IIR - Workshop

Vulnerability Management (Training)

29. November - 1. Dezember 2004

René Pfeiffer

pfeiffer@technikum-wien.at

rene.pfeiffer@paradigma.net

pfeiffer@luchs.at

Technikum Wien

Fachbereich Elektronische Informationsdienste

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Wichtiger Hinweis:

Die Logfileauszüge dieses Vortrags enthalten zum Teil „echte“ IP Adressen und Hostnamen, die von Providern oder anderen Personen in Verwendung sind. Ich bitte diesen Umstand nicht als Anschuldigung zu verstehen oder daraus Maßnahmen oder Empfehlungen abzuleiten. Die Beispiel-Logs wurden bereits ausgewertet und in Einzelfällen wurden Schritte unternommen bzw. wurde der entsprechende Vorfall in Abstimmung mit der zutreffenden Security Policy behandelt. Ich bitte das Erscheinen der IP Adressen in keinster Weise als Bewertung, Kritik oder Anschuldigung zu sehen. Weiterhin bitte ich darum, diese IP Adressen keinen speziellen Untersuchungen wie Portscans, Security Audits oder ähnlichem ohne Zustimmung des Eigentümers zu unterziehen.

Dieses Dokument ist Copyright 2004 René Pfeiffer und darf über jedes Medium beliebig zitiert oder verteilt werden, sofern dieser Hinweis erhalten bleibt.

1 Firewalling mit Linux Netfilter und Kernel 2.6.x

BOUNDARY, n. In political geography, an imaginary line between two nations, separating the imaginary rights of one from the imaginary rights of the other.

– *„The Devil's Dictionary"*, *Ambrose Bierce*

1.1 Netfilter im Kern 2.4.x/2.6.x

- **zustandgesteuert / stateful inspection**
  - Source/Destination NAT
  - reichhaltiger Protokoll Support (FTP, TFTP, etc.)
- **modular implementiert seit Kernel 2.4.x**
- **Filtern auf Layer 2, 3 und 4**
  - IPV4, IPV6 und IPsec
  - Layer 7 Filter Projekt
- **Paketmanipulationen**
  - Markieren
  - Umschreiben von Kopfinformationen
- **ergänzt durch exzellenten Routing Code in 2.6.x**
- **multiple Plattformen (IA32, IA64, PPC, S390, etc.)**

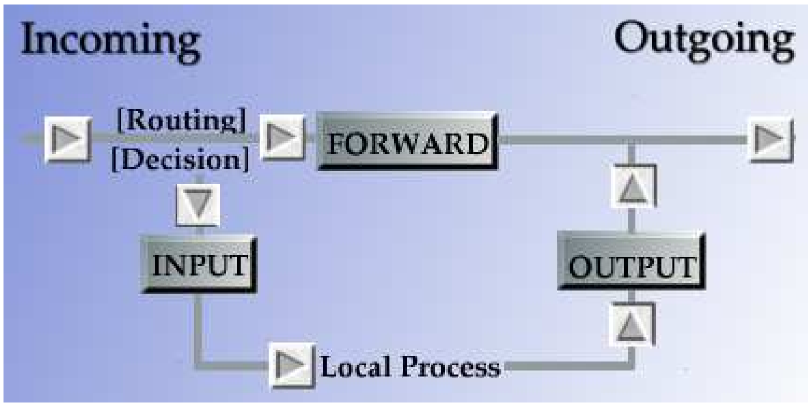

Abbildung 1: Das Diagramm zeigt den Fluß der Pakete durch eine Linux Netfilter Firewall. `INPUT` wird von eingehenden Paketen passiert, die für die Firewall selbst bestimmt sind. `FORWARD` betrifft ausschließlich weitergeleitete Pakete und durch `OUPUT` müssen alle ausgehenden Pakete, die die Firewall selbst generiert hat. Durch dieses Design wird vermieden, daß passierende Pakete mehr als eine dieser Chains passieren müssen.

1.2 Paketfluß durch den Netfilter

1.3 Möglichkeiten des Netfilter Codes

Eingriffe sind möglich durch

- **Paketfilter (filter)**
  - `INPUT`, `OUTPUT` - Pakete an bzw. von Filtermaschine
  - `FORWARD` - Pakete, die Filtermaschine durchqueren
- **Paketmanipulationen (mangle)**
  - `PREROUTING`, `POSTROUTING` - Pakete vor bzw. nach dem Routen
  - `FORWARD`, `INPUT`, `OUTPUT`
- **Network Address Translation (NAT)**
  - `PREROUTING`, `POSTROUTING` - Pakete vor bzw. nach dem Routen
  - `OUTPUT` - lokal generierte Pakete
- **verschiedene Aktionen**
  `ACCEPT`, `DROP`, `REJECT`, `DNAT`, `SNAT`, `MASQ`, `LOG`, etc.

1.4 Aktionen

- **ACCEPT** - Paket wird durchgelassen
- **DROP** - Paket wird gelöscht
- **REJECT** - Paket wird mit Fehlermeldung quittiert
  - TCP RST
  - ICMP Net/Host Unreachable
  - ICMP Port/Protocol Unreachable
  - ICMP Net/Host/Admin Prohibited
- **SNAT**, **DNAT**, **MASQUERADE** - Paket wird durch NAT umgeschrieben
- **REDIRECT** - Paket wird umgeleitet
- **LOG** - Paket wird durch System geloggt
- **ULOG** - Logeintrag wird an Applikation weitergereicht
- **QUEUE** - Paket wird an Applikation weitergereicht
- **TOS** - Type of Service des Pakets wird verändert

1.5 Kriterien und Extensions

Filterregeln können unter anderem basiert auf

- **MAC-Adresse**[1]
- **IP-Adresse und Port**
- **ein-/ausgehende Netzwerkkarte**
- **SPI**[2] **von AH**[3] **und ESP**[4] **Paketen (IPsec)**
- **DSCP (differentiated services) und TOS Feld**
- **ICMP, TCP, UDP**
  - Typen und Codes
  - TCP Flags und Optionen
- **Paketlänge**
- **Benutzer bzw. Gruppe** - falls das Paket lokal erzeugt wurde
- **Paketart** - Unicast, Broadcast, Multicast
- **Paketrate** - zur Begrenzung von Paketdurchsatz
- **TTL**

formuliert werden.

[1] MAC = Media Access Control
[2] Security Parameter Index
[3] Authentication Header
[4] Encapsulated Security Payload

1.6 Netfilter Beispiele mit iptables

```
# Setzen der Default Policy auf DROP für alle Chains
/sbin/iptables --policy INPUT   DROP
/sbin/iptables --policy FORWARD DROP
/sbin/iptables --policy OUTPUT  DROP
```

SSH Verbindungen mit stateful inspection erlauben

```
iptables --append FORWARD --protocol tcp \
         --source $EVERYWHERE --source-port $DYNA_PORTS \
         --destination $DMZ_NET --destination-port $SSH \
         --match state --state NEW,ESTABLISHED \
         --jump ACCEPT
iptables --append FORWARD --protocol tcp \
         --source $DMZ_NET --source-port $SSH \
         --destination $EVERYWHERE --destination-port $DYNA_PORTS \
         --match state --state ESTABLISHED \
         --jump ACCEPT
```

1.7 FTP Datenverbindungen filtern

```
# FTP Datenkanal Port 20 für aktives FTP
$IPTABLES --append FORWARD --protocol tcp \
          --source $DMZ_NET --source-port $FTPDATA \
          --destination $EVERYWHERE --destination-port $DYNA_PORTS \
          --match state --state ESTABLISHED,RELATED \
          --jump ACCEPT
$IPTABLES --append FORWARD --protocol tcp \
          --source $EVERYWHERE --source-port $DYNA_PORTS \
          --destination $DMZ_NET --destination-port $FTPDATA \
          --match state --state ESTABLISHED \
          --jump ACCEPT

# Datenübertragung bei passivem FTP
$IPTABLES --append FORWARD --protocol tcp \
          --source $EVERYWHERE --source-port $DYNA_PORTS \
          --destination $DMZ_NET --destination-port $DYNA_PORTS \
          --match state --state ESTABLISHED,RELATED \
          --jump ACCEPT
$IPTABLES --append FORWARD --protocol tcp \
          --source $DMZ_NET --source-port $DYNA_PORTS \
          --destination $EVERYWHERE --destination-port $DYNA_PORTS \
          --match state --state ESTABLISHED \
          --jump ACCEPT
```

1.8 Weitere Regelbeispiele

MAC-basiertes Filtern mit TCP Regel

```
iptables --insert INPUT --protocol tcp --source $SERVER \
         --destination $SELF --destination-port $SSH \
         --match mac --mac-source 00:60:97:11:d9:02 \
         --jump ACCEPT
```

Begrenzung der Paketrate

```
iptables --append INPUT --protocol udp \
         --source 0/0 --destination $SELF --destination-port $DNS \
         --match limit --limit 50/second --jump ACCEPT
```

IP Blacklisting für die Dauer von 60 Sekunden, falls jemand IP Spoofing über die externe Netzwerkkarte versucht

```
iptables -A FORWARD -m recent --update --seconds 60 -j DROP
iptables -A FORWARD -i eth0 -d 127.0.0.0/8 -m recent --set -j DROP
```

1.9 QoS und Traffic Shaping

Der Linux Kern besitzt einen exzellenten Routing Code[5]

- **feine Einstellung von Routing Queues**
  - PFIFO mit ToS (Default für Linux Router)
  - Token Bucket Filter (TBF)
  - Stochastic Fairness Queueing (SFQ)
  - Random Early Drop (RED)
  - Ingress Policer für eingehende Pakete
  - ...
- **Verteilen von Bandbreite nach Services und Maschinen**
  Netfilter kann Pakete durch Regeln markieren
- **Abfangen von Lastspitzen**
  - Begrenzen von eingehendem SMTP auf 1 Mbit/s
  - lokale Webserver bekommen maximal 75% der Anbindung
  - Reservieren von 10% für kritische Applikationen
- **Zerteilen einer Anbindung**
  - Verteilen einer Standleitung auf 10 Filialen
  - 10 Mbit/s $\longrightarrow$ 5 $\times$ 2 Mbit/s

