Angriffe auf Computersysteme

Angriffsmöglichkeiten, Schutz vor Angriffen und die Analyse erfolgreicher Einbrüche

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

Rechtlicher Hinweis

- Die vorgestellten Programme sind nicht illegal
- Besitz in Deutschland nicht strafbar
- Einsatz gegen fremde Systeme kann aber belangt werden (auch / oder gerade am Arbeitsplatz)
- Vertragsbedingungen von Providern (z.B. Portscans sind Mitgliedern verboten)

- Zugangskontrolldiensteschutzgesetz (ZKDSG)

- Netzwerk Scan Techniken
- Systemschwachstellen ausnutzen
- Aufbau von Sicherheitssystemen
- Digitale Forensik

Typischer Einbruchszyklus

- Vergleich: Einbruch in eine Bank

- Beschaffen von Informationen
  - Öffentliche Infos (DNS-Einträge, Webseiten des Ziels, ...)
  - Aktive Verfahren (Scans)
  - Passive Verfahren (Sniffing)
- Einbruch in das System
- Verstecken und Festsetzen
- Missbrauch des Systems

- Host Mapping (IP-Adresse aktiv?)
- Protokoll Scans (Welche Protokolle werden angeboten?)
- Port Scans (Welche Dienste werden angeboten?)
- OS Detection
- Vulnerability Scans (Banner Grabbing, RPC Queries, ...)

- Interessant nur im lokalen Netz
  - Physical Layer (1)
  - Data Link Layer (2)
- Interessant für Internet-Verbindungen
  - Network Layer (3)
  - Transport Layer (4)
  - Applikation Layer (5)

- Physical Layer
  - Protokolle: Ethernet (MAC-Adresse)
  - Werkzeuge: Sniffer (tcpdump, ethereal, ...)

- Data Link Layer
  - Protokolle: ARP (Zuordnung IP- zu MAC-Adresse), RARP, BOOTP
  - Werkzeuge: Sniffer (s.o.) und zusätzlich Paketinjektion (Ettercap, Nemesis, ...)

- Network Layer
  - Protokolle: IP, ICMP, IGMP (IP-Multicasting), RIP (dynamisches Routing)
  - Werkzeuge: Scanner, Sniffer, Paketinjektion
- Transport Layer
  - Protokolle: TCP, UDP
  - Werkzeuge: Scanner, Sniffer, ...
- Applikation Layer
  - Protokolle: FTP, SMTP, DNS, HTTP, ...
  - Werkzeuge: Schwachstellenscanner, ...

- Scan Techniken und Werkzeuge unter Linux
  - ICMP Scans
  - TCP Scans
  - UDP Scans
- Schwachstellenscanner

Wichtige Begriffe

- False Positive (Protokoll, Dienst, Port)
- False Negative (Protokoll, Dienst, Port)

- Spoofing (IP-Datagramm)

- Slow Scan
- Stealth Scan

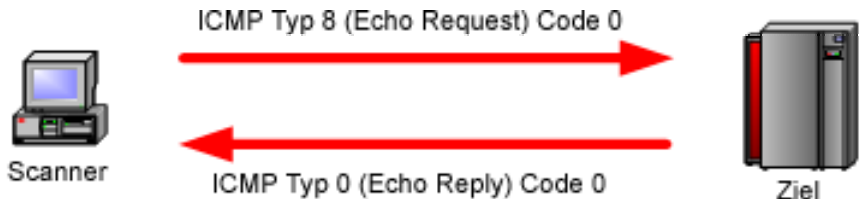
- Decoys („Ablenker“)

- Definition in RFC 792, Ergänzungen in RFC 1122 und 1812
- Protokoll zur Fehlermeldung an andere Netzgeräte (ICMP ***Error*** Messages)
- Austausch von Informationen in Netzwerken durch ICMP ***Query*** und ***Reply*** Messages
- Fehler bei der Bearbeitung von ICMP-Fehlermeldungen erzeugen niemals neue ICMP-Fehlermeldungen

- Programmiertechnisch auf Raw-Sockets

ICMP Scans

- Echo Request Scans (Host aktiv?)
  - ICMP Typ: 8 (Echo Request) Code: 0
  - ICMP Typ: 0 (Echo Reply) Code: 0
  - Tools: ping, fping, nmap



- Weitere Scans möglich: ICMP Broadcasts, ...

Host Fingerprinting mit ICMP (Beispiele)

- ICMP Response Rate
  - UDP/TCP Pakete an geschlossene Ports -> ICMP Destination Unreachable (Type 3, Code 3 (Port Unreachable)), die Rate gibt Hinweis auf OS
- Datenmenge bei ICMP Errors
  - 8 oder mehr Byte aus auslösendem Paket
- Antwort auf ICMP Query Broadcasts
- Auswertung des IP TTL / IP ID Feldes

- Tools: icmpenum, X/xprobe, nmap

xprobe gegen Red Hat 7.3 (Kernel 2.4.18)

Angriffe auf Computersysteme: Netzwerk Scan Techniken

```
[root@fls_nb root]# uname -a
Linux fls_nb.ias-berlin 2.4.18 #2 Thu Apr 25 14:42:24 CEST 2002 i686 unknown
[root@fls_nb root]# xprobe -v -i lo localhost
X probe ver. 0.0.2
-----------------
Interface: lo/127.0.0.1

LOG: Target: 127.0.0.1
LOG: Netmask: 255.255.255.255
LOG: probing: 127.0.0.1
LOG: [send]-> UDP to 127.0.0.1:32132
LOG: [98 bytes] sent, waiting for response.
TREE: Cisco IOS 11.x-12.x! Extreme Network Switches.Linux
2.0.x!2.2.x!2.4.x.
TREE: Linux kernel 2.0.x!2.2.x!2.4.x! Based.
TREE: Linux kernel 2.2.x!2.4.x! Based.
LOG: [send]-> ICMP echo request to 127.0.0.1
LOG: [68 bytes] sent, waiting for response.
TREE: ICMP echo/echo reply are not filtered
FINAL:[ Linux 2.2.x/2.4.5+ kernel ]
```

- Open Source Systeme wie Linux ändern ihr Verhalten oft innerhalb weniger Patch Releases
- Linux erlaubt die Änderung vieler Parameter unter /proc/sys/net/ipv4/
- Beispiele für den 2.4er Kernel
  - icmp_echo_ignore_broadcasts
  - icmp_echoreply_rate
  - ip_default_ttl
- Patch für 2.2er Kernel (www.innu.org/~sean/) der Modifikationen am TCP/IP Stack erlaubt

- Definition in RFC 793, Ergänzungen in RFC 1122 und 1213
- Überprüft empfangene Pakete auf Verfälschung
- Wiederholt verlorene Pakete
- Sortiert die Pakete in die richtige Reihenfolge
- Bremst Sender bei Netzüberlastung

- Wichtige Zustände einer Verbindung: LISTEN, ESTABLISHED und CLOSED
- Three Way Handshake: SYN, SYN/ACK, ACK

TCP Connect Scan (1)

- Versuch eines „normalen Verbindungsaufbaus“
  - SYN senden
  - Ziel: SYN / ACK (Port offen) oder RST (Port geschlossen)
  - ACK senden

- Paketfilter
  - DROP/DENY -> Timeout
  - ICMP Dest. Unreachble
  - RST -> False Negative

