ogg/.asf-Dateien werden nicht richtig verarbeitet
-> Buffer Overflow
- Mallory kann beliebigen eigenen Code ausführen, wenn Alice die präparierte Musik-/Filmdatei abspielt
(-> vgl. VLC Browser Plugin)
- Verfügbarkeit von Patches? CVE-2013-1954:
 - Meldung am 11.01.2013, Patch am 17.01.2013, Release am 11.04.2013

- Passworteingabe ist das am weitesten verbreitete Authentifizierungsverfahren
- Ziel des Angriffs: „Erraten“ von Benutzernamen und Passwort
- Varianten:
 - Brute-Force Angriff
 - Dictionary Attack (Wörterbuchangriff)
 - Brechen des Hash-/Verschlüsselungsalgorithmus für das Passwort
 - Social Engineering
- Password Cracking am Beispiel älterer UNIX-Systeme:
 - Administrator (`root`) vergibt Benutzernamen
 - Eintrag in `/etc/passwd`
 - Datei für **alle** lesbar
 - Format des Eintrags

```
huber:Ad9%y?SmW+zP&:23:17:Herbert Huber:/home/huber:/bin/bash
Username:Password:UID:GID:Gecko-String:Home-Verzeichnis:Shell
```

- Benutzer wählt Passwort
 - Passwort wird mit sich selbst als Schlüssel verschlüsselt und verschlüsselt gespeichert in `/etc/passwd`:
z.B. `:Ad9%y?SmW+zP&:`
 - Auch `root` kennt Passwort **nicht**
- Authentisierung:
 - Eingegebenes Passwort wird mit sich selbst verschlüsselt und mit dem in `/etc/passwd` verglichen.
- Verschlüsselungsalgorithmus
`crypt(pwd, salt)` bekannt
- Dictionary Attack:
 - Angreifer verschlüsselt Wörter aus Wörterbuch und vergleicht verschlüsselte Strings mit Einträgen in `/etc/passwd`
- Verhinderung der Dictionary Attack
 - Zus. Parameter `salt` in `crypt`
 - 12 Bit Zahl: $0 \leq \text{salt} < 4096$
 - Bei Initialisierung zufällig gewählt
 - Die ersten 2 Zeichen im Passwort String sind `salt`; im Beispiel: `Ad`
- Brute Force Dictionary Attack:
 - Angreifer muss Wörterbuch für **jeden** Benutzer mit dessen `salt` verschlüsseln und vergleichen
 - Bei heutiger Rechenleistung kein echtes Problem.
- Verhinderung z.B. durch:
 - Shadow Password System (nur `root` kann verschl. Passwort lesen)
 - One-Time Passwords
 - Alternativen zu `crypt()`

- In die Verschlüsselung fließen zwei zufällig gewählte Zeichen ("Salt") ein.
- Salt wird in der Ausgabe im Klartext hinterlegt.
- Angreifer müsste 4096 Werte pro Wörterbuch-Eintrag vorab berechnen.
- (Aus heutiger Sicht kein großer Aufwand mehr)

```
1 #include <stdio.h>
2 #include <unistd.h>
3
4 int main(void)
5 {
6     char *ergebnisAA, *ergebnisxy;
7
8     ergebnisAA = crypt("GeheimesPasswort", "AA");
9     printf("Salt AA: %s\n", ergebnisAA);
10
11     ergebnisxy = crypt("GeheimesPasswort", "xy");
12     printf("Salt xy: %s\n", ergebnisxy);
13
14     return 0;
15 }
```

Ausgabe: `Salt AA: AA3w0THiFXV1A`
`Salt xy: xyj.4bikXtQ1o`

- Neuerer Ansatz:
 - Verschlüsselte / gehashte Passwörter in /etc/shadow ausgelagert.
 - Nur noch „root“ hat überhaupt Lesezugriff, reguläre Benutzer kommen nicht an die verschlüsselten / gehashten Passwörter heran.
 - Längeres Salt.
 - Aufwendigere Hashverfahren, z.B. SHA-512, in mehreren Runden angewandt.
 - Nutzung von "Slow Hash Functions" wie PBKDF2, bcrypt, scrypt.

- Ziel: Angreifer will dauerhaften Zugang (Hintereingang) zu einer bereits kompromittierten Maschine
 - An der Betriebssystem-Authentisierung vorbei
 - Mit speziellen Rechten (z.B. root)
- Mechanismen z.B.:
 - „Verstecktes“ eigenes SUID-root Programm mit „shellcode“.
 - SUID-root Systemprogramm durch eigene Version mit versteckter Funktionalität austauschen.
 - Installation eines “versteckten” Netzdienstes, der zu bestimmten Zeiten einen Netzwerk-Port öffnet und auf Kommandos wartet.
 - Eintrag in `.rhosts`-Datei von root bzw. `authorized_keys` für SSH-Zugang
- Detektion durch Integritäts-Checks:
 - Kryptographische Prüfsummen:
 - aller installierten Programme
 - Konfigurationsdateien
 - regelmäßige Überprüfung
 - Überprüfung der offenen Ports und der aktivierten Netzdienste
 - Suche nach ungewöhnlichen SUID/SGID-Programmen
- Reaktion bei erkannten Hintertüren:
 - Vollständiges Entfernen der Schadsoftware wirklich möglich?
 - Ggf. Maschine neu bzw. aus „sauberem“ Backup aufsetzen.
 - Verwundbarkeit, die zur Kompromittierung geführt hat, muss behoben werden!

■ Begriffsbildung:

- Zusammensetzung aus *root* (= Administratorkennung unter UNIX/Linux) und *Toolkit* (= Werkzeugkasten)
- Ursprünglich Bezeichnung für zueinander komplementäre UNIX-Systemprogramme mit eingebauten Backdoors (1. Generation Rootkits)

■ Typischer Ablauf:

- Angreifer kompromittiert Maschine und erlangt root-Berechtigung
- Angreifer installiert Rootkit
 - Werkzeuge aus dem Rootkit bereinigen Spuren u.a. in Logfiles
 - Backdoors ermöglichen kontinuierlichen root-Zugang für Angreifer
- Rootkits der 1. Generation bestehen aus eigenen Varianten von Kommandos und Programmen wie *ps*, *ls*, *top*, *du*, *find*, *netstat*, *passwd*, *sshd*, ...
- Alle ersetzten Systembefehle verstecken Prozesse, Dateien etc. des Angreifers.