[5] http://www.lartc.org/

1.10 Traffic Shaping im Einsatz

Szenario

- **Standleitung mit 2320 kbit/s**
- **Anbindung hat Services und LAN**
  HTTP, HTTPS, POP3/IMAP, SMTP, DNS, SSH, LAN
- **Spitzen zerstören Interaktivität**
- **Aufteilung der Bandbreite in folgende Klassen**
  - HTTP bekommt 594 kbit/s
  - HTTPS bekommt 464 kbit/s
  - DNS und SSH bekommen 48 kbit/s
  - POP3 und IMAP bekommen 348 kbit/s
  - SMTP bekommt 403 kbit/s
  - Rest bekommt 464 kbit/s

  Wenn 2320 kbit/s nicht ausgeschöpft, so können Klassen Bandbreite von anderen ausborgen.
- **Hierarchical Token Bucket (HTB)**
  - HTTP, HTTPS, DNS, SSH, POP3, IMAP und Rest bekommen Stochastic Fairness Queueing (SFQ) Queue
  - SMTP bekommt Random Early Drop (RED) Queue
- **eingehender SMTP Verkehr wird auf 403 kbit/s reduziert**

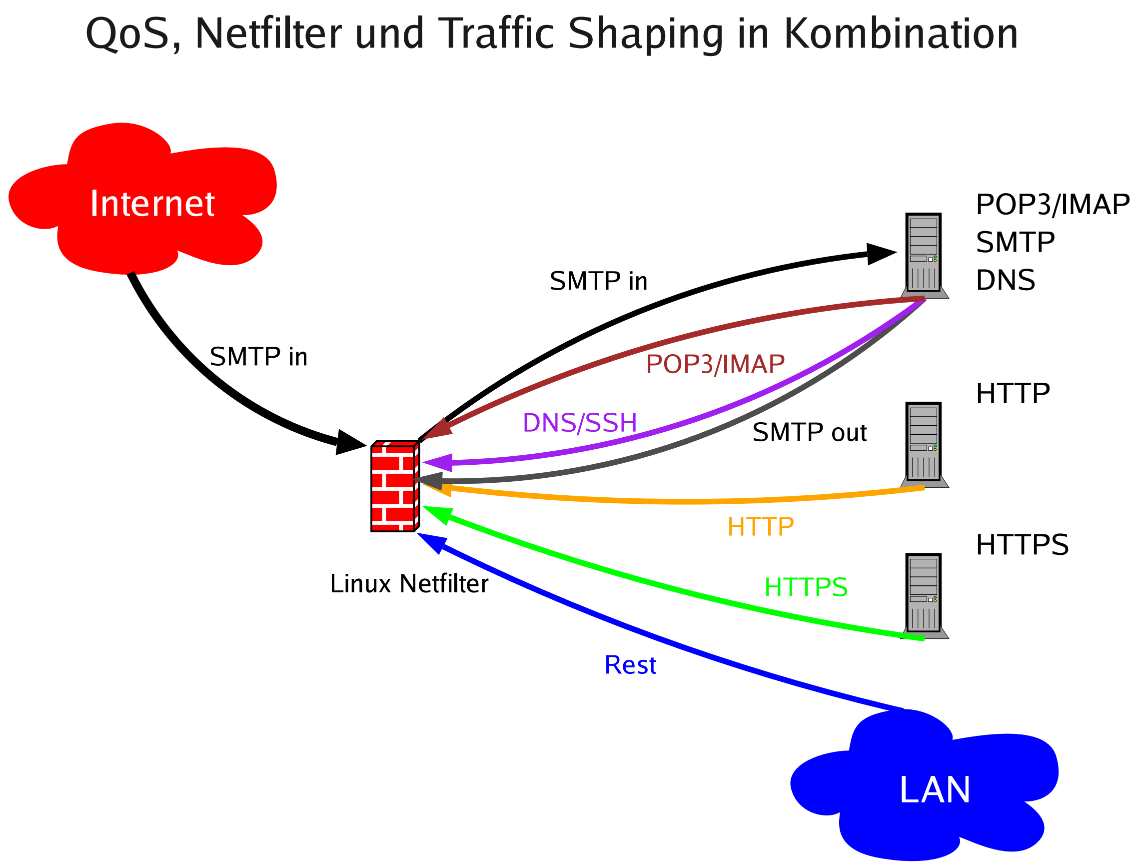

Abbildung 2: Das Diagramm zeigt Traffic Shaping an einem Linux Kern. Begrenzt sind SMTP Daten eingehend. Ebenso begrenzt sind DNS/SSH, POP3/IMAP, HTTP, HTTPS, SMTP und restliche Pakete ausgehend, jeweil in einer eigenen Bandbreitenklasse. Die einzelnen Klassen können sich von den anderen Klassen Bandbreite ausleihen, sollten die Limits pro Klasse nicht überschritten sein. Mit dieser Methode läßt sich der Datendurchfluß beliebig aufteilen und regeln.

1.11 Traffic Shaping im Einsatz - Überblick

1.12 IPsec im Kernel 2.6.x

- **IPsec in 2.6 basiert auf dem USAGI Projekt**[6]
- **große Ähnlichkeit mit IPsec unter FreeBSD und NetBSD**
- **benutzt Linux Cryptographic API**
- **Userspace Applikationen** `setkey` **und** `racoon`
- **mehrere Möglichkeiten**
  - manuelle Verbindung im Transportmodus oder Tunnelmodus
  - automatische Verbindung via Internet Key Exchange (IKE)[7] Protokoll
  - Preshared Keys (PSKs)
  - X.509 Zertifikate[8]
- **Interoperabilität**

[6] http://www.linux-ipv6.org/
[7] http://www.faqs.org/rfcs/rfc2409.html
[8] http://www.ipsec-howto.org/x507.html

1.13 IPsec Beispiele

Verschlüsselung allen Traffics ohne Tunnel

```
#!/usr/sbin/setkey -f

# Configuration for 192.168.1.100

# Flush the SAD and SPD
flush;
spdflush;

# Attention: Use this keys only for testing purposes!
# Generate your own keys!

# AH SAs using 128 bit long keys
add 192.168.1.100 192.168.2.100 ah 0x200 -A hmac-md5 \
0xc0291ff014dccdd03874d9e8e4cdf3e6;
add 192.168.2.100 192.168.1.100 ah 0x300 -A hmac-md5 \
0x96358c90783bbfa3d7b196ceabe0536b;

# ESP SAs using 192 bit long keys (168 + 24 parity)
add 192.168.1.100 192.168.2.100 esp 0x201 -E 3des-cbc \
0x7aeaca3f87d060a12f4a4487d5a5c3355920fae69a96c831;
add 192.168.2.100 192.168.1.100 esp 0x301 -E 3des-cbc \
0xf6ddb555acfd9d77b03ea3843f2653255afe8eb5573965df;

# Security policies
spdadd 192.168.1.100 192.168.2.100 any -P out ipsec
           esp/transport//require
           ah/transport//require;

spdadd 192.168.2.100 192.168.1.100 any -P in ipsec
           esp/transport//require
           ah/transport//require;
```

Quelle: IPsec HOWTO[9]

[9] http://www.ipsec-howto.org/x247.html

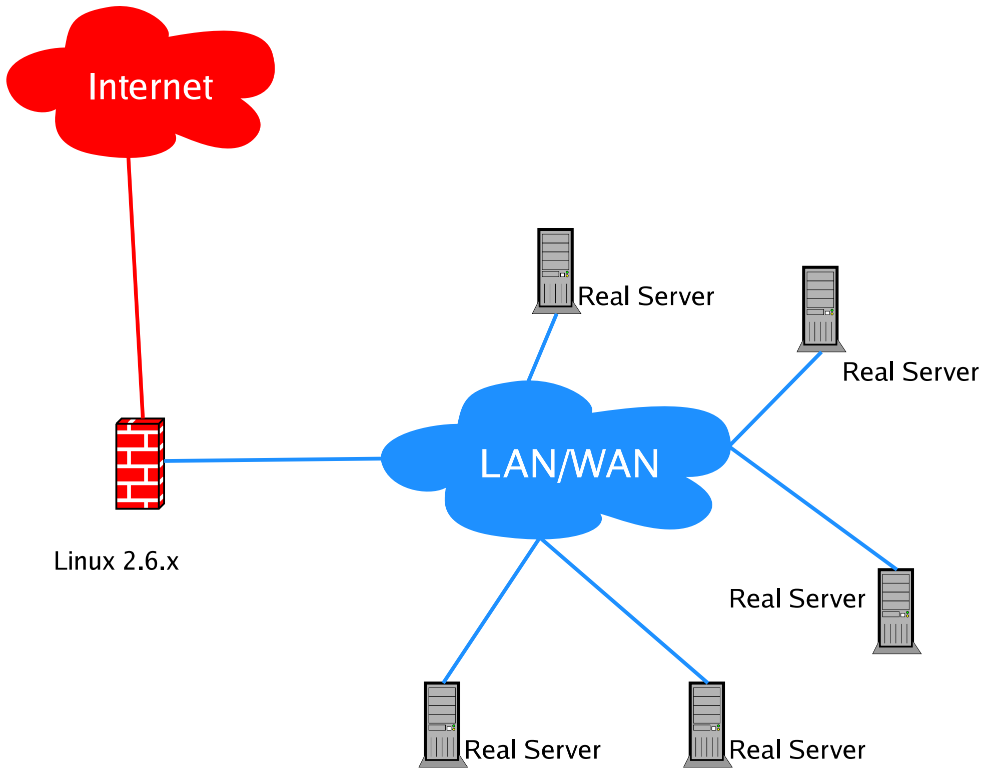

Abbildung 3: IP Virtual Server Load Balancing versteckt hinter eine IP mehrere physikalische Server. Das Verteilen der Anfragen und der Antworten kann über drei verschiedene Wege gelöst werden, abhängig von den Anforderungen an die Konfiguration.

1.14 IP Virtual Server Load Balancing

1.15 IP Virtual Server Load Balancing im Kernel 2.6.x

- **Virtual Server über NAT**
  - Serverapplikation beliebig (muß TCP/IP sein)
  - bis zu 20 Knoten sinnvoll wegen Last am LinuxDirector
- **Virtual Server über IP Tunnel**[10]
  - LinuxDirector hat IP Tunnel zu Knoten
  - Knotenserver können verteilt sein
- **Virtual Server über direktes Routing**[11]
  - Funktionsweise ähnlich IBM NetDispatcher
  - MAC wird umgeschrieben, Server antworten direkt

[10] http://www.linux-vs.org/VS-IPTunneling.html
[11] http://www.linux-vs.org/VS-DRouting.html

2 Open Source / Freie Software im Sicherheitsbereich

We have a policy that we are not being hacked.

