

# IT-Sicherheit

- Sicherheit vernetzter Systeme -

## Kapitel 3: Security Engineering

Aktualisierte Folien vom 28.10.2011

# Inhalt (1)

1. Security Engineering – Ziel und Vorgehensmodell
2. Notation von Sicherheitsproblemen: Handelnde Personen
3. Angreifermodelle
4. Bedrohungen, Angriffe und Gefährdungen
  1. Denial of Service
  2. Distributed Denial of Service
  3. Malicious Code
    - Viren
    - Würmer
    - Trojanische Pferde
  4. Hoax und Spam
  5. Mobile Code
    - ActiveX
    - JavaScript

# Inhalt (2)

4. Bedrohungen, Angriffe und Gefährdungen (Forts.)
  6. Buffer Overflow
  7. Account-/Password-Cracking
  8. Hintertüren / Falltüren
  9. Rootkits
  10. Cross Site Scripting (XSS)
  11. Sniffer
  12. Port Scanner
  13. Social Engineering
5. Rechtliche Regelungen: Strafgesetzbuch
6. „Top Cyber Security Risks“
7. Zusammenfassung

# Security Engineering - Zielsetzung

- Analogie zum Software Engineering
- Zielsetzung: Sicheres IT-System bzw. sichere IT-Infrastruktur, in der die gesteckten Sicherheitsziele (CIA) auch unter „widrigen Bedingungen“ erfüllt werden.
- Hauptaufgabe: Strukturierte Herangehensweise an die szenarienspezifische Umsetzung von Sicherheitsmaßnahmen.
  - Koordinierte Anwendung etablierter Methoden und Konzepte
  - Auswahl von Maßnahmen immer abhängig vom konkreten Bedarf
  - Praxisbezug, Pragmatismus

# Security Engineering - Vorgehensmodell

## ■ Lebenszyklus-Orientierung:

- Sicherheit von Anfang an adäquat berücksichtigen
- Fundierte Methoden und erprobte Werkzeuge für
  - Design
  - Implementierung
  - Test
  - Betrieb und Wartung
- Laufende Anpassung von Systemen an sich ändernde Anforderungen

## ■ Interdisziplinäre Aspekte

- Wirtschaftlichkeit von Maßnahmen
- Berücksichtigung der Benutzerfreundlichkeit
- Gesetzliche Auflagen

# Verzahnung mit dem Systementwicklungszyklus

- Am Beispiel eines einfachen Wasserfallmodells
  - Gilt sinngemäß für andere Vorgehensmodelle
  
- Planung
  - Machbarkeitsanalyse und Kostenabschätzung für Sicherheitsmaßnahmen
  
- Anforderungsanalyse
  - Analyse bestehender Sicherheitsvorgaben und zu schützender Assets
  - Analyse relevanter Angriffe und vorhandener Sicherheitsmaßnahmen
  - Dokumentation von „misuse cases“ analog zu „use cases“
  - Analyse rechtlicher Auflagen
  - Sicherheitsspezifische Risikoanalyse
  
- Architekturkonzept
  - Entwicklung einer Sicherheitsarchitektur
  - Spezifikation von Reaktionen auf Sicherheitsvorfälle

# (Fortsetzung)

## ■ Implementierungskonzept

- ❑ Auswahl der technischen Komponenten für die Umsetzung der Sicherheitsarchitektur
- ❑ Anwendung von Security Design Patterns
- ❑ Spezifikation sicherheitsspezifischer Tests und Definition von Abnahmekriterien
- ❑ Review und Genehmigung des Sicherheitskonzepts

## ■ Implementierung und Test

- ❑ Integration der ausgewählten Sicherheitsmaßnahmen
- ❑ Durchführung von Code-Reviews
- ❑ Sicherheitsspezifische Tests und Abnahme
- ❑ Schulung / Sensibilisierung von Benutzern

## ■ Betrieb und Wartung

- ❑ Kontinuierliche Anpassung z.B. an neuartige Angriffe, veränderte Anforderungen und Randbedingungen, ...
- ❑ Regelmäßige proaktive Analysen, z.B. durch Penetration Testing

# Anforderungen an Off-the-shelf-Komponenten

- Komplexe IT-Infrastrukturen bestehen zu großen Teilen aus nicht selbst entwickelten, sondern zugekauften Komponenten.
  
- Anforderungen aus dem Security Development Lifecycle:
  - Secure by design:  
Sicherheit von Anfang an, nicht nachträglich eingebaut.
  - Secure by default:  
Auslieferung in einer als sicher eingeschätzten Konfiguration; z.B. nicht alle Features von vornherein aktiviert.
  - Secure in deployment:  
Komponente bietet während der Inbetriebnahme keine Angriffsfläche.



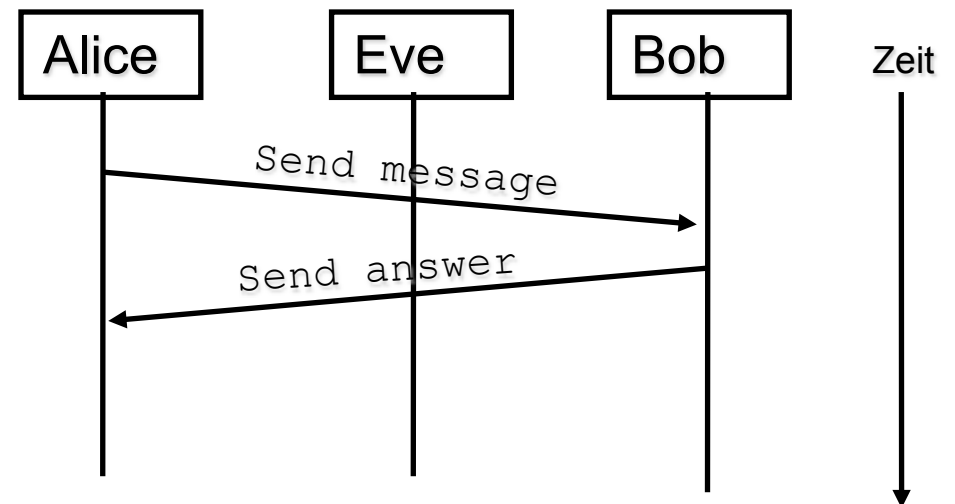
# Inhalt (1)

1. Security Engineering – Ziel und Vorgehensmodell
2. Notation von Sicherheitsproblemen: Handelnde Personen
3. Angreifermodelle
4. Bedrohungen, Angriffe und Gefährdungen
  1. Denial of Service
  2. Distributed Denial of Service
  3. Malicious Code
    - Viren
    - Würmer
    - Trojanische Pferde
  4. Hoax und Spam
  5. Mobile Code
    - ActiveX
    - JavaScript

# Notation: Handelnde Personen

- 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) and Dave (D)**  
sind ggf. weitere gutartige Teilnehmer
  - **Trent (T)**  
Vertrauenswürdiger Dritter (Trusted arbitrator)
  - **Walter (W)**  
Wächter (Warden), bewacht insb. Alice und Bob

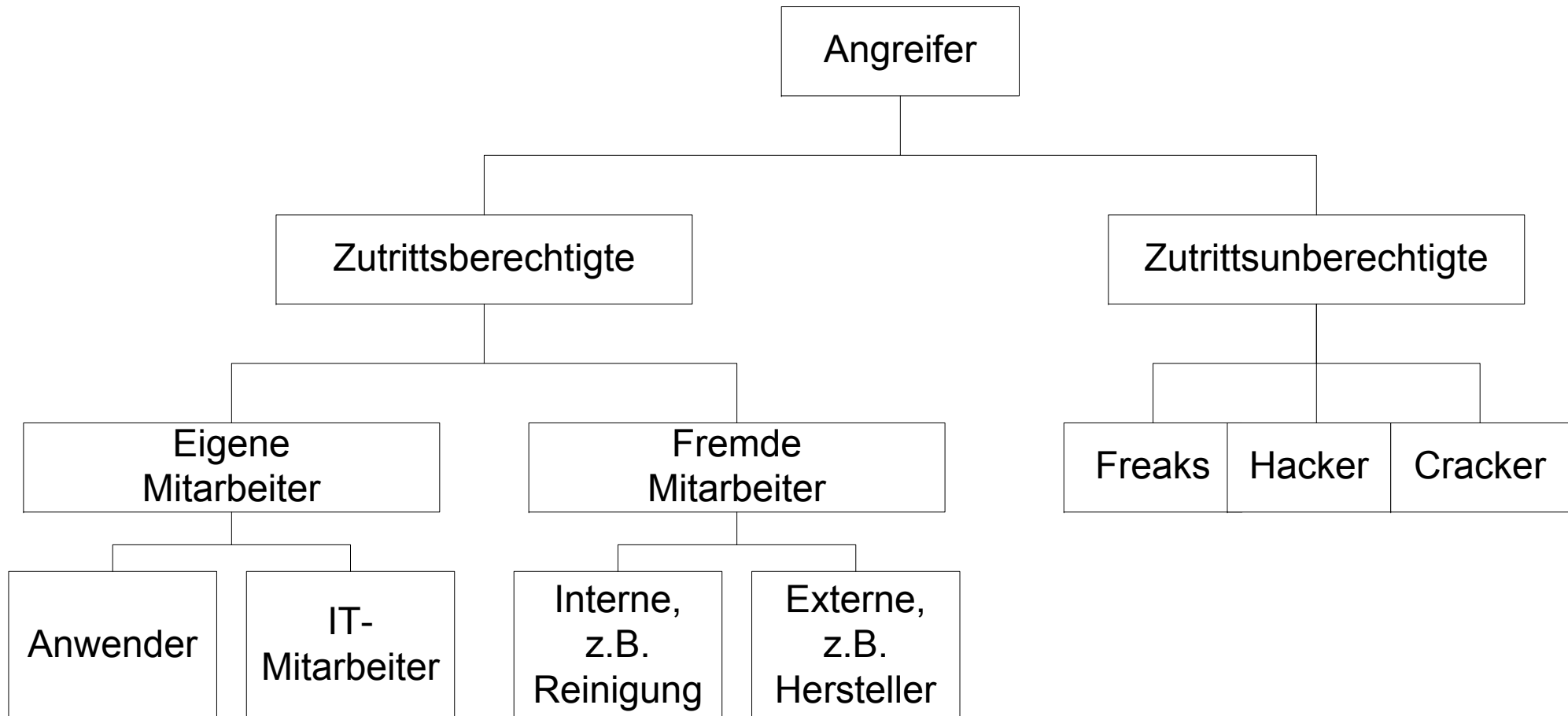
- Die „Bösen“:
  - **Eve (E)**  
Eavesdropper; Abhörender
  - **Mallory, Mallet (M)**  
Malicious attacker; Angreifer
- Bsp.: Abhören der Kommunikation zwischen A. und B. (UML Sequence Diagram)



