Intrusion Detection und digitale Forensik am Beispiel von Rootkits

Wilhelm Dolle, Head of Networking Unit & IT-Security (wilhelm.dolle@interActive-Systems.de)
www.interActive-Systems.de/security

Agenda

- Rootkits
- Intrusion Detection
- Digitale Forensik
- Rootkits finden

Nach einem erfolgreichen Angriff

Rootkits

- Was geschieht nach einem erfolgreichen Einbruch?
  - Überwachungsmethoden analysieren
  - Verwischen von Spuren / Entdeckung vermeiden
  - Installation eines Rootkits
  - Einrichtung von Backdoors
  - Angriff auf weitere Systeme

Was sind Rootkits?

- Oft leichte Installation
- Verbergen Spuren (Dateien, Prozesse, Netzverbindungen, ...) des Einbrechers
- Bringen trojanisierte Systemprogramme und andere Tools mit („Admin Bausatz“)
- Schaffen dauerhaften Zugang zum kompromittierten System (Backdoor)
- Enthalten zusätzlich oft Keylogger, Sniffer, Logfilecleaner

- Dateibasierte/-modifizierende Rootkits
- Kernelbasierte/-modifizierende Rootkits

- Ende 80‘er: manipulieren von Logdateien
- 1989: Phrack-Magazin: Umgehen von Unix-Überwachung
- 1994: erster CERT Hinweis auf eine Sammlung von Programmen
- Erste Erwähnung von „Rootkits“ in Zusammenhang mit SunOS
- Mitte der 90‘er: LKM-Rootkits für Linux (später auch für andere Unix-Varianten)
- 1997 heroin, 1999 knark / adore
- 1999 Kernel Rootkits für Windows (NT-Rootkit)
- 2001 KIS, SucKIT manipulieren den Kernel direkt im Hauptspeicher (Technik 1998 beschrieben)

Dateibasierte Rootkits

- Ersetzen bzw. verändern Systembefehle und Überwachungsprogramme im Dateisystem
- Klassische Rootkits (z.B. lrk3, lrk4, lrk5, t0rnkit)
- Unterart: Library Rootkits
  - t0rnkit tauscht zum Beispiel libproc.so aus, die dafür verantwortlich ist, dass Prozessinformationen über das /proc-Dateisystem vom Kernel an anfragende Prozesse weitergegeben werden können

Dateibasierte Rootkits - t0rnkit

- syslogd wird angehalten
- Hash-Wert eines Passwortes (Hintertür) in */etc/ttyhash*
- Trojanisierter SSH-Daemons unter */usr/sbin/nscd*
- Start als „# Name Server Cache Daemon..“ in der Datei */etc/rc.d/rc.sysinit* (Defaultport 47017)
- Konfigurationsdateien unter */usr/info/.t0rn*
- Austausch von: *login*, *ls*, *netstat*, *ps*, *ifconfig*, *top*, *du* und *find* (Zeitstempel & Originalgrößen werden zurückgesetzt)
- *in.fingerd* öffnet eine Shell am Port 2555
- */usr/src/.puta* enthält verschiedene Binarys (unter anderem einen Sniffer)
- *telnet*, *rsh* und *finger* werden in */etc/inetd.conf* aktiviert
- *syslogd* wird wieder gestartet

- LKM oder direkte Modifikation des Kernels im Speicher

- Vorteile:
  - Privilegierter Zugriff auf das System
  - Behandlung von Netzwerkpaketen vor lokaler Firewall (stealth Backdoor)
  - Manipulation der Systemsprungtabelle oder direkt der Systemfunktionen
  - Keine Änderung von Systemprogrammen nötig

Interessante Systemfunktionen

Rootkits

- open() – lesender Zugriff Original, ausführender Zugriff trojanisierte Datei
- getdents(), mkdir(), chdir(), rmdir() – Verstecken von Verzeichnissen/Dateien
- execve(), clone(), fork() – Ausführen von Programmen mit bestimmten Eigenschaften (verstecken), und Vererbung an Kindprozesse
- stat() – Manipulation der Dateieigenschaften
- ioctl() – Device-Kontrolle, z.B. kein promisc-Bit (Sniffer) zu sehen