– Lisa Kopp, Friendster rep, commenting on a security flaw[12]

Im Bereich der Computer- und Netzwerksicherheits gibt es neben Lösungen mit proprietärer Software auch Softwareangebote, die auf Open Source[13] oder Freier Software[14] aufbauen. Kommerzielle Produkte gibt es mit allen Softwarearten.
Diese Sektion stellt einige Softwarepakete vor, mit der man sein Netzwerk bzw. seine Server überprüfen kann, sei es in einem kompletten Auditing, zum Testen von vorhandenen Schutzmaßnahmen oder auch nur zur Diagnose von Problemen.

[12] http://www.wired.com/wired/archive/12.06/dating_pr.html
[13] http://www.opensource.org/docs/definition.php
[14] http://ffs.or.at/artikel/faq/

2.1 hping2 - Paketgenerator

`hping2`[15] ermöglicht es beliebige ICMP, TCP oder UDP Pakete an einen Host zu senden.

- Testen von Packetfilterregeln
- Port Scannen (auch mit gefälschten Source-Adressen, „Spoofing“)
- Testen der Netzperformance durch Variation
  - der Packetgröße
  - des Protokolls
  - der Type of Service (ToS) Kennung
  - der Fragmentierung
- MTU Ermittlung
- Traceroute mit verschiedenen Protokollen
- File Transfer (auch durch Firewalls hindurch)
- Pakete mit ungültiger Checksumme

[15]http://www.kyuzz.org/antirez/hping/

2.2 nmap - Network Mapper

- Ursprung: OS Identifikation anhand des TCP/IP „Fingerabdrucks“
- Scan-Arten umfassen
  - TCP connect() Scan
  - TCP SYN, FIN, Xmas oder NULL Scan
  - TCP Scanning per FTP Proxy (Bounce Attack)
  - SYN/FIN Scannen mit IP Fragmenten
  - TCP ACK und Window Scannen
  - UDP Scannen (ICMP Port Unreachable)
  - ICMP Scannen (Ping Sweep)
  - direktes RPC Scannen (nicht über `rpcinfo`)
  - OS Identifikation
- Versions-Identifikation durch Banneranalyse
- aktiv unterstützte OS Fingerabdruck Datenbank
- Angabe von Wildcards beim Ziel-Host
  z.B. `192.168.0.*,212.17.1-230.54-168`
- XML Output zur Verwendung in anderen Tools
- Vielfalt an Optionen bezüglich Timing für bessere Performance

2.2.1 nmap Scans

Scannen ausgewählter Ports mit TCP connect()

```
[root@agamemnon ~]# nmap -sT -sR -p 21,22,23,25,110,111,113,143,80,2049,3128 \
                    -P0 -O paladin.luchs.at

Starting nmap 3.77 ( http://www.insecure.org/nmap/ ) at 2004-11-26 12:54 CET
Interesting ports on chello084114134246.3.15.vie.surfer.at (212.17.78.195):
PORT     STATE    SERVICE          VERSION
21/tcp   filtered ftp
22/tcp   open     ssh
23/tcp   filtered telnet
25/tcp   open     smtp
80/tcp   filtered http
110/tcp  filtered pop3
111/tcp  filtered rpcbind
113/tcp  closed   auth
143/tcp  filtered imap
2049/tcp filtered nfs
3128/tcp filtered squid-http
Device type: general purpose
Running: Linux 2.4.X|2.5.X|2.6.X
OS details: Linux 2.4.18 - 2.6.7, Linux 2.4.20 (Itanium),
            Linux 2.4.3 SMP (RedHat), Linux 2.4.7 through 2.6.3,
            Linux 2.6.0 (x86), Linux 2.6.0-test5 - 2.6.0 (x86),
            Linux 2.6.3 - 2.6.7, Linux kernel 2.6.4 (x86)

Uptime 0.442 days (since Fri Nov 26 02:18:16 2004)

Nmap run completed -- 1 IP address (1 host up) scanned in 14.693 seconds
```

UDP Scan mit RPC- und Versioninfo

```
# nmap 3.77 scan initiated Fri Nov 26 14:15:05 2004 as:
# nmap -sU -sV -sR -r -O -oN /tmp/bazaar_udp.log -P0 192.168.15.2
Warning:  OS detection will be MUCH less reliable because we did
          not find at least 1 open and 1 closed TCP port
Interesting ports on server.foomatic.at (192.168.15.2):
(The 1462 ports scanned but not shown below are in state: closed)
PORT      STATE         SERVICE                  VERSION
53/udp    open          domain
67/udp    open|filtered dhcpserver
111/udp   open          rpcbind (rpcbind V2)     2 (rpc #100000)
123/udp   open|filtered ntp
137/udp   open          netbios-ns               Microsoft Windows netbios-ssn
                                                 (host: SERVER workgroup: GROUP)
138/udp   open|filtered netbios-dgm
631/udp   open|filtered unknown
817/udp   open          mountd (mountd V1-3)     1-3 (rpc #100005)
957/udp   open|filtered unknown
960/udp   open          status (status V1)       1 (rpc #100024)
2049/udp  open          nfs (nfs V2-4)           2-4 (rpc #100003)
3130/udp  open|filtered squid-ipc
3401/udp  open|filtered squid-snmp
32770/udp open|filtered sometimes-rpc4
32771/udp open|filtered sometimes-rpc6
32772/udp open          nlockmgr (nlockmgr V1-4) 1-4 (rpc #100021)
MAC Address: 00:10:5A:68:E9:95 (3com)
Too many fingerprints match this host to give specific OS details

# Nmap run completed at Fri Nov 26 14:41:09 2004:
# 1 IP address (1 host up) scanned in 1564.294 seconds
```

Ping Sweep

```
nmap -sP 10.2.2.*

Starting nmap V. 2.30BETA21 by fyodor@insecure.org ( www.insecure.org/nmap/ )
Host subdomains.somewhere.lan (10.2.2.1) appears to be up.
Host prank.somewhere.lan (10.2.2.2) appears to be up.
Host dazzled.somewhere.lan (10.2.2.3) appears to be up.
Host deregulated.somewhere.lan (10.2.2.10) appears to be up.
Host appliers.somewhere.lan (10.2.2.11) appears to be up.
Host rotary.somewhere.lan (10.2.2.12) appears to be up.
Host telegraphic.somewhere.lan (10.2.2.15) appears to be up.
Host swept.somewhere.lan (10.2.2.19) appears to be up.
Host sitting.somewhere.lan (10.2.2.20) appears to be up.
Host functionals.somewhere.lan (10.2.2.22) appears to be up.
Host Freetown.somewhere.lan (10.2.2.27) appears to be up.
Host followed.somewhere.lan (10.2.2.38) appears to be up.
Host yanked.somewhere.lan (10.2.2.40) appears to be up.
Host contender.somewhere.lan (10.2.2.42) appears to be up.
Host robberies.somewhere.lan (10.2.2.43) appears to be up.
Host parrots.somewhere.lan (10.2.2.48) appears to be up.
Host ingestion.somewhere.lan (10.2.2.51) appears to be up.
Host load.somewhere.lan (10.2.2.59) appears to be up.
Host outskirts.somewhere.lan (10.2.2.61) appears to be up.
Host designs.somewhere.lan (10.2.2.68) appears to be up.
Host winded.somewhere.lan (10.2.2.200) appears to be up.
Host bucolic.somewhere.lan (10.2.2.201) appears to be up.
Host besmirched.somewhere.lan (10.2.2.202) appears to be up.
Host crumbled.somewhere.lan (10.2.2.203) appears to be up.
Host requester.somewhere.lan (10.2.2.205) appears to be up.
Host Scot.somewhere.lan (10.2.2.250) appears to be up.
Nmap run completed -- 256 IP addresses (26 hosts up) scanned in 6 seconds
```

Suchen von SMTP Servern mittels `nmap -sT -p 25 10.2.2.1-254`

```
Starting nmap V. 2.54BETA1 by fyodor@insecure.org ( www.insecure.org/nmap/ )
Interesting ports on subdomains.somewherem.lan (10.2.2.1):
Port       State       Service
25/tcp     open        smtp

The 1 scanned port on prank.somewherem.lan (10.2.2.2) is: closed
Interesting ports on dazzled.somewherem.lan (10.2.2.3):
Port       State       Service
25/tcp     open        smtp

Interesting ports on deregulated.somewherem.lan (10.2.2.10):
Port       State       Service
25/tcp     open        smtp

Interesting ports on appliers.somewherem.lan (10.2.2.11):
Port       State       Service
25/tcp     open        smtp

Interesting ports on rotary.somewherem.lan (10.2.2.12):
Port       State       Service
25/tcp     open        smtp

The 1 scanned port on telegraphic.somewherem.lan (10.2.2.15) is: closed
The 1 scanned port on swept.somewherem.lan (10.2.2.19) is: closed
The 1 scanned port on sitting.somewherem.lan (10.2.2.20) is: closed
The 1 scanned port on functionals.somewherem.lan (10.2.2.22) is: closed
The 1 scanned port on Freetown.somewherem.lan (10.2.2.27) is: closed
The 1 scanned port on followed.somewherem.lan (10.2.2.38) is: closed
The 1 scanned port on contender.somewherem.lan (10.2.2.42) is: closed
The 1 scanned port on robberies.somewherem.lan (10.2.2.43) is: closed
The 1 scanned port on parrots.somewherem.lan (10.2.2.48) is: closed
The 1 scanned port on ingestion.somewherem.lan (10.2.2.51) is: closed
Interesting ports on load.somewherem.lan (10.2.2.59):
Port       State       Service
25/tcp     open        smtp

The 1 scanned port on outskirts.somewherem.lan (10.2.2.61) is: closed
The 1 scanned port on besmirched.somewherem.lan (10.2.2.202) is: closed
The 1 scanned port on crumbled.somewherem.lan (10.2.2.203) is: closed
The 1 scanned port on requester.somewherem.lan (10.2.2.205) is: closed
The 1 scanned port on Scot.somewherem.lan (10.2.2.250) is: closed
Nmap run completed -- 254 IP addresses (26 hosts up) scanned in 6 seconds
```