TCP Connect Scan (2)

- Leicht zu implementieren durch Std. Calls (socket(), connect(), ...)
- Keine Root Rechte nötig, da der TCP/IP Stack normal genutzt und nicht umgangen wird
- False Negative nur bei Paketfiltern
- Keine False Positive
- Schnell

- Leicht zu entdecken in Logfiles oder durch Intrusion Detection Systeme

TCP SYN Scan (half open)

- Verbindung wird nicht vollständig aufgebaut
  - SYN senden
  - Ziel: SYN / ACK (Port offen) oder RST (Port geschlossen)
  - RST senden
- Zustand der Verbindung geht nicht über das halboffene Stadium (Zustand SYN-SEND) hinaus
- Muss am TCP-Stack des Betriebssystems vorbei implementiert werden

TCP SYN-ACK Scan (1)

- Offener Server-Port antwortet nicht auf SYN / ACK
- Bei geschlossenem Port wird ein Überbleibsel einer schon geschlossenen oder abgebrochenen Verbindung vermutet und mit RST beantwortet

- Vorgehen
  - SYN / ACK senden
  - Ziel: Paket wird verworfen (Port offen) oder mit RST beantwortet (Port geschlossen)

TCP SYN-ACK Scan (2)

- Kein definierter TCP Zustand (Stealth Scan)
- Ergebnis invers zu Connect und SYN-Scan
  - False Positive bei Paketverlust oder DROP / DENY
  - False Negative bei RST von Paketfilter
- Manche OS schicken auch für offene Ports RST (Cisco, Windows, ...)

Weitere TCP Scans

- FIN Scan
  - Scanner schickt Paket mit FIN Flag; Ziel gibt RST bei geschlossenem oder nichts bei offenem Port zurück
- XMAS Scan
  - Alle Flags gesetzt (mindestens aber FIN, PSH und URG)
- Null Scan
  - Kein Flag gesetzt

- Reaktion auf FIN Pakete auf offenen Port (korrekt: keine Antwort)
- Reaktion auf RST Pakete auf geschlossene Ports
- TCP Optionen: Unterstützung, Reihenfolge
- Wert des Acknowledgement Number Fields im ersten Paket (SN oder SN+1)
- Muster bei der Generierung der Initial Sequence Number (ISN)
- Größe des Window Size Fields

nmap gegen Red Hat 7.3 (Kernel 2.4.18)

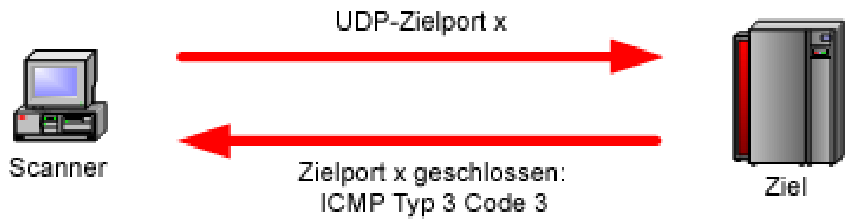
```
[root@fls_nb root]# uname -a
Linux fls_nb.ias-berlin 2.4.18 #2 Thu Apr 25 14:42:24 CEST 2002 i686 unknown

[root@fls_nb root]# nmap -sT -vv -O localhost

Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )
[...]
Remote operating system guess: Linux Kernel 2.4.0 - 2.4.17 (X86)
OS Fingerprint:
TSeq(Class=RI%gcd=1%SI=398B3E%IPID=Z%TS=100HZ)
T1(Resp=Y%DF=Y%W=7FFF%ACK=S++%Flags=AS%Ops=MNNTNW)
T2(Resp=N)
T3(Resp=Y%DF=Y%W=7FFF%ACK=S++%Flags=AS%Ops=MNNTNW)
T4(Resp=Y%DF=Y%W=0%ACK=O%Flags=R%Ops=)
T5(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=Y%DF=Y%W=0%ACK=O%Flags=R%Ops=)
T7(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
PU(Resp=Y%DF=N%TOS=C0%IPLEN=164%RIPTL=148%RID=E%RIPCK=E%UCK=E%UL
    EN=134%DAT=E)
```

User Datagram Protocol (UDP)

- Definition in RFC 768
- Überprüft empfangene Pakete auf Verfälschung
- Verbindungs- und zustandslos
- Unsicher (prüft nicht ob die Pakete auch ankommen)

UDP Scans (1)

- UDP Paket wird an einen bestimmten Port geschickt
  - Zielport geschlossen: ICMP Destination Unreachable, Port Unreachable (Type 3, Code 3)
  - Zielport offen: Paket wird verworfen (oder mit UDP Paket beantwortet)



UDP Scans (2)

- Große Gefahr für False Positive
  - Paketverlust, DROP / DENY durch Paketfilter
- Recht langsam im Vergleich zu TCP Scans durch ICMP Rate Limits

- Definition eines Port Scans ist schwierig
  - Einfachster Fall: „Mehrere Pakete, welche innerhalb einer kurzen Zeitperiode an verschiedene Ports gerichtet sind und von derselben Ursprungsadresse gesendet wurden“

- Scanlogd von Solar Designer (www.openwall.com/scanlogd/)
- PortSentry (www.psionic.com/products/portsentry.html)

Schwachstellenscanner (Historisch)

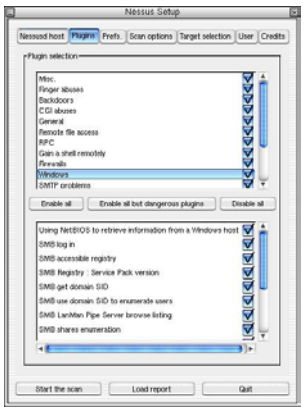
- Der Oldie: SATAN (Security Administrator Tool for Analyzing Networks)
  - 1995 von Wietse Venema und Dan Farmer veröffentlicht
  - Verschiedene Tools vereint unter einer leicht zu bedienenden Oberfläche (Webbrowser)
  - Spaltete die Security-Szene

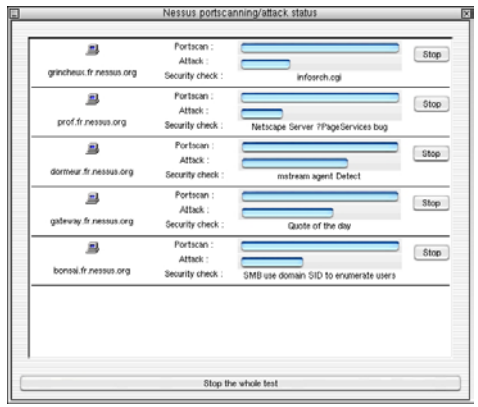
Schwachstellenscanner (aktuell)