Rootkits die LKM benutzen

- Rootkits benutzen ladbare Kernelmodule (benötigen: insmod, lsmod, rmmod)
- Verbreitete Exemplare
  - Knark (für Kernel 2.2) – speziell zur Täuschung von Programmen wie Tripwire oder md5sum
  - Adore (für Kernel 2.2 und 2.4)
    - Module: adore.o und cleaner.o
    - Komandozeilentool: ava

- 1999 von Greg Hoglund als Proof of Concept
- Gerätetreiber _root_.sys
- Startdatei deploy.exe
- Starten: net start _root_
- Stoppen: net stop _root_

- Eigener TCP/IP-Stack
- Telnet Backdoor unter der IP 10.0.0.166 (beliebiger Port)
- Verstecken von Dateien und Prozessen
- Möglichkeit zum Umleiten von Programmaufrufen

Rootkits direkt über den Hauptspeicher

- Bringen eigene Modulloader mit oder benötigen keine LKM Unterstützung
- Greifen direkt auf den im Hauptspeicher befindlichen Kernel über /dev/kmem zu
- Technik wurde schon 1998 beschrieben
- Monolitischer Kernel hilft in einigen Fällen nicht mehr

Kernel Intrusion System (KIS)

Rootkits

- DefCon 9 / 2001: Kernel Intrusion System (KIS) von optyx
- Bringt eigenen Modullader mit (Kernel-Memory-Patching)
- Versteckte Hintertür: lauscht erst auf einem Port nachdem ein spezielles Paket (beliebiger Port) an den Rechner geschickt wurde (Stealth-Backdoor)
- Graphischer Client und Server
- Interface für Plug-ins (leicht erweiterbar)

Rootkits starten

- Modifikation des *init*-Programms
- Modifikation der Startscripte
- Modifikation von Serverprogrammen die oft beim Systemstart aufgerufen werden (*sshd*, *httpd*, ...)

Wie fällt ein kompromittiertes System auf?

Rootkits

- Intrusion Detection Systeme (host- / netzwerkbasierend)
- File System Integrity Checker

- Systemabstürze
- Erhöhter Netzwerkverkehr
- Verstoß gegen die Sicherheitsrichtlinien (nicht erlaubte Protokolle, Zugriffszeiten, ...)
- Dateisystem wird stärker genutzt
- Höhere Prozessorlast
- Geänderte Passwörter / neue Benutzer

Agenda

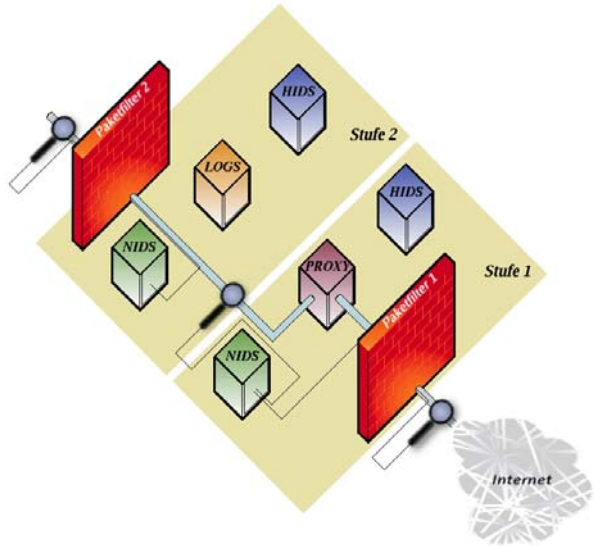
- Rootkits
- Intrusion Detection
- Digitale Forensik
- Rootkits finden

Wozu benötigen wir Intrusion Detection?

- Intrusion detection is needed in today's computing environment because it is impossible to keep pace with the current and potential threats and vulnerabilities in our computing systems. – SANS Institute ID FAQ

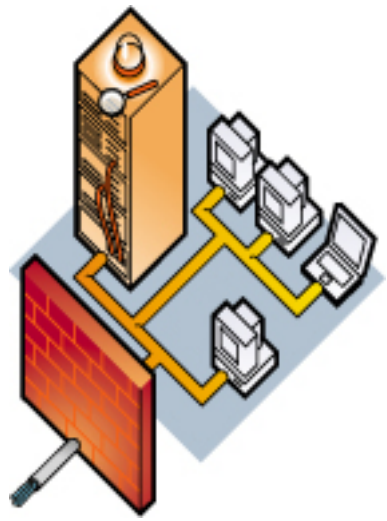
Intrusion Detection Systems

- Erkennen Angriffe und Einbrüche
- Erkennen Veränderungen an Dateien
- Erkennen verbotene Aktionen
- Erkennen unerlaubten Netzwerkverkehr