2.3 Nessus

2.3.1 Eigenschaften

- Client/Server Architektur
  ⟶ Client und Server kommunizieren verschlüsselt
  ⟶ Einsatz von Nessus Scannern mit Remote Zugriff
- Möglichkeit von Plug-In Modulen
  Vulnerability Datenbank - sehr aktuell
- NASL - Nessus Attack Scripting Language
- gleichzeitiges Testen beliebiger Hosts
  ⟶ abhängig von der Performance des Nessus Servers
  ⟶ konfigurierbar
- Service Erkennung
  Nessus prüft Service bei Connect ⟶ Entdecken von Services auf Nicht-Standard-Ports
  ⟶ entsprechendes Wiederholen von Tests
- Knowledge Base beim Scannen
  Plugins teilen die ermittelten Informationen über Hosts ⟶ Optimierung der Scans
- übersichtliche Reports
  NSR, ASCII Text, HTML, XML, LaTeX
- unabhängige Developer, breiter Support
  Test-Plugins sind Stunden nach Veröffentlichung eines Problems auf BugTraq, NTBugTraq, Incidents, Vuln-dev, Technotronic, ... verfügbar
- Scannen von Cron Jobs aus
  - Nessus Client läßt sich im Batch Modus einsetzen
  - automatischer Ablauf von periodischen Checks

2.3.2 Ablauf eines Nessus Scans

- Auswahl des `nessusd` Servers
- Auswahl der durchzuführenden Security Tests
- Auswahl der zu verwendenden Port Scanner
- Begrenzen der Tests
  Vorsicht: bei inkorrekter Netzwerkauswahl kann Nessus „entwischen“
  Wichtig bei Scans auf Zone Transfers
  Vorsicht: Nessus kann Server, Router, Firewalls und Workstations durch den Scan deaktivieren; daher Tests besonders gut auswählen.
- Setzen der plugin-spezifischen Optionen
- Durchführen des Scans
- Auswertung
  - Identifizieren von Fehlalarmen
  - Vergleich mit früheren Scans
  - Ableiten von Verbesserungen

Nessus Client Login

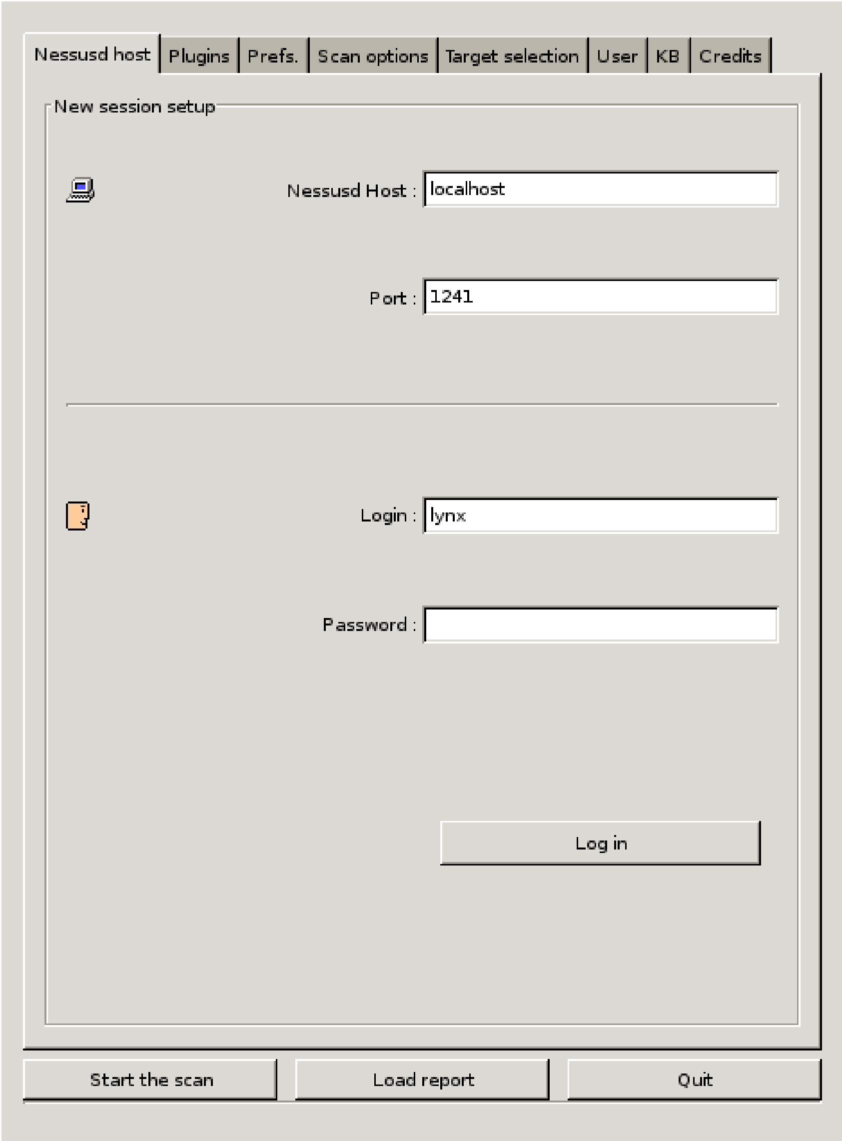

Abbildung 4: Login mit dem Nessus Client. Nessus verfügt über eine eigene Paßwortdatenbank zur Authorisierung. Die Kommunikation zwischen Server und Client ist ebenso verschlüsselt. Zugriff kann mit Zertifikaten und einer Benutzerdatenbank geregelt werden.

Auswahl der Nessus Plugins

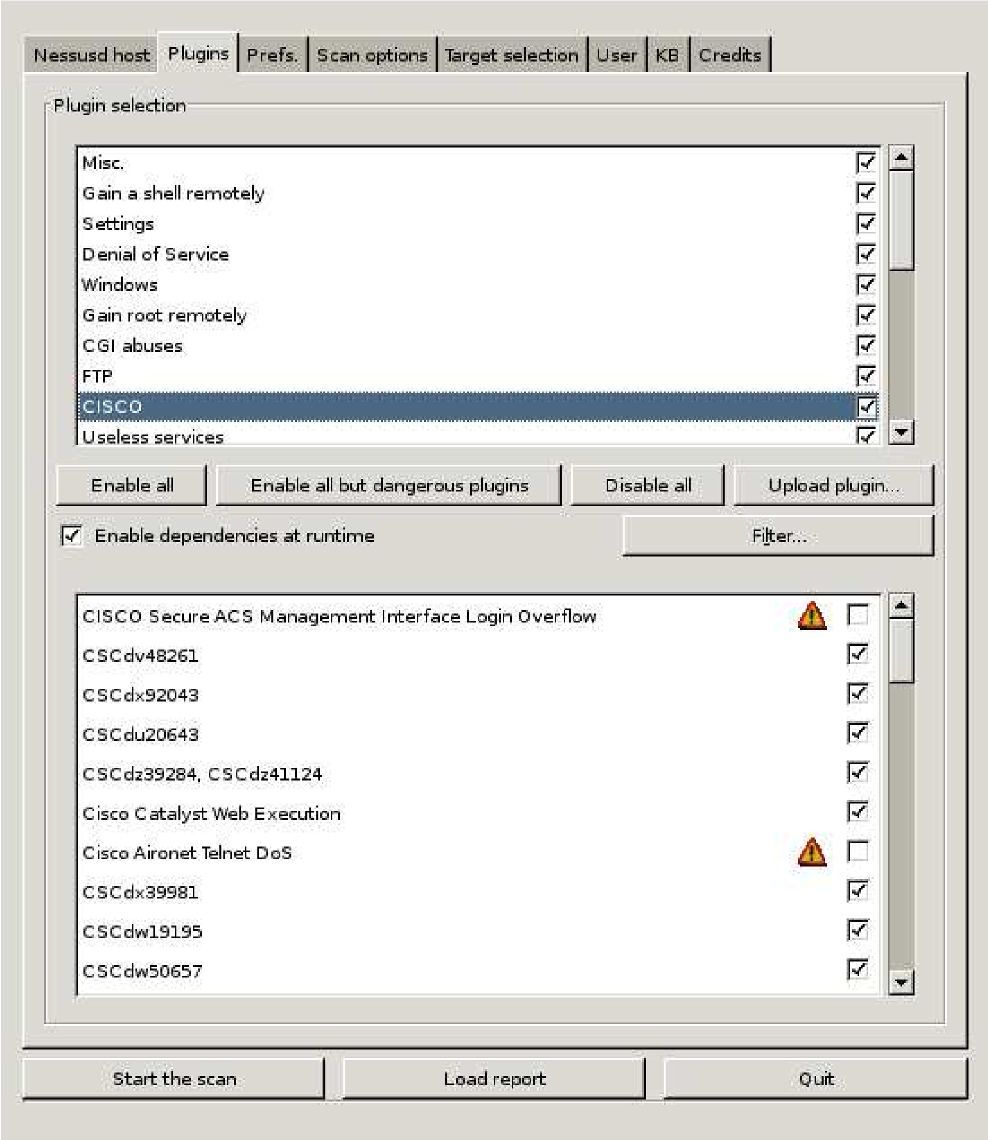

Abbildung 5: Auswählen der zu verwendenen Plugins für den Scan. **Wichtig:** Manche Module können Systeme zum Absturz bringen oder Netzwerk-Equipment wie Router oder Print-HUBs lahmlegen. Die Auswahl sollte daher sehr sorgfältig sein.

Abbildung 6: Zu jedem Plugin lassen sich detailliertere Informationen abrufen. Oft finden sich URLs zu Bug Reports oder standardisierten Beschreibungen.