■ Detektion über Host-IDS und Tools wie *chkrootkit*

■ Rootkits der 2. Generation

- Motivation: Alle Systemprogramme einzeln auszutauschen ist aus Angreifersicht aufwendig und fehleranfällig.
- Neuer Lösungsansatz: Betriebssystemkern (Kernel) modifizieren
→ Dateien, Prozesse etc. des Angreifers werden vor allen Systemprogrammen versteckt

■ LKM-Rootkits unter Linux

- Loadable Kernel Module → OS-Kern wird zur Laufzeit erweitert
- Kernelmodul ersetzt Systemfunktionen z.B. zum
 - Auslesen von Verzeichnisinhalten (Verstecken von Dateien)
 - Zugriff auf die Prozessliste (Verstecken von Malware)
- Ggf. mit Backdoor (spezieller Funktionsaufruf liefert root-Berechtigung)

■ Prävention

- Nachladen von Kernelmodulen komplett deaktivieren

■ Detektion

- „Sauberes“ System nur nach Booten z.B. von USB-Stick oder CD

- **Sony BMG copy protection rootkit (2005)**
 - Musik-CDs mit Abspiel-Software für Windows-PCs
 - Heimlich wird ein Rootkit mit installiert, das zum DRM-Enforcement den Zugriff auf die CD einschränkt.
 - Versteckt alle Dateien, deren Name mit "\$sys\$" beginnt.
 - In der Folge taucht vermehrt Malware auf, die sich mit solchen Dateinamen tarnt.

- **Banker-Rootkit (64-Bit-Variante 2011)**
 - Deaktiviert Signatur-Zwang für Windows-Treiber.
 - Installiert eigenen Filesystem-Treiber.
 - Installiert gefälschtes Wurzelzertifikat und modifiziert HOSTS-Datei.
 - Benutzer landet auf einer nachgebauten Online-Banking-Website des Angreifers, die vom Browser als vertrauenswürdig eingestuft wird.

■ Hypervisor-level Rootkits:

- ❑ Rootkit übernimmt das komplette System
- ❑ Ursprüngliches Betriebssystem wird als virtuelle Maschine ausgeführt
- ❑ Beispiel: Blue Pill (2006)

■ Bootkits:

- ❑ Angreifer ersetzt Bootloader durch Malware
- ❑ Hebelt auch Schutz durch komplett verschlüsselte Festplatten aus
- ❑ Beispiele: Evil Maid Attack, Stoned Bootkit, Alureon

■ Hardware- / Firmware-Rootkits:

- ❑ Rootkit installiert sich z.B. im BIOS oder in der Firmware der Netzwerkkarte (Beispiel: Delugré-NetXtreme Rootkit 2010)

■ Zuverlässige Detektion schwierig

- ❑ Timing: Erkennen der rootkit-virtualisierten Umgebung durch veränderte Dauer z.B. von Systemaufrufen. (Problem: zu viele False-Positives)
- ❑ Externe Analyse (Booten von CD)

Lenovos Service Engine: BIOS-Rootkit direkt vom Hersteller

 heise Security 13.08.2015 16:01 Uhr – Dennis Schirmmacher  vorlesen

(Bild: dpa, Diego Aubel)

Lenovos Service Engine ist im BIOS von einigen Desktop-PCs und Laptops des Herstellers verankert, funkt nach Hause und lädt ohne zu fragen Software nach.

Im BIOS von vielen Desktop-PCs und Laptops von Lenovo versteckt sich hinter der Option Lenovo Service Engine (LSE) im Grunde ein Rootkit, über das der Hersteller Computer aushorchen und Anwendungen direkt aus dem BIOS auf Geräte pushen kann. Das bemerkte ein Besitzer eines betroffenen Laptops, als [er Windows 8 ohne Internetverbindung neu installierte und sich dennoch ein Lenovo-Dienst im System eingenistet hat](#).

Über die LSE kann der Hersteller zudem prüfen, ob bestimmte Lenovo-Dienste noch laufen und diese gegebenenfalls wiederherstellen. Das Ganze soll anonymisiert ablaufen, erklärte Lenovo. Aufgrund der Kritik von Nutzern hat der Hersteller eigenen Angaben zufolge bereits Ende Juli reagiert; neue Computer sollen seitdem nicht mehr mit über die LSE-Funktion verfügen.

Angreifer könnten Schadcode in Systeme schmuggeln

Neben der Schnüffel-Kritik haben wohl auch Sicherheitsbedenken Lenovo zur Abschaffung von LSE bewegt, denn Angreifer könnten unter gewissen Umständen diesen Kanal missbrauchen, um Schadcode auf Systeme zu schmuggeln.

Bei der Software-Installation setzt Lenovo auf Microsofts Windows Platform Binary Table (WPBT). Dabei werden im Zuge einer Windows-Installation Binärdaten aus dem BIOS nachgeladen, die Dritte dort abgelegt haben.

<http://www.heise.de/newsticker/meldung/Lenovos-Service-Engine-BIOS-Rootkit-direkt-vom-Hersteller-2778547.html>

1. Grundlegendes zur Angriffsanalyse

- ❑ Notation von Sicherheitsproblemen
- ❑ Angreifermodelle
- ❑ Begriffe und Zusammenhänge

2. Ausgewählte technische Angriffsvarianten

- ❑ Denial of Service (DoS und DDoS)
- ❑ Schadsoftware (Malicious Code - Viren, Würmer, Trojanische Pferde)
- ❑ E-Mail-Security (Hoaxes und Spam)
- ❑ Mobile Code (ActiveX, JavaScript, ...)
- ❑ Systemnahe Angriffe (Buffer Overflows, Backdoors, Rootkits, ...)
- ❑ Web-basierte Angriffe (XSS, ...)
- ❑ Netzbasierte Angriffe (Sniffing, Portscans, ...)

3. Bewertung von Schwachstellen

- ❑ Common Vulnerability Scoring System (CVSS)
- ❑ Untersuchungen zu “Top Security Risks” nach SANS

- Einbetten von Schadcode in (vertrauenswürdigen) anderen Code
- Beispiel:
 - Alice betreibt eine Webseite mit Gästebuch-Funktion.
 - Mallet hinterlässt einen Gästebuch-Eintrag, der JavaScript enthält.
 - Bob ruft die Gästebuch-Einträge auf der Website von Alice ab und führt dabei den JavaScript-Code von Mallet aus.
- Typisches Ziel bei XSS:
 - Sensible Daten, z.B. Cookies, an den Angreifer übertragen.
 - Mallet kann sich damit als Bob ausgeben
(Identitätsdiebstahl, impersonation attack)
- Häufig im Zusammenhang mit HTML und JavaScript, betrifft aber nicht nur Webbrowser
(z.B. Skype 2011: JavaScript in Profildfeldern)

- Anwendung prüft Benutzereingaben nicht ausreichend
- Im Gästebuch-Beispiel:
 - Gästebuch-Webanwendung sollte Einträge mit (Schad-)Code nicht akzeptieren
 - Client (Bob) kann von Server (Alice) gewünschten Code nicht von böartigem Code (Mallet) unterscheiden.
- Im Skype-Beispiel:
 - Skype-Client von Mallet erlaubt JS-Code im Feld „Mobiltelefonnummer“.
 - Skype-Client von Bob führt diesen Code ungeprüft aus.
- JavaScript kann u.a. HTML-Formulare automatisch ausfüllen und abschicken; wurde missbraucht z.B. für
 - Sofort-Kauf von Ebay-Angeboten
 - Beleidigende oder Spam-artige Einträge in Internet-Foren
 - Generieren von URLs, die Benutzer auf fremde Webseiten umleiten und dabei sensible Daten als Parameter übergeben.