Netzwerkbasiertes IDS (NIDS)

- Überwacht ein komplettes Netzsegment
- Signaturbasierte Erkennung
- Anomaliebasierte Erkennung
- Protokollanalyse



Hostbasierte IDS (HIDS)

- Überwachen nur einen Rechner
- File System Integrity Checker
- Audit des Systems

Mögliche Fehler eines IDS

- Falsch positiv (eine erlaubte Aktion wird als Angriff identifiziert)
- Falsch negativ (ein Angriff wird vom IDS nicht erkannt oder als unbedenklich klassifiziert)
- Subversionsfehler (sehr komplexe Fehler; der Angreifer kann das IDS unterminieren)

Ursprungszustand ermitteln:

- */usr/bin/find / -type f -perm +6000 –exec /bin/ls –ail {} \; > setuidgid.original*
- */bin/ls –ailR /etc > etc.original*

Überprüfung:

- */usr/bin/find / -type f -perm +6000 –exec /bin/ls –ail {} \; | diff setuidgid.original -*
- */bin/ls –ailR /etc | diff etc.original -*

- Logdateien können eine wertvolle Hilfe sein (Angreifer versuchen meist sie zu löschen oder zu manipulieren)
- Sicherung der Protokolle auf einem weiterem Rechner
  - syslogd.conf: *.* @log_backup
  - Auf dem Rechner @log_backup den syslogd mit der Option –r (remote) starten
- Evtl. ssyslogd mit PEO-1 Protokoll (Verschlüsselung) einsetzten

