Firewalls und Intrusion Detection Systeme

Grundlagen, Planung und Realisierungsvorschläge unter Linux

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

- Allgemeines zur Sicherheit von IT-Systemen
- Paketfilter und Proxys
- Intrusion Detection
- Zusätzliche Maßnahmen
- Zusammenfassung

Was möchte ich schützen?

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

- Vertraulichkeit, Integrität, Verfügbarkeit und Authentizität

- 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

- Erstellen von Sicherheitsrichtlinien
- Risikomanagement
  - Risikoanalysen
  - Vorgehen bei „Katastrophen“
- Authentisierung der Benutzer
- Zugriffskontrollen
- Filtern von kritischen Inhalten
- Verschlüsselung
- Benutzerschulungen

Detection Phase

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

- Detection kann wichtiger als Protection sein

Response Phase

- (Automatische?) Reaktion auf Vorfälle
- Analyse der Vorfälle
- Disaster Recovery
- Bestimmen der verantwortlichen Schwachstellen

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

Sicherheitsrichtlinien (1)

niedergeschriebene(!) Regeln über

- Vertraulichkeit
- Integrität
- Verfügbarkeit

- Philosophie (bezüglich Sicherheit)
- Strategie um die Ziele zu erreichen
- Sicherheitsregeln (was darf man, was nicht)
- praktische Anleitungen

Warum Sicherheitsrichtlinien?

- Unterschied zwischen „Technik“ und „Verantwortlichkeit“
- Entwicklung der Richtlinien macht dem Management die Bedeutung von Sicherheit klar
- zeigt Kunden/Partnern/Investoren Seriosität
- Messlatte für Umsetzung von Sicherheitsmaßnamen
- gibt Sicherheitspersonal die Rückendeckung des Managements

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

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

- Allgemeines zur Sicherheit von IT-Systemen
- Paketfilter und Proxys
- Intrusion Detection
- Zusätzliche Maßnahmen
- Zusammenfassung

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 Zwischennetz (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, ablehnen, modifizieren oder loggen nach Kriterien wie IP Quell-/Zieladresse, Protokoll, TCP/UDP Quell-/Zielport, ICMP-Typ, Paketgröße/-validität, Fragmentierung, Zustand der Verbindung

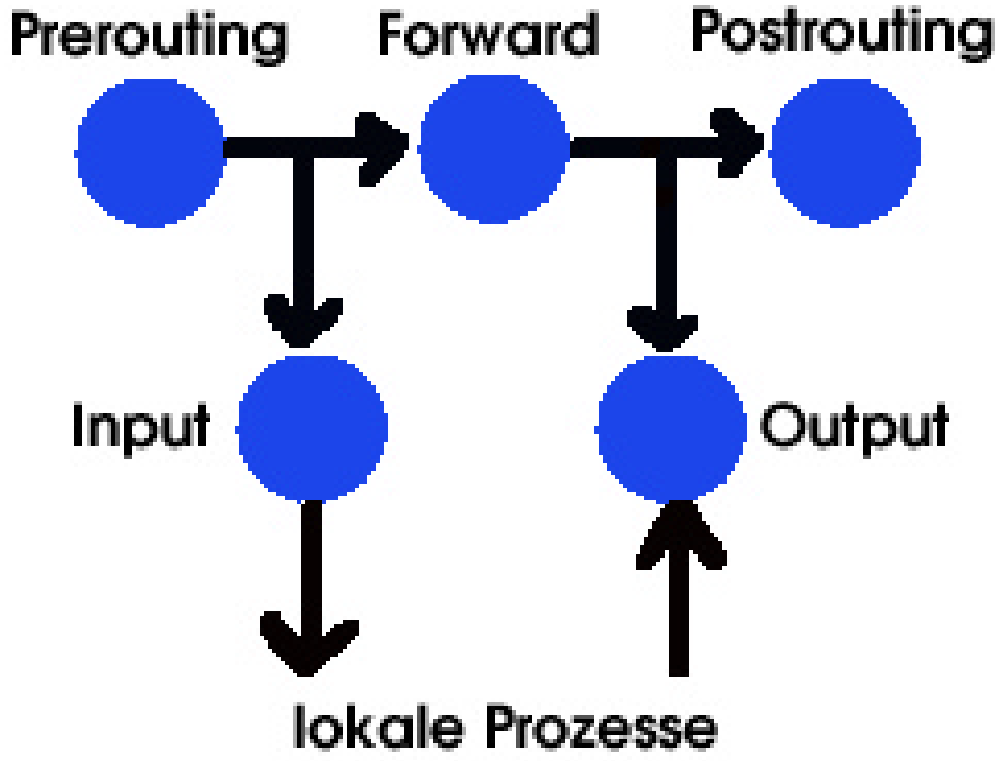
- Netfilter / Iptables ab Linux-Kernel 2.4

- Userspace Tool zum Filtern von Paketen im Kernel
- ersetzt Ipchains
- benötigt Kernel ab 2.4 mit Netfilter-Unterstützung
- Informationen unter netfilter.samba.org