Auswahl der Voreinstellungen

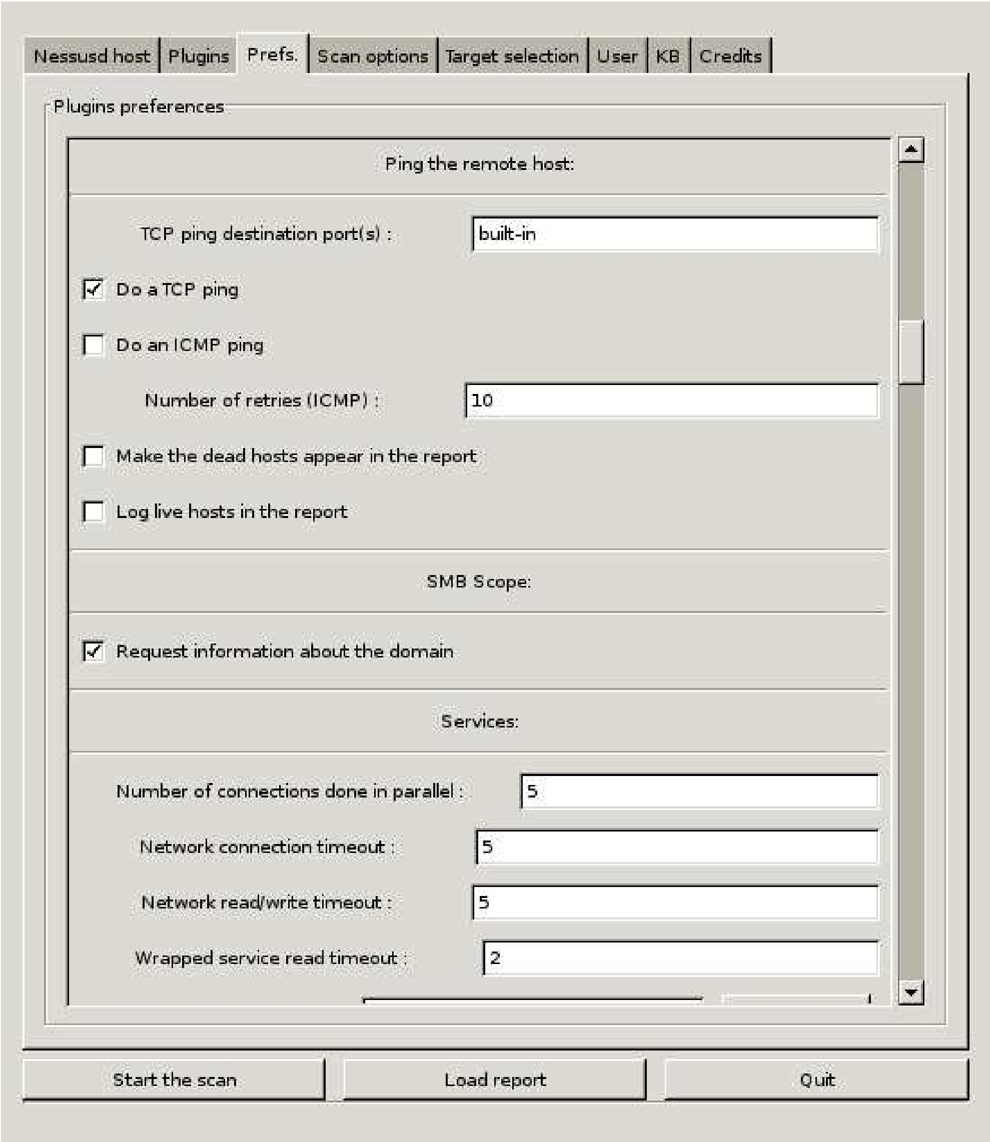

Abbildung 7: Einstellen der Voreinstellungen für die verwendeten Plugin-Module.

Auswahl der Ziele

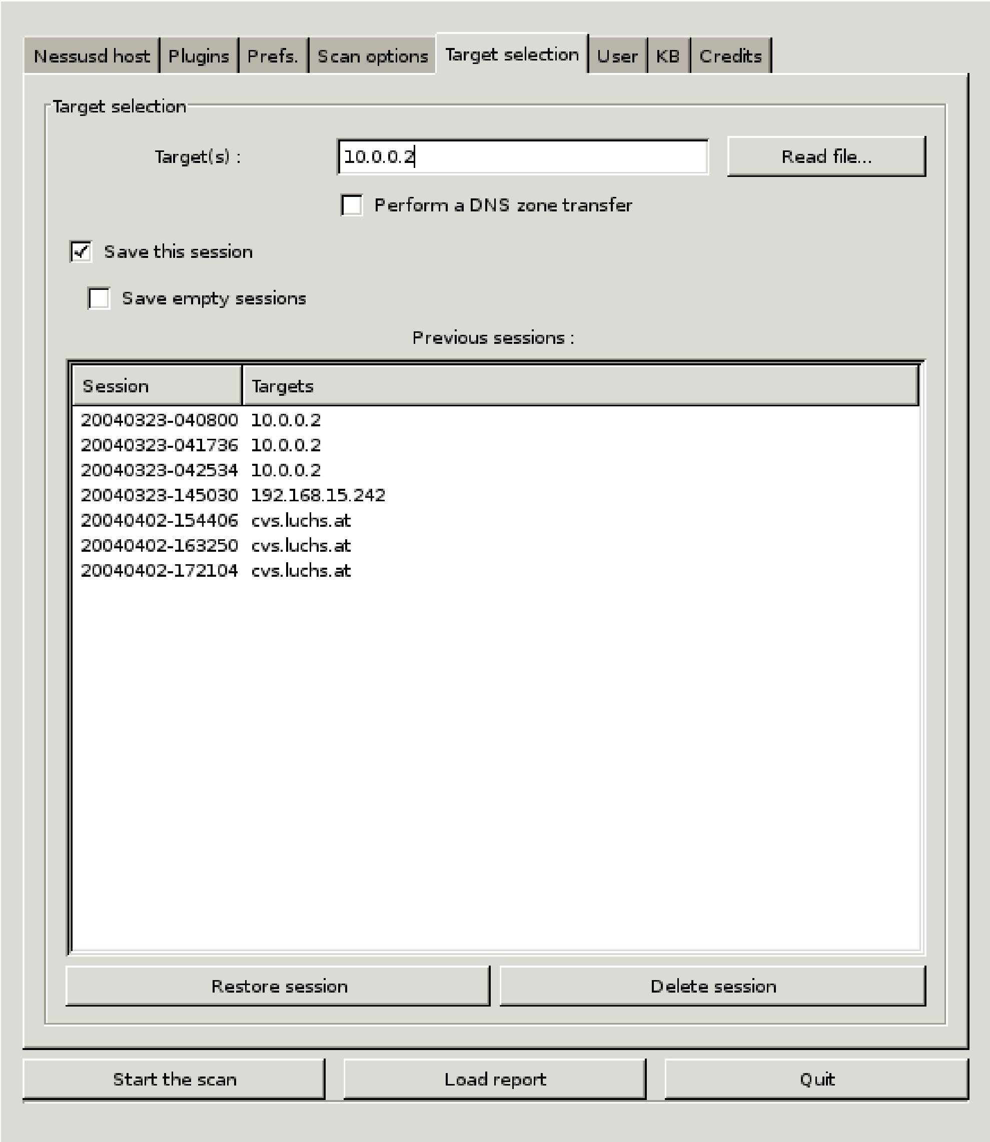

Abbildung 8: Auswahl der Ziele für den bevorstehenden Scan. Es können auch vorbereitete Textfiles und DNS Zone Transfers benutzt werden.

2.4 Network Intrusion Detection

Gezielte Fragen an Intrusion Detection Systeme

- **Welche Applikationen sollen überwacht werden?**
- **Welche Teile des Netzwerks sollen überwacht werden?**
- **Gibt es genug Expertise in house für die Betreuung?**
  $\longrightarrow$ *IDS müssen betreut werden*
  $\longrightarrow$ *Administratoren müssen eingebunden werden*
- **Wie groß darf das Backlog werden?**
  $\longrightarrow$ *Festlegung eines Zeitfensters*
  $\longrightarrow$ *Wahl geeigneter Auswertemethoden*
- **Wie schnell muß aus den Daten ein Alarm generiert werden?**
  $\longrightarrow$ *Anforderungen an die Auswertung*
- **Möchte man NIDS Daten als Sonden für Trends einsetzen?**
  $\longrightarrow$ *Warnungen vor bevorstehenden Attacken*

2.4.1 Network Intrusion Detection mit Snort

Die Fähigkeiten von Snort[16] umfassen

- **Real-Time Traffic Analyse**
- **Analyse von aufgezeichnetem Netzwerkverkehr**
- **Schreiben von Log-Daten in externe SQL Datenbank**
  normaler Betriebsmodus arbeitet mit Logs im Dateisystem
- **frei programmierbar durch Rule Sets**
  $\longrightarrow$ *beliebige Definition von Alerts*
  $\longrightarrow$ *Konfigurieren von automatischen Benachrichtigungen*
- **Flexible Response Option**
  Snort kann auf bestimmte Pakete mit einer Reihe von Antworten reagieren
- **Schnelligkeit**
  - Entkoppeln von Detektieren und Auswerten
  - Erfassen von 100 Mbit/s Link mit 80 Mbit/s

  Es gibt noch weitere Anstrengungen die Rate zu verbessern und schnellere Netzwerke zu beobachten.
- **Stealth Modus**
  Snort benötigt keinen TCP/IP Stack auf dem System

[16] http://www.snort.org/

2.4.2 Snort Filterregeln

- **Grundfunktionen** `alert / log / pass`
- unterstützt derzeit TCP, UDP & ICMP
  In Zukunft geplant: ARP, IGRP, GRE, OSPF, RIP, IPX
- **Snort kann die folgenden Paketinformationen testen**
  - `TTL` - Wert des IP Pakets
  - `ID` - IP Header Fragment ID
  - `DSIZE` - Datenlänge des IP Pakets
  - `Content` - bestimmter Inhalt in den Daten des IP Pakets (Pattern Matching)
  - `Flags` - TCP Flags
  - `SEQ` - TCP Sequenznummern
  - `ACK` - TCP ACK-Nummer
  - `Session` - beobachtet einzelne Sessions (Telnet, rlogin, FTP, HTTP)
  - `IType` - ICMP Typ
  - `ICode` - ICMP Code
  - `ICMP_Id` - ICMP Echo ID
  - `ICMP_Seq` - ICMP Echo Sequenznummer
  - `IPoption` - IP Options
    * rr - Record Route
    * eol - End of list
    * nop - No op
    * ts - Time Stamp
    * sec - IP security option
    * lsrr - Loose source routing
    * ssrr - Strict source routing
    * satid - Stream identifier
  - `RPC` - RPC Service/Applikations Aufrufe
- **Flexible Response -** `resp`
  Snort kann eine sich aufbauende Verbindung trennen

2.4.3 Snort Flexible Response

- **Senden von TCP-RST** an Empfänger, Sender oder beide
- **Senden von ICMP Nachrichten an den Sender**
  - ICMP Network Unreachable
  - ICMP Host Unreachable
  - ICMP Port Unreachable
- **Gegenmaßnahmen mit Checkpoint Firewall-1 über Snortsam**[17]
  - Liste von IPs, die nicht geblockt werden sollen
  - Kontrolle über Zeitintervalle
  - Rollback Support zur Aufhebung von Blocks
  - verschlüsselte Two-Fish Kommunikation zwischen Snortsam und Firewall-1
  - Plugin-Möglichkeit für andere Firewalls

Damit ist es möglich auf bestimmte Kriterien und Datenpakete zu reagieren.