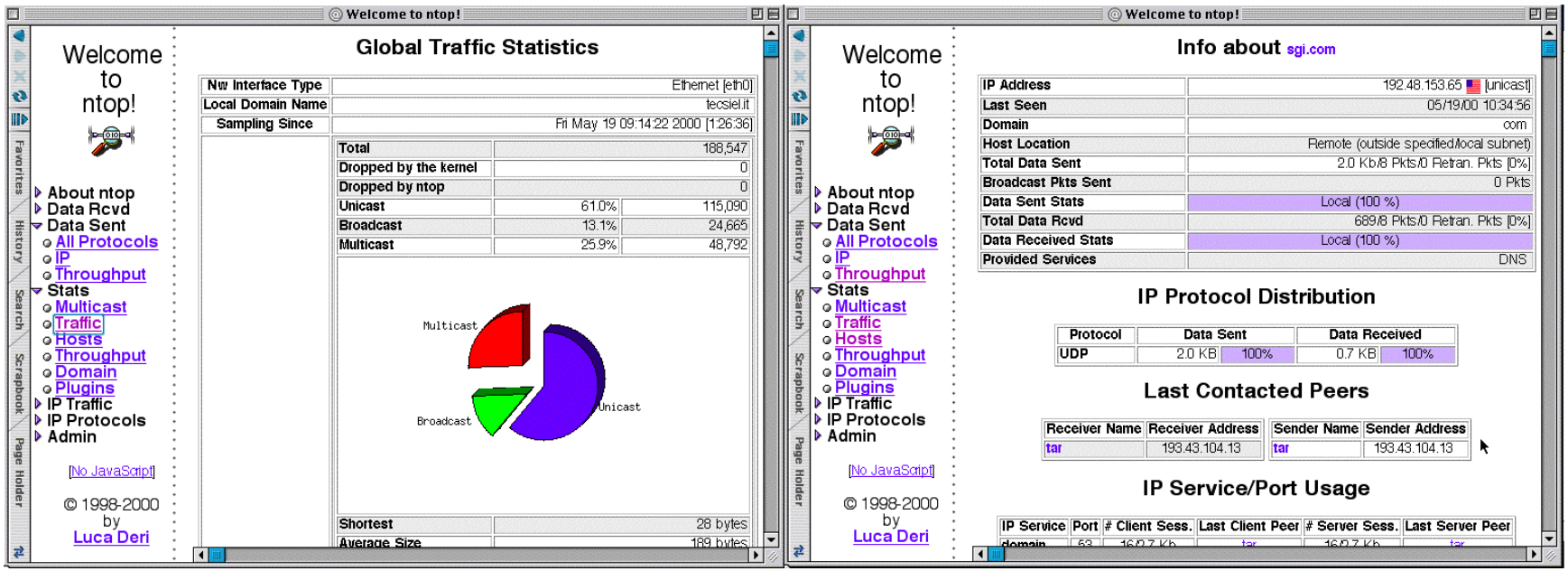
- www.tripwire.org

- Seit Oktober 2000 unter GNU GPL
- Testet die Integrität von Dateien und erkennt Manipulationen am Filesystem.
- Sowohl die Regeln, als auch die Datenbank werden kryptographisch verschlüsselt um Manipulationen zu verhindern.

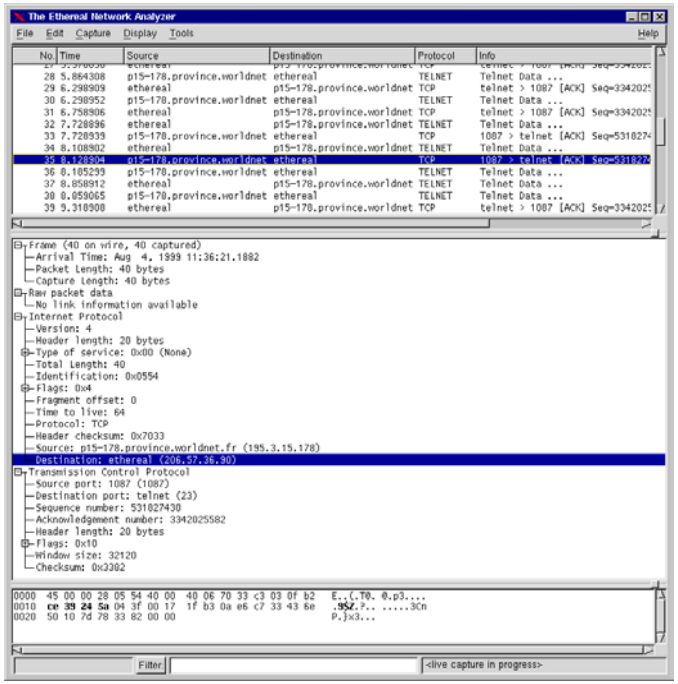
Netzbasiertes IDS – Vorarbeiten

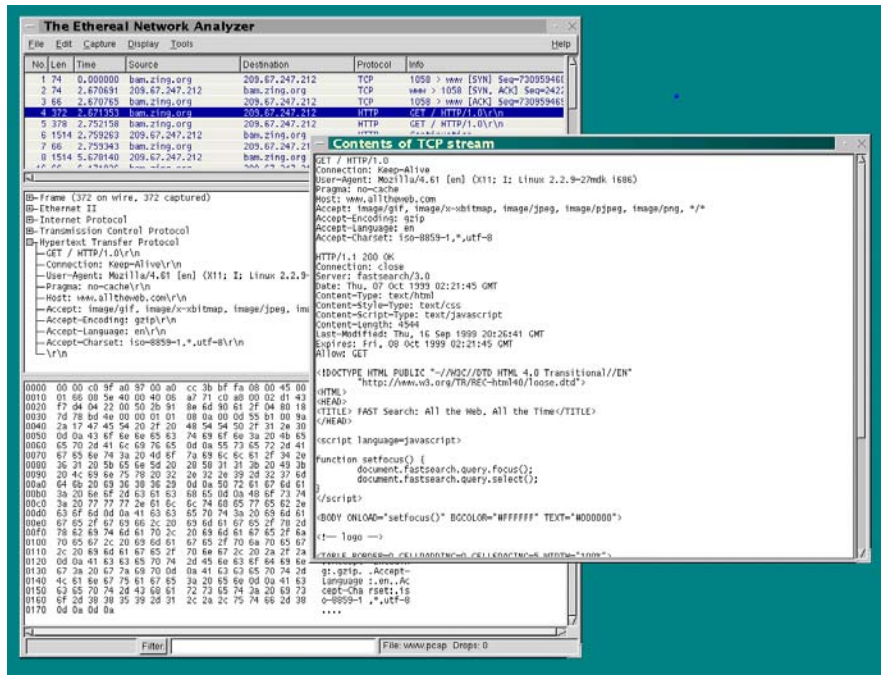
- Übersicht über den Normalzustand im Netzwerk bekommen
  - ntop
  - Ethereal
  - arpwatch – Kontrolle der MAC-Adressen im Netzwerk