- grosse Menge an neuem Code
- enthält evtl. noch kritische Fehler

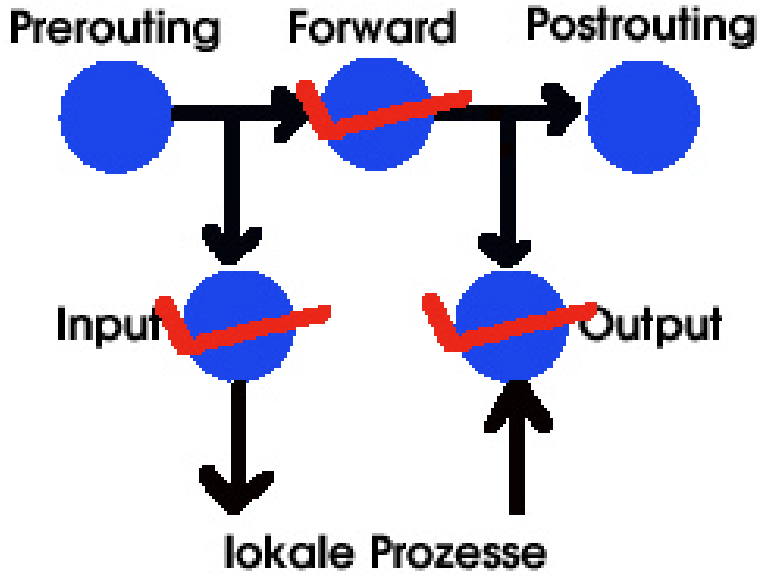
- Paketfilter
- Connection tracking / stateful inspection
- Network address translation (NAT)
- Packet mangling
- einfache Bandbreitenbeschränkung
- Limitierungen
- erweitertes Logging möglich

- Chains (Ketten) im Kernelspace
  - PREROUTING
  - INPUT
  - OUTPUT
  - FORWARD
  - POSTROUTING
- Tables (Tabellen) im Userscape
  - filter
  - nat
  - mangle

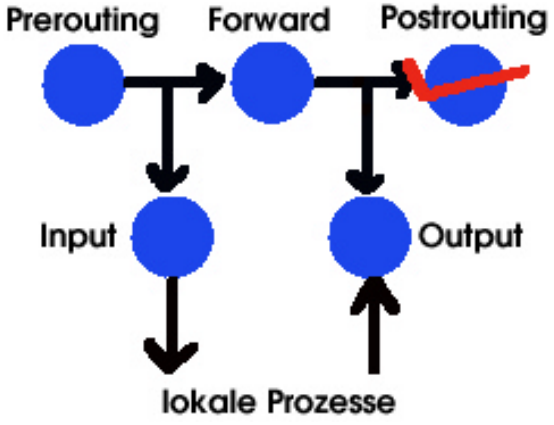

Prerouting
Forward
Postrouting
Input
Output
lokale Prozesse

- verantwortlich für das Filtern von Paketen ist die filter Tabelle in Abhängigkeit von
  - IP Adresse, Quell-MAC-Adresse
  - ICMP – Typ und Code
  - UDP – Quell- und Zielport
  - TCP
    - Quell- und Zielport
    - Flags (UAPRSF)
    - Options
    - Benutzer (uid, pid, gid)
  - Fragmente

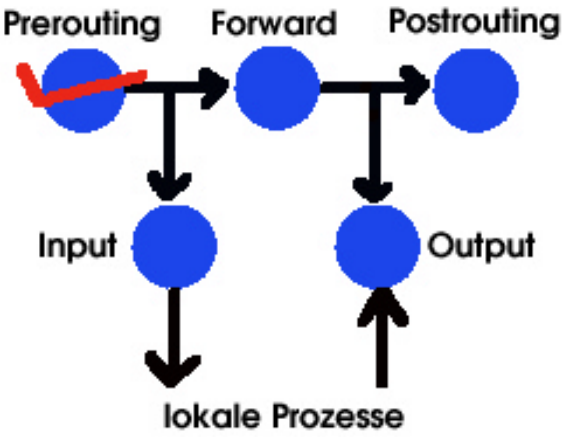
- Filtern für lokale Prozesse in der INPUT und der OUTPUT Kette
- Filtern für remote Prozesse in der FORWARD Kette (Unterschied zu Ipchains)



- Source rewrite NAT (SNAT)
  - Quelladresse des Paketes wird verändert
  - schließt Masquerading ein
  - immer nach dem Routing/ Filtern in der POSTROUTING Kette
  - Beispiele:
    - iptables –t nat –A POSTROUTING –j SNAT --to 192.168.100.1-192.168.100.5
    - iptables –t nat –A POSTROUTING –j MASQUERADE

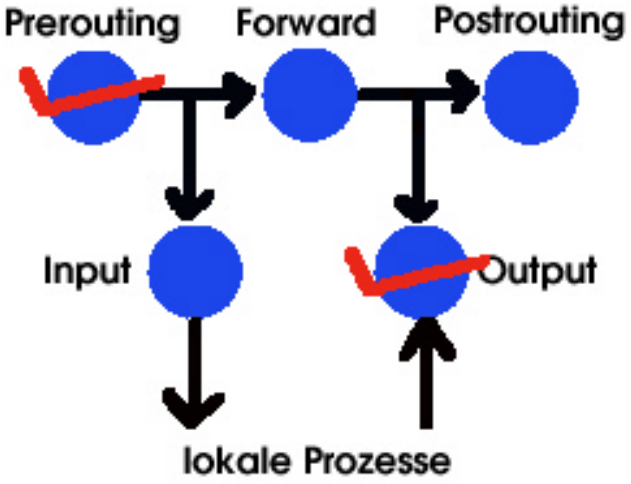


- Destination rewrite NAT (DNAT)
  - Zieladresse des Paketes wird verändert
  - Immer vor dem Routing/ Filtern in PREROUTING Kette



  - schließt Port Forwarding und Redirection ein
  - Beispiele:
    - iptables –t nat –A PREROUTING --dport 80 –j REDIRECT --to-port 3000
    - iptables –t nat –A PREROUTING –j DNAT--to 192.168.100.10-192.168.100.15