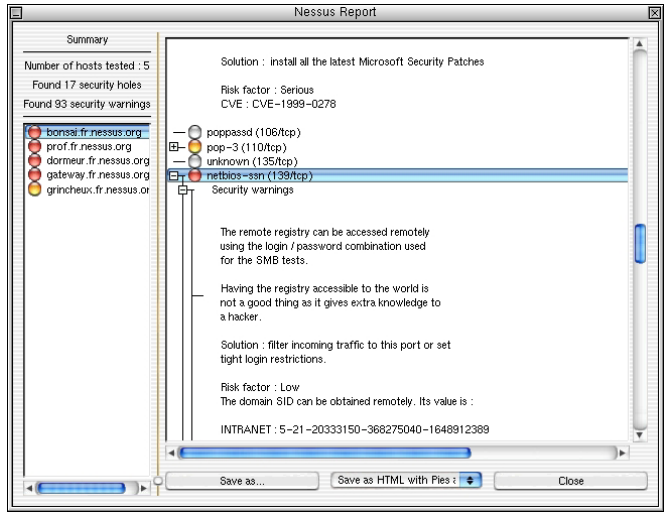
- Kommerziell:
  - ISS (Internet Security Systems)
  - SAINT (Security Administrator's Integrated Network Tool) - erst ab Version 3.4 kommerziell
- Frei:
  - Nessus
  - SARA (Security Auditor's Research Assistant)

Schwachstellenscanner (Nessus)

- Plug-in Architektur
- NASL (Nessus Attack Scripting Language)
- Umfangreiche, täglich aktualisierte Datenbank
- Client-Server Architektur (Server scannt Ziel, Client dient als Frontend)
- Sehr umfangreiche Reports







Reaktionen auf Scans

- Logdaten aufheben (Beweise sichern)
- Früherkennung von neuen Sicherheitslücken bei Zunahme spezifischer Scans
- Kompromittierte Hosts (auch im eigenen Netz!) erkennen (die meisten Scans benötigen Root Rechte)

- (Automatische Reaktion?)
- (Scannen der Scanquelle?)

- Resorcenverbrauch vs. Sicherheitsgewinn?

- Netzwerk Scan Techniken
- Systemschwachstellen ausnutzen
- Aufbau von Sicherheitssystemen
- Digitale Forensik

Angriffsmöglichkeiten

Angriffe auf Computersysteme: Systemschwachstellen ausnutzen

- unsichere Passwörter
- unsichere Protokolle
- fehlerhafte Programme / Dienste
  - Buffer Overflow Angriffe
  - Format String Angriffe
- unvorsichtige Benutzer (!)

SANS / FBI Top 10 List Unix (2002)

Angriffe auf Computersysteme: Systemschwachstellen ausnutzen

- Remote Procedure Calls (RPC)
- Apache Web Server
- Secure Shell (SSH)
- Simple Network Management Protocol (SNMP)
- File Transfer Protocol (FTP)
- R-Services - Vertrauensbeziehungen
- Line Printer Daemon (LPD)
- Sendmail
- BIND / DNS
- Accounts ohne oder mit schwachen Passwörtern

Buffer-Overflow Schwachstellen

Angriffe auf Computersysteme: Systemschwachstellen ausnutzen

- Häufigste Sicherheitslücke unter Unix/Linux
- Programmierfehler (meist C)
- Überschreiben von Speichergrenzen
- Ermöglicht durch Einsatz von Funktionen die nicht auf Bereichsüberschreitung testen (z.B. strcpy())
- Absturz des Programms bei ungezieltem Überschreiben
- Bei geeigneter Manipulation Ausführen von speziellen Befehlen (Exploit)
- Erste Angriffe dieser Art wahrscheinlich schon 1988

Format-String Schwachstellen (1)

- In C Format der Ausgabe bei printf() und scanf() durch Format-Strings bestimmbar
- Beispiel: printf(„Name = %s“, name)
  - Format-String: „Name = %s“
  - Formatangabe: %s
  - Argument: name
- Existiert kein Argument für eine Formatangabe, und kann ein Angreifer die Zeichenkette beeinflussen, wird diese Schwachstelle ausnutzbar

Format-String Schwachstellen (2)

- Ausnutzen von: printf(text)
- Angreifer setzt: text=„Datei: %x %s %n“
- Es existieren keine Argumente für die Formatangaben
- Stattdessen wird bei der Ausgabe jeweils der oberste Wert auf dem Stack benutzt (hier ausgegeben) – z.B. Inhalt von lokalen Variablen oder Zeiger
- Manipulation und Ausnutzen (Exploit) wie bei Buffer-Overflows möglich

- Quelltext eines Programms manuell oder automatisch prüfen

- Ausnutzen der Schwachstellen auf Betriebssystemebene verhindern
- Ausnutzen der Schwachstellen innerhalb der Laufzeitumgebung des Programms verhindern

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

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

- Klassische Rootkits (z.B. lrk3, lrk4, lrk5, t0rnkit)
- Ersetzen beziehungsweise verändern Systembefehle und Sicherheitsprogramme

- Weit verbreitetes Rootkit für Unix und Linux
- Ersetzt unter anderem die Programme du, find, ifconfig, ls, netstat, ps, top und login
- Über Konfigurationsdatei anpassbar (z. B. /dev/.hidden/psconf)
- login: bei bestimmten IP-Adressen root-Zugang ohne Passwort und Logging möglich
- Trojanisierte Programme alle 31.336 Bytes groß
- In Standardinstallation TCP-Port 47017 offen

- LKM oder direkte Modifikation des Kernels im Speicher
- Vorteile:
  - Privilegierter Zugriff auf das System
  - Behandlung von Netzwerkpaketen vor lokaler Firewall
  - Manipulation der Systemsprungtabelle oder direkt der Systemfunktionen
  - Keine Änderung von Systemprogrammen nötig

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

Rootkits die LKM benutzen

- Erstes Auftauchen vor etwa 5 Jahren
- Rootkits benutzen ladbare Kernelmodule (benötigen: insmod, lsmod, rmmod)

- Verbreitete Exemplare
  - Knark (für Kernel 2.2)
  - Adore (für Kernel 2.2 und 2.4)

- 2001: KIS
- 2002: SucKIT

- Benötigen keine LKM Unterstützung
- Greifen direkt auf den im Hauptspeicher befindlichen Kernel über /dev/kmem zu

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

- Modifikation der Startscripte
- Austausch von Serverprogrammen (sshd, httpd, ...)

Rootkits finden (Konzepte)

- Offene Ports (Scan von außen mit netstat vergleichen – hidden Backdoor?)
- Modulliste (wenn nicht versteckt)
- Vergleich der Systemsprungtabelle mit System.map
- 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)

- Netzwerk Scan Techniken
- Systemschwachstellen ausnutzen
- Aufbau von Sicherheitssystemen
- Digitale Forensik

Was möchte ich schützen?

- Daten (Dokumente, Emails, ...)
- Ressourcen (Speicherplatz, Rechenzeit, Netzbandbreite, ...)
- Reputation

- Vertraulichkeit, Integrität, Verfügbarkeit und Authentizität

Gegen wen oder was schütze ich?

- Intrusion (das Eindringen in ein System selber, evtl. Datendiebstahl, ...)
- Denial of Service – DoS (Zugang zu Diensten eines Systems verhindern)
- Illegale Benutzung der Ressourcen (unter anderem als Plattform für weitere Einbrüche oder zum Verbreiten von Spam-Mail)

- Abschätzung Aufwand gegen Nutzen

Sicherheit ist ein kontinuierlicher Prozess, der eine ständige Überwachung und Verfeinerung benötigt um zu funktionieren.