- Lokal bedeutet hier: Ohne Beteiligung eines Webserver
- Auslöser: JavaScript-Funktion prüft übergebene Parameter nicht
- Beispiel:

```
<HTML>
<TITLE>HTML-Beispieldokument DOM-XSS</TITLE>
Hallo
<SCRIPT>
  var pos=document.URL.indexOf("username")+9;
  document.write(document.URL.substring(pos,document.URL.length));
</SCRIPT>
<BR/>
Dies ist ein Beispiel-HTML-Dokument.
</HTML>
```

- Aufruf mit:

`http://www.example.com/index.html?username=<script>alert("XSS-Problem!")</script>`

- Als Parameter übergebener Code wird von anfälligen Browsern ausgeführt.

■ Ablauf:

- Webserver liefert Webseite mit Inhalt aus, der vom Benutzer übergebene (und somit nicht-persistente) Parameter (inkl. JavaScript-Code) enthält.
- Mallet bringt Alice dazu, einen Link mit entsprechenden Parametern anzuklicken

■ Beispiel:

- Alice klickt folgenden Link an:

```
http://suchmaschine.example.com/?suchbegriff=<script type="text/
javascript">alert("Alice, Du hast ein XSS-Problem!")</script>
```

- Webserver liefert folgendes Dokument aus:

```
<HTML>
  <TITLE>Suchmaschine: Ergebnisse</TITLE>
  Ihr Suchbegriff war: <script type="text/javascript">alert(„Alice,
    Du hast ein XSS-Problem!“)</script>
  <BR/>
  Hier sind Ihre Ergebnisse: ...
</HTML>
```

- Schadcode wird vom Webserver gespeichert und bei jeder Anfrage ausgeliefert.
- Vgl. Gästebuch-Beispiel; Eintrag enthält z.B.

Tolle Webseite!

```
<script type="text/javascript">alert("Aber mit XSS-Problem!")</script>
```

- Dadurch sehr breit gestreuter Angriff
- Besonders problematisch, wenn der „verseuchte“ Webserver als besonders vertrauenswürdig konfiguriert ist
- Gegenmaßnahme:
 - Webapplikation muss Script-Code aus Benutzereingaben entfernen oder „ungefährlich“ machen.
 - Script-Code kann anhand der Meta-Zeichen, z.B. <, erkannt werden.
 - Client-seitig: JavaScript deaktivieren oder Plugins wie NoScript verwenden

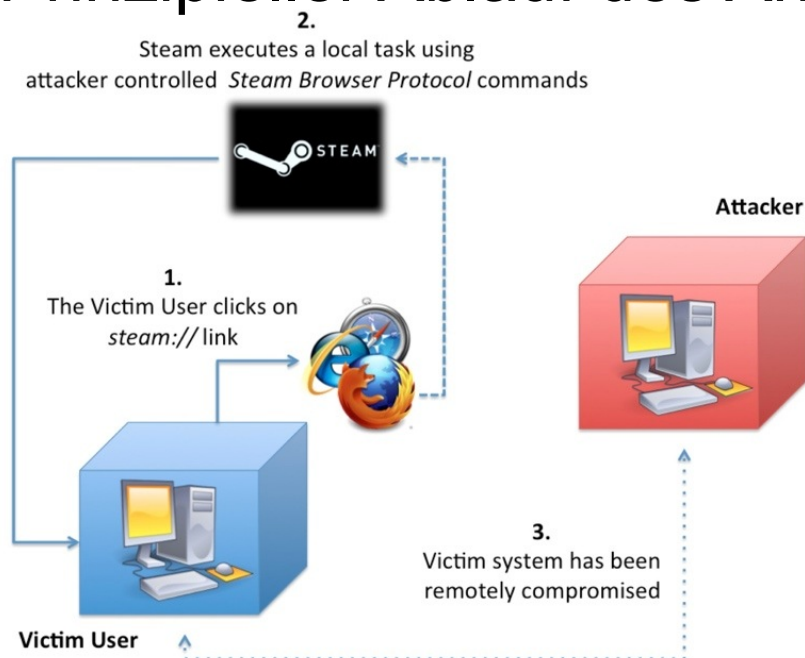
XSS Beispiel: Angriff auf Issue Tracking System von Apache

- apache.org nutzt Atlassian JIRA als Issue Tracking System
- 5. April 2010: Angreifer legt neue Issue (INFRA-2591) an:
ive got this error while browsing some projects in jira <http://tinyurl.com/XXXXXXXXXX> [obscured]
 - tinyurl: Dienst, um URLs zu kürzen und „dauerhaft“ zu machen
 - Lange URL enthält XSS, um Cookies von JIRA-Usern zu stehlen
 - Auch Administratoren von JIRA werden Opfer
 - Gleichzeitig startet Angreifer Brute-force Passwort-Angriff gegen Anmeldeseite login.jsp
- 6. April 2010: Angreifer hat Administrator-Rechte auf JIRA
 - Angreifer deaktiviert Benachrichtigung für ein Projekt
 - Ändert Pfad für den Upload von Attachments; Pfad erlaubt Ausführung von JSP und ist schreibbar für JIRA-User
 - Erzeugen neuer Issues mit einer Vielzahl von Attachments
 - Eines der Attachments ist JSP zum Durchsuchen und Kopieren von Dateisystem-Inhalten

- 6. April 2010: (Fortsetzung)
 - Angreifer kopiert sich damit viele Home-Directories von JIRA-Usern
 - Weiteres JSP mit Shell-Backdoor für das System wird installiert
- 9. April 2010: Angreifer hat jar-Datei installiert, die alle Benutzernamen und Passwörter protokolliert
 - Eines der Passwörter funktioniert auch für den Rechner `brutus.apache.org`
 - Für diesen Account ist `sudo` aktiviert.
 - Damit voller Root-Zugriff auf `brutus` (Server für JIRA u. Wiki).
 - Angreifer nutzt dies zum Ausspähen von Logindaten für den Subversion-Server `minotaur`
 - Angreifer beginnt, JIRA-Passwort-Reset-Mails zu verschicken, damit sich noch mehr Benutzer auf dem kompromittierten System anmelden.
 - Dadurch wird der Angriff schließlich erkannt.
 - Infos: https://blogs.apache.org/infra/entry/apache_org_04_09_2010

Beispiel: Steam Browser Protocol Insecurity (10/2012)

- Quelle: Luigi Auriemma, Donato Ferrante, <http://revuln.com>
- Steam:
 - Software-Plattform der Valve Corporation für Download und Digital Rights Management von kommerziellen und kostenlosen Spielen.
 - Installiert eigene URL-Handler für steam://*-URLs u.a. zum Installieren und Starten von Software.
- Prinzipieller Ablauf des Angriffs:



Internet Explorer	Warning including the URL, and in case of IE9 a possible additional warning ("protected mode") without any detail
Firefox	No URL visualized, only request for confirmation (no warnings)
Chrome	Warning with a detailed description or the URL and the program to call
Opera	Warning with only 40 chars of the URL visualized
Safari	Direct execution without warnings
Webkit	Direct execution without warnings
MaxThon	Direct execution without warnings
Avant	Direct execution without warnings
Lunaspice	Direct execution without warnings
SeaMonkey	See Firefox
PaleMoon	See Firefox
SRWare Iron	See Chrome

- Anklicken eines steam:// - URL reicht, um ein System in folgenden Fällen zu kompromittieren:
 - *retailinstall*-Kommando von Steam unterstützt remote hosts und zeigt TGA-Bilddatei an. TGA-Parser ist anfällig für Buffer Overflow und erlaubt somit das Ausführen beliebiger Kommandos.
 - *run*-Kommando erlaubt das Ausführen installierter Spiele und übergibt Kommandozeilenparameter:
 - Z.B. über die Logfile-Unterstützung der Valve Source Engine können .bat-Dateien im Autostart-Ordner des Benutzers angelegt werden.
 - Einigen Spielen mit Auto-Update-Feature kann über Kommandozeile ein Server vorgegeben werden, von dem ein vermeintliches Update geladen werden soll (keine Integritätsprüfung!).