- findet in der PREROUTING oder OUTPUT Kette statt
- defragmentiert alle Pakete
- Zustandstabellen enthalten
  - IP Protokoll
  - Socketpaar
  - Zustand der Verbindung (NEW, ESTABLISHED, RELATED (z.B. bei ICMP oder FTP), INVALID)
  - Timeouts
  - Sequenznummer



- Beispiele für Regeln
  - iptables –A INPUT –m state --state ESTABLISHED,RELATED –j ACCEPT
  - iptables –A OUTPUT –m state --state ESTABLISHED,RELATED,NEW –j ACCEPT

- Beispiel für einen Zustandstabelleneintrag
  - tcp 6 1995 ESTABLISHED src=ip1 dst=ip2 sport=1311 dport=80 src=ip2 dst=ip1 sport=80 dport=1311 [ASSURED] use=1

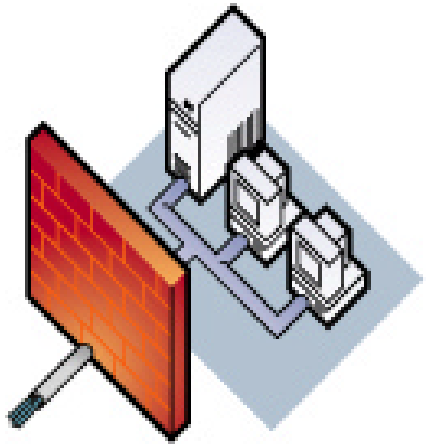
- Schutz gegen manche DoS-Angriffe
- greift sowohl beim Verbindungsaufbau als auch beim Logging
- kann die Anzahl der Aktionen pro Sekunde und die Maximalanzahl begrenzen
- Beispiel für eine SYN-Flood Schutzregel:
  - iptables –A INPUT –p tcp --syn –m limit --limit 1/s --limit-burst 5 –j ACCEPT

- Manipulation von Paketen in der mangle Tabelle
- Beispiel für das Ändern des TOS Flags
  - iptables –t mangle –A OUTPUT –p tcp --dport 20 –j TOS --set-tos Maximize-Throughput
- Pakete können für Load-Balancing per DNAT markiert werden
- Beispiel einer Paketmarkierungsregel
  - iptables –t mangle –A PREROUTING –p tcp --dport 25 –j MARK --set-mark 1

- REJECT: aktive Ablehnung einer Verbindungsanfrage durch ICMP Paket vom Inhalt "Admin hat's verboten" oder "Dienst nicht verfügbar„
- DROP/DENY: kommentarloses Wegwerfen der Verbindungsanfrage (Timeout für Anfragenden)

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

- Anwendungsbeispiel: Linux-PC als Router für eine einfache ISDN-Wählverbindung, Fernwartung nur aus dem internen Netz (192.168.100.0) über SSH



- Variablen und Kernelparameter setzen:

```
IPTABLES=/sbin/iptables
DEV_LOC=eth0
DEV_EXT=ippp0
for IF in $DEV_LOC $DEV_EXT ; do
  echo "1" > /proc/sys/net/ipv4/conf/$IF/rp_filter
  echo "0" > /proc/sys/net/ipv4/conf/$IF/accept_redirects
  echo "0" > /proc/sys/net/ipv4/conf/$IF/accept_source_route
  echo "1" > /proc/sys/net/ipv4/conf/$IF/log_martians
done
```

- alles was nicht ausdrücklich erlaubt ist wird verboten:

```
$IPTABLES -P INPUT REJECT
$IPTABLES -P OUTPUT REJECT
$IPTABLES -P FORWARD REJECT
```

- evtl. gespeicherte Regelketten löschen:

```
$IPTABLES -F
$IPTABLES -t nat -F
$IPTABLES -X
```

- Kernel auf Umgang mit dynamischen IP-Adressen vorbereiten und IP-Forwarding aktivieren

```
echo "1" > /proc/sys/net/ipv4/ip_dynaddr
echo "1" > /proc/sys/net/ipv4/ip_forward
$IPTABLES -t nat -A POSTROUTING -o $DEV_EXT -j
    MASQUERADE
```

- FORWARD: zwei Regeln für bereits bestehende ein- und ausgehende Verbindungen und eine die erlaubt, dass neue Verbindungen von innen nach außen aufgebaut werden können:

```
$IPTABLES -A FORWARD -i $DEV_LOC -o $DEV_EXT -m
    state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A FORWARD -i $DEV_EXT -o $DEV_LOC -m
    state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A FORWARD -i $DEV_LOC -o $DEV_EXT -m
    state --state NEW -j ACCEPT
```

Iptables – Sehr einfache Filterregeln (6)

- Filterregeln sollen nur eingehende SSH-Verbindungen lokal zulassen:

```
$IPTABLES -A INPUT -i $DEV_LOC -s
    192.168.100.0/255.255.255.0 -p tcp --sport 1000:1023 --
    dport ssh –m state --state NEW,ESTABLISHED,RELATED
    -j ACCEPT
$IPTABLES -A OUPUT -o $DEV_LOC -d
    192.168.100.0/255.255.255.0 -p tcp --dport 1000:1023 --
    sport ssh –m state --state ESTABLISHED,RELATED -j
    ACCEPT
```

- beliebig komplexe Erweiterungen möglich

- FTP auf dem Paketfilterrechner hinzufügen (schlechte Idee, aber nur als Beispiel)

```
$IPTABLES -A INPUT -p tcp --sport 21 -m state --state
    ESTABLISHED -j ACCEPT
$IPTABLES -A OUTPUT -p tcp --dport 21 -m state --state
    NEW,ESTABLISHED -j ACCEPT
```

- nicht nur Steuerungskanal (Port 21) sondern auch noch Datenkanal nötig (aktives/passives FTP)