Allgemein unterteilt man den Prozess in drei Phasen:

- Protection Phase
- Detection Phase
- Response Phase

Protection Phase

- Erstellen von Sicherheitsrichtlinien
- Authentisierung der Benutzer
- Zugriffskontrollen
- Filtern von Inhalten
- Verschlüsselung
- Sicherheitssysteme
- Benutzerschulungen

- Netz- und Hostbasierte Intrusion Detection Systeme
- Netzwerk- und Hostüberwachung
- Auditing (Logfiles)

- Erkenntnisse in neue (verfeinerte) Mechanismen einfließen lassen
- Richtlinien anpassen

Ein gutes Sicherheitssystem (Firewall) sollte mehrere Abwehrreihen beinhalten (Angriffe werden verhindert und/oder erkannt):

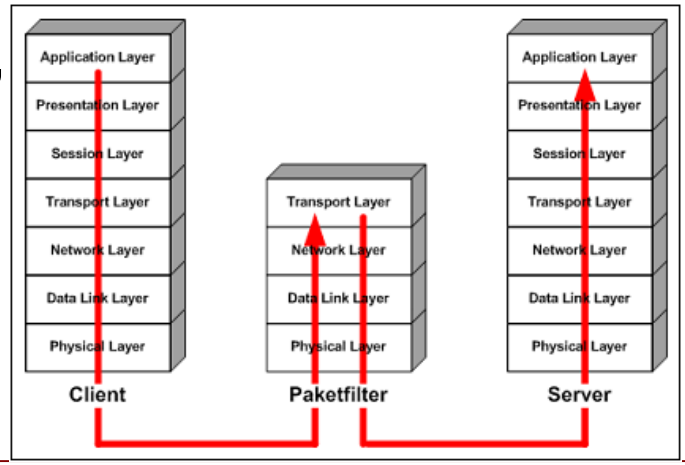
- Paketfilter und Proxys
- Hostbasierte IDS / IRS
- Netzbasierte IDS / IRS
- Sicherheitsmechanismen auf Kernelebene

Definition einer Firewall

- organisatorisches und technisches Konzept zur Trennung von Netzbereichen, dessen korrekte Umsetzung und dauerhafte Pflege

- typische Umsetzung
  - je ein Paketfilter zwischen zwei Netzen
  - dazwischen liegt ein Grenznetz (DMZ)
  - Paketfilter lassen nur Daten vom direkt angebundenen Netz in die DMZ und zurück durch
  - direkte Verbindung aus einem Netz zu dem Paketfilter des anderen Netzes oder gar ins andere Netz ist verboten

- Routing von Paketen zwischen Netzen
- Reine Paketfilter arbeiten auf den Schichten 3 (Network Layer) und 4 (Transport Layer) des OSI Modells
- Reine Paketfilter sehen keine Applikationsdaten
- Pakete weiterleiten, verwerfen, zurückweisen, modifizieren oder loggen nach Kriterien wie IP Quell-/Zieladresse, Protokoll, TCP/UDP Quell-/Zielport, ICMP-Typ, Fragmentierung, Zustand der Verbindung

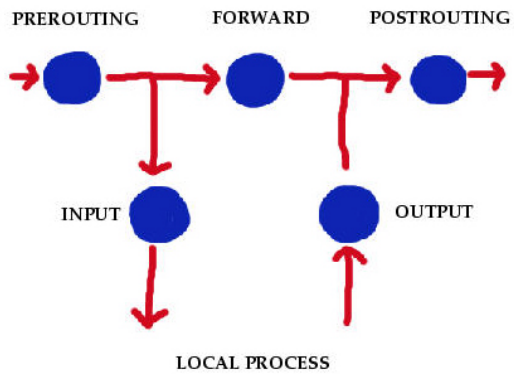


- Userspace Tool zum Filtern von Paketen im Kernel
- Ersetzt Ipchains
- Modularer Aufbau
- Benötigt Kernel mit Netfilter-Unterstützung
- Informationen unter netfilter.samba.org

- Große Menge an neuem Code (alle Fehler schon gefunden?)

Iptables - Features

- Paketfilter
- Connection tracking / stateful packet filtering
- Network address translation (NAT)
- Packet mangling
- Einfache Bandbreitenbeschränkung
- Limitierungen
- Erweitertes Logging möglich
- Erweitertes REJECT
- Übersichtlicher Paketfluss



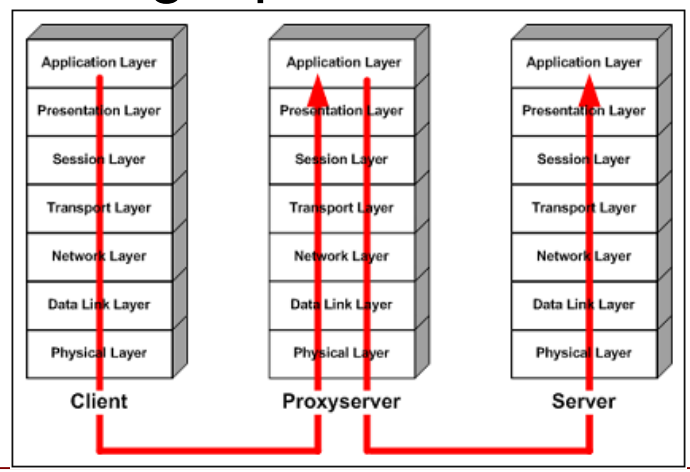
Iptables – REJECT oder DROP/DENY?

- REJECT: aktive Ablehnung einer Verbindungsanfrage durch ICMP oder TCP-Reset
- DROP/DENY: kommentarloses Wegwerfen der Verbindungsanfrage (Timeout für Anfragenden)

- REJECT besser (z.B. schnellere Fehlersuche), DROP/DENY kostet nur Zeit (z.B. ident)
- „Verstecken“ des Rechners durch DROP/DENY weder sinnvoll noch möglich

Proxys (Application Level Gateways)

- Proxys arbeiten auf den Schichten 5 bis 7 des OSI-Modells und können daher Applikationsdaten berücksichtigen
- Erweiterte Filtermöglichkeiten im Vergleich zu einem reinen Paketfilter
- Logische Trennung der Clients vom Netz (und damit von den Servern die angesprochen werden)
- Beispiele: TIS Firewall Toolkit, Squid, ...



- Erkennt (detects) Einbrüche (Intrusions)
- Erkennt Veränderungen wichtiger Dateien
- Erkennt die Installation von Hintertüren
- Erkennt verbotene Aktionen im Logfile
- Erkennt unerlaubten Netzwerkverkehr

- Ermöglicht (automatische) Reaktionen

Charakteristika eines guten IDS

- Läuft kontinuierlich ohne Überwachung
- Es sollte fehlertolerant sein
- Es sollte resistent gegen Eingriffe sein
- Sollte minimalen Overhead produzieren
- Soll Abweichungen von der Norm erkennen
- Sollte leicht in das Netz integrierbar sein
- Es sollte schwer zu täuschen sein