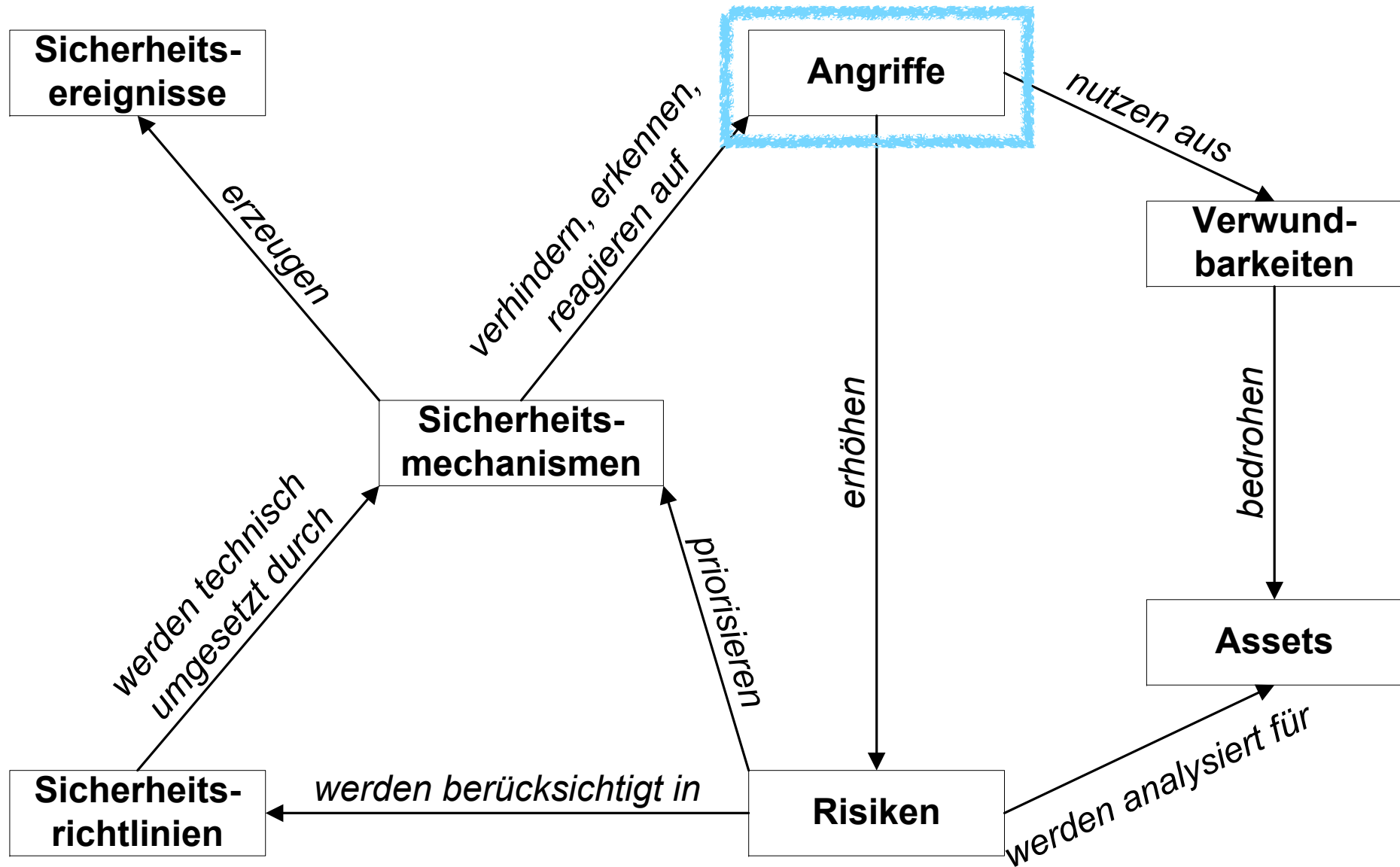
# Angreifermodelle

- Was können Eve, Mallory und Mallet?
- Angreifermodell umfasst u.a. Angaben zu
  - Position des Angreifers
    - Innentäter
    - Besucher, Einbrecher, ...
    - Internet
  - Fähigkeiten des Angreifers (= Wissen + finanzielle Möglichkeiten), z.B. bei
    - experimentierfreudigen Teenagern
    - IT-Sicherheitsexperten
    - erfahrenen Industriespionen
  - Motivation bzw. Zielsetzung des Angreifers
    - Spieltrieb
    - Geltungsbedürfnis
    - Geld
    - Vandalismus

# Tätertypisierung



# Begriffe und Zusammenhänge



# Bedrohungsanalyse: Bedrohungen und Angriffe

## ■ IT-Systeme sind **Bedrohungen (Threats)** ausgesetzt:

- ❑ Unberechtigter Zugriff auf Daten
- ❑ Diebstahl, Modifikation, Zerstörung
- ❑ Störung der Verfügbarkeit, Ressourcen unbenutzbar machen

## ■ **Angriff (Attack; konkrete Bedrohungsinstanz)**

Unberechtigter Zugriff auf ein System (oder der Versuch)

- ❑ **Passiver Angriff**  
Angreifer kann nur Informationen erlangen, diese aber **nicht ändern**
- ❑ **Aktiver Angriff**  
Angreifer kann Daten und/oder Systeme manipulieren
- ❑ **Social Engineering**  
Angreifer erlangt Daten, indem er „menschliche Schwäche“ ausnutzt

## ■ Beispiele für Angriffe

- ❑ Abhören (Eavesdropping, wiretapping, sniffing)
- ❑ Maskerade (Masquerade)  
Mallet behauptet, Alice zu sein
- ❑ Kompromittieren von Accounts (z.B. *root*)
- ❑ Ressourcenmissbrauch; escalation of privileges
- ❑ Kommunikationsangriffe: Ändern, Kopieren, Wiedereinspielen, Fälschen, Umleiten, Verzögern, Löschen von Nachrichten
- ❑ Denial of service (DoS), Distributed DoS (DDoS): Berechtigte Nutzer können Dienste nicht mehr (uneingeschränkt) nutzen

# Inhalt (1)

1. Security Engineering – Ziel und Vorgehensmodell
2. Notation von Sicherheitsproblemen: Handelnde Personen
3. Angreifermodelle
4. Bedrohungen, Angriffe und Gefährdungen
  1. Denial of Service
  2. Distributed Denial of Service
  3. Malicious Code
    - Viren
    - Würmer
    - Trojanische Pferde
  4. Hoax und Spam
  5. Mobile Code
    - ActiveX
    - JavaScript

# Denial of Service (DoS) and DDoS

- Angriff versucht, das Zielsystem oder Netzwerk für berechnigte Anwender unbenutzbar zu machen, z.B. durch:
  - Überlastung
  - Herbeiführen einer Fehlersituation
  - Ausnutzung von Programmierfehlern oder Protokollschwächen (Vulnerabilities)
- 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” “Vacation” Mail mit cc: an interne Mailingliste  
Außerdem konfiguriert er automatische Bestätigung durch Empfänger  
⇒ **Mailstorm**



# DoS Beispiele

## ■ 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

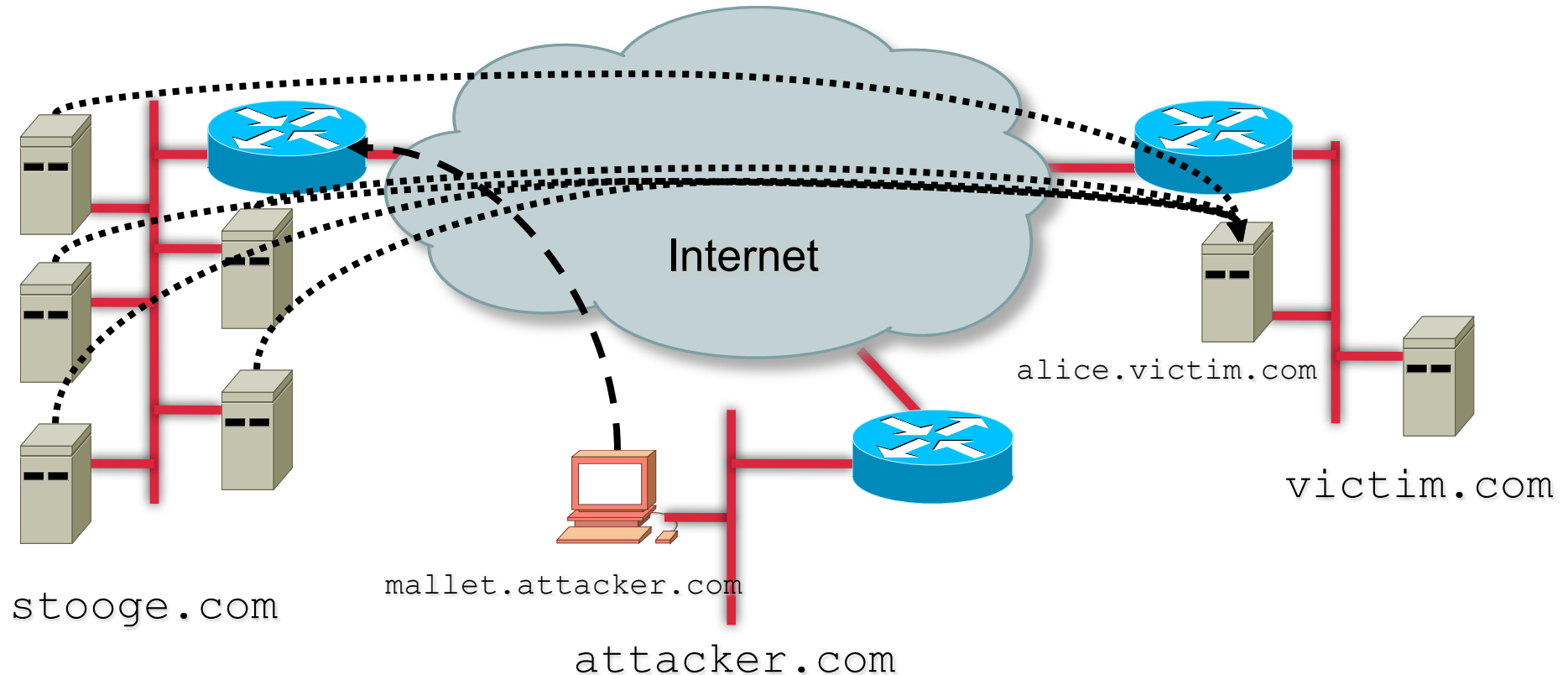
- **Land:** gefälschtes IP-Paket mit *IP Source Adr. = IP Destination Adr.* und *Source Port = Dest. Port*  
⇒ 100 % CPU Last bei best. Implementierungen (1997)
- **Teardrop:** Fragmentierte Pakete enthalten Feld `Fragment Offset`  
Hier Manipulation, so dass sich Fragmente „überlappen“  
⇒ u.U. Absturz des Systems (Win95, WinNT, Linux 2.0)