- Typisches Problem: Viele Steam-Benutzer arbeiten mit Administrator-Accounts

Erwarteter Aufruf

`http://webserver/cgi-bin/find.cgi?ID=42`

Erzeugte SQL-Abfrage

`SELECT autor, text FROM artikel WHERE ID=42`

Aufruf mit SQL-Injektion

`http://webserver/cgi-bin/find.cgi?
ID=42;UPDATE+USER+SET+TYPE="admin"+WHERE+ID=2`

Erzeugte SQL-Abfrage: 2 Befehle!

`SELECT autor, text FROM artikel WHERE ID=42; UPDATE
USER SET TYPE="admin" WHERE ID=2`

1. Grundlegendes zur Angriffsanalyse

- ❑ Notation von Sicherheitsproblemen
- ❑ Angreifermodelle
- ❑ Begriffe und Zusammenhänge

2. Ausgewählte technische Angriffsvarianten

- ❑ Denial of Service (DoS und DDoS)
- ❑ Schadsoftware (Malicious Code - Viren, Würmer, Trojanische Pferde)
- ❑ E-Mail-Security (Hoaxes und Spam)
- ❑ Mobile Code (ActiveX, JavaScript, ...)
- ❑ Systemnahe Angriffe (Buffer Overflows, Backdoors, Rootkits, ...)
- ❑ Web-basierte Angriffe (XSS, ...)
- ❑ Netzbasierte Angriffe (Sniffing, Portscans, ...)

3. Bewertung von Schwachstellen

- ❑ Common Vulnerability Scoring System (CVSS)
- ❑ Untersuchungen zu “Top Security Risks” nach SANS

■ Local Area Network (LAN):

- ❑ Oft gemeinsam genutztes Medium (shared medium), z.B. WLAN, Ethernet ohne Switches
- ❑ Netzwerk-Karten können im Prinzip gesamten Verkehr mithören, aber
- ❑ geben nur die an den Rechner adressierten Pakete weiter
- ❑ Gefahr: “Promiscuous Mode”:
 - Einstellung der Karte
 - Im Promiscuous Mode werden **alle** Pakete gelesen und ans OS durchgereicht

■ Wide Area Network (WAN):

- ❑ Jeder Vermittlungsrechner kann Nachrichten „mitlesen“, z.B. Mirror-Ports an Routern
- ❑ “Anzapfen” von Leitungen durch Geheimdienste

■ Tools:

- ❑ Übergang zwischen Werkzeug des System- sowie Netzadministrators und Cracker-Tool sind fließend
- ❑ tcpdump, ngrep
Standard-Werkzeuge in vielen UNIX-/Linux-Distributionen
- ❑ wireshark
Packet-Analyzer (Linux, Windows)
- ❑ snoop
Sun Solaris

```

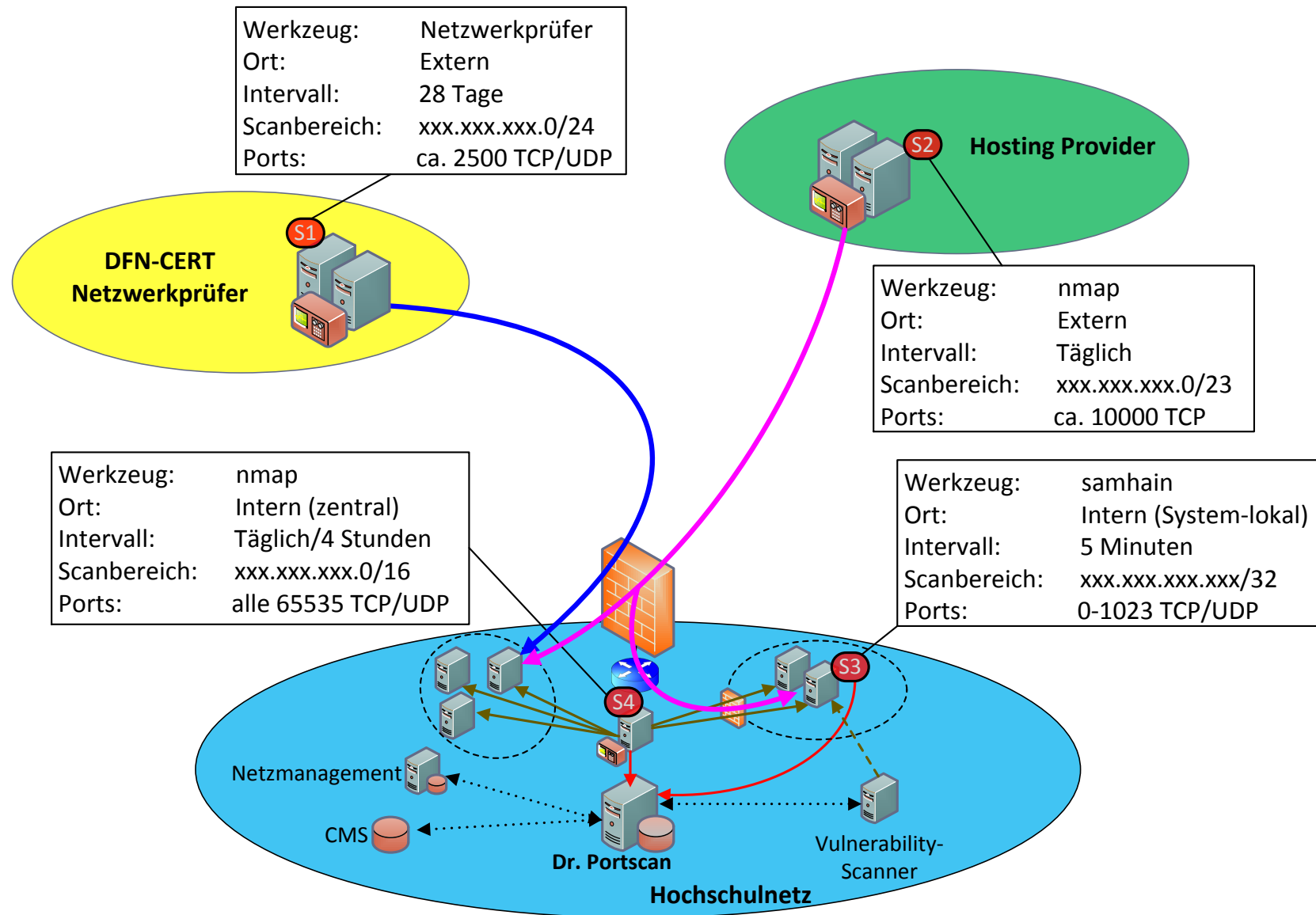
.....
match: nm.ifi.lmu.de
#####
U 129.187.15.116:60267 -> 10.156.33.53:53
.....www.nm.ifi.lmu.de.....
#
U 129.187.15.116:51458 -> 10.156.33.53:53
.....www.nm.ifi.lmu.de.....
#
U 10.156.33.53:53 -> 129.187.15.116:51458
.....www.nm.ifi.lmu.de.....Q....pchege01.....Q....(dns0...hostmaster...U...*0.....Q.
#
U 10.156.33.53:53 -> 129.187.15.116:60267
.....www.nm.ifi.lmu.de.....Q....pchege01...../.....Q...T.....Q.....dns2.lrz.....dns1.....Q...dns1.
.....Q...dns3\.....Q...gcher00.....Q...dns0.....Q...T.....Q.....Q.....Q.....Q.....S.
.....W.....Q.....(C..W.....Q.....{.....Q.....#.....Q.....Q.....I.0.....).....S
###
T 129.187.15.116:50224 -> 141.84.218.31:80 [AP]
GET /itsec HTTP/1.1..Host: www.nm.ifi.lmu.de..User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_2) AppleWebKit/534.51.22 (KHTML,
like Gecko) Version/5.1.1 Safari/534.51.22..Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8..Accept-Language: d
e-de..Accept-Encoding: gzip, deflate..Connection: keep-alive....
#
T 141.84.218.31:80 -> 129.187.15.116:50224 [AP]
HTTP/1.1 302 Found..Date: Thu, 27 Oct 2011 07:38:14 GMT..Server: Apache/1.3.29 (Unix) PHP/4.3.4..X-Powered-By: PHP/4.3.4..Location: htt
p://www.nm.ifi.lmu.de/teaching/Vorlesungen/2011ws/itsec..Keep-Alive: timeout=15, max=100..Connection: Keep-Alive..Transfer-Encoding: ch
unked..Content-Type: text/html....1....0....
#
T 129.187.15.116:50224 -> 141.84.218.31:80 [AP]
GET /teaching/Vorlesungen/2011ws/itsec HTTP/1.1..Host: www.nm.ifi.lmu.de..User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_2) Ap
pleWebKit/534.51.22 (KHTML, like Gecko) Version/5.1.1 Safari/534.51.22..Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*
/*;q=0.8..Accept-Encoding: gzip, deflate..Accept-Language: de-de..Connection: keep-alive....
#
T 141.84.218.31:80 -> 129.187.15.116:50224 [AP]
HTTP/1.1 301 Moved Permanently..Date: Thu, 27 Oct 2011 07:38:14 GMT..Server: Apache/1.3.29 (Unix) PHP/4.3.4..Location: http://www.nm.ifi
lmu.de/teaching/Vorlesungen/2011ws/itsec/..Keep-Alive: timeout=15, max=99..Connection: Keep-Alive..Transfer-Encoding: chunked..Conten
t-Type: text/html; charset=iso-8859-1...151...<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">.<HTML-><HEAD>.<TITLE>301 Moved Permanen
tly</TITLE>.</HEAD><BODY>.<H1>Moved Permanently</H1>..The document has moved <A HREF="http://www.nm.ifi.lmu.de/teaching/Vorlesungen/2011
ws/itsec/">here</A>.<P>.</P>.</HR>.<ADDRESS>Apache/1.3.29 Server at www.nm.ifi.lmu.de Port 80</ADDRESS>.</BODY></HTML>...0....

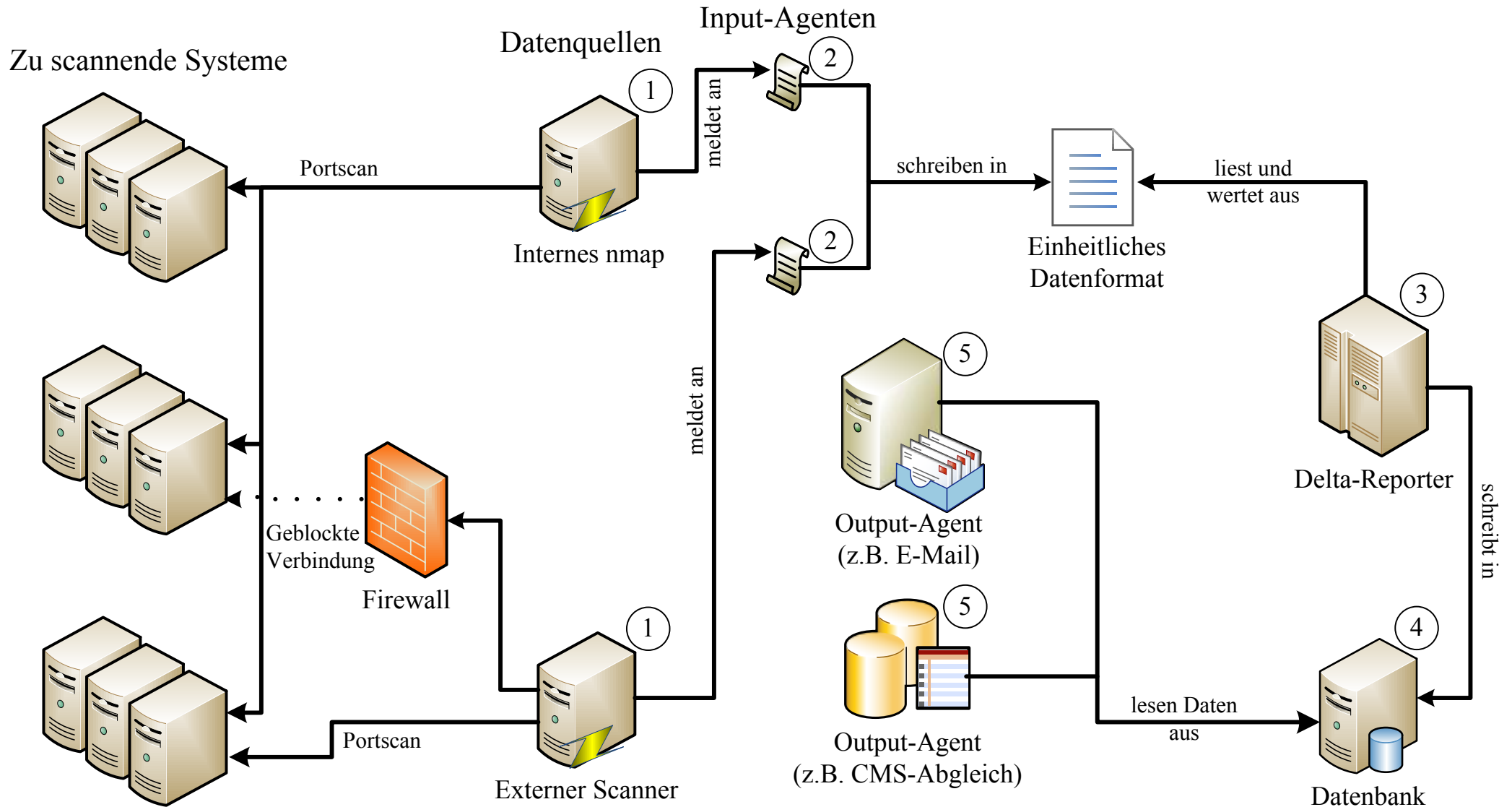
```