- Netzwerkmonitor (www.ntop.org)
  - Konsolenausgabe
  - Eigener Webserver, Benutzerpasswörter, OpenSSL



- Network Protocol Analyzer (www.ethereal.com)
  - GUI
  - Capture Files von vielen anderen Tools analysieren (z.B. tcpdump)





Netzbasiertes IDS - Snort (1)

- www.snort.org

- Buffer Overflows
- stealth port scans
- cgi-Angriffe
- SMB und NetBIOS Tests
- Portscanner (wie nmap)
- DDoS Clients

Netzbasiertes IDS - Snort (2)

- TCP-Stream Reassemblierung (stream4 Präprozessor)
- IP-Defragmentierung (frag2 Präprozessor)
- SPADE (statistical packet anomaly detection engine)
- HTTP Präprozessor erkennt UNICODE

- Sicherheitsfeatures (chroot, User snort/snort)

- Flexible Response (SNORT kann direkt Gegenmaßnamen einleiten)

- Netzwerkverkehr wird anhand von Regeln nach bekannten Signaturen untersucht

alert TCP $EXTERNAL 80 -> $INTERNAL any (msg: "IDS215/client-netscape47-overflow-retrieved"; content: "|33 C9 B1 10 3F E9 06 51 3C FA 47 33 C0 50 F7 D0 50|"; flags: AP;)

Snort - Plazieren der Sensoren

- Sensor muss zu überwachenden Verkehr „sehen“ können
- Weitere aktive Snort-Prozesse (Sensoren nach Bedarf)
- „Vor“ einem Paketfilter (Angriffserkennung)
- „Hinter“ einem Paketfilter (Einbruchserkennung, sehr hohe Empfindlichkeit)