## ■ Aufbrauchen von Bandbreite bzw. Betriebssystem-Ressourcen

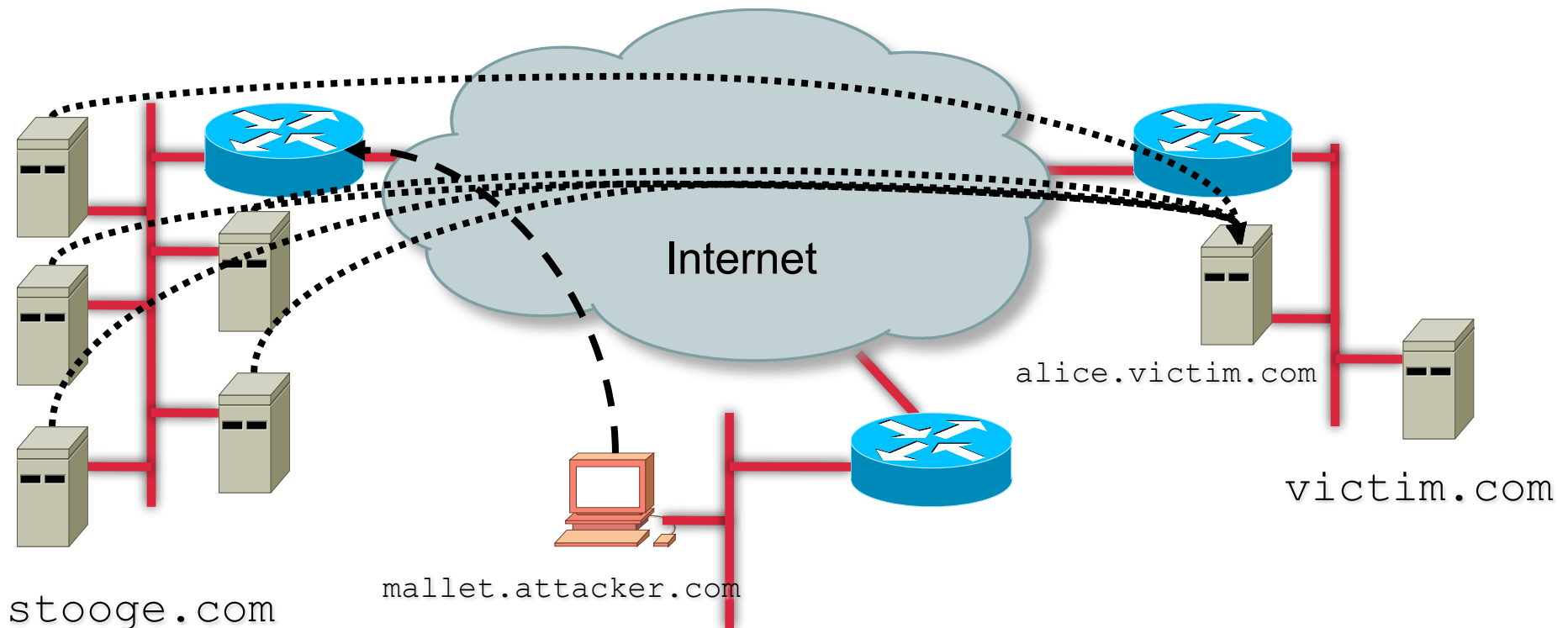
- Fluten des Netzwerkes des Opfers (z.B. SMURF)
- SYN-Flooding
- Low Orbit Ion Cannon (LOIC)

# 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)



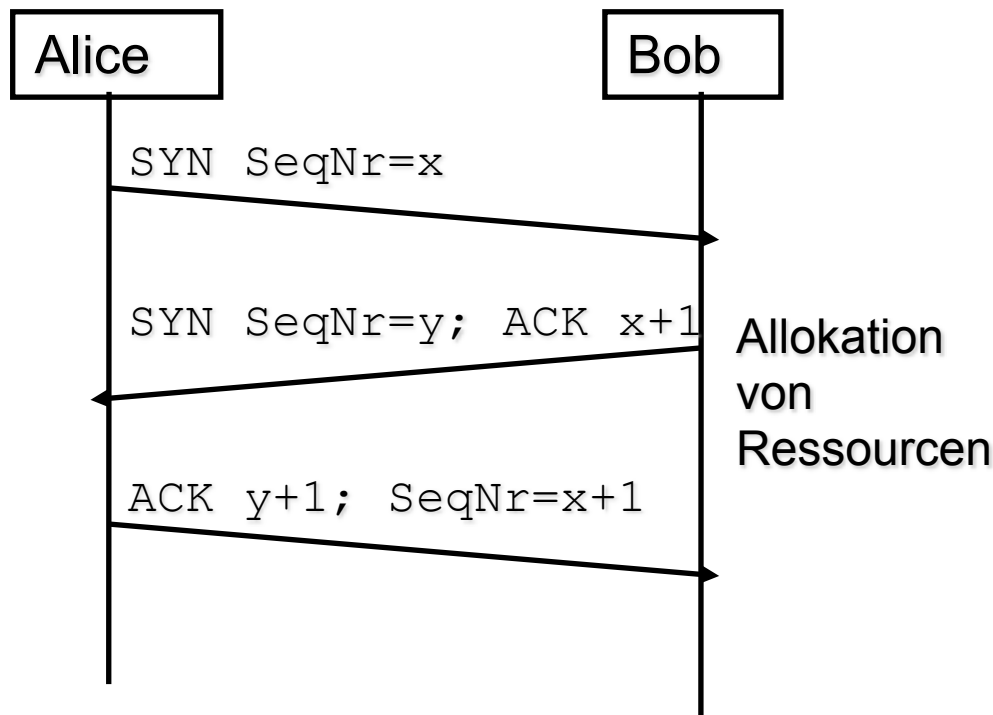
# SMURF: Gegenmaßnahmen?



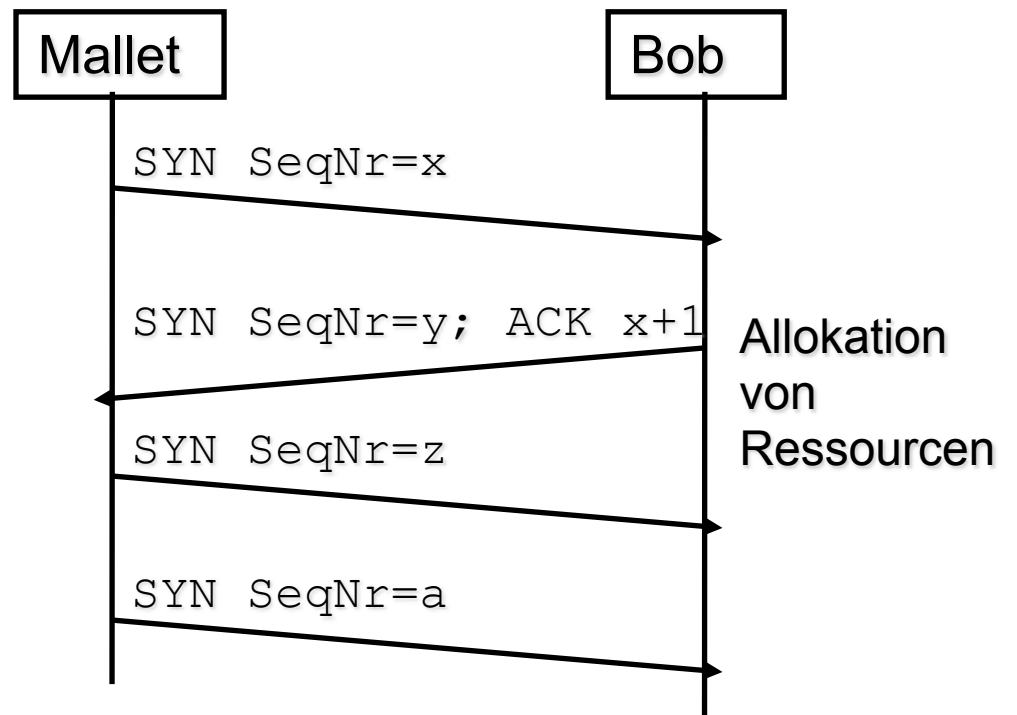
- Überkompensation:  
ICMP oder IP-Broadcast am Router komplett deaktivieren
- Besser:
  - ❑ Server so konfigurieren, dass sie nicht auf Broadcast-Pings antworten
  - ❑ Router so konfigurieren, dass sie von außen an die Broadcast-Adresse gerichtete Pakete nicht weiterleiten

# DoS-Techniken: SYN Flooding

- TCP 3-Way-Handshake zum Verbindungsaufbau



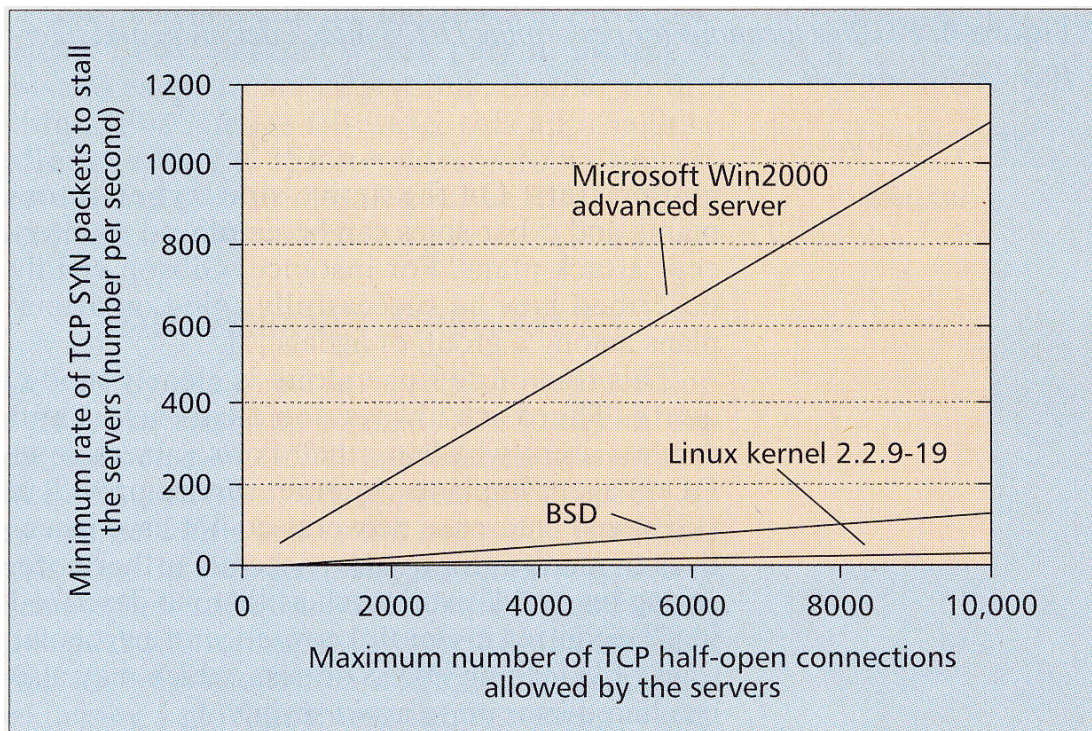
- SYN Flooding



- ❑ „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]