- Suchen auf entferntem Rechner nach „offenen“ Ports
 - Versuch eines Verbindungsaufbau / pro Port
 - Falls erfolgreich: Port ist „offen“
- Damit Identifikation von Diensten
- Gezielte Suche nach Rechnern, die Dienste mit bekannten Schwächen anbieten
- Auch hier ist der Übergang zwischen nützlichem Werkzeug und Cracker Tool fließend
- Port-Scans werden oft als Angriff gewertet und deshalb getarnt durchgeführt
- Beispiel:
 - nmap

```
Nmap scan report for www.nm.ifi.lmu.de (141.84.218.31)
Host is up (0.018s latency).
rDNS record for 141.84.218.31: pcheger01.nm.ifi.lmu.de
Not shown: 65532 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
443/tcp    open  https
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.15 - 2.6.27, Linux 2.6.24 - 2.6.26 (Debian), Linux 2.6.27 (Ubuntu 8.10)
Uptime guess: 74.352 days (since Sun Aug 14 01:24:27 2011)
```

Proaktive Netzüberwachung mit Portscans





1. Grundlegendes zur Angriffsanalyse

- ❑ Notation von Sicherheitsproblemen
- ❑ Angreifermodelle
- ❑ Begriffe und Zusammenhänge

2. Ausgewählte technische Angriffsvarianten

- ❑ Denial of Service (DoS und DDoS)
- ❑ Schadsoftware (Malicious Code - Viren, Würmer, Trojanische Pferde)
- ❑ E-Mail-Security (Hoaxes und Spam)
- ❑ Mobile Code (ActiveX, JavaScript, ...)
- ❑ Systemnahe Angriffe (Buffer Overflows, Backdoors, Rootkits, ...)
- ❑ Web-basierte Angriffe (XSS, ...)
- ❑ Netzbasierte Angriffe (Sniffing, Portscans, ...)

3. Bewertung von Schwachstellen

- ❑ Common Vulnerability Scoring System (CVSS)
- ❑ Untersuchungen zu “Top Security Risks” nach SANS

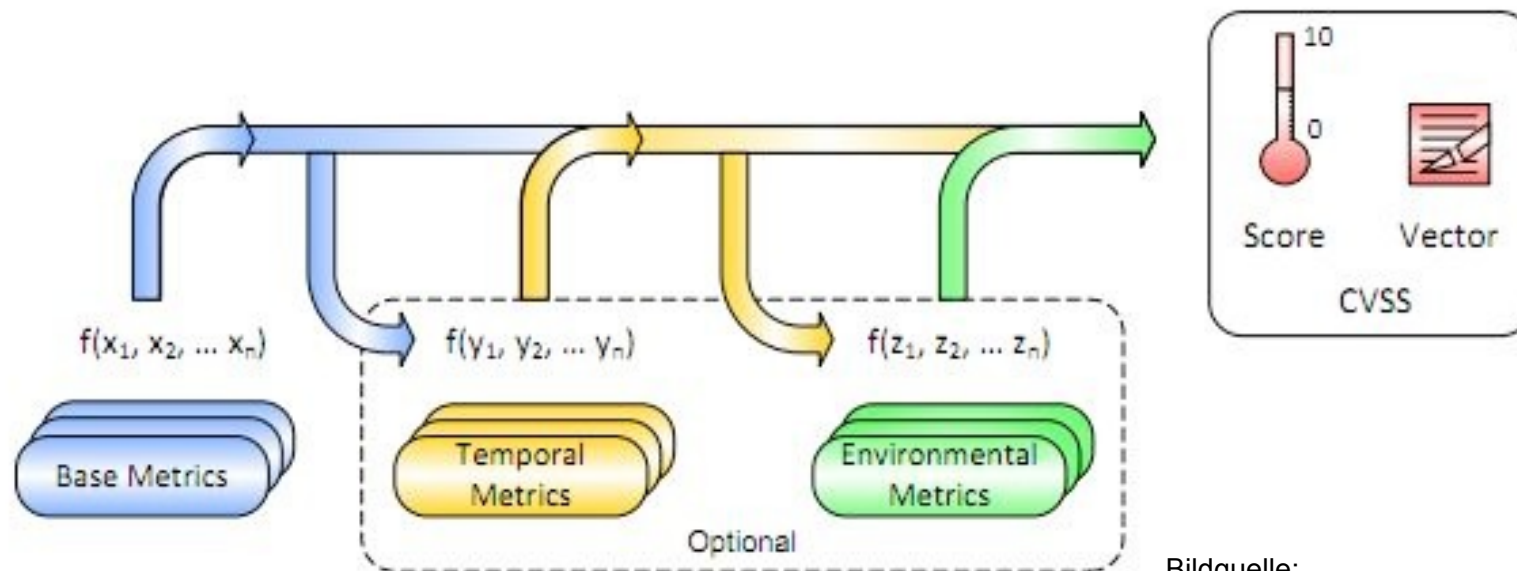
- **Hauptziel: Priorisierung**
 - Wie wichtig ist es, eine Schwachstelle (schnell) zu beseitigen?
 - Welche von mehreren gleichzeitig bekannten Schwachstellen ist die dringendste?

- **Betrifft sowohl die Entwickler als auch die Betreiber von Software/Systemen.**

- **Idee: Quantitative Bewertung von Schwachstellen** anhand einer definierten Menge verschiedener Charakteristika
=> Jeder Schwachstelle wird ein Zahlenwert zugeordnet.
Problem: Objektivität / Einheitlichkeit.

- **CVSS-Ansatz: Dreiteilung in unveränderliche bzw. zeitlich und räumlich variable Charakteristika.**

- CVSS ist an der CMU entstanden und wird inzwischen von FIRST (Forum of Incident Response and Security Teams) gepflegt. v2 ist aktuell de-facto Standard, v3 im Juni 2015 erschienen.
- Drei Gruppen von Bewertungskennzahlen:
 - **Base Metrics:** Grundlegende Eigenschaften der Verwundbarkeit
 - **Temporal Metrics:** Zeitabhängige Eigenschaften der Verwundbarkeit
 - **Environmental Metrics:** Anwenderspezifische Eigenschaften der Verwundbarkeit



Bildquelle:

<http://www.first.org/cvss/cvss-guide.html>

- Base Metrics werden oft von Herstellern / Sicherheitsunternehmen veröffentlicht

■ Input:

- Einfache Bewertung von Schwachstellen durch vorgegebene Fragen und Antwortmöglichkeiten.