Mögliche Fehler eines IDS

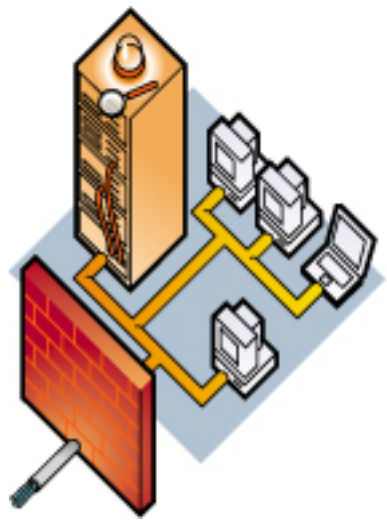
- Falsch positiv (eine erlaubte Aktion wird als Angriff identifiziert)
- Falsch negativ (ein Angriff wird vom IDS nicht erkannt oder als unbedenklich klassifiziert)
- Subversionsfehler (sehr komplexe Fehler; der Angreifer kann das IDS unterminieren)

Hostbasierte IDS überwachen nur einen einzigen Rechner

- Beobachtet systemrelevante Dateien und Befehle
- Meldet ungewöhnliche Dateien und Admistrationsvorgänge
- Beobachtet offene Ports und Netzwerkverbindungen

Netzbasierte IDS sammeln Informationen aus einem ganzen Netzsegment

- Untersucht und protokolliert (bei Bedarf) sämtliche Netzwerkpakete
- Erkennt mögliche Angriffe an „Fingerabdrücken“
- Unterzieht die protokollierten Daten statistischen Analysen (Anomalieerkennung)



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 -*

syslogd

- 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

xinetd statt inetd (1)

- Erweiterte Protokollfunktionen
  - Uhrzeit, Dauer
  - fehlerhafte Zugriffe werden protokolliert
  - anfragender Client und Benutzer
- Erweiterte Zugriffskontrolle
- Bindung einzelner Dienste an spezielle IP-Adressen
- Schutz gegen einfache Denial-of-Service-Attacken

xinetd statt inetd (2)

- Konfiguration nicht mit inetd kompatibel (einzelne Dateien für jeden Service)

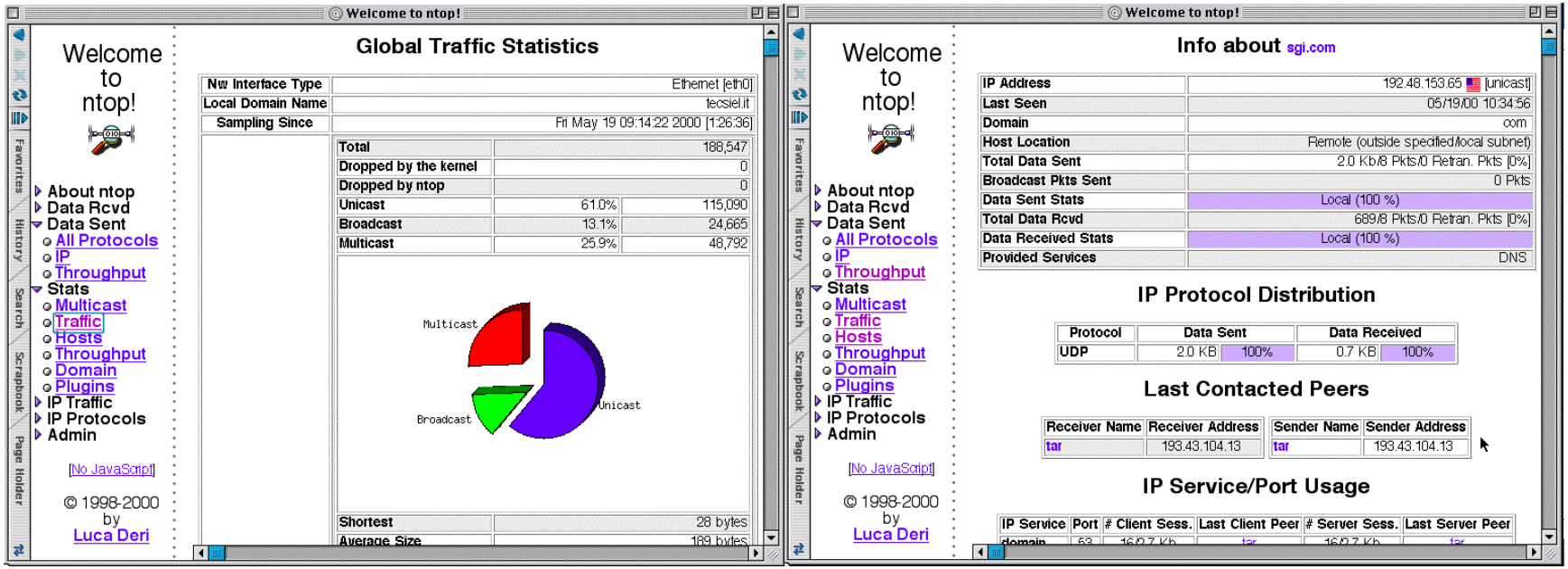
```
service telnet
{
log_on_success          = HOST PID DURATION USERID
log_on_failure          = HOST PID USERID
no_access               = 192.168.100.35
socket_type             = stream
protocol                = tcp
port                    = 23
log_type                = SYSLOGD daemon
max_load                = 2.5
access_times            = 10:00-22:00
instances               = 5
}
```

- www.tripwire.org

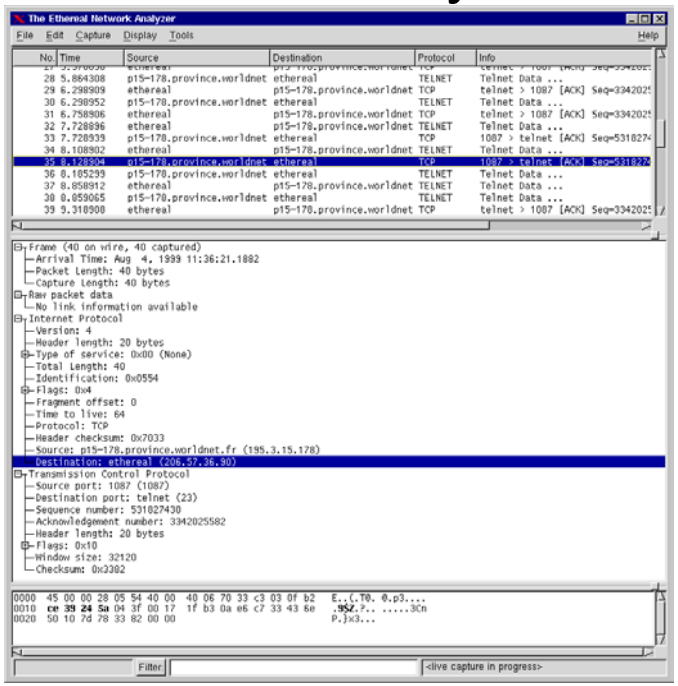
- Seit Oktober 2000 unter GNU GPL
- Testet die Integrität von Dateien und erkennt Manipulationen am Filesystem.
- Sowohl die Regeln, als auch die Datenbank werden kryptographisch verschlüsselt um Manipulationen zu verhindern.