Iptables – aktives FTP

- FTP-Client sendet Portnummer (größer 1023) über PORT-Kommando an Server
- Server verbindet sich von Port 20 auf diesen Port um Daten zu senden (Gegenrichtung)
- ipchains: alle Ports über 1023 müssen für Zugriff von Remote-Port 20 auf FTP-Clients freigegeben werden (sehr unsicher)
- iptables besitzt connection tracking

```
$IPTABLES -A INPUT -p tcp --sport 20 -m state --state
    ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A OUTPUT -p tcp --dport 20 -m state --state
    ESTABLISHED -j ACCEPT
```

- FTP-Server benutzt PORT Kommando
- Client kontaktiert den Server auf gewünschtem Port (gleiche Richtung wie Verbindungsaufbau, viel sicherer)
- Quell- und Zielport sind nun unbekannt
- connection tracking erkennt aber die Verwandtschaft der Verbindungen

```
$IPTABLES -A INPUT -p tcp --sport 1024: --dport 1024: -m
    state --state ESTABLISHED -j ACCEPT
$IPTABLES -A OUTPUT -p tcp --sport 1024: --dport 1024: -m
    state --state ESTABLISHED,RELATED -j ACCEPT
```

- 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, ...

- Allgemeines zur Sicherheit von IT-Systemen
- Paketfilter und Proxys
- Intrusion Detection
- Zusätzliche Maßnahmen
- Zusammenfassung

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

- erkennt (detects) Einbrüche (Intrusions)
- erkennt Veränderungen wichtiger Dateien
- erkennt die Installation von Hintertüren
- erkennt die Veränderung von Systemregeln
- 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

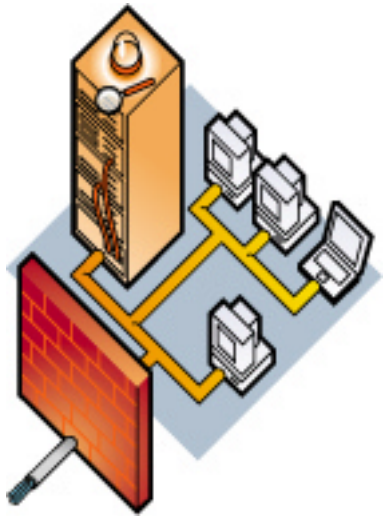
- Falsch positiv (eine erlaubte Aktion wird als Angriff identifiziert)
- Falsch negativ (ein Angriff wird vom IDS nicht erkannt oder als unbedenklich klassifiziert)
- Subversionsfehler (sehr komplexe Fehler; der Angreifer kann das IDS unterminieren)

Hostbasierte IDS überwachen nur einen einzigen Rechner

- beobachtet systemrelevante Dateien und Befehle
- meldet ungewöhnliche Dateien und Admistrationsvorgänge
- analysiert und sichert Protokolle
- beobachtet offene Ports und Netzwerkverbindungen
- wird auch oft als Audit bezeichnet

Netzbasierte IDS sammeln Informationen aus einem ganzen Netzsegment

- untersucht und protokolliert (bei Bedarf) sämtliche Netzwerkpakete
- erkennt mögliche Angriffe an „Fingerabdrücken“
- untersucht die protokollierten Daten auf Trends
- kann die Paketfilterregeln anpassen



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 (Übertragung aber unsicher)
  - 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
- evtl. syslog-ng (pattern matching) einsetzen