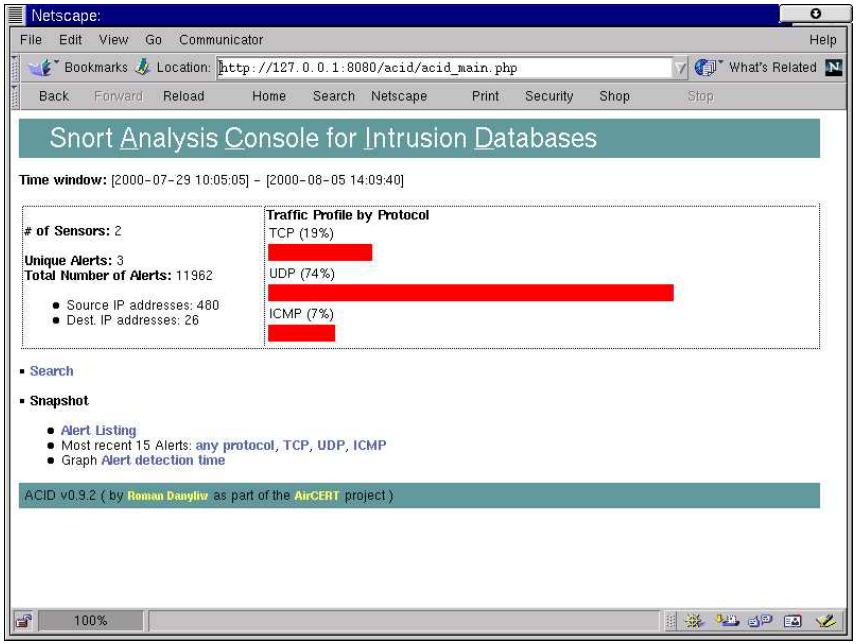
Snort - Datenbanken

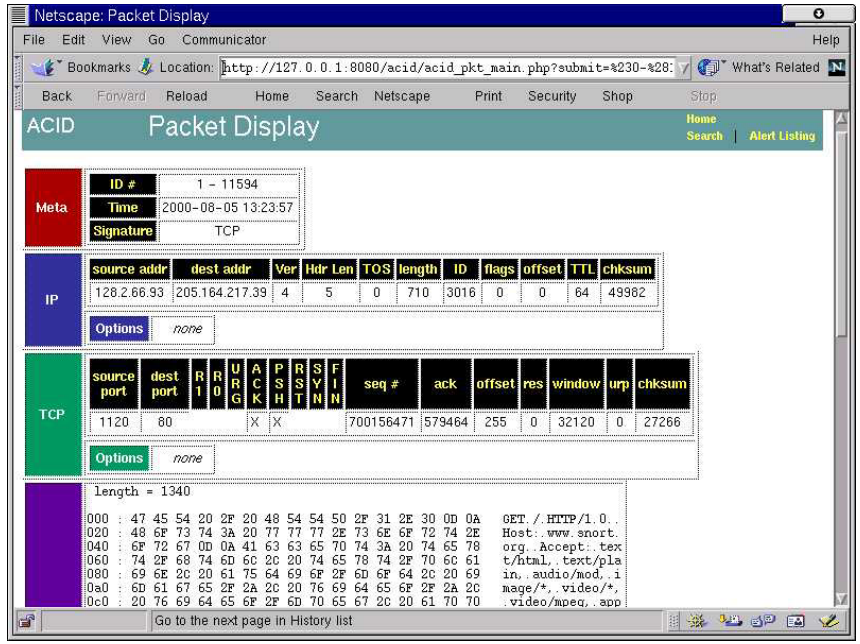
- Auswertung von „rohen“ Daten ist recht mühsam
- Loggen in lokale Dateien skaliert nicht

- Oracle, MySQL, PostgreSQL, ODBC
- Zentrale Datenbank (evtl. zusätzlich zur lokalen Datenerfassung)

- Loggen per Output-Plugin im Binär-Modus (Unified)
- Mit Barnyard Unified-Daten einlesen und an Datenbank übergeben

- Analysis Console for Intrusion Databases (ACID, www.cert.org/kb/acid)





Agenda

- Rootkits
- Intrusion Detection
- Digitale Forensik
- Rootkits finden

Grundsätze der digitalen Forensik

- Änderungen am Originalsystem soweit möglich vermeiden
- Uhrzeit und Datum notieren (Vergleich von Realzeit zu System/BIOS-Zeit)
- Änderungen dokumentieren (mit Uhrzeit)
- Hardware-Inventur des Originalsystems
- Geringe Personenzahl im Untersuchungsteam
- Zuerst alle Daten sammeln und dann erst Analyse beginnen
- Niemals mit Originaldaten arbeiten

Momentaufnahme des Systems

- Daten in der Reihenfolge ihrer Vergänglichkeit sichern:
  - Registerwerte, Cacheinhalte
  - Hauptspeicher
  - Aktueller Zustand des Netzwerkes
  - Laufende Prozesse
  - Daten auf Festplatten
  - Daten auf Disketten, CD-RW, Streamer, ...
  - Daten auf CD-R, Papier, ...

- **VOR** der Untersuchung zusammengestellte bootbare CD mit Mini-Linuxsystem (bzw. dem zu untersuchenden System)
  - Alle Programme statisch gelinkt
  - Typische Unix/Linux Programme:
    - dd, cp, cat, ls, ps, lsof, strings, find, file, bash, grep, less, vi, perl, ifconfig, kill, nc/netcat, tcpdump, arp, des, df, diff, du, last, lsmod, md5, mv, netstat, rpcinfo, showmount, top, uname, uptime, w, who, fdisk, gzip
  - Spezielle Programme zur forensischenDatensammlung und Analyse
    - TCT, TCTUtils, Autopsy, cryptcat, perl

- Keine Panik!

- System nicht abschalten (flüchtige Speicher, laufende Prozesse)
- Keinen Netzwerkstecker ziehen (aktuelle Netzverbindungen)
- Kein Backup einspielen (Analyse unmöglich, Schwachstelle nicht beseitigt, welches ist das letzte nicht kompromittierte Backup?)