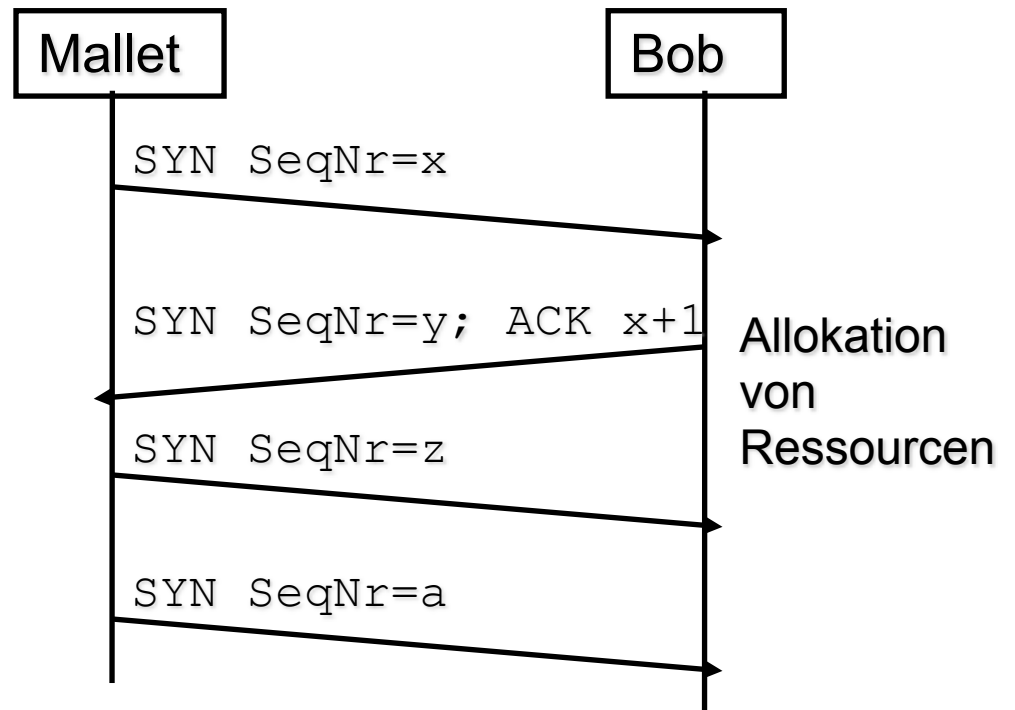


- Wiederholung von „verlorenen“ SYN-Paketen:

- 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.



# Distributed Denial of Service (DDoS)

## ■ 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

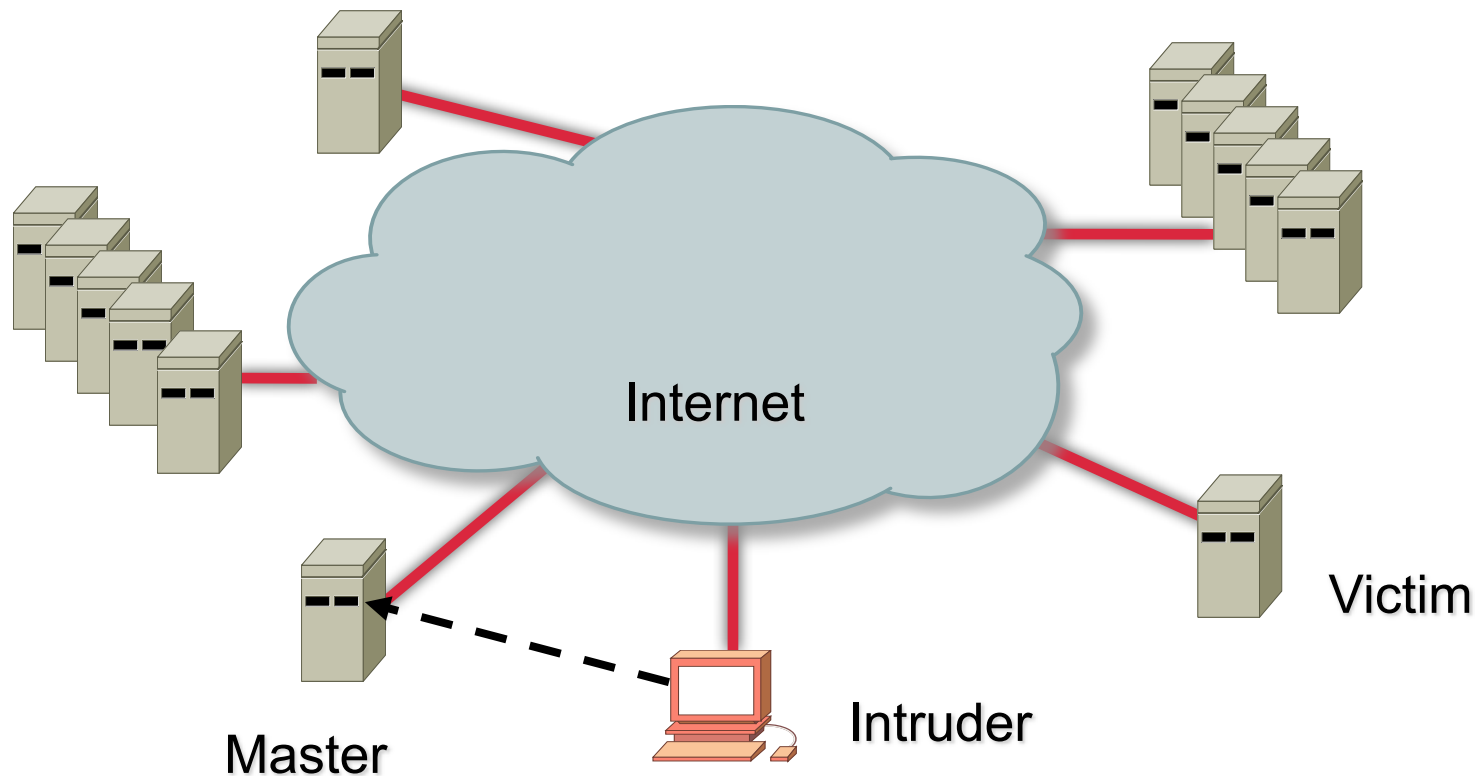
## ■ Beispiele:

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

# DDoS: Grundsätzlicher Ablauf

## ■ Dreistufiges Verfahren:

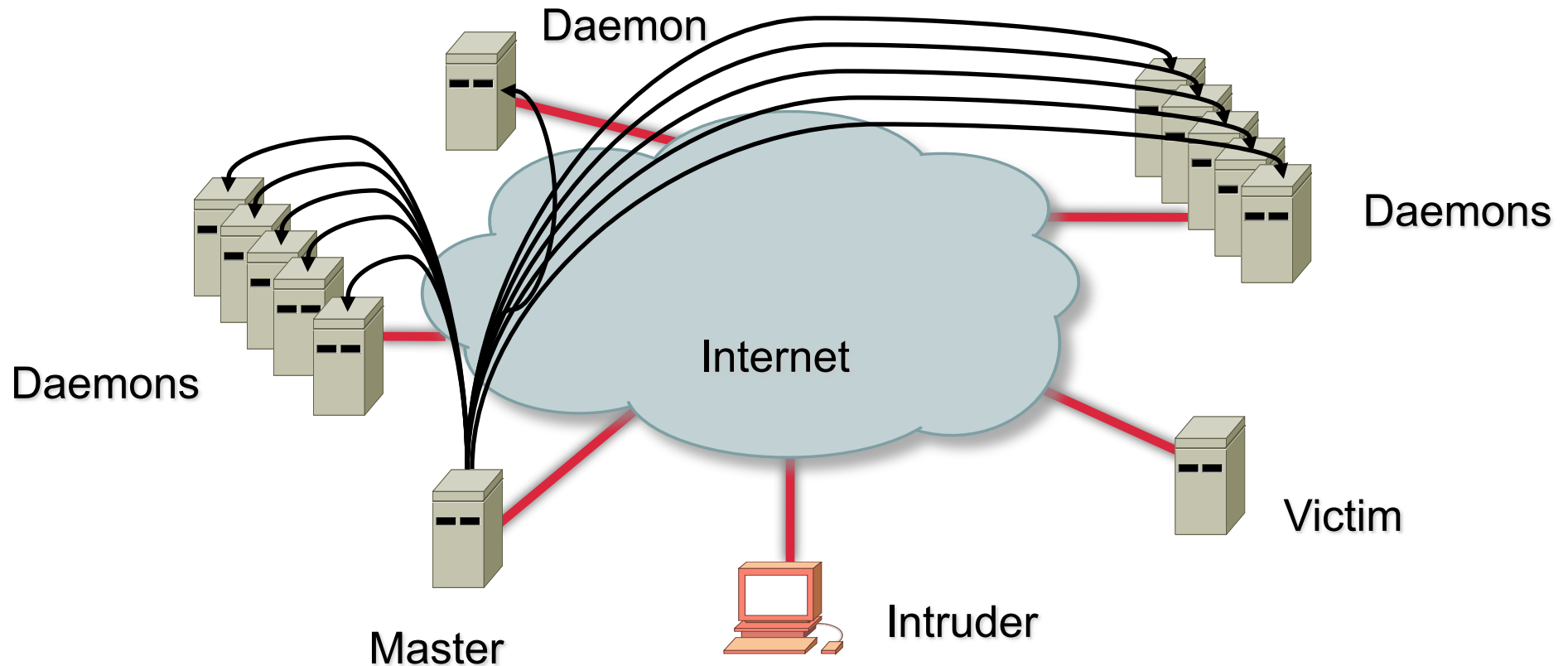
1. Intruder findet Maschine(n) die kompromittiert werden können; Hacking-Werkzeuge, Scanner, Rootkits, DoS/DDoS-Tools werden installiert; ⇨ Maschine wird Master