[17] http://www.snortsam.net/

2.4.4 Snort Preprocessors

- **Flow Tracking**
  Portscans
- **IP Defragmentation**
  Detektiren von Fragmenten, DoS Erkennung
- **Stateful Inspection / Stream Reassembly**
  Detektieren von Portscans, Fingerprinting, ECN, Paketanomalien
- **HTTP Transaktionen normalisieren und prüfen**
  Dekodieren von Unicode-URIs, Abstimmen auf Server Capabilities
- **RPC Datenpakete normalisieren**
- **Back Orifice Detection**
- **Telnet und FTP Normalisierung**
- **Flow-Portscan Detektor**
- **ARP Spoofing Detektor**
  Überwachen bestimmter MAC/IP-Kopplungen

2.4.5 Aufbau von Snort Filterregeln

- **Grundfunktionen**
  - `alert` - generiert einen Alarm und loggt das Paket anschließend
  - `log` - loggt das Paket
  - `pass` - ignoriert das Paket
  - `activate` - generiert einen Alarm und aktiviert eine `dynamic` Regel
  - `dynamic` - bleibt untätig bis durch `activate` Rule aktiviert, fungiert dann als `log` Regel

  gefolgt von

  - IP Adressen
  - Ports
  - Richtung
- unterstützt derzeit IP, TCP, UDP & ICMP

2.4.6 Regeloptionen

Regeloptionen sind das Herz von Snort. Sie betreffen

- **Metadaten** `meta-data`
  - beschreiben die Regel
  - haben keinen Einfluß auf Wirkungsweise
- **Paketdaten** `payload`
  - Inspizieren der Daten (Text und binäre Muster möglich)
  - Perl-kompatible Regular Expressions
- **weitere Daten** `non-payload`
  - Kopfdaten (TTL, Protokoll,
  - Fragmentinformationen
  - RPC
- **Phase nach Detektierung** `post-detection`
  - TCP Session Extraktion
  - Flexible Response
  - Logging in bestimmte Ablagen

2.4.7 Beispielregeln

mögliche Attacke auf bzw. mit einem Webserver CGI:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
 (msg:"WEB-CGI HyperSeek hsx.cgi directory traversal attempt";
  flow:to_server,established; uricontent:"/hsx.cgi";
  content:"../../"; content:"%00"; distance:1;
  reference:bugtraq,2314;
  reference:cve,CAN-2001-0253;
  classtype:web-application-attack; sid:803; rev:8;)
```

BIND TSIG Buffer Overflow

```
alert udp $EXTERNAL_NET any -> $HOME_NET 53
 (msg:"DNS EXPLOIT named tsig overflow attempt";
  content:"|80 00 07 00 00 00 00 00 01|?|00 01 02|";
  reference:bugtraq,2303; reference:cve,CVE-2001-0010;
  classtype:attempted-admin; sid:314; rev:8;)
```

Fehlgeschlagener Oracle Login

```
alert tcp $EXTERNAL_NET any -> $SQL_SERVERS $ORACLE_PORTS
 (msg:"ORACLE misparsed login response";
  flow:from_server,established;
  content:"description=|28|"; nocase; content:!"connect_data=|28|sid=";
  nocase; content:!"address=|28|protocol=tcp"; nocase;
  classtype:suspicious-login; sid:1675; rev:4;)
```

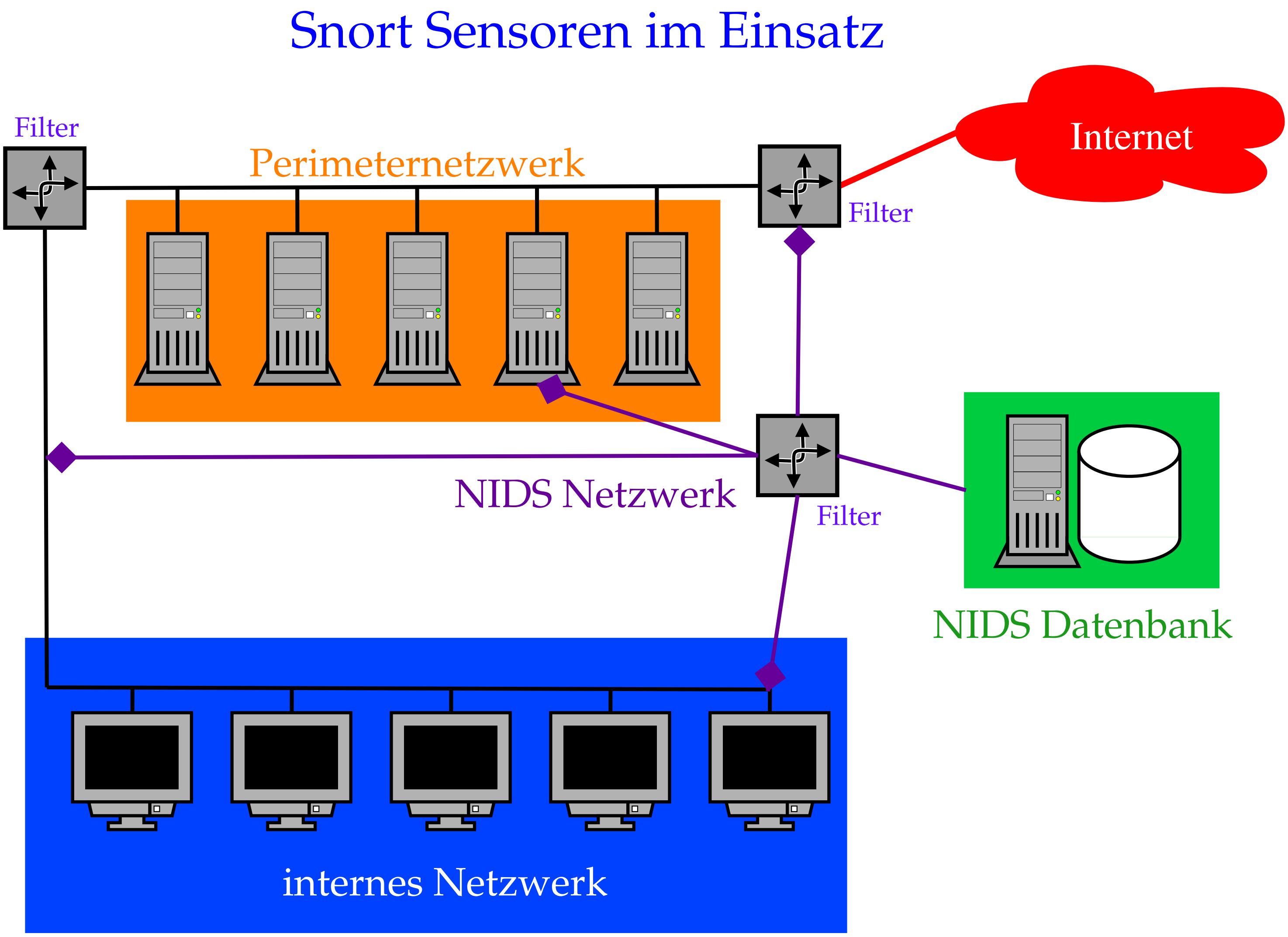

Abbildung 9: Schematischer Einsatz von Snort Sensoren in einem Netzwerk. Man kann Sensoren auf Paketfiltern, Servern oder auf speziellen Sondenservern unterbringen. Der sinnvollste Einsatz geschieht auf einer eigenen Maschine, die ausgewählten Netzwerkverkehr über einen dafür abgestellten Port an einem Switch erhält.

2.5 AIDE - Advanced Intrusion Detection Environment

- Host-based Integrity Checker
- generiert Datenbank mit Signaturen von Dateien und Verzeichnissen
  ⟶ AIDE kann in Postgres DB schreiben
- Generierung einer Signatur aus
  - Benutzer und Gruppe
  - Dateigröße
  - Inode Blocknummer
  - Permissions
  - Zeitmerkmale
    - atime - Access Time, letzter Zugriff
    - ctime - Creation Time, Erstellungsdatum
    - mtime - Modification Time, letzte Veränderung
- Berechnung von Checksummen
  - SHA1
  - MD5
  - RMD160
  - Tiger
  - CRC32 (optional mit mhash Library)
  - Haval (optional mit mhash Library)
  - Gost (optional mit mhash Library)

Einsatzgebiet von AIDE

- Installation und Konfiguration von AIDE
- Festlegen der Verzeichnisse, die mit Signatur versehen werden sollen
- Festlegen des Signaturgrades
  - Anzahl Merkmale
  - Anzahl Prüfsummen
- Generierung einer AIDE Datenbank als Referenz
- Sicherung von AIDE Konfiguration und Datenbank auf seperatem Datenträger

Beispielkonfiguration für AIDE

```
# Here are all the things we can check - these are the default rules
#
#p:      permissions
#i:      inode
#n:      number of links
#u:      user
#g:      group
#s:      size
#b:      block count
#m:      mtime
#a:      atime
#c:      ctime
#S:      check for growing size
#md5:    md5 checksum
#sha1:   sha1 checksum
#rmd160: rmd160 checksum
#tiger:  tiger checksum
#R:      p+i+n+u+g+s+m+c+md5
#L:      p+i+n+u+g
#E:      Empty group
#>:      Growing logfile p+u+g+i+n+S