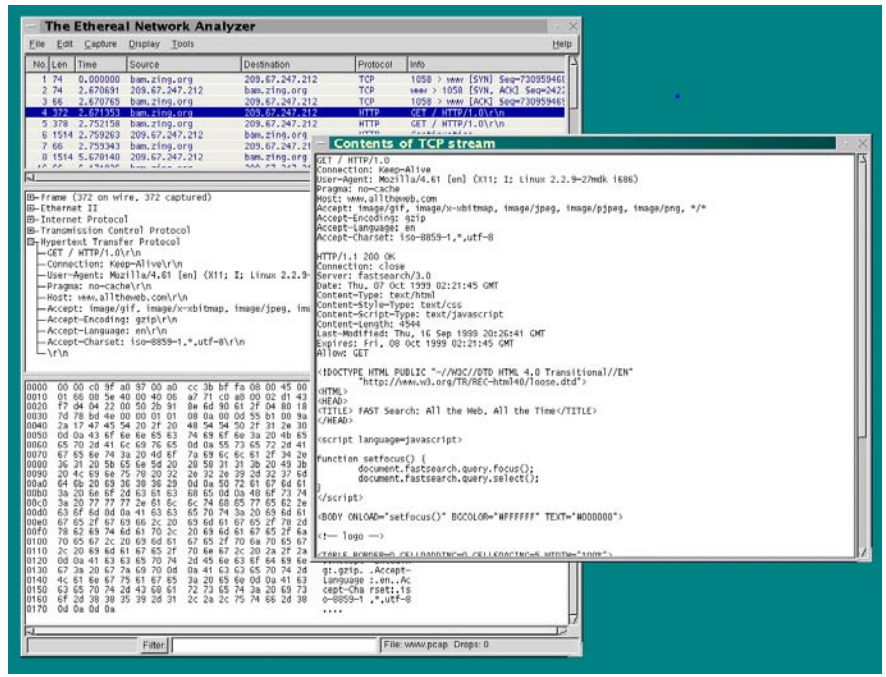
- Übersicht über den Normalzustand im Netzwerk bekommen
  - ntop
  - Ethereal
  - arpwatch – Kontrolle der MAC-Adressen im Netzwerk

- 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

- TCP-Stream Reassemblierung
- IP-Defragmentierung (ab Version 1.7)
- SPADE (statistical packet anomaly detection engine – ab Version 1.7)
- HTTP Präprozessor erkennt UNICODE
- Sicherheitsfeatures (chroot, User snort/snort)

- Flexible Response (SNORT kann direkt Gegenmaßnamen einleiten)

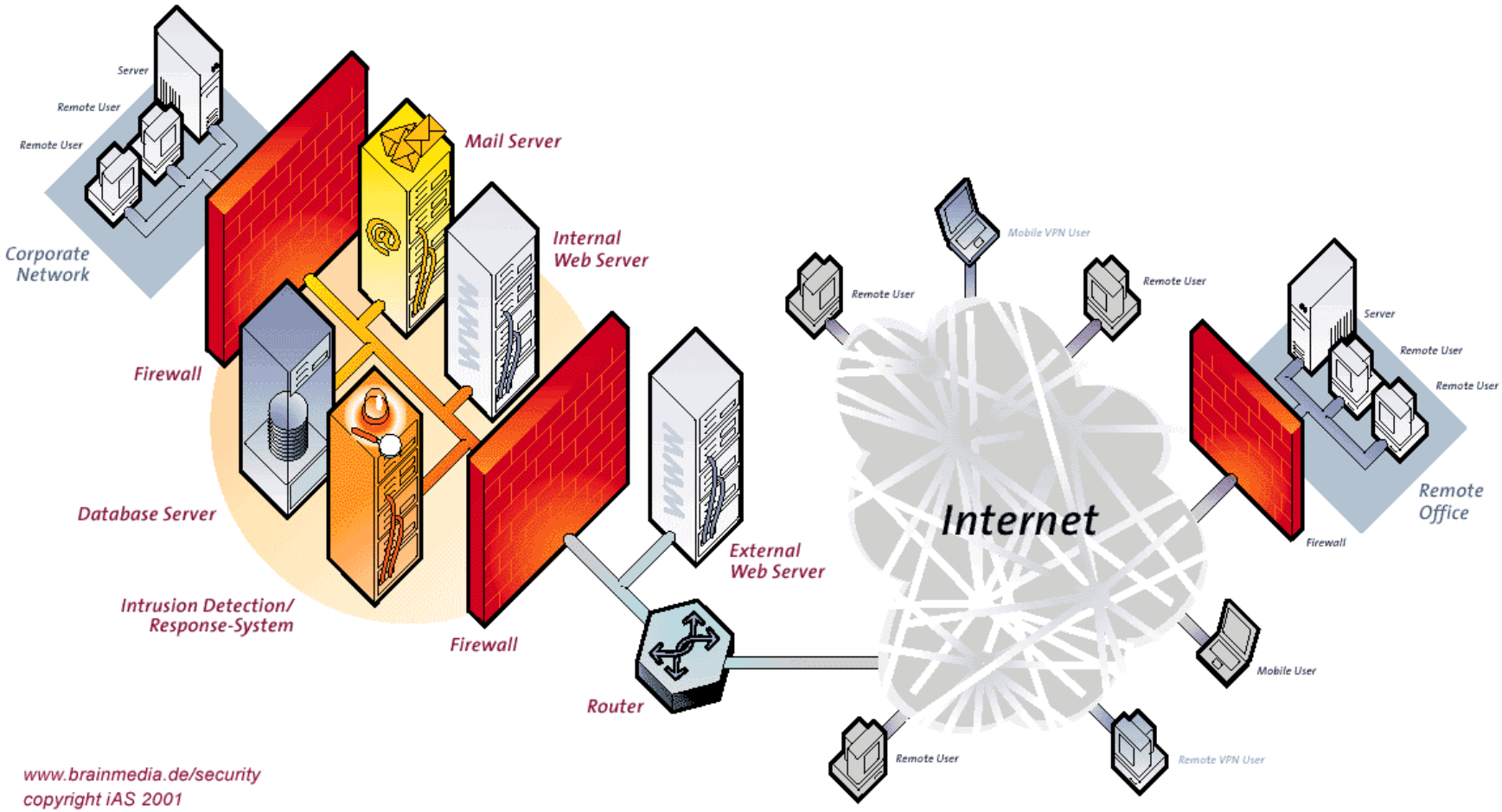
- Netzwerkverkehr wird anhand von Regeln nach bekannten Signaturen untersucht

```
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;)
```

- 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 - Sinnvolles Plazieren der Sensoren

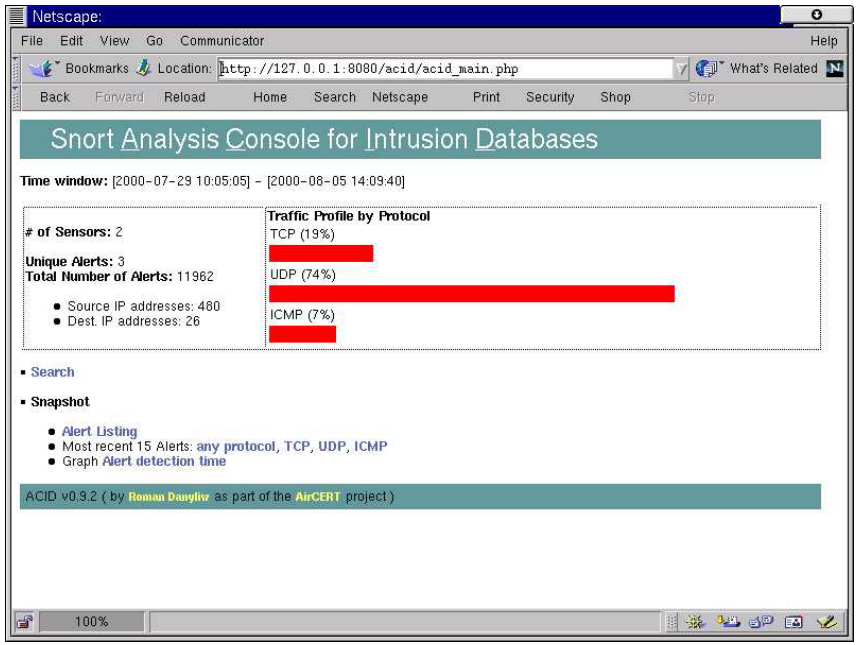
Angriffe auf Computersysteme: Aufbau von Sicherheitssystemen