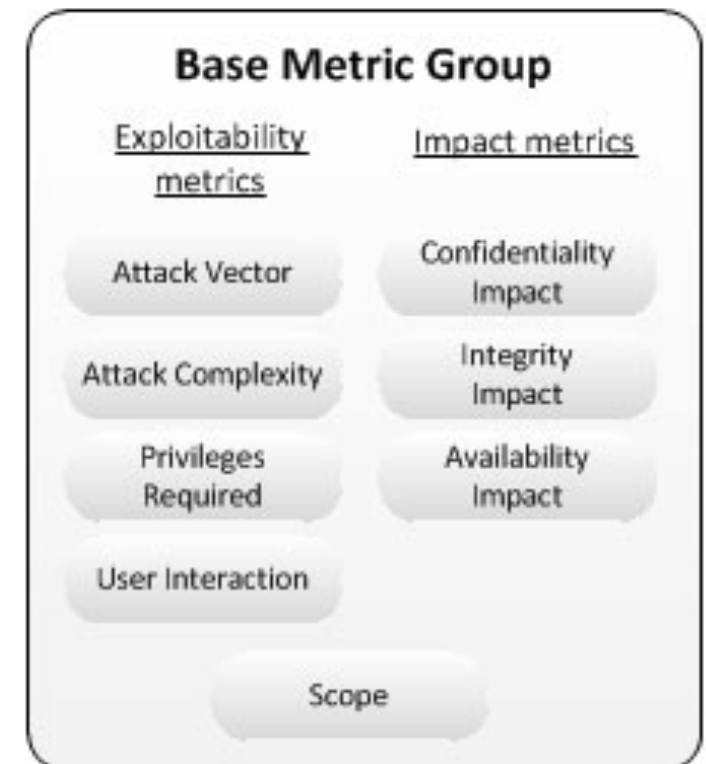
■ Outputs:

- **CVSS-Score** (= Zahl) zwischen 0,0 (harmlos) und 10,0 (Katastrophe)
- **CVSS-Vektor** = kompakter String, Kurzfassung des gesamten Inputs

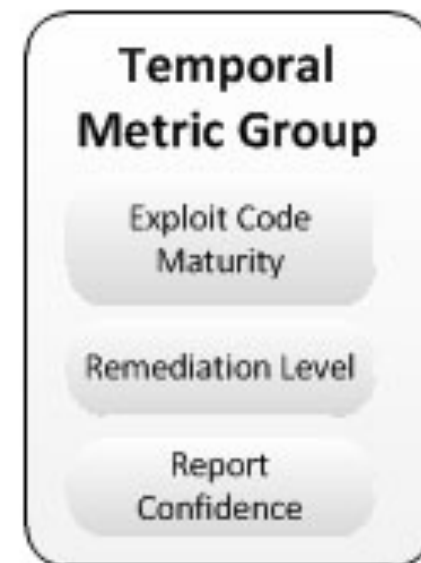
■ Scoring-Formel:

- Die pro Frage (Metric) gewählte Antwort beeinflusst den Score.
- Der konkreter Einfluss-Wert ist jeweils empirisch definiert; hierin steckt die "Intelligenz" bzw. Praxiserfahrung von CVSS.
- Als CVSS-Anwender muss man "nur" die zur Schwachstelle pro Metric passende Antwortmöglichkeit auswählen.
- Den numerische CVSS-Score gibt der "CVSS Calculator" aus.

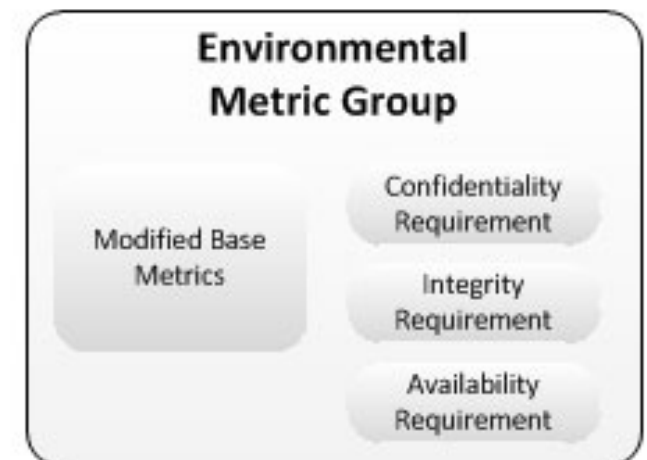
- Die **Base Metrics** bewerten die intrinsischen, konstanten Eigenschaften von Schwachstellen; drei Untergruppen: **Exploitability metrics**, **Impact metrics** und **Scope**.
- **Exploitability metrics** (technische Aspekte der Schwachstellenausnutzung):
 - ❑ **Attack vector**: Physisch, auf dem Rechner, vom LAN aus, via Internet?
 - ❑ **Attack complexity**: Trivial, anspruchsvoll?
 - ❑ **Privileges required**: Jeder, reg. User, Admin?
 - ❑ **User interaction**: Angreifer braucht User?
- **Impact metrics** (Auswirkung der Schwachstelle):
 - ❑ **Auswirkungen auf C, I, A**
 - ❑ Jeweils **gar nicht**, **gering** oder **stark**?
- **Scope**: Wirkt sich die Schwachstelle auf andere Systeme aus?



- Die **Temporal Metrics** bewerten den jeweils (zeitlich) aktuellen und damit variablen Stand der Schwachstelle.
- **Exploit code maturity?**
 - ❑ **Not defined**: Soll nicht in Auswertung einfließen; oder:
 - ❑ Abstufung: Reine Theorie; Proof of Concept; gut funktionierendes Exploit; Wurm oder anderweitiges Massenproblem im Umlauf.
- **Remediation level?**
 - ❑ Not defined; oder:
 - ❑ Abstufung: Keine Lösung; Workaround; offizieller temporärer Fix; offizieller dauerhaft Fix verfügbar
- **Report confidence?**
 - ❑ Not defined; oder:
 - ❑ Abstufung: Gerücht; dokumentiert; offiziell bestätigt



- Die **Environmental Metrics** bewerten die Schwachstelle im Hinblick auf das Einsatzgebiet des betroffenen Systems; sie unterscheiden sich also z.B. je nach Organisation.
- **C, I, A Requirements?**
 - Undefined; oder:
 - Abstufung: Gering; mittel; hoch
- **Modified Base Metrics?**
 - Falls Sicherheitsmaßnahmen im Einsatz sind, die sich auf die Base-Metric-Eigenschaften der Schwachstelle auswirken, können diese individuell modifiziert werden.
 - Default: not defined.



■ CVSS Score:

- ❑ Man muss **Base Score**, **Temporal Score** und **Environmental Score** unterscheiden.
- ❑ “CVSS Score” ist oft Synonym für **CVSS Base Score**

■ Bei CVSS v2 galt:

- ❑ **Base Score** \geq **Temporal Score** \geq **Environmental Score**
- ❑ Beispiele:
 - Praxisrelevanter Score ist niedriger, wenn offizieller Fix verfügbar ist.
 - Praxisrelevanter Score ist niedriger, wenn CIA Requirements niedrig sind.