# You can alse create custom rules - my home made rule definition goes like this
#
MyRule = p+i+n+u+g+s+b+m+c+md5+sha1

# Next decide what directories/files you want in the database

/etc p+i+u+g     #check only permissions, inode, user and group for etc
/bin MyRule      # apply the custom rule to the files in bin
/sbin MyRule     # apply the same custom rule to the files in sbin
/var MyRule
!/var/log/.*     # ignore the log dir it changes too often
!/var/spool/.*   # ignore spool dirs as they change too often
!/var/adm/utmp$  # ignore the file /var/adm/utmp
```

2.6 Tripwire

Tripwire[18] ist ebenso wie Aide ein Werkzeug, um Veränderungen an Dateien festzustellen. Es ist älter als Aide, stammt ursprünglich von der Purdue University. Mittlerweile gibt es Tripwire mit und ohne kostenpflichtigen Support. Der Code selbst unterliegt der GPL. Die Konfigurationen sind analog zu Aide, wobei sich die möglichen Optionen und Features im Detail unterscheiden.

[18] http://www.tripwire.org/

3 Content Filtering

Es gibt mehrere Methoden ein Content Filtering für Email- und HTTP/FTP-Proxies umzusetzen. Bei Servern wie Sendmail[19], Postfix[20] oder Squid[21] läßt sich sehr viel über geeignete Plugins steuern. Mit diesen Methoden lassen sich Anti-Virus-Maßnahmen, Spamfilter, MIME-Typ-Prüfungen und andere Dinge sehr leicht implementieren.

3.1 Content Filtering mit Squid Proxy

- **Squid[22]ist ein HTTP/FTP Forward & Reverse Proxy**
  - Web Proxy für Browser als Clients
  - Reverse Proxy vor Web Servern
    - Load Balancer
    - Entlastung des Web Servers bei statischem Content
- **Koppelung mehrerer Squids als Parent/Child Cluster**
- **Benutzung von externen Programmen als URL Filter**
  - URL Request wird vom Squid nach Prüfung der ACLs angenommen
  - URL wird an ein externes Programm weitergegeben
  - externes Programm kann URL beliebig modifizieren
  - Squid holt tatsächlich die URL, die der Filter zurückgibt

  Einsatz zum Schutz von Web Servern

[19]http://www.sendmail.org/
[20]http://www.postfix.org/
[21]http://www.squid-cache.org/
[22]http://www.squid-cache.org/

3.2 Squid als Reverse Proxy

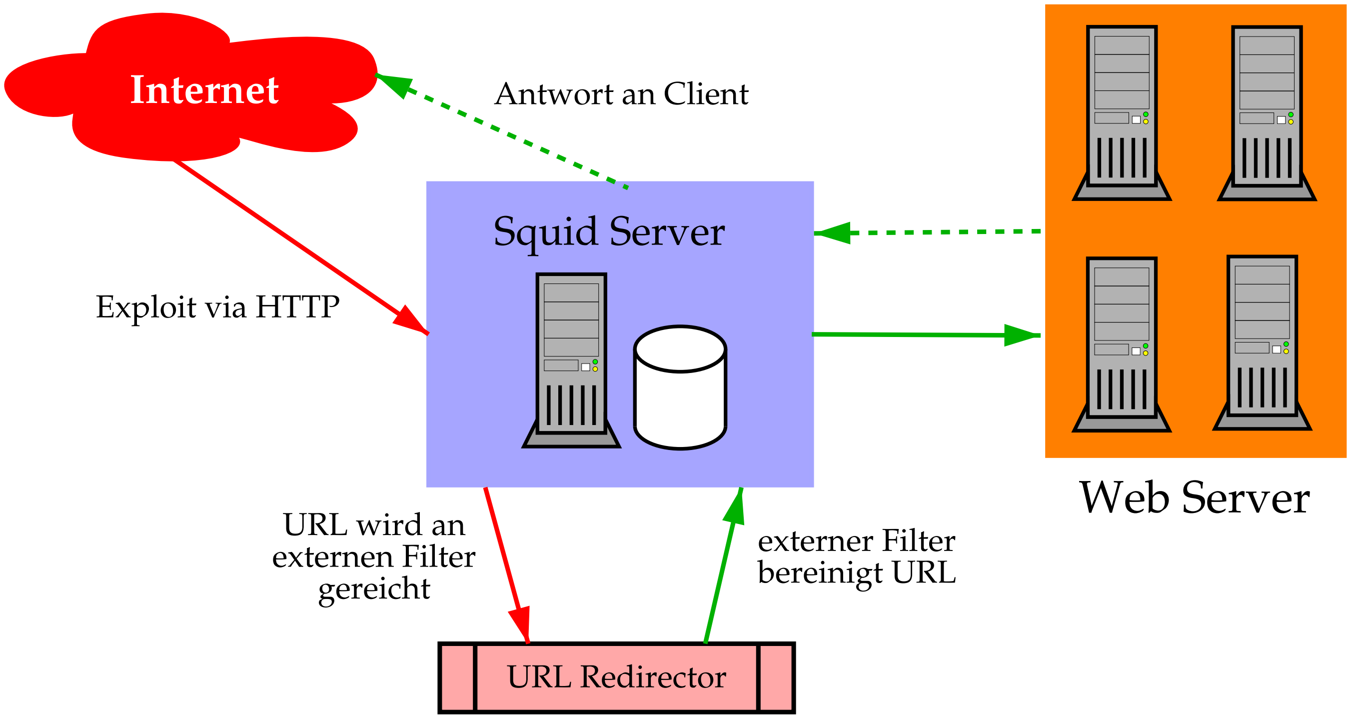

Abbildung 10: Ein Squid als Reverse Proxy mit URL Filtern vor einer Reihe von Web Servern. Das externe Filterprogramm am Squid prüft URLs auf Attacken und leitet diese um bzw. blockt sie. Zusätzlich stehen Loginformationen zur Verfügung, die abgeführt werden können.

3.3 Sendmail & Mail Filter API (Milter)

- **Sendmail[23] besitzt API um Mails in-transit zu prüfen**
  - Mail Filter API[24] (Milter) erlaubt Plugins
  - Sendmail MTA[25] stellt
    * Verbindungsinformation (Hostname & Adresse)
    * `HELO/EHLO` Parameter
    * Sender und Empfänger
    * Mailkopf
    * Mailkörper

    zur Verfügung
- **Filterplugin kann**
  - Verbindung, Sender oder Empfänger akzeptieren oder abbrechen
  - in die Nachricht eingreifen und sie verändern
    * Kopfdaten verändern oder hinzufügen
    * Sender und Empfänger hinzufügen oder entfernen
    * Nachricht ersetzen

[23] http://www.sendmail.com/

[24] API = Application Programmer's Interface, Schnittstelle für Programmierer

[25] MTA = Mail Transport Agent, Begriff für Mailtransportapplikation

3.4 MIMEDefang MTA Plugin

- **MIMEDefang[26] ist ein milter-fähiges Plugin**
- **implementiert in Perl und C**
- **erlaubt Verbindung zu anderen Filtern**
  - Spamassassin[27]
  - Antivirus Software (File::Scan, NAI McAfee uvscan, F-Secure, OpenAntiVirus, H+BEDV Antivir, Sophos, AvpLinux)
  - HTML Cleaner
- **blockiert gefährliche Dateitypen**
  - Microsoft Knowledge Base Article - 290497[28]
  - Attachments werden entfernt und durch Hinweis ersetzt
- **sehr gut konfigurierbar**

[26] http://www.mimedefang.org/
[27] http://www.spamassassin.org/
[28] http://support.microsoft.com/default.aspx?scid=kb;EN-US;290497

3.5 Filternder Mailproxy

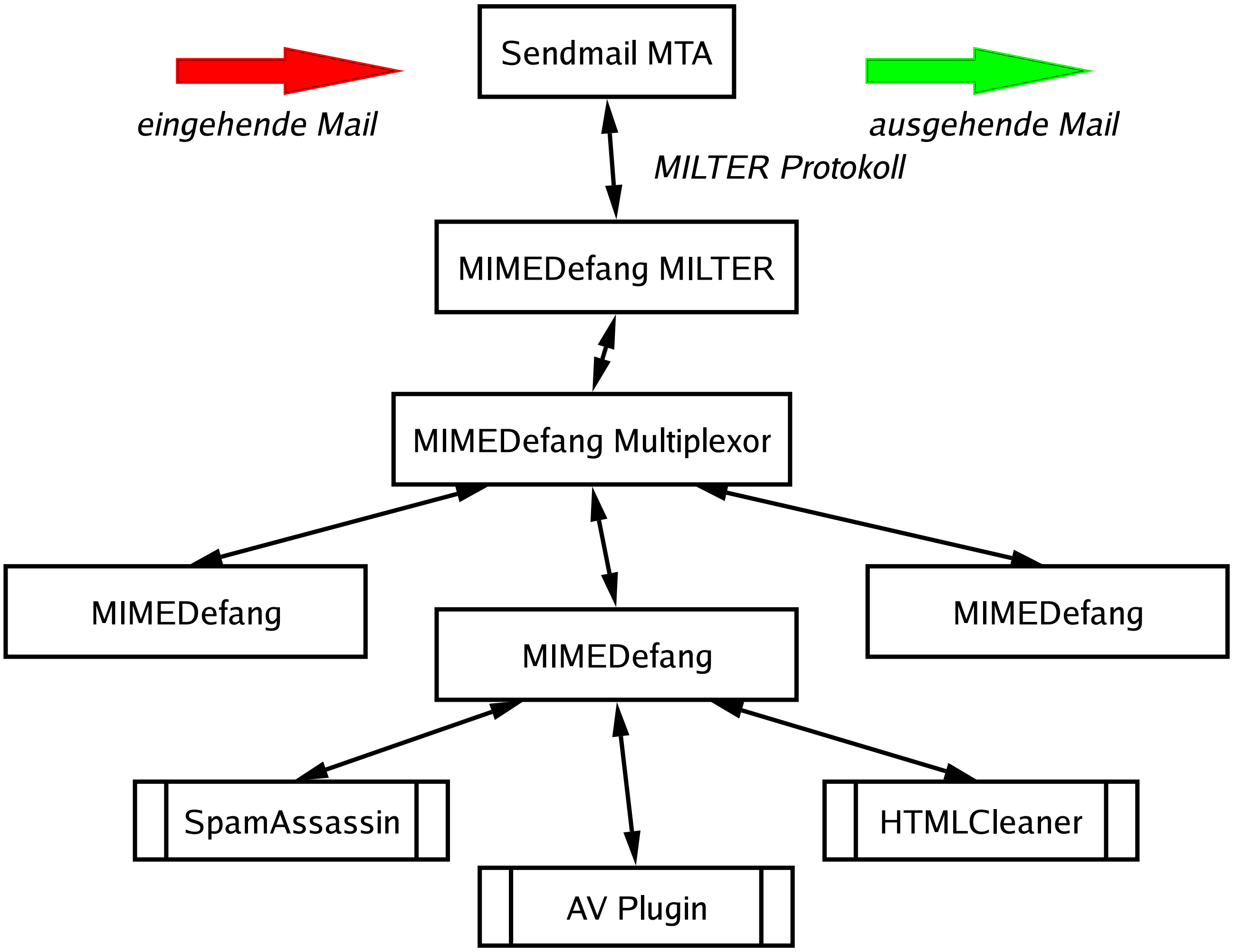

Abbildung 11: Schema eines Mailproxy mit Filterfunktionen. Die Emails werden über die Milter API des MTA weitergereicht. Der daran angeschlossene MIMEDefang Multiplexor verteilt die durchgeschleusten Nachrichten an einzelne MIMEDefang Instanzen, die eine Prüfung des Inhalts durchführen. Jede einzelne Instanz kann weitere Plugins einbinden. Damit ist eine leichte Anbindung an gängige Antivirus-Programme möglich.

3.6 Postfix MTA mit Zusätzen

- **filternder Mailproxy mit Postfix[29] ebenso möglich**
- **Anbindung der Filter über Local Mail Transfer Protocol (LMTP)[30]**
- **Möglichkeiten mit Regular Expressions zu Filtern**
  - Postfix hat eingebaute Kopf-/Inhaltsprüfungen[31]
  - Dateiendungen und andere Muster können so entfernt oder abgelehnt werden
- **Filtermöglichkeiten ähnlich reichhaltig wie bei Sendmail**
  - AMaViS[32]

[29] http://www.postfix.org/
[30] http://www.ietf.org/rfc/rfc2033.txt