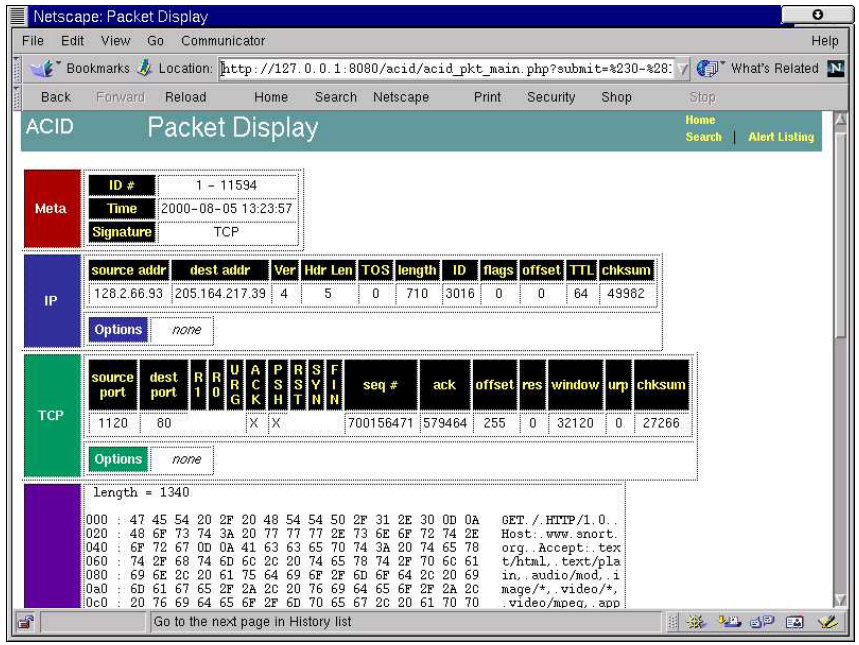


- Auswertung von „rohen“ Daten ist recht mühsam
- Loggen in lokale Dateien skaliert nicht

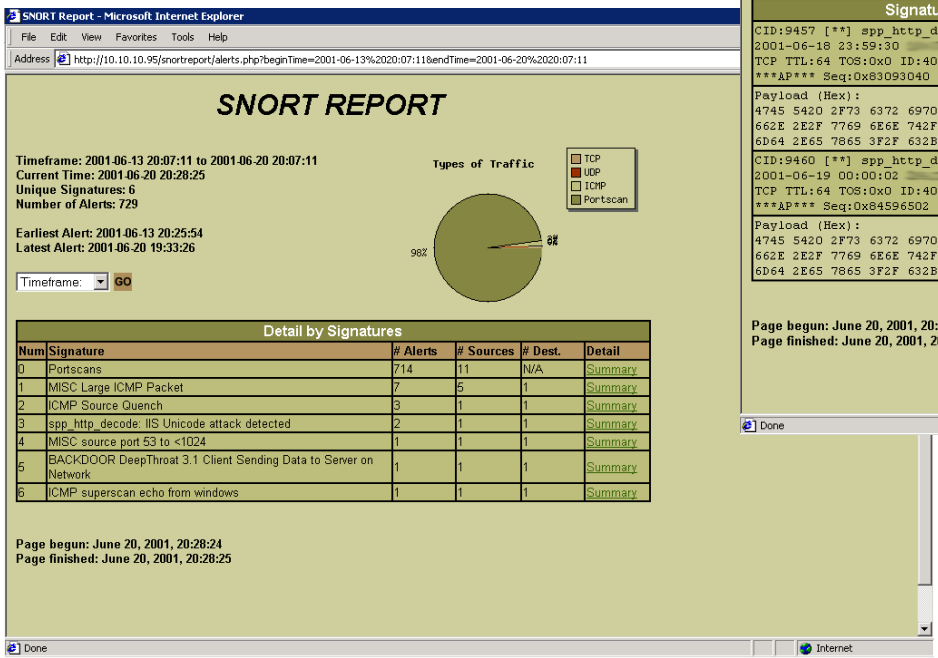
- Oracle, MySQL, PostgreSQL, ODBC
- Zentrale Datenbank (evtl. zusätzlich zur lokalen Datenerfassung)

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





- SNORT Report (www.circuitsmaximus.com)


SNORT Report - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://10.10.10.95/snortreport/alerts.php?beginTime=2001-06-13%2020:07:11&endTime=2001-06-20%2020:07:11

SNORT REPORT

Timeframe: 2001-06-13 20:07:11 to 2001-06-20 20:07:11
Current Time: 2001-06-20 20:28:25
Unique Signatures: 6
Number of Alerts: 729