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

- 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
}
```

- Scan: extensives nachforschen welche Dienste ein Rechner oder eine Rechnergruppe anbietet
- Portscan fragt höflich und formgerecht alle möglichen Dienste (65535 statusbehaftete (TCP) und noch mal so viele statuslose (UDP)) Quellen auf Basis von IP ab
- praktisch ungefährlich (keine unnötigen Dienste)
- Portscan ist nicht notwendigerweise eine Angriffsvorbereitung (schon gar kein Angriff)
- sogar Bannerscans (Serversoftware und Versionsnummer abfragen) sind legal
- trotzdem loggen?

- PortSentry (www.psionic.com/abacus/portsentry/) entdeckt Portscans und kann in Echtzeit darauf reagieren
- Unterstützt TCP und UDP
- Unter Linux werden auch Stealth Scans, wie zum Beispiel SYN/half-open, FIN, NULL, X-MAS und oddball entdeckt

- Startart auswählen (TCP/UDP Stealth-Modus, Advanced-Stealth-Modus)
- Angreifer blocken? (ja/nein)
- Welche Methoden werden zum Blocken benutzt? (Eintrag in /etc/hosts.deny oder per Paketfilter (KILL_ROUTE) alle Pakete des Angreifers verwerfen und/oder loggen)

- Tripwire (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.

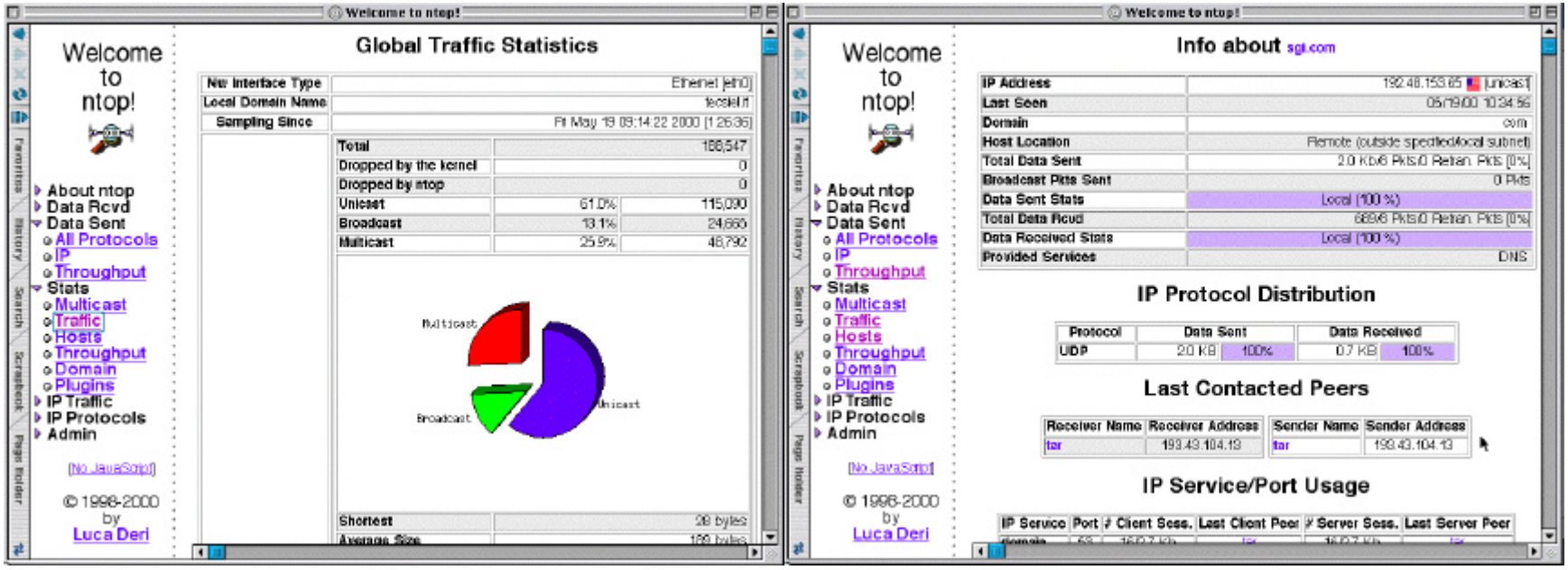
Tripwire einrichten

- allgemeine Konfigurationsdatei (twcfg.txt) und Richtliniendatei (twpol.txt) nach eigenen Bedürfnissen anpassen (gut dokumentiert)
- twinstall.sh erfragt dann Passphrases und verschlüsselt die beiden Dateien (distributionsabhängig)

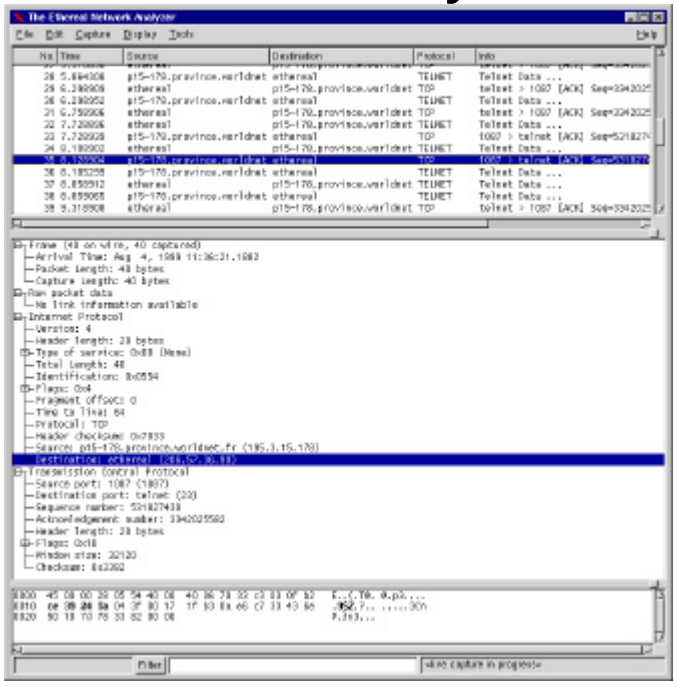