■ Bei CVSS v3 gilt o.g. Ungleichung nicht mehr zwingend wg. optionaler **Modified Base Metrics**...

- ❑ Die Praxis wird zeigen, ob davon Gebrauch gemacht wird.
- ❑ CVSS v3 ist hoffentlich besser, jedenfalls auch komplexer geworden als v2.

- Adobe Acrobat Buffer Overflow (CVE-2009-0658)
 - Fehler beim Parsen von JBIG2-Bildern in PDFs.
 - Beim Öffnen präparierter PDFs in Adobe Acrobat wird beliebiger Schadcode ausgeführt.
- CVSS v3 Base Score: 7.8

Metric	Value	Comments
Access Vector	Local	A flaw in the local document software that is triggered by opening a malformed document.
Attack Complexity	Low	
Privileges Required	None	
User Interaction	Required	The victim needs to open the malformed document.
Scope	Unchanged	
Confidentiality Impact	High	Assuming a worst-case impact of the victim having High privileges on the affected system.
Integrity Impact	High	Assuming a worst-case impact of the victim having High privileges on the affected system.
Availability Impact	High	Assuming a worst-case impact of the victim having High privileges on the affected system.

Beispiel aus / ergänzend:
<https://www.first.org/cvss/examples>

1. Grundlegendes zur Angriffsanalyse

- ❑ Notation von Sicherheitsproblemen
- ❑ Angreifermodelle
- ❑ Begriffe und Zusammenhänge

2. Ausgewählte technische Angriffsvarianten

- ❑ Denial of Service (DoS und DDoS)
- ❑ Schadsoftware (Malicious Code - Viren, Würmer, Trojanische Pferde)
- ❑ E-Mail-Security (Hoaxes und Spam)
- ❑ Mobile Code (ActiveX, JavaScript, ...)
- ❑ Systemnahe Angriffe (Buffer Overflows, Backdoors, Rootkits, ...)
- ❑ Web-basierte Angriffe (XSS, ...)
- ❑ Netzbasierte Angriffe (Sniffing, Portscans, ...)

3. Bewertung von Schwachstellen

- ❑ Common Vulnerability Scoring System (CVSS)
- ❑ Untersuchungen zu “Top Security Risks” nach SANS

- SANS Institute (gegründet 1989)
(System Administration, Audit, Networking and Security)
- Herausgeber von
 - SANS/FBI Annual Top 20 Internet Security Vulnerability List (bis 2007)
 - Top Cyber Security Risks (seit 2008); letzte Aktualisierung: 09/2009
- Quellen:
 - Angriffsdaten von IDS-Systemen aus über 6.000 Organisationen
 - Schwachstellenberichte von rund 9 Mio. Systemen
 - Information aus Internet Storm Center des SANS
- Identifizierte Hauptprobleme:
 - Zeitnahe Software-Aktualisierung auf Client-Systemen
 - Unsichere Webanwendungen

■ Client-seitig:

- #1: Schwachstellen in Web-Browsern
- #2: Schwachstellen in Office-Programmen
- #3: E-Mail (Phishing, Spam, DoS, Malware)
- #4: Media Players (u.a. Flash Player, Quicktime, ...)

■ Server-seitig:

- #1: Webanwendungen (u.a. XSS, SQL Injection, ...)
- #4: Angriffe über Backup-Software (lokale Berechtigungen; zentrale Ablage)
- #5: Programmierfehler in Anti-Virus-Software (code execution vulnerabilities)
- #6: Unzureichend gesicherte Management-Server (u.a. Monitoring, Softwareverteilung, ...)

■ Faktor Mensch:

- #1: Nutzung privater Endgeräte am Arbeitsplatz
- #2: Phishing
- #3: Unverschlüsselte mobile Geräte und Speichermedien

Rank	Score	ID	Name
[1]	93.8	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
[2]	83.3	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
[3]	79.0	CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
[4]	77.7	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
[5]	76.9	CWE-306	Missing Authentication for Critical Function
[6]	76.8	CWE-862	Missing Authorization
[7]	75.0	CWE-798	Use of Hard-coded Credentials
[8]	75.0	CWE-311	Missing Encryption of Sensitive Data
[9]	74.0	CWE-434	Unrestricted Upload of File with Dangerous Type
[10]	73.8	CWE-807	Reliance on Untrusted Inputs in a Security Decision
[11]	73.1	CWE-250	Execution with Unnecessary Privileges
[12]	70.1	CWE-352	Cross-Site Request Forgery (CSRF)
[13]	69.3	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
[14]	68.5	CWE-494	Download of Code Without Integrity Check
[15]	67.8	CWE-863	Incorrect Authorization
[16]	66.0	CWE-829	Inclusion of Functionality from Untrusted Control Sphere
[17]	65.5	CWE-732	Incorrect Permission Assignment for Critical Resource
[18]	64.6	CWE-676	Use of Potentially Dangerous Function
[19]	64.1	CWE-327	Use of a Broken or Risky Cryptographic Algorithm
[20]	62.4	CWE-131	Incorrect Calculation of Buffer Size
[21]	61.5	CWE-307	Improper Restriction of Excessive Authentication Attempts
[22]	61.1	CWE-601	URL Redirection to Untrusted Site ('Open Redirect')
[23]	61.0	CWE-134	Uncontrolled Format String
[24]	60.3	CWE-190	Integer Overflow or Wraparound
[25]	59.9	CWE-759	Use of a One-Way Hash without a Salt

- “Before we knew it - An empirical study of zero-day attacks in the real world”, Bilge/Dumitras, Oktober 2012

http://users.ece.cmu.edu/~tdumitra/public_documents/bilge12_zero_day.pdf

- Wie lange werden Sicherheitslücken ausgenutzt, bevor sie allgemein bekannt (und beseitigt) werden?

- Untersuchung für 11 Millionen Windows-PCs mit Symantec-Software
- Dauer schwankt zwischen 19 Tagen und 30 Monaten
- Durchschnitt liegt bei 312 Tagen (!)

- Wie wirkt sich die Veröffentlichung einer Sicherheitslücke aus?

- Anzahl an Malware-Varianten steigt um das bis zu 85.000-fache
- Anzahl beobachteter Angriffe steigt um das bis zu 100.000-fache

- Mehrwert und Seiteneffekte von “Full Disclosure”?

- Angreifermodelle beschreiben Fähigkeiten, Motivation usw.

- Angriffe zielen darauf ab, den individuellen Schutzbedarf (Vertraulichkeit, Integrität, Verfügbarkeit) zu verletzen:
 - Malware-/Rootkit-infizierte Systeme bieten keine Vertraulichkeit mehr
 - Buffer Overflow Exploits zerstören die Integrität von Software
 - DoS-Angriffe stören die Verfügbarkeit
 - ...

- Zu jedem Angriff gibt es mehr oder weniger effektive / kostspielige Gegenmaßnahmen:
 - Ziel ist aber kein Flickenwerk aus einzelnen Maßnahmen, sondern ein von Grund auf sicheres Design (=> Security Engineering).
 - Kenntnis von Angriffsvarianten und -wegen ist Voraussetzung für die Konzeption adäquater Sicherheitsmaßnahmen.