- Möglichst auf eigenen Analyserechner
- Über das Netzwerk per netcat
  - Ziel: netcat –l –p 6666 >> log.txt
  - Quelle: [Daten] | netcat –w 2 [Ziel-IP] 6666
- Bei nicht vertrauenswürdigen Netzen verschlüsseln mit des ...
  - netcat –l –p 6666 | des –d –c –k [Schlüssel] >> log.txt
  - [Daten] | des –e –c –k [Schlüssel] | netcat –w 2 [Ziel-IP] 6666
- ... oder Benutzung von cryptcat

Datensammlung (Hauptspeicher)

- Ziel:
  - netcat –l –p 6666 > kmem.img
- Quelle
  - dd bs=1024 < /dev/kmem | netcat –w 2 [Ziel-IP] 6666

Datensammlung (Netzwerkverbindungen)

- Ziel:
  - netcat –l –p 6666 > netstat.txt
- Quelle
  - netstat -an | netcat –w 2 [Ziel-IP] 6666

Datensammlung (weitere Infos)

- last (Wer war zuletzt eingeloggt?)
- who (Wer ist eingeloggt?)
- w (Wer ist eingeloggt und was macht er?)
- ps (laufende Prozesse)
- lsof (Welche Applikation auf welchem Port?)
- arp (MAC-Adressen im Cache)
- netstat (Routen und Netzwerkverbindungen)
- ....

Datensammlung (Festplatteninhalt)

- Bitstream Image einer Partition

- Ziel:
  - netcat –l –p 6666 | dd of=hda1.img
- Quelle
  - dd if=/dev/hda1 | netcat –w 2 [Ziel-IP] 6666

Vorteile des Images gegenüber Backup

- File Slack (restlicher Speicherplatz bei nicht vollständig beschriebenen Clustern; kann Teile von überschriebenen Dateien enthalten)
- Unallokierte Datenblöcke (evtl. gelöschte Dateien)
- Keine Modifikation der Zugriffszeiten („MACtimes“ bei Unix Systemen)
  - M – mtime: Änderung am Inhalt
  - A – atime: letzter lesender Zugriff
  - C – ctime: Änderungen am Inode (Rechte, Eigentümer)

Prozesse ohne zugehörige Binär Datei

- Angreifer löschen oft das Binary nach dem Ausführen
- Wiederherstellung bei installiertem proc-Filesystem
  - cat /proc/[PID]/exe > [Datei]

Externe Datenquellen

- Dedizierte Logserver
- Firewall-Logs
- Router-Logs
- Intrusion Detection Systeme
  - netzwerkbasierte (z.B. Snort)
  - hostbasierte
  - System Integrity Verifier (z.B. Tripwire)

Hilfsmittel für die Analyse

- Kommerziell
  - z.B. EnCase
- Freeware
  - The Coroner‘s Toolkit (1999, Dan Farmer und Wietse Venema)
  - TCT-Utils (2001, Brian Carrier)
  - Autopsy (2001, Brian Carrier)
  - TASK (2002, Brian Carrier)

Agenda

- Rootkits
- Intrusion Detection
- Digitale Forensik
- Rootkits finden

System untersuchen (1)

Rootkits

- Wichtige Dateien überprüfen (inetd.conf / xinetd, services, Startdateien in rc.d, ...)
- Oft „trojanisierte“ Programme
  - ps, ls, find, ifconfig, netstat, du, df
  - sshd, httpd
  - login, passwd
  - inetd, tcpd
- Verdächtige MACtimes in /sbin/ oder /usr/sbin? (verdächtige Binaries mit „strings“ ansehen)
- Dateien mit Link Count kleiner 1 (lsof +L1)
- Historys ansehen (z.B. .bash_history)
- Normale Dateien im Device File System (*find /dev –type f)*

System untersuchen (2)

Rootkits

- Geladene Module prüfen
- Nach Interfaces im „promiscous mode“ suchen (Sniffer)
- Analyse der installierten Pakete mit RPM

```
# rpm --verify –all
S.5.....    /bin/ls
S.5....T    /usr/bin/named
S.5.....    /bin/netstat
```

- Logfiles auf verdächtige Einträge durchsuchen; Beispiel:

  Oct 31 16:52:33 opfer telnetd: connect from x.x.x.x

  Oct 31 16:52:34 opfer telnetd: ttloop: peer died: ...