- *tripwire --init* erstellt (nach Eingabe der Passphrase) die verschlüsselte Datenbank
- *tripwire --check* erstellt einen Bericht mit allen Änderungen (minimum täglich, kann per Email verschickt werden)
- nach (gewünschten) Änderungen am Dateisystem muss die DB aktualisiert werden (*tripwire --update –r Reportdatei*)

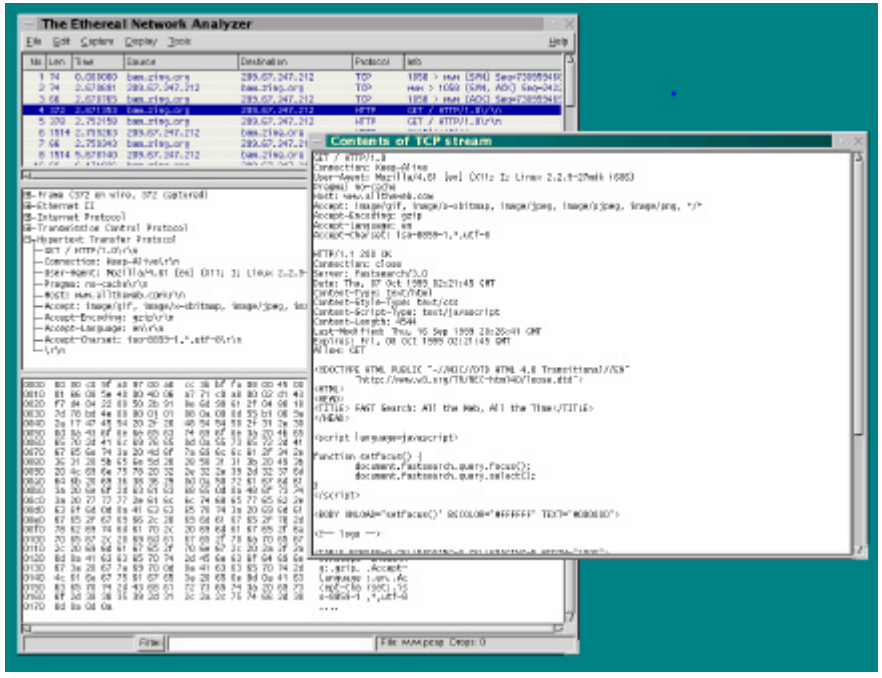
- Übersicht über den Normalzustand im Netzwerk bekommen
  - ntop
  - Ethereal
  - arpwatch – Kontrolle der MAC-Adressen im Netzwerk

- 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)





- kann unter anderem erkennen:
  - buffer overflows
  - stealth port scans
  - cgi-Angriffe
  - SMB und NetBIOS Tests
  - Portscanner (wie nmap)
  - DDoS Clients
- Protokollierung
  - syslogd
  - SMB WinPopUp über Samba

- Dynamische Regelsätze
- TCP-Stream Reassemblierung
- IP-Defragmentierung (ab Version 1.7)
- HTTP Präprozessor erkennt UNICODE
- Ausführliche Regelsätze, individuell zusammenstellbar
- Sicherheitsfeatures (chroot, User snort/snort)

- Flexible Response (SNORT kann direkt Gegenmaßnamen einleiten)

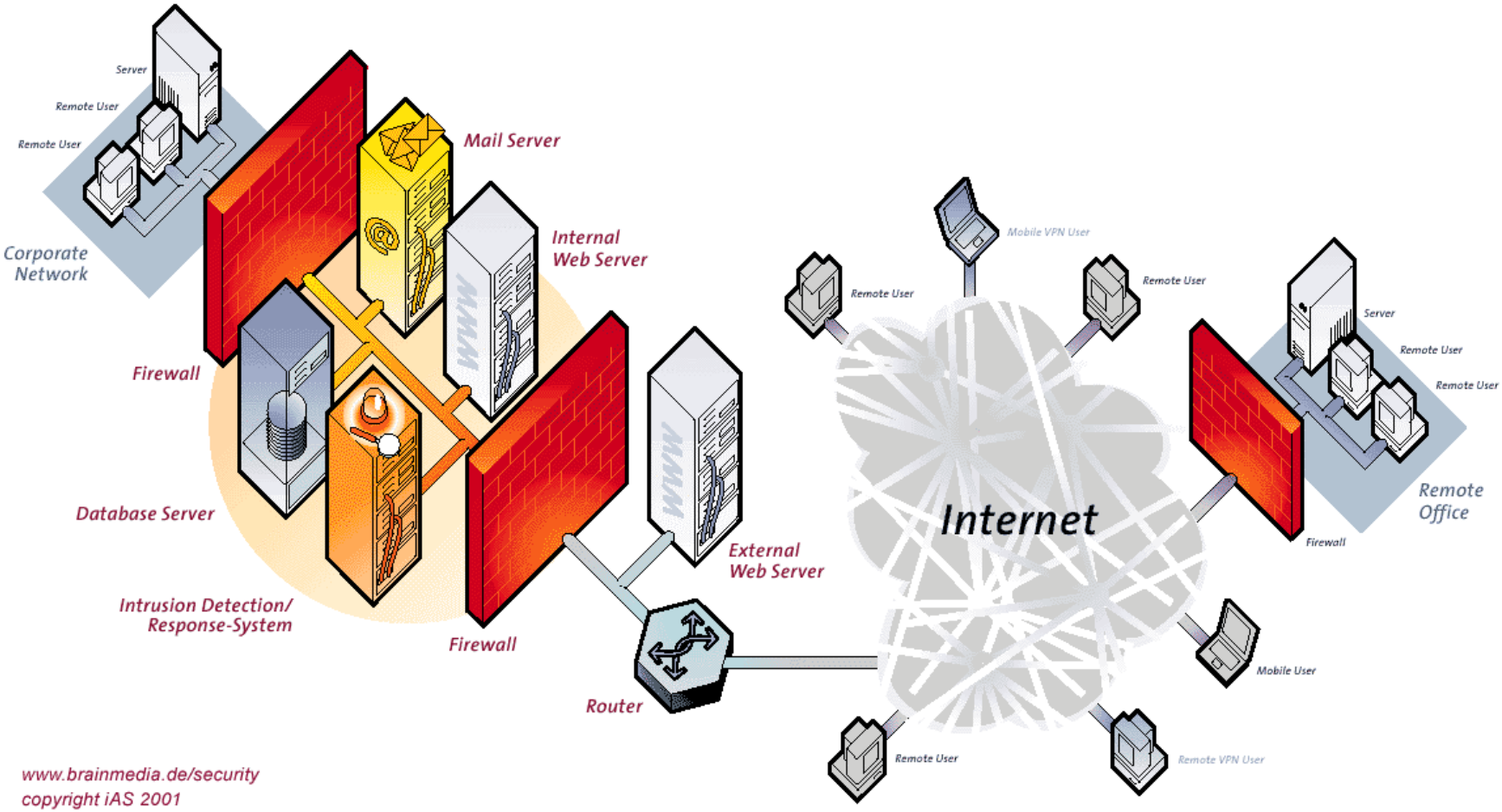
- www.snort.org
- unter /etc/snort/ gewünschte Regeln aktivieren (lieber mehr als weniger)
- *snort –u snort –g snort –s –d –D –i eth0 –l /var/log/snort –c /etc/snort/rules.base*
- nach und nach Regeln die falsch positive Ergebnisse liefern entfernen
- aktuelle Regeln/Signaturen aus der ArachNIDS Datenbank auf www.whitehats.com (geht auch automatisch)

- Netzwerkverkehr wird anhand von Regeln nach Signaturen (Fingerprints) 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;)
