SecurITy 2004 - Workshop

Firewalling, Filtering, Intrusion Monitoring & Anomaly Detection

23. Juni 2004

René Pfeiffer

pfeiffer@technikum-wien.at

rene.pfeiffer@paradigma.net

Technikum Wien

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.

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.)**

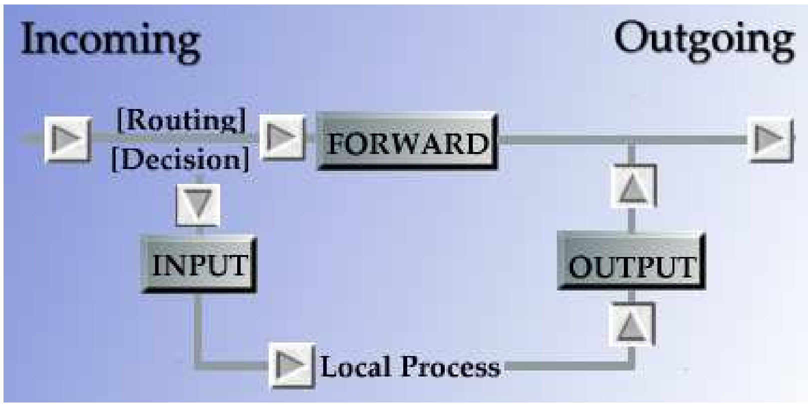

Abbildung 1: Das Diagramm zeigt den Fluß der Pakete durch eine Linux Netfilter Firewall. `INPUT` wird von eingehenden Paketen passiert, die für die Firewall selbst bestimmt sind. `FORWARD` betrifft ausschließlich weitergeleitete Pakete und durch `OUPUT` müssen alle ausgehenden Pakete, die die Firewall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