[31] http://www.postfix.org/header_checks.5.html
[32] http://www.amavis.org/

3.7 AMaViS

- **Perl-basierendes Plugin**
- **eigener Server kommuniziert mit MTA**
- **Vielzahl von Anti-Virus Engines können verwendet werden**
  - Sophos Anti Virus via Sophie[33], ClamAV[34], OpenAntiVirus Scanner-Daemon (OAV)[35], TrendMicro Anti Virus via Trophie[36], AVG Anti-Virus[37], FRISK F-Prot Daemon[38], KasperskyLab AVP[39], H+BEDV AntiVir[40], CentralCommand Vexira Antivirus[41], Command AntiVirus[42], Symantec CarrierScan und AntiVirus Scan Engine[43], DrWeb Antivirus[44], F-Secure Antivirus[45], CAI InoculateIT, MkS_Vir, ESET Software NOD32[46], Norman Virus Control[47], Panda Antivirus[48], NAI McAfee AntiVirus[49], VirusBuster[50], CyberSoft VFind[51], Ikarus Anti-Virus[52], BitDefender[53]
- **Schnittstelle zu SpamAssassin**

[33] http://www.vanja.com/tools/sophie/
[34] http://www.clamav.net/
[35] http://www.openantivirus.org/
[36] http://www.vanja.com/tools/trophie/
[37] http://www.grisoft.com/
[38] http://www.f-prot.com/
[39] http://www.kaspersky.com/
[40] http://www.hbedv.com/
[41] http://www.centralcommand.com/
[42] http://www.commandsoftware.com/
[43] http://www.symantec.com/
[44] http://www.sald.com/
[45] http://www.f-secure.com/products/anti-virus/
[46] http://www.nod32.com/
[47] http://www.norman.com/products_nvc.shtml
[48] http://www.pandasoftware.com/
[49] http://www.nai.com/
[50] http://www.virusbuster.hu/en/
[51] http://www.cyber.com/
[52] http://www.ikarus-software.com/
[53] http://www.bitdefender.com/

A IDS in komplexen Netzwerken

> It's easy to cry 'bug' when the truth is that you've got a complex system and sometimes it takes a while to get all the components to co-exist peacefully.

– *Doug Vargas*

A.1 Was? - Sichten der Logquellen

- **Kombinieren von mehreren Logfiles verschiedener Maschinen**
  - Telefonlogs
  - `utmp` und `wtmp` Logs für Login-Zeiten
  - Prozeß Accounting Logs
  - Shell History Logs
  - `syslog`, NT Ereignisse, MTA Logs, etc.
  - Logs von Intrusion Detection Tools
  - Logs von anderen Sniffern & Sonden
- **Betriebsdaten via SNMP**
- **zeitliche Korrelation ist wichtig (Zeitserver)**
- **möglichst vielschichtig loggen, damit man die Integrität testen kann**
  - IP-Adresse **und** Hostname
    Forward und reverse DNS; Logging von IP-Adresse „fälschungssicherer"
  - MAC- und IP-Adresse
    Monitoring Tools für ARP Aktivität

A.2 Anomalien - Was sind Unregelmäßigkeiten?

- **Performanceveränderungen**
  $\longrightarrow$ *CPU Last, E/A Durchsatz, Plattenplatz*
- **Zustandsänderungen („Phasenübergänge“) von Systemen**
- **Interaktionen - Logins**
  - zu ungewöhnlichen Zeiten
  - von bestimmten Benutzern
  - von bestimmten Maschinen
- **ausbleibende Status Reports von Subsystemen**

Probleme:

- *Welche Log Informationen sind Indikatoren? Welche nicht?*
- *Wie sieht der „normale“ Betriebszustand aus?*
- *Wie schauen schlechte Nachrichten* wirklich *aus?*

A.3 Architektonische Überlegungen

- **Ausdehnung des Netzwerks**
  - Standorte und Standleitungen
    $\longrightarrow$ *insbesondere VPNs*
  - Zuständigkeitsbereiche
- **Einteilung der Systeme in Gruppen bzw. Sicherheitsstufen**
  - lokale Netze
  - Perimeternetze / DMZ
  - Funktionen der Server
    $\longrightarrow$ *Sichten der aktiven Applikationen*
- **Festlegen von Intrusion Detection Zonen**
  $\longrightarrow$ *Isolieren der Zugänge*
- **Planung des Logdatentransports**
  $\longrightarrow$ *zentrales Log Repository*
  $\longrightarrow$ *Absicherung der Transportkanäle zum Repository*

A.4 Einsatz von Data Mining Verfahren

- **Data Mining ist kein Real Time Verfahren**
  ⟶ *Daten stehen nicht sofort im richtigen Format bereit*
  ⟶ *Verzögerung durch Aufbereitung und Transport der Daten*
- **Etablierung einer Baseline**
  - Grundzustand der überwachten Systeme muß bekannt sein
  - Simulation von Angriffen bzw. Ausnahmesituationen muß möglich sein
  - ausreichend Zeit für Kalibrierung muß vorhanden sein
- **Organisation des Datenflusses muß geplant sein**
  ⟶ *Weg der Daten & Delta Load muß feststehen*
- **aufbereitete Daten können zur Visualisierung dienen**
  - Online Analytical Processing (OLAP)
  - direkte Aufbereitung der Datenbank in Graphen
  - Anwendung von Textminig Methoden auf Logs

  sehr nützliches Hilfsmittel für Systemadministration
- **Infrastruktur ermöglicht leichteres Erstellen von Statusberichten**

A.5 Automationsmöglichkeiten - Übersicht

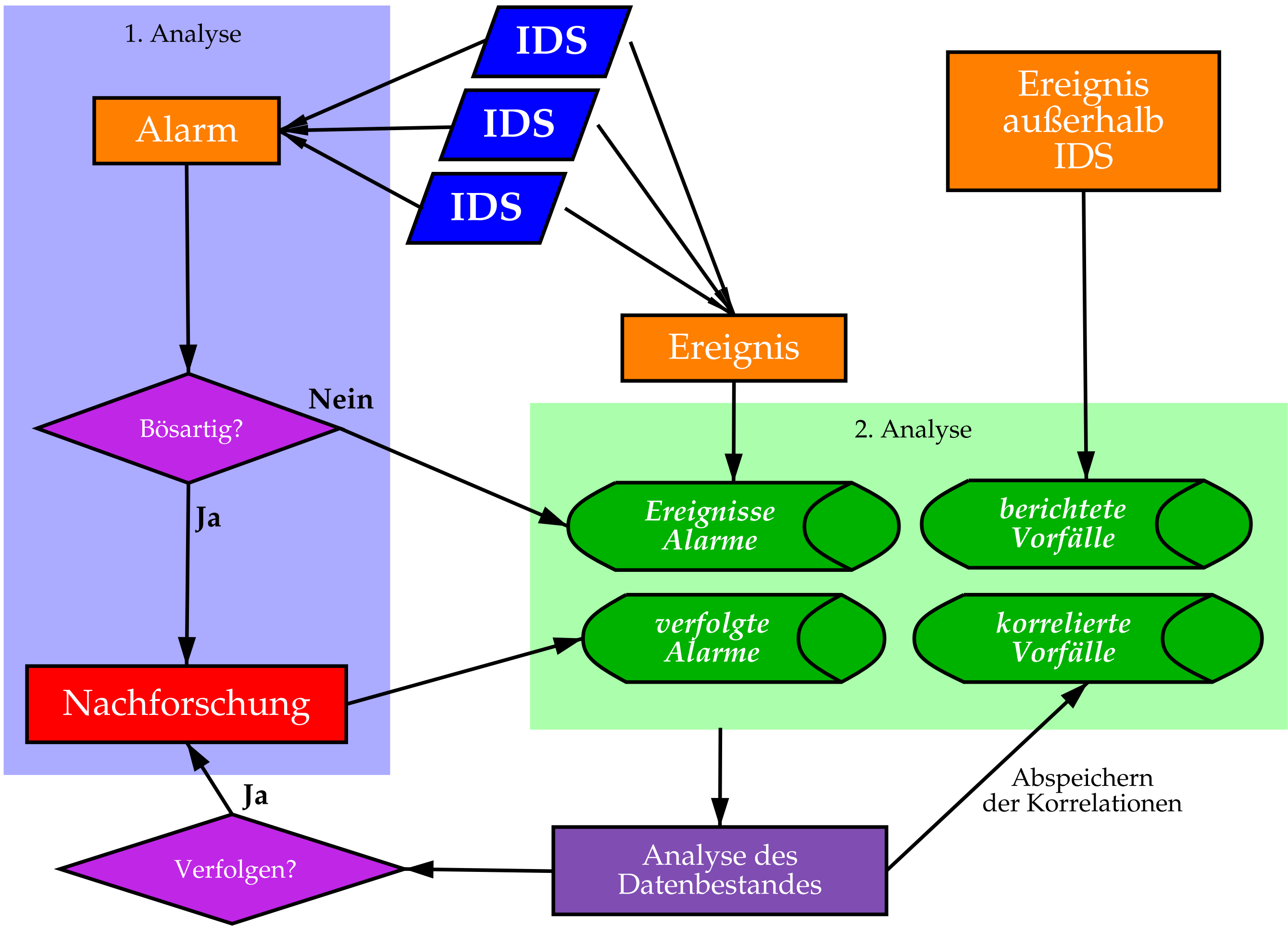

Abbildung 12: Die Abbildung zeigt die mehrstufige Auswertung von IDS Daten. In der ersten Stufe wird unmittelbar auf einen Alarm reagiert. Die Daten werden in Datenbanken gesammelt und durch weitere Prozesse analysiert. Dadurch können sich weitere Alarme ergeben, die behandelt werden müssen.
(`http://www.securityfocus.com/infocus/1201`)