- Meldung in /var/log/messages oder /var/log/secure
- Paket in /var/log/snort/Rechner-IP/

- loggen in Datenbanken (z.B. mySQL) möglich

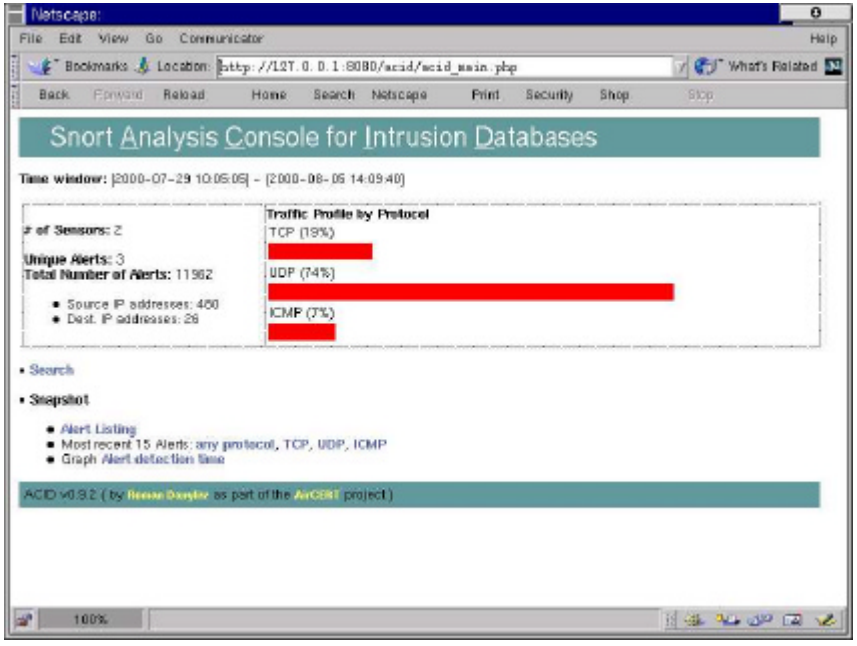
Sinnvolles Plazieren der Sensoren

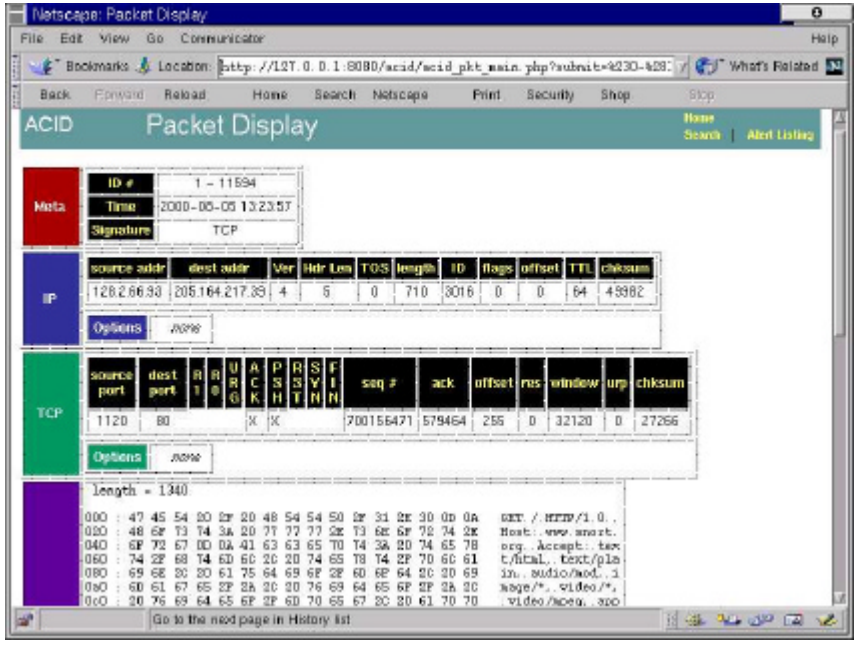
Firewalls und Intrusion Detection Systeme: Intrusion Detection



SNORT – Arbeitserleichterungen (1)

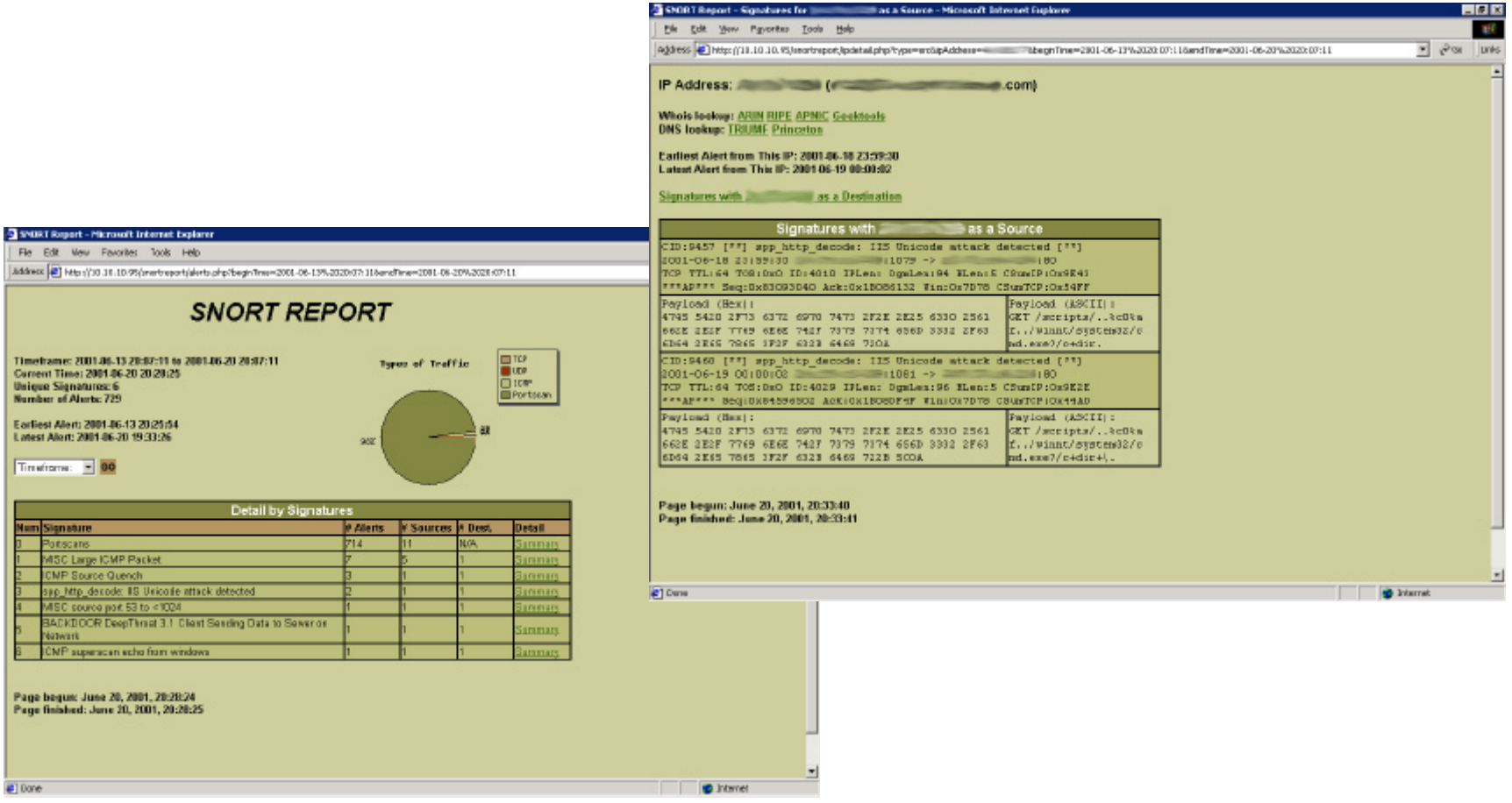
- Analysis Console for Intrusion Databases (ACID, www.cert.org/kb/acid)





SNORT – Arbeitserleichterungen (2)

- SNORT Report (www.circuitsmaximus.com)



Intrusion Detection - Fazit

- Intrusion Detection bietet keinen Schutz
- erlaubt die Erkennung von Angriffen wenn vorgeschaltete Sicherheitssysteme versagen
- Möglichkeit das Vorgehen des Angreifers zu erlernen
- Sensoren eines IDS sollten sinnvoll plaziert werden (z.B. vor dem externen Paketfilter, in einer DMZ, hinter dem internen Paketfilter)
- Kopplung von IDS und Paketfilter muss gut überlegt sein (Möglichkeit zum DoS-Angriff)

- Allgemeines zur Sicherheit von IT-Systemen
- Paketfilter und Proxys
- Intrusion Detection
- Zusätzliche Maßnahmen
- Zusammenfassung

Absichern des Standardkernels

Firewalls und Intrusion Detection Systeme: Zusätzliche Maßnahmen

- Wenn mit einem Standardkernel gearbeitet werden soll, sollte man diesen absichern
- Um Kernel-Based Root-Kits zu blocken sollte man auf einen monolithischen Kernel setzen
- Kein kompletter Schutz, da Linux unter anderem den Zugriff auf den Hauptspeicher (/proc/kmem) zulässt