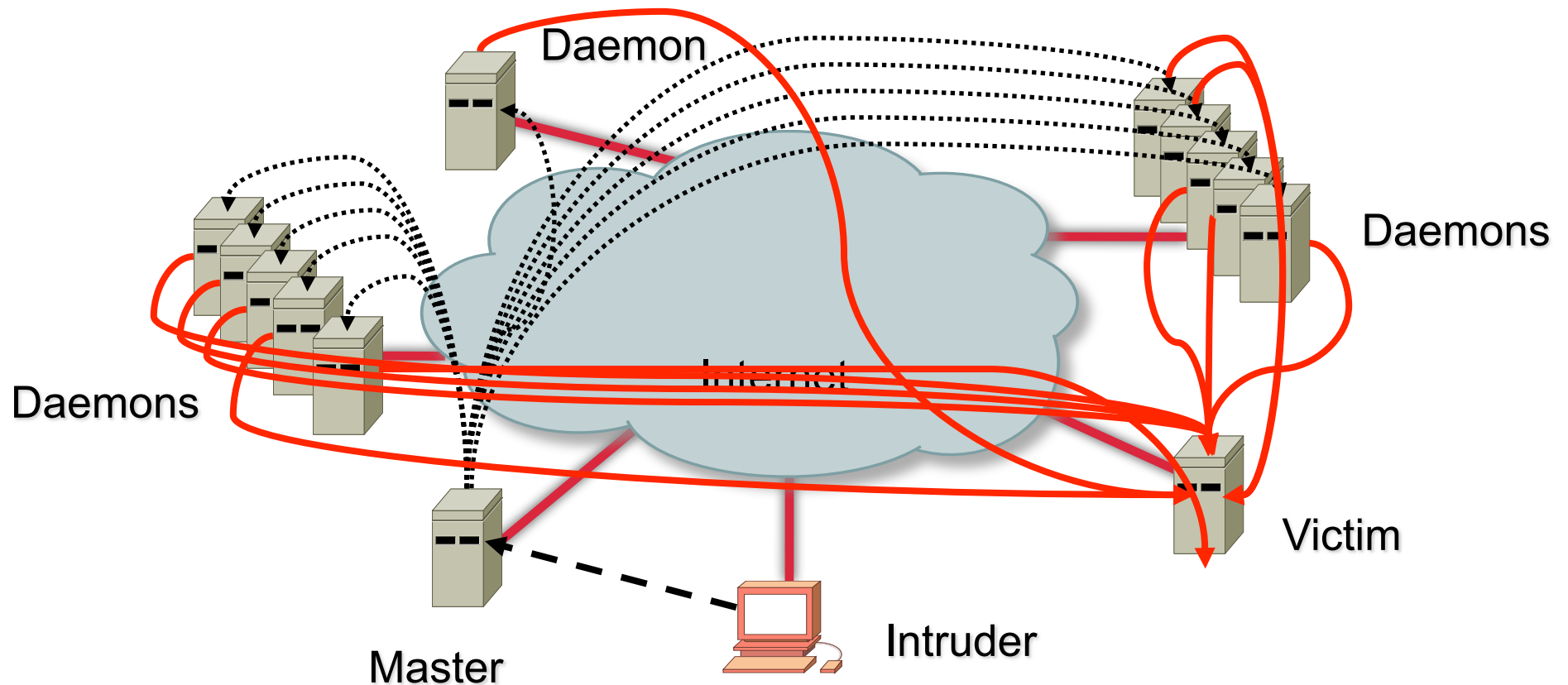
## (Fortsetzung)

2. Master versucht automatisiert, weitere Maschinen zu kompromittieren, um DDoS-Software (Daemon) zu installieren



## (Fortsetzung)

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



# DDoS Beispiele: Trin00, TFN, Stacheldraht

## ■ 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

Details unter

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

## ■ Tribe Flood Network (TFN)

- Master kompromittiert UNIX-Systeme über RPC-Buffer-Overflow
- SYN-, ICMP-, UDP-Flooding
- SMURF-Angriff

## ■ Kommunikation:

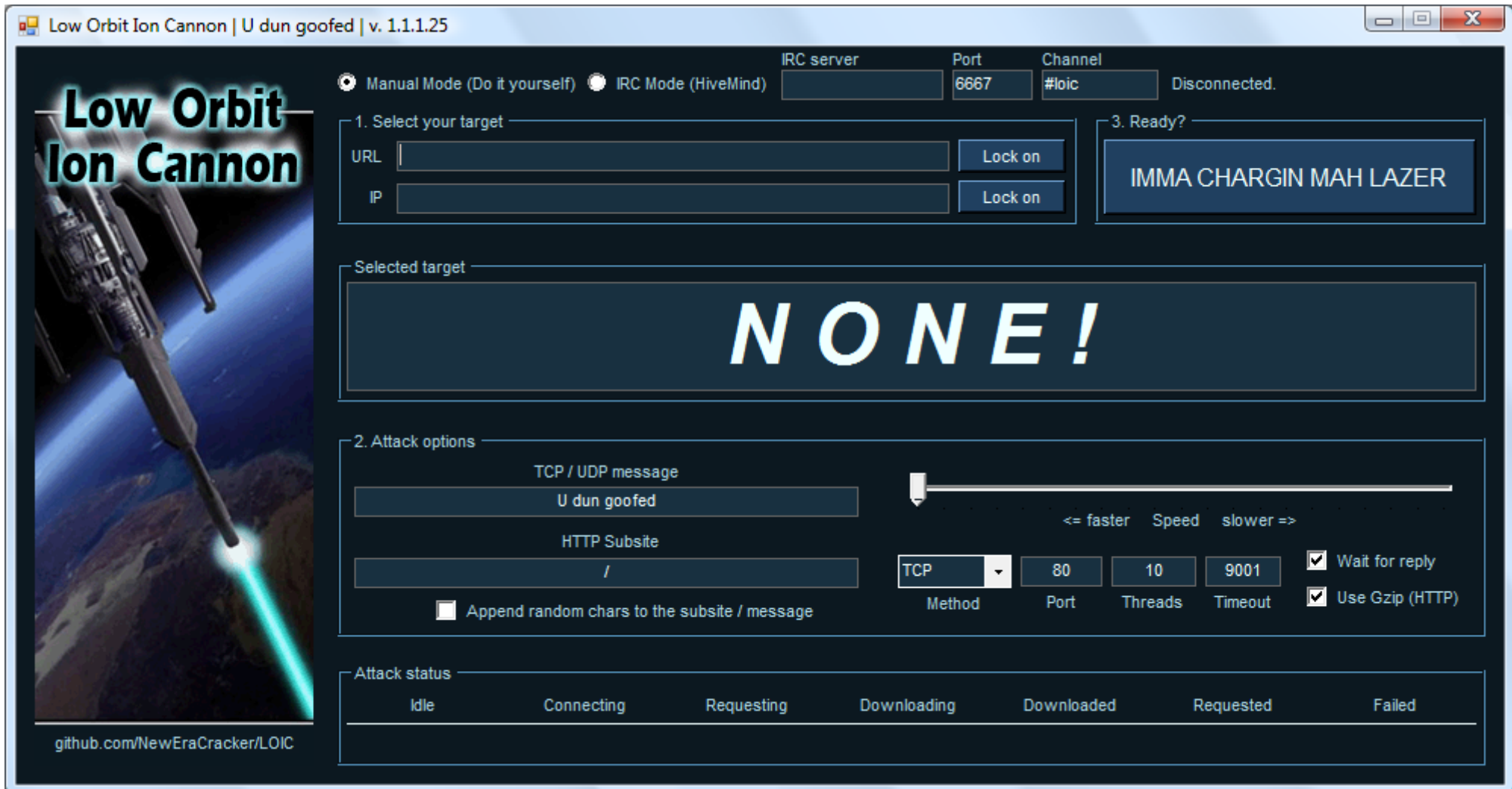
- wird vollständig in ICMP ECHO und ICMP REPLY Nachrichten „versteckt“:  
Kommando wird im Identifier Feld des ICMP Paketes kodiert; z.B.  
345 -> SYN-Flooding;  
890 -> UDP-Flooding;
- Kein Passwort-Schutz

## ■ Stacheldraht = Trinoo + TFN + verschlüsselte Kommunikation + Auto-Update des Agenten

# Low Orbit Ion Cannon (LOIC)

- Open Source „Network Stress Testing Application“
- Flooding mit TCP- oder UDP-Paketen
- Weltweit 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 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

# LOIC GUI



# (D)DoS: Schutz- und Gegenmaßnahmen

- **Generell:**
  - Pauschaler Schutz gegen (D)DoS-Angriffe ist praktisch fast unmöglich
  - Aber: Spezifika einzelner Angriffe erlauben oft gute Schutzmaßnahmen
- **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
  - Bug- und Sicherheitswarnungen (z.B. CERT) verfolgen

# Beispiel: Erpressungsversuch mit DDoS-Drohung

**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](http://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
  - „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
```

# Inhalt (1)

1. Security Engineering – Ziel und Vorgehensmodell
2. Notation von Sicherheitsproblemen: Handelnde Personen
3. Angreifermodelle
4. Bedrohungen, Angriffe und Gefährdungen
  1. Denial of Service
  2. Distributed Denial of Service
  3. Malicious Code
    - Viren
    - Würmer
    - Trojanische Pferde
  4. Hoax und Spam
  5. Mobile Code
    - ActiveX
    - JavaScript



# Malicious Code: Virus

## ■ 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 &amp;&amp; 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)

Name der Datei	Faxsend.exe
Länge der Datei	9488
Einsprungadresse	1004
Programmcode	1004: load... 1005: add... ..... 9488: Ret

- Datei nach der Infektion

Name der Datei	Faxsend.exe
Länge der Datei	9889
Einsprungadresse	9489
Programmcode	1004: load... 1005: add... ..... 9488: Ret  Viruscode 9489: mult... ..... 9889: jmp 1004

# Viren: Klassifikation

## ■ 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.

# Einschub: False-Positives bei Virensignaturen

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>

# Malicious Code: Wurm

## ■ 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)
- ❑ .....

# Malicious Code: Trojanisches Pferd

## ■ 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."
```

# Trojanische Pferde: Beispiele

- 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“

# „Staatstrojaner“

- Veröffentlichte Analyse (08.10.2011)  
<http://www.ccc.de/system/uploads/76/original/staatstrojaner-report23.pdf>
  
- Chaos Computer Club (CCC) analysiert zugespielte 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 in Ohio USA



# „Staatstrojaner“: Analyse

## ■ 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

# „Staatstrojaner“ Befehlssatz C&C

- Nicht alle Kommandos konnte 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

# Malicious Code heute

- Grenzen zwischen Klassen verschwinden
- Heutige Schadsoftware umfasst i.d.R. mehrere Klassen, z.B.
  - Virus mit Wurmfunktionalität
  - Wurm mit Trojanischem Pferd und Backdoor
  - Schadsoftware mit DoS- bzw. DDoS-Funktionalität
  - Schadsoftware mit eigenem Mail-Server für E-Mail-Spamming
  - usw.