Earliest Alert: 2001-06-13 20:25:54
Latest Alert: 2001-06-20 19:33:26

Types of Traffic

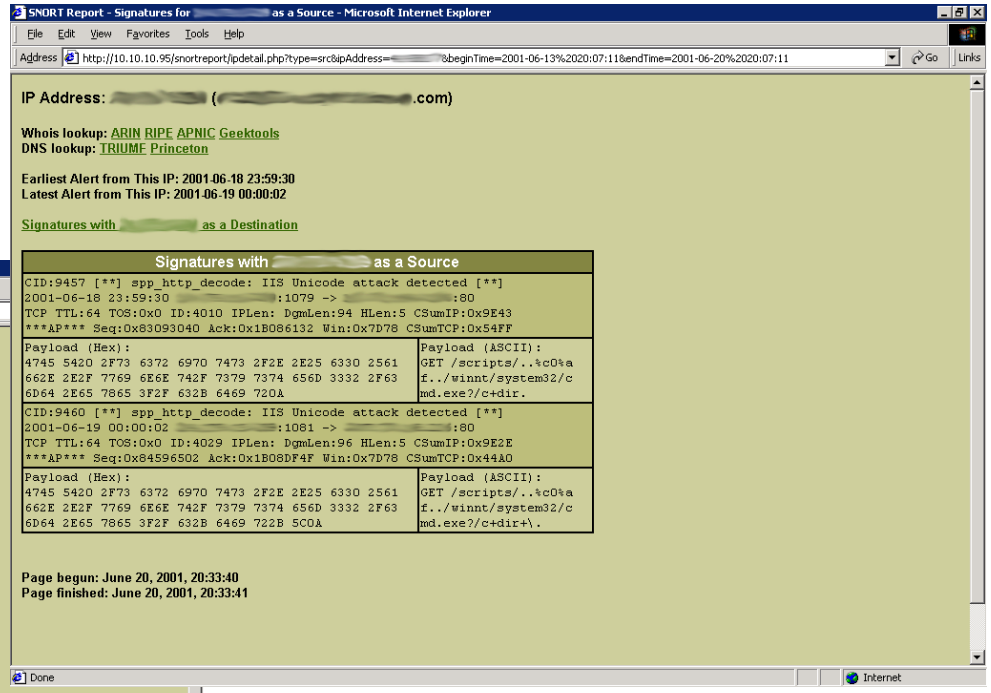
Timeframe: GO

Detail by Signatures

Num	Signature	# Alerts	# Sources	# Dest.	Detail
0	Portscans	714	11	N/A	Summary
1	MISC Large ICMP Packet	7	5	1	Summary
2	ICMP Source Quench	3	1	1	Summary
3	spp_http_decode: IIS Unicode attack detected	2	1	1	Summary
4	MISC source port 53 to <1024	1	1	1	Summary
5	BACKDOOR DeepThroat 3.1 Client Sending Data to Server on Network	1	1	1	Summary
6	ICMP superscan echo from windows	1	1	1	Summary

Page begun: June 20, 2001, 20:28:24
Page finished: June 20, 2001, 20:28:25

Done Internet


SNORT Report - Signatures for [redacted] as a Source - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://10.10.10.95/snortreport/ipdetail.php?type=src&ipAddress=[redacted]&beginTime=2001-06-13%2020:07:11&endTime=2001-06-20%2020:07:11

IP Address: [redacted] ([redacted].com)

Whois lookup: ARIN RIPE APNIC Geektools
DNS lookup: TRIUMF Princeton

Earliest Alert from This IP: 2001-06-18 23:59:30
Latest Alert from This IP: 2001-06-19 00:00:02

Signatures with [redacted] as a Destination

Signatures with [redacted] as a Source

CID:9457 [**] spp_http_decode: IIS Unicode attack detected [**]
2001-06-18 23:59:30 [redacted]:1079 -> [redacted]:80
TCP TTL:64 TOS:0x0 ID:4010 IpLen: DgmLen:94 HLen:5 CSumIP:0x9E43
AP Seq:0x83093040 Ack:0x1B086132 Win:0x7D78 CSumTCP:0x54FF

Payload (Hex):	Payload (ASCII):
4745 5420 2F73 6372 6970 7473 2F2E 2E25 6330 2561	GET /scripts/..%c0%a
662E 2E2F 7769 6E6E 742F 7379 7374 656D 3332 2F63	f../winnt/system32/c
6D64 2E65 7865 3F2F 632B 6469 720A	md.exe?/c+dir.

CID:9460 [**] spp_http_decode: IIS Unicode attack detected [**]
2001-06-19 00:00:02 [redacted]:1081 -> [redacted]:80
TCP TTL:64 TOS:0x0 ID:4029 IpLen: DgmLen:96 HLen:5 CSumIP:0x9E2E
AP Seq:0x84596502 Ack:0x1B08DF4F Win:0x7D78 CSumTCP:0x44A0

Payload (Hex):	Payload (ASCII):
4745 5420 2F73 6372 6970 7473 2F2E 2E25 6330 2561	GET /scripts/..%c0%a
662E 2E2F 7769 6E6E 742F 7379 7374 656D 3332 2F63	f../winnt/system32/c
6D64 2E65 7865 3F2F 632B 6469 722B 5C0A	md.exe?/c+dir+\.

Page begun: June 20, 2001, 20:33:40
Page finished: June 20, 2001, 20:33:41

Done Internet

Härten der Systeme (Beispiele)

- Keine unnötigen Dienste installieren/anbieten bzw. nur an nötiges Interface binden
- Monolithischer Linux-Kernel (kernelbased Rootkits) – trotzdem Zugriff auf /proc/kmem

- Buffer-Overflow / Format-String Schwachstellen
  - Nicht ausführbare Speichersegmente: OpenWall (Linux), PaX (Linux), Solaris ab 2.6
  - Schutz vor Überschreiben von Zeigern: StackGuard, FormatGuard (beides Patches für den gcc), StackShield, Libsafe, Libverify

- 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/kcore) 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

- Netzwerk Scan Techniken
- Systemschwachstellen ausnutzen
- Aufbau von Sicherheitssystemen
- Digitale Forensik

Was ist Digitale Forensik?

- Analyse nach der Tat
- Spurensicherung
  - Wer?
  - Was?
  - Wann?
  - Wie?
- Beweissicherung
  - Für eventuelle (juristische) Verfolgung
- Dokumentation

Nach einem erfolgreichen Angriff

- Überwachungsmethoden analysieren
- Verwischen von Spuren / Entdeckung vermeiden
- Installation eines Root Kits
- Einrichtung von Backdoors
- Angriff auf weitere Systeme

Wie fällt ein kompromittiertes System auf?

- Intrusion Detection Systeme (host- / netzwerkbasierend)

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

- 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

- 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

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

- 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]

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

Analyse (1)

- Wichtige Dateien überprüfen (/etc/passwd, /etc/group, inetd.conf / xinetd, services, Startdateien in rc.d, ...)
- Verdächtige MACtimes in /sbin/ oder /usr/sbin? (verdächtige Binaries mit „strings“ ansehen)
- Geladene Module prüfen
- Per ifconfig nach Interfaces im „promiscous mode“ suchen (Sniffer)
- Historys ansehen (z.B. .bash_history)

Analyse (2)

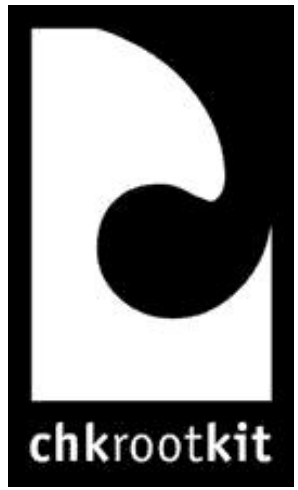
- Logfiles auf verdächtige Einträge durchsuchen; Beispiel:

  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

- Beliebte Plätze für Root-Kits
  - /dev/, /usr/man, /lib/modules, /usr/lib
- Oft „trojanisierte“ Programme
  - ps, ls, find, ifconfig, netstat, du, df
  - sshd, httpd
  - login, passwd
  - inetd, tcpd
- www.chkrootkit.org (erkennt sehr viele Root-Kits auf lokalem Rechner automatisch)



- 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)

- Netzwerk Scan Techniken
- Systemschwachstellen ausnutzen
- Aufbau von Sicherheitssystemen
- Digitale Forensik

Ein wenig Literatur

- TCP/IP Illustrated Volume 1; W. Richard Stevens; Addison Wesley; 1994; ISBN 0-201-63346-9
- Linux System Security; Scott Mann & Ellen L. Mitchell; Prentice Hall; 2002; ISBN 0-13-015807-0
- Linux Firewalls; 2. Auflage, Robert L. Ziegler; Markt & Technik; 2002; ISBN 3-827-26257-7
- Network Intrusion Detection. An Analyst's Handbook; Second Edition; Stephen Northcutt & Judy Novak; New Riders; 2001; ISBN 0-7357-1008-2
- Intrusion Detection für Linux-Server; Ralf Spenneberg; Markt & Technik, 2002