- Absicherung des Rechners selber wird vorausgesetzt (z.B. keine unnötigen Dienste installieren/anbieten)

- LIDS (www.lids.org) ist ein Patch für den Kernel
- es werden Features implementiert, die Linux im Vergleich zu anderen OS fehlen (Capabilities)
  - 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

- Hewlett-Packards HP-LX erweitert den Linux-Kernel um einige Sicherheitsfeatures
- Compartments schotten Applikationen in einer Art Sandbox voneinander und vom System ab
- Kommunikation zwischen den Compartments und zum System kann gezielt erlaubt/verboten werden
- Kernel-Modules und Kernel-Patches unterliegen der GPL
- Administrationsprogramme kosten im kommerziellen Einsatz (recht viel) Geld

- OpenWall Patch (www.openwall.com/linux) verhindert das Ausführen von Code vom (User) Stack; schützt zu etwa 95% vor dem Ausnutzen von Bufferoverflows; bisher nur für Kernel 2.2 verfügbar (2.4 ab etwa 2.4.15 laut Homepage)
- WireX Stackguard Compiler (vom gcc abgeleitet, schreibt eine Prüfsumme auf den Stack (Canary) und prüft den vor Rücksprung); Immunix (www.immunix.org) bietet eine komplett neu übersetzte RedHat 7.0 an
- Libsafe (www.research.avayalabs.com/project/libsafe)

- Allgemeines zur Sicherheit von IT-Systemen
- Paketfilter und Proxys
- Intrusion Detection
- Zusätzliche Maßnahmen
- Zusammenfassung

- Sicherheit ist ein kontinuierlicher Prozess

- Ein Sicherheitssystem sollte die folgenden Abwehrreihen bereitstellen:
  - Paketfilter und Proxys
  - Hostbasierte IDS
  - Netzbasierte IDS
  - abgesicherter Kernel