# Malicious Code: Schutz- und Gegenmaßnahmen

- 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.)*

# Inhalt (1)

1. Security Engineering – Ziel und Vorgehensmodell
2. Notation von Sicherheitsproblemen: Handelnde Personen
3. Angreifermodelle
4. Bedrohungen, Angriffe und Gefährdungen
  1. Denial of Service
  2. Distributed Denial of Service
  3. Malicious Code
    - Viren
    - Würmer
    - Trojanische Pferde
  4. Hoax und Spam
  5. Mobile Code
    - ActiveX
    - JavaScript

# E-Mail: Falsche Virenwarnungen; Hoaxes

## ■ 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

# Hoax, mögliche Erkennungszeichen

- Warnung vor „extrem gefährlichem Virus“
- “Keine AV 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>

# SPAM-E-Mail

- Unerwünschte Werbemails (unsolicited commercial e-mail, UCE)
- Begriff SPAM
  - SPAM eingetragenes Warenzeichen von Hormel Food
  - „Spam“-Sketch aus Monty Python's Flying Circus
- SPAM-Aufkommen
  - Am Beispiel LRZ, 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,....)





# 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

# Phishing

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.

# Grenzen zw. Spam, Hoax und Phishing verwischen

## ■ Abmahnung:

**Betreff: Ermittlungsverfahren gegen Sie**

Guten Tag,

in obiger Angelegenheit zeigen wir die anwaltliche Vertretung und Interessenwahrung der Firma Videorama GmbH, Munchener Str. 63, 45145 Essen, an.

Gegenstand unserer Beauftragung ist eine von Ihrem Internetanschluss aus im sogenannten Peer-to-Peer-Netzwerk begangene Urheberrechtsverletzung an Werken unseres Mandanten. Unser Mandant ist Inhaber der ausschliesslichen

Nutzungs- und Verwertungsrechte im Sinne der §§ 15ff UrhG bzw. § 31 UrhG an diesen Werken, bei denen es sich um geschützte Werke nach § 2 Abs 1 Nr. 1 UrhG handelt.

Durch das Herunterladen urheberrechtlich geschützter Werke haben sie sich laut § 106 Abs 1 UrhG i.V. mit §§ 15,17,19 Abs. 2 pp UrhG nachweislich strafbar gemacht. Bei ihrem Internetanschluss sind mehrere Downloads von musikalischen Werken dokumentiert worden.

Aufgrund dieser Daten wurde bei der zuständigen Staatsanwaltschaft am Firmensitz unseres Mandanten Strafanzeige gegen Sie gestellt.

**Aktenzeichen: XXX Js XXX/14 Sta Essen**

**Ihre IP Adresse zum Tatzeitpunkt: XXX.XXX.XXX.XXX**

**Ihre E-Mail Adresse: info@XXXXXXXXX.de**

**Illegal heruntergeladene musikalische Stücke (mp3): 18**

**Illegal hochgeladene musikalische Stücke (mp3): 24**

Wie Sie vielleicht schon aus den Medien mitbekommen haben, werden heutzutage Urheberrechtverletzungen erfolgreich vor Gerichten verteidigt, was in der Regel zu einer hohen Geldstrafe sowie Gerichtskosten führt.

Link: Urheberrecht: Magdeburger muss 3000 Euro Schadensersatz zahlen

<http://www.petanews.de/it-news/urheberrecht-magdeburger-muss-3000-euro-schadensersatz-fur-132-musiktitel-zahlen/>

# (Fortsetzung)

Genau aus diesem Grund unterbreitet unsere Kanzlei Ihnen nun folgendes Angebot: Um weiteren Ermittlungen der Staatsanwaltschaft und anderen offiziellen Unannehmlichkeiten wie Hausdurchsuchungen, Gerichtsterminen aus dem Weg zu gehen, gestatten wir Ihnen den Schadensersatzanspruch unseres Mandanten aussergerichtlich zu lösen.

Wir bitten Sie deshalb den Schadensersatzanspruch von 100 Euro bis zum 18.10.2010 sicher und unkompliziert mit einer UKASH-Karte zu bezahlen. Eine Ukash ist die sicherste Bezahlmethode im Internet und für Jedermann anonym an Tankstellen, Kiosken etc. zu erwerben. Weitere Informationen zum Ukash-Verfahren erhalten Sie unter:

<<http://www.ukash.com/de> <<http://www.ukash.com/de/de/where-to-get.aspx>>

Senden Sie uns den 19-stelligen Pin-Code der 100 Euro Ukash an folgende E-Mailadresse [zahlung@rechtsanwalt-giese.info](mailto:zahlung@rechtsanwalt-giese.info)

Geben Sie bei Ihrer Zahlung bitte Ihr Aktenzeichen an!

Sollten Sie diesen Bezahlvorgang ablehnen bzw. wir bis zur angesetzten Frist keinen 19-stelligen Ukash PIN-Code im Wert von 100 Euro erhalten haben, wird der Schadensersatzanspruch offiziell aufrecht erhalten und das Ermittlungsverfahren mit allen Konsequenzen eingeleitet. Sie erhalten dieses Schreiben daraufhin nochmals auf dem normalen Postweg.

Hochachtungsvoll,  
Rechtsanwalt Florian Giese

# GIESE RECHTSANWÄLTE

Hamburg, 20.10.2010

## Hinweis und Mitteilung vom 20.10.2010

Rechtsanwalt Florian Giese ist nicht Urheber von betrügerischen E-Mails mit dem Betreff "Ermittlungsverfahren gegen Sie". Hierbei handelt es sich um Spam-Mails von Betrügern.

Beweismittel sind hier ausreichend vorhanden. Bitte sehen Sie von Telefonanrufen und E-Mails ab und lesen Sie zunächst diese Mitteilung. Ich danke für Ihr Verständnis.

Die Kanzlei Giese Rechtsanwälte in Hamburg und insbesondere Rechtsanwalt Florian Giese stehen nicht im Zusammenhang mit den betrügerischen E-Mails, welche angeblich im Auftrag der Firma **Videorama GmbH** wg. Urheberrechtsverletzung in Filesharing-Netzwerken versendet werden.

Weder hat die Firma Videorama GmbH hier einen solchen Auftrag erteilt, noch sind durch RA Florian Giese unter der E-Mail Adresse "**giese@rechtsanwalt-giese.info**" oder "**zahlung@rechtsanwalt-giese.info**" derartige Mails versendet worden. Unsere Web- oder Mailserver sind zu keine Zeit Gegenstand von Hacker-Angriffen gewesen.

Auch ist die Kanzlei Giese oder RA Florian Giese nicht Inhaber der Domain "**rechtsanwalt-giese.info**" und daher auch nicht in Besitz der vorgenannten Absende-Adressen. Rechtsanwalt Florian Giese hat zudem keinerlei Einfluss auf die Inhalte, welche auf der Domain "**rechtsanwalt-giese.info**" platziert sind. Die Kanzlei Giese Rechtsanwälte unterhält Internetpräsentationen lediglich unter den Domains **rechtsanwalt-giese.de** und **kanzlei-giese.com**.

>> [STARTSEITE](#)

>> [RECHTSANWÄLTE](#)

>> [BERATUNG](#)

>> [ONLINE-BERATUNG](#)

>> [SCHWERPUNKTE](#)

>> [MEDIALAW-NEWS](#)

>> [KONTAKT](#)

>> [IMPRESSUM](#)

### Markenanmeldung

Markenschutz – einfach und schnell: Markenanmeldung (DE) inkl. Identitätsrecherche (DE, EU, IR) ab 179,00 Euro zzgl. 19% USt. und Amtsgebühren des DPMA. mehr ...

### Rechtsprechung

Urheberrecht:  
Urheberrechtsschutz für  
Heiratsannoncen

# Spam, klassische Gegenmaßnahmen: Spamfilter

- 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. Maschinelles Lernen 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).

# Spamfilter

## ■ 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)?

## ■ Beispiel: SpamAssassin (<http://spamassassin.apache.org/>)

- ❑ Implementiert alle Filterarten (Blacklist, Regelbasis, Bayessches Netz)
- ❑ Zentral und dezentral einsetzbar
- ❑ feingranular konfigurierbar

# Weitere Anti-SPAM-Maßnahmen

- Eigenes Kapitel



# Inhalt (1)

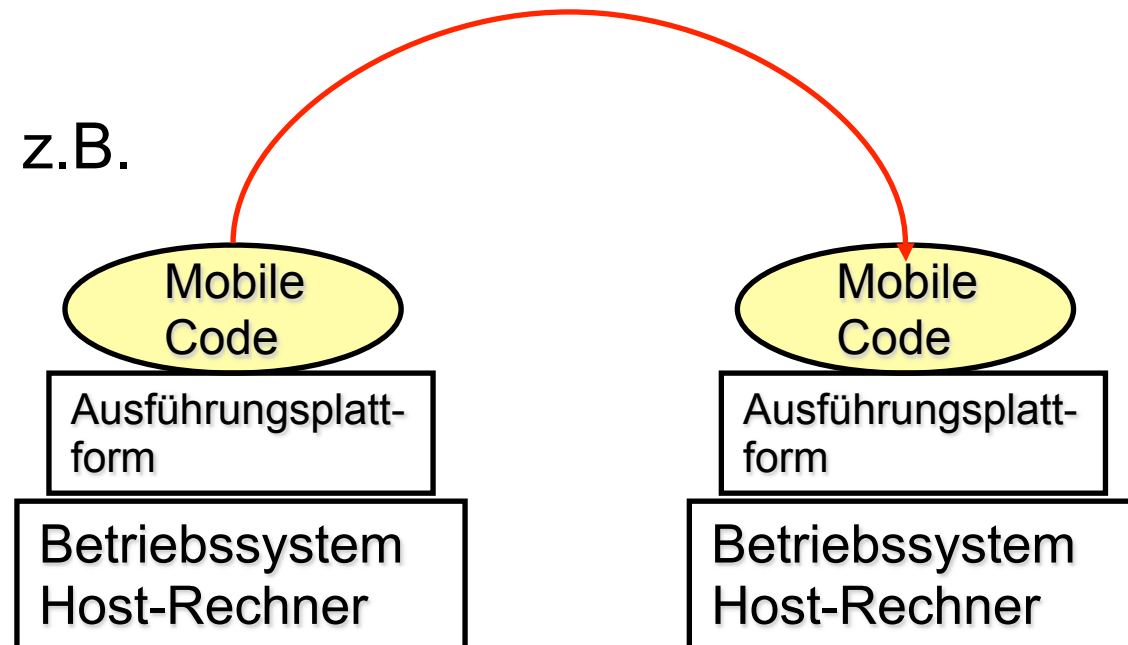
1. Security Engineering – Ziel und Vorgehensmodell
2. Notation von Sicherheitsproblemen: Handelnde Personen
3. Angreifermodelle
4. Bedrohungen, Angriffe und Gefährdungen
  1. Denial of Service
  2. Distributed Denial of Service
  3. Malicious Code
    - Viren
    - Würmer
    - Trojanische Pferde
  4. Hoax und Spam
5. Mobile Code
  - ActiveX
  - JavaScript

# Malicious Code: „Mobile Code“

- 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



# Mobile Code: ActiveX

- 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](http://www.iks-jena.de/mitarb/lutz/security/activex.html)

# Mobile Code: JavaScript

- 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/](http://www.peacefire.org/security/jscookies/)

# JavaScript: Ein Sicherheits-Dauerbrenner

- 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 from March 16, 2011. The article title is "RIM: Blackberry-Nutzer sollen JavaScript deaktivieren". The author is Ryan Naraine and Stefan Beiersmann. The article discusses a security vulnerability in the BlackBerry browser where JavaScript can be used to execute remote code. It mentions that RIM has advised users to deactivate JavaScript in their browsers to protect themselves. The article also includes a sidebar with related news items and a Facebook widget.