  - => Buffer-Overflow-Angriff auf telnetd

Rootkits finden (Konzepte)

- Offene Ports (Scan von außen mit netstat vergleichen – hidden Backdoor?)
- Modulliste (wenn nicht versteckt)
- Vergleich der Systemsprungtabelle mit System.map
- Vergleich der Kernelsymboltabelle (/proc/ksyms) mit der eines „sauberen“ Kernels
- Signatur (wenn bekannt und nicht verschlüsselt) im Speicher suchen (z.B. strings /dev/kmem)
- PID-Test (versuchen alle „freien“ PIDs durch einen Testprozess zu belegen)

- www.s0ftpj.org
- Ursprünglich für Linux Kernel 2.2, inzwischen portiert auf 2.4
- Kompilieren mit sauberem Kernel (passend zum Zielsystem)
- Findet geladene Module (die zB. mit *lsmod* nicht angezeigt werden) und kann sie entfernbar machen
  - kstat –M
- Zeigt veränderte Systemcalls an
  - kstat –S

- www.chkrootkit.org
- Lokal ausführbares Script
- Erkennt nahezu alle bekannten Rootkits
- Benutzt lokal vorhandene Befehle
- Analoge Probleme wie mit *ifconfig* bei der Erkennung des promisc-Modus (besser *ip link show*)
- Evtl. Einsatz von einem schreibgeschützten Medium mit allen notwendigen (statisch gelinkten) Befehlen:

  chkrootkit –p /mnt/cdrom/bin

ChkRootKit Komponenten

- chkrootkit – lokales Script benutzt die anderen Programme

- ifpromisc – sucht nach Netzwerkkarten im Promiscuous-Modus
- chklastlog – sucht nach Löschvorgängen in lastlog
- chkwtmp – sucht nach Löschvorgängen in wtmp
- check_wtmpx – sucht auf Solarissystemen nach Löschvorgängen in wtmpx
- chkproc – sucht nach Anzeichen für trojanisierte LKMs
- chkdirs – sucht nach Anzeichen für trojanisierte LKMs
- strings – simple und schnelle Implementation des strings-Kommando

ChkRootKit Ausgabe

```
# ./chkrootkit
...
Checking ‘su’ ... not infected
Checking ‘ifconfig’ ... INFECTED
...
Checking ‘identd’ ... not found
Checking ‘init’ ... INFECTED
...
Searching for Adore Worm... nothing found
...
Checking ‘sniffer’ ...
eth0 is not promisc
...
```

Entfernen von Rootkits

- Backup einspielen erst nach digitaler Forensik (wann war das letzte „saubere“ Backup?)

- Manipulierte Dateien durch Originale ersetzen
- Module entfernen
- Startscripte säubern / Startpunkte des Rootkits entfernen

- Wenn möglich: System komplett neu aufsetzen!

Proaktive Schutzmaßnamen (1)

Rootkits

- Gut ausgebildete Systembetreuer (sowie ausreichend Zeit)
- Verhindern des Ausnutzens von Buffer-Overflows und Format-String Angriffen
- Auditing von in den Kernel eingefügten Modulen (Modifikation von insmod)

Proaktive Schutzmaßnamen (2)

Rootkits

- Problem der Unix-Architektur: der allmächtige Administrator *root*

- LIDS (Linux Intrusion Detection System)
- NSA Security-Enhanced Linux
- GrSecurity
- RSBAC (Rule Set Based Access Control)

- LIDS (www.lids.org) ist ein Patch für den Kernel
- Es werden Features implementiert, die Linux im Vergleich zu anderen OS fehlen
  - Schutz/Kontrolle/Verstecken von Dateien (z.B. /etc/passwd), Geräten (z.B. /dev/hda), Speicher (/dev/kmem) und Prozessen(z.B. tripwire) (auch vor root!)
  - Integrierter Portscan-Detector
  - ACLs für User, Einschränkungen für root (z. B. keine Module laden, kein Neustart)
- Administration erst nach RIPE-MD Kennwort oder Neustart

Fazit

- Dateibasierte/-modifizierende Rootkits
- Kernelbasierte/-modifizierende Rootkits

- Intrusion Detection
- Digitale Forensik

- Rootkits finden (Konzepte, kstat, chkrootkit)
- Proaktive Schutzmaßnamen

- Upcoming Events ;-)
  - Auto Rooter
  - Sniffer Backdoors