**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







# RSA Security Hack: Einfallstor Adobe Flash

- 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

# Silverlight: Marketing

**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  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](http://www.pcworld.com/article/131472/silverlight_declared_secure.html)

# Silverlight: Realität

The screenshot shows a Cisco security alert page. At the top is the Cisco logo and navigation menu. The breadcrumb trail is: HOME > SECURITY INTELLIGENCE OPERATIONS > Latest Threat Information > Multivendor Security Alerts. The alert title is "Microsoft Silverlight Pointer Handling Memory Corruption Vulnerability". The threat type is "Unintended Weakness: Arbitrary Code Execution". The alert details include: IntelliShield ID: 21034, Version: 1, First Published: August 10, 2010 02:54 PM EDT, Last Published: August 10, 2010 02:54 PM EDT, Vector: Network, Authentication: None, Exploit: Unproven, Port: Not Available, CVE: CVE-2010-0019. The severity is Moderate Damage (CVSS Base: 9.3, CVSS Temporal: 6.9). The urgency is Unlikely Use. The credibility is Confirmed. The page also features a "Powered by" section and a "Related Links" section with solutions like Security Solutions, E-mail Security, Threat Control, and Threat Control. The version summary states: "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."

Worldwide Log

CISCO 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

Threat Type: **Unintended Weakness: Arbitrary Code Execution**

IntelliShield ID:	21034	Urgency:	Unlikely Use	
Version:	1	Credibility:	Confirmed	
First Published:	August 10, 2010 02:54 PM EDT	Severity:	Moderate Damage	
Last Published:	August 10, 2010 02:54 PM EDT	CVSS Base:	9.3	<a href="#">CVSS Calculator</a>
Vector:	Network	CVSS Temporal:	6.9	<a href="#">CVSS Version 2</a>
Authentication:	None			
Exploit:	Unproven			
Port:	Not Available			
CVE:	CVE-2010-0019			

Powered by

Related Links

Solutions

- [Security Solutions](#)
- [E-mail Security](#)
- [Threat Control](#)
- [Threat Control](#)

Products & Services

- [Cisco Security Manager Security](#)
- [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>



# Inhalt (2)

## 4. Bedrohungen, Angriffe und Gefährdungen (Forts.)

6. Buffer Overflow

7. Account-/Password-Cracking

8. Hintertüren / Falltüren

9. Rootkits

10. Cross Site Scripting (XSS)

11. Sniffer

12. Port Scanner

13. Social Engineering

## 5. Rechtliche Regelungen: Strafgesetzbuch

6. „Top Cyber Security Risks“

7. Zusammenfassung

# Exploits: Buffer Overflow (hier: stack smashing)

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

■ Speicherabbild eines Programms (am Bsp. Unix)

■ 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

0xFFFFFFFF



0x00000000

# Beispiel: Anfälliger C-Code

```
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“).

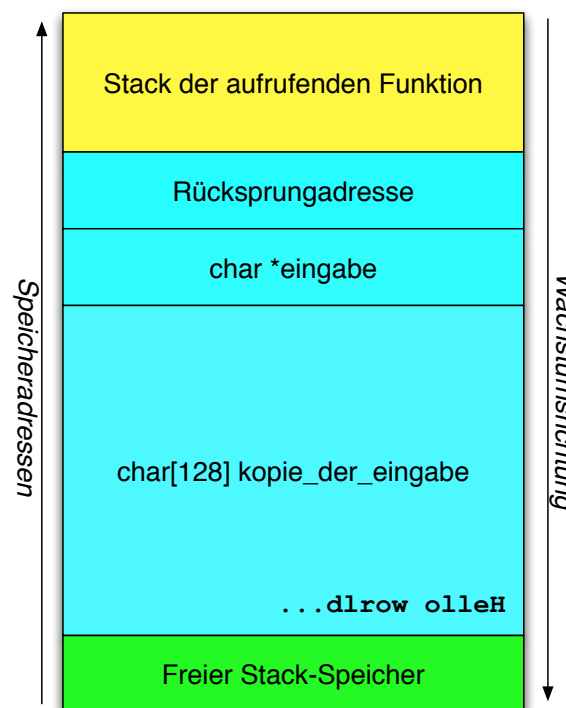
# Ausnutzen von Buffer Overflows in Stack-Segmenten

- 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)

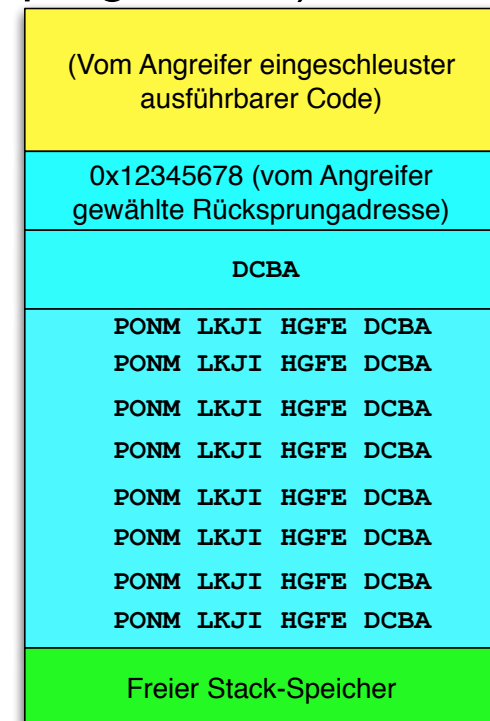
```
1 #include <string.h>
2
3 void kopiere_eingabe (char *eingabe)
4 {
5     char kopie_der_eingabe[128];
6     strcpy(kopie_der_eingabe, eingabe);
7 }
8
9 int main (int argc, char **argv)
10 {
11     kopiere_eingabe(argv[1]);
12 }
```

Quelltext

Anmerkung: Darstellung des Stack-Aufbaus vereinfacht!



Stack bei  
regulärer Eingabe



Stack bei  
Buffer Overflow

# Kleinere Hürden beim Stack-Smashing

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

```
Rücksprung erfolgt  
„irgendwo“ hierhin:  →  NOP  
                       →  NOP  
                       →  NOP  
                       →  NOP  
                       →  NOP  
                       →  NOP  
Schadcode beginnt  
ab hier:              →  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

# Shellcode: Beispiel

## ■ 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](http://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()`).

# Schutz vor Stack-Smashing

- 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.

# Buffer-Overflows: Weitere Aspekte

## ■ 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



# Inhalt (2)

## 4. Bedrohungen, Angriffe und Gefährdungen (Forts.)

6. Buffer Overflow

7. Account-/Password-Cracking

8. Hintertüren / Falltüren

9. Rootkits

10. Cross Site Scripting (XSS)

11. Sniffer

12. Port Scanner

13. Social Engineering

## 5. Rechtliche Regelungen: Strafgesetzbuch

6. „Top Cyber Security Risks“

7. Zusammenfassung

# Account/Password Cracking

- Passworteingabe ist das am weitesten verbreitete Authentifizierungsverfahren
- Ziel des Angriffs: „Erraten“ von Benutzername 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

```
reiser:Ad9%y?SmW+zP&:23:17:Helmut Reiser:/home/reiser:/bin/tcsh
Username:Password:UID:GID:Gecko-String:Home-Verzeichnis:Shell
```

# Bsp. UNIX-Authentisierung: User / Password

## ■ 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
- ....

# Back Doors, Trap Doors

- 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!

# Rootkits

## ■ 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 (Fortsetzung)

## ■ 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

# Rootkits: Beispiele

- 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.

# Rootkits: Moderne Ausprägungen

## ■ 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: false positives)
- ❑ Externe Analyse (Booten von CD)



# Inhalt (2)

4. Bedrohungen, Angriffe und Gefährdungen (Forts.)
  6. Buffer Overflow
  7. Account-/Password-Cracking
  8. Hintertüren / Falltüren
  9. Rootkits
  10. Cross Site Scripting (XSS)
  11. Sniffer
  12. Port Scanner
  13. Social Engineering
5. Rechtliche Regelungen: Strafgesetzbuch
6. „Top Cyber Security Risks“
7. Zusammenfassung

# Cross Site Scripting (XSS)

- 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 Profildern)

# XSS: Grundproblem

- 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 lässt JavaScript-Code im Feld „Mobiltelefonnummer“ zu.
  - Skype-Client von Bob führt diesen Code ungeprüft aus.
- JavaScript kann u.a. HTML-Formulare automatisch ausfüllen und abschicken; Missbrauch 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.

# DOM-basiertes (lokales) XSS

- 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")+5;
    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>](http://www.example.com/index.html?username=<script>alert()

- Als Parameter übergebener Code wird von anfälligen Browsern ausgeführt.

# Reflexives (nicht-persistentes) XSS

## ■ 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>
```

# Persistentes (stored) XSS

- 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 neue Issues mit einer Vielzahl von Attachments
  - Eines der Attachments ist JSP zum Durchsuchen und Kopieren von Dateisystem-Inhalten

# (Fortsetzung)

- 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](https://blogs.apache.org/infra/entry/apache_org_04_09_2010)



# Inhalt (2)

## 4. Bedrohungen, Angriffe und Gefährdungen (Forts.)

- 6. Buffer Overflow
- 7. Account-/Password-Cracking
- 8. Hintertüren / Falltüren
- 9. Rootkits
- 10. Cross Site Scripting (XSS)

11. Sniffer

12. Port Scanner

13. Social Engineering

## 5. Rechtliche Regelungen: Strafgesetzbuch

6. „Top Cyber Security Risks“

7. Zusammenfassung

# Sniffer: Abhören des Netzverkehrs

- 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

- 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...pcheger01.....Q...C.dns0...hostmaster..w.U..*0.....Q...
#
U 10.156.33.53:53 -> 129.187.15.116:60267
.....www.nm.ifi.lmu.de.....Q...pcheger01.....Q...T.....Q...dns2.lrz.....Q...dns1.....Q...dns1.
\.....Q...dns3\.....Q...acheron.....Q...dns0.....Q...T...n.....Q...T.....Q...L.....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: de-
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> .</> .<HR> .<ADDRESS>Apache/1.3.29 Server at www.nm.ifi.lmu.de Port 80</ADDRESS> .</BODY> .</HTML> .0....
```

# Port-Scanner

- 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)
```

# Social Engineering

- Sammelbegriff für Angriffe, die sich nicht direkt gegen Technik wenden, sondern den „Faktor Mensch“ ausnutzen.
- Eher passive Varianten, u.a.
  - Belauschen von Gesprächen
  - Beim Tippen „über die Schulter schauen“ (shoulder surfing)
  - Durchsuchen von Papiertonnen u.ä. (dumpster diving)
  - USB-Sticks gezielt „liegen lassen“ (baiting)
- Aktive Angriffe (soziale Manipulation), u.a.
  - Am Telefon als Mitarbeiter der IT-Abteilung oder guter Bekannter/Assistent des Chefs ausgeben (pretexting)
    - Hilfsbereitschaft ausnutzen, scheinbar harmlose Informationen erfragen
    - Mit diesen Informationen Vertrauen weiterer Zielpersonen erschleichen
  - Spear-Phishing-E-Mails
  - Internet-Bekanntschäften, z.B. über fingiertes Facebook-Konto

# Beispiele für Social Engineering

## ■ Robin Sage (2010)

- ❑ Fake-Account auf Facebook, Twitter, ...
- ❑ Angebliche IT-Sicherheitsberaterin mit MIT-Abschluss
- ❑ Opfer: Mitarbeiter von US-Militär, -Regierung und Rüstungsindustrie
- ❑ Innerhalb von zwei Monaten zahlreiche Kontakte zu hochrangigen Mitarbeitern, Zugang zu vertraulichen Dokumenten, Bankkonten, ...
- ❑ Operation „Robin Sage“ ist eine vierwöchige Übung für US-Spezialeinheiten („unconventional warfare exercise“).
- ❑ Experiment von Thomas Ryan zur Vertrauensseligkeit in Social Networks



## ■ USB-Sticks für Bankangestellte (<http://www.darkreading.com/security/article/208803634/index.html>)

- ❑ Bank beauftragt Assessment inkl. SE; Angestellte wissen sogar davon
- ❑ 20 USB-Sticks mit Malware auf Parkplatz, Weg zur Kantine, etc. „verloren“
- ❑ 15 USB-Sticks werden gefunden, alle 15 werden am Arbeitsplatz ausprobiert

## ■ Kevin Mitnick (Buch: Die Kunst der Täuschung, Risikofaktor Mensch)

# Inhalt (2)

## 4. Bedrohungen, Angriffe und Gefährdungen (Forts.)

- 6. Buffer Overflow
- 7. Account-/Password-Cracking
- 8. Hintertüren / Falltüren
- 9. Rootkits
- 10. Cross Site Scripting (XSS)
- 11. Sniffer
- 12. Port Scanner
- 13. Social Engineering

## 5. Rechtliche Regelungen: Strafgesetzbuch

- 6. „Top Cyber Security Risks“
- 7. Zusammenfassung

# Rechtliche Regelungen: Strafgesetzbuch

- Strafgesetzbuch (StGB) regelt Strafrecht
- Verletzungen der Normen werden im Strafverfahren verhandelt
- Antragsdelikt: Tat wird nur auf Antrag (Anzeige) i.d.R. durch den „Verletzten“ (§ 77) verfolgt (§ 202a, 202b, 303a, 303b)
- Offizialdelikt: Tat wird „von Amts wegen“ (Staatsanwaltschaft) verfolgt (§ 202c)
  
- § 202a: Ausspähen von Daten
- § 202b: Abfangen von Daten (seit 11.08.2007)
- § 202c: Vorbereiten des Ausspähens und Abfangens von Daten (seit 11.08.2007)
- § 303a: Datenveränderung
- § 303b: Computersabotage
- § 303c: Strafantrag

## § 202a StGB: Ausspähen von Daten

- (1) Wer unbefugt sich oder einem anderen Zugang zu Daten, die nicht für ihn bestimmt und die gegen unberechtigten Zugang besonders gesichert sind, unter Überwindung der Zugangssicherung verschafft, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.
  
- (2) Daten im Sinne des Absatzes 1 sind nur solche, die elektronisch, magnetisch oder sonst nicht unmittelbar wahrnehmbar gespeichert sind oder übermittelt werden.



## § 202b StGB: Abfangen von Daten

Wer unbefugt sich oder einem anderen unter Anwendung von technischen Mitteln nicht für ihn bestimmte Daten (§ 202a Abs. 2) aus einer nichtöffentlichen Datenübermittlung oder aus der elektromagnetischen Abstrahlung einer Datenverarbeitungsanlage verschafft, wird mit Freiheitsstrafe bis zu zwei Jahren oder mit Geldstrafe bestraft, wenn die Tat nicht in anderen Vorschriften mit schwererer Strafe bedroht ist.

# § 202c StGB: Vorbereitung des Abfangens oder Ausspähens von Daten („Hackerparagraph“)

- (1) Wer eine Straftat nach § 202a oder § 202b vorbereitet, indem er
1. Passwörter oder sonstige Sicherungscodes, die den Zugang zu Daten (§ 202a Abs. 2) ermöglichen, oder
  2. Computerprogramme, deren Zweck die Begehung einer solchen Tat ist, herstellt, sich oder einem anderen verschafft, verkauft, einem anderen überlässt, verbreitet oder sonst zugänglich macht, wird mit Freiheitsstrafe bis zu einem Jahr oder mit Geldstrafe bestraft.
- (2) § 149 Abs. 2 und 3 gilt entsprechend.  
(Vorbereitung der Fälschung von Geld und Wertzeichen;  
längere Haftstrafen)

- **Offizialdelikt**

# Diskussion von § 202c StGB

- Ist der Einsatz von IT-Sicherheitswerkzeugen generell illegal?
  - „Dual use tools“: Fast alles, was gutartig eingesetzt werden kann, kann auch missbraucht werden.
- Reaktionen:
  - Rechtsausschuss des Deutschen Bundestages: Gutwilliger Umgang mit solchen Werkzeugen durch IT-Sicherheitsexperten wird nicht von §202c erfasst.
  - Bundesjustizministerium: Unter Strafe werden nur Vorbereitungshandlungen zu *Computerstraftaten* gestellt.
- Verfahren für mehrere Selbstanzeigen wurden eingestellt bzw. abgelehnt.
- Rechtsunsicherheit besteht weiterhin
- EICAR-Empfehlung: Sorgfalt, Dokumentation, Einwilligung  
[http://www.eicar.org/files/jlussi\\_leitfaden\\_web.pdf](http://www.eicar.org/files/jlussi_leitfaden_web.pdf)

# §205 StGB: Strafantrag

(1) In den Fällen des § 201 Abs. 1 und 2 und der §§ 201a, 202, 203 und 204 wird die Tat nur auf Antrag verfolgt.  
Dies gilt auch in den Fällen der §§ 202a und 202b, es sei denn, dass die Strafverfolgungsbehörde wegen des besonderen öffentlichen Interesses an der Strafverfolgung ein Einschreiten von Amts wegen für geboten hält.

- § 202c fehlt in dieser Aufzählung; d.h. 202c ist Offizialdelikt
- „Besonderes öffentliches Interesse“ liegt im Ermessen der Staatsanwaltschaft.

## § 303a StGB: Datenveränderung

- (1) Wer rechtswidrig Daten (§ 202a Abs. 2) löscht, unterdrückt, unbrauchbar macht oder verändert, wird mit Freiheitsstrafe bis zu zwei Jahren oder mit Geldstrafe bestraft.
  
- (2) Der Versuch ist strafbar.
  
- (3) Für die Vorbereitung einer Straftat nach Absatz 1 gilt § 202c entsprechend.

## § 303b StGB: Computersabotage

(1) Wer eine Datenverarbeitung, die für einen anderen von wesentlicher Bedeutung ist, dadurch erheblich stört, dass er

1. eine Tat nach § 303a Abs. 1 begeht,
2. Daten (§ 202a Abs. 2) in der Absicht, einem anderen Nachteil zuzufügen, eingibt oder übermittelt oder
3. eine Datenverarbeitungsanlage oder einen Datenträger zerstört, beschädigt, unbrauchbar macht, beseitigt oder verändert,

wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

(2) Handelt es sich um eine Datenverarbeitung, die für einen fremden Betrieb, ein fremdes Unternehmen oder eine Behörde von wesentlicher Bedeutung ist, ist die Strafe Freiheitsstrafe bis zu fünf Jahren oder Geldstrafe.

## § 303b StGB: Computersabotage (Forts.)

(3) Der Versuch ist strafbar.

(4) In besonders schweren Fällen des Absatzes 2 ist die Strafe Freiheitsstrafe von sechs Monaten bis zu zehn Jahren. Ein besonders schwerer Fall liegt in der Regel vor, wenn der Täter

1. einen Vermögensverlust großen Ausmaßes herbeiführt,
2. gewerbsmäßig oder als Mitglied einer Bande handelt, die sich zur fortgesetzten Begehung von Computersabotage verbunden hat,
3. durch die Tat die Versorgung der Bevölkerung mit lebenswichtigen Gütern oder Dienstleistungen oder die Sicherheit der Bundesrepublik Deutschland beeinträchtigt.

(5) Für die Vorbereitung einer Straftat nach Absatz 1 gilt § 202c entsprechend.

## § 303c StGB: Strafantrag

In den Fällen der §§ 303, 303a Abs. 1 und 2 sowie § 303b Abs. 1 bis 3 wird die Tat nur auf Antrag verfolgt, es sei denn, dass die Strafverfolgungsbehörde wegen des besonderen öffentlichen Interesses an der Strafverfolgung ein Einschreiten von Amts wegen für geboten hält.



# Inhalt (2)

4. Bedrohungen, Angriffe und Gefährdungen (Forts.)
  6. Buffer Overflow
  7. Account-/Password-Cracking
  8. Hintertüren / Falltüren
  9. Rootkits
  10. Cross Site Scripting (XSS)
  11. Sniffer
  12. Port Scanner
  13. Social Engineering
5. Rechtliche Regelungen: Strafgesetzbuch
  6. „Top Cyber Security Risks“
  7. Zusammenfassung

# „Top Cyber Security Risks“

- 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

# Beispiele aus den Top 20 Internet Security Problems

## ■ 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

# Zusammenfassung

## ■ Security Engineering

- Grundlegende Zielsetzung:  
Entwicklung und Betrieb möglichst sicherer Systeme
- Kontinuierliche Verbesserung: Sicherheit als Prozess, nicht als Produkt
- Voraussetzungen:
  - Bedrohungsanalyse (Schwerpunkt dieses Kapitels)
  - Risikopriorisierung (vgl. Risikomanagement nach ISO/IEC 27000)

## ■ In den nächsten Kapiteln:

- Bewertung und Auswahl von Sicherheitsmechanismen zur Durchsetzung von Sicherheitsanforderungen
- Viele Sicherheitsmechanismen verwenden kryptographische Verfahren insb. zur Sicherstellung von Vertraulichkeit und Integrität
- Deshalb im Folgenden zuerst grundlegende Techniken der Kryptologie