1.2 Paket■uß 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**

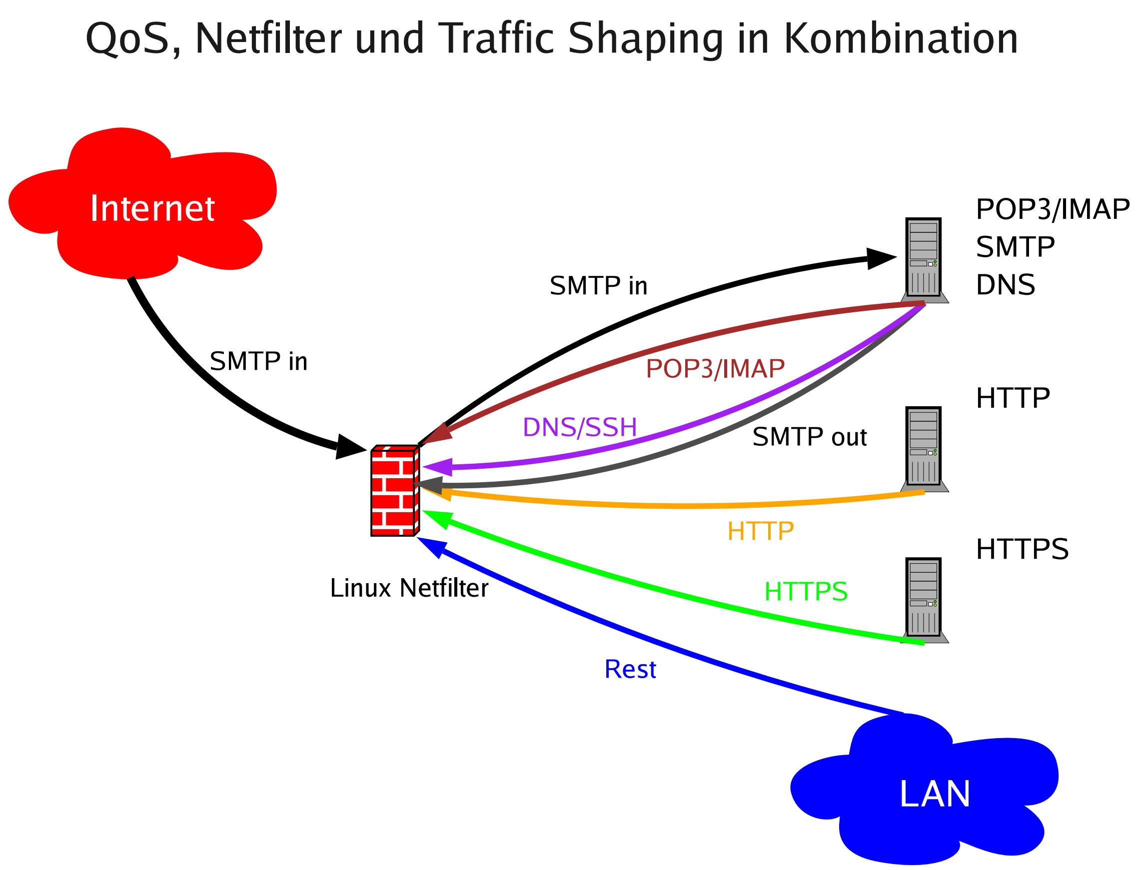

Abbildung 2: Das Diagramm zeigt Traffic Shaping an einem Linux Kern. Begrenzt sind SMTP Daten eingehend. Ebenso begrenzt sind DNS/SSH, POP3/IMAP, HTTP, HTTPS, SMTP und restliche Pakete ausgehend, jeweil in einer eigenen Bandbreitenklasse. Die einzelnen Klassen können sich von den anderen Klassen Bandbreite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

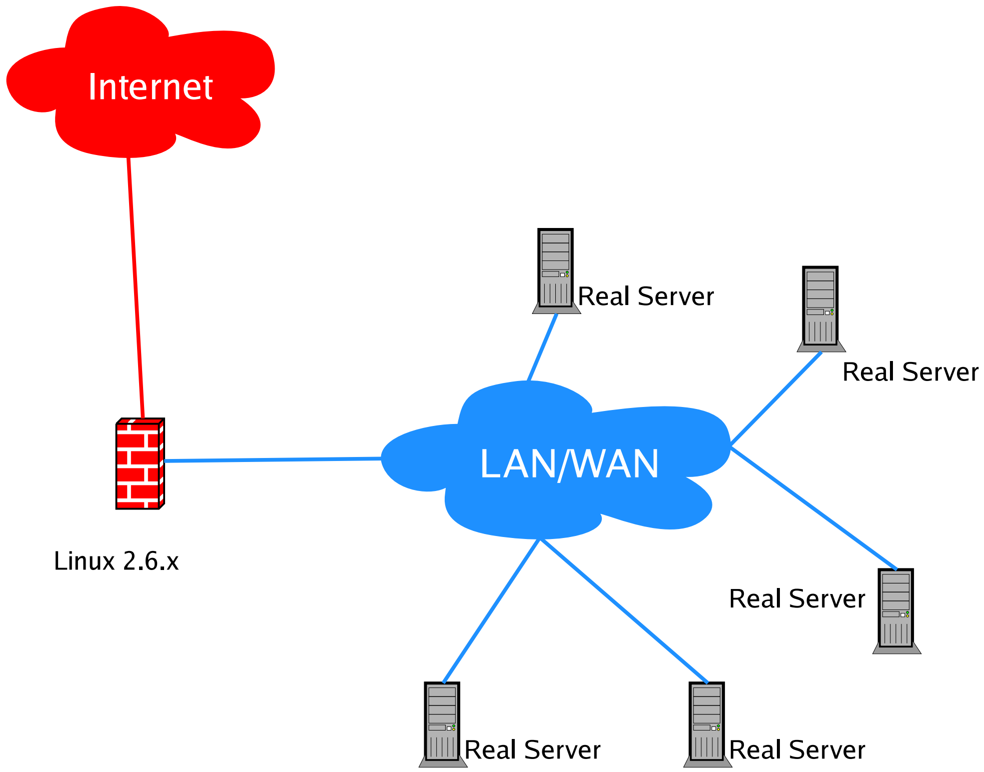

Abbildung 3: IP Virtual Server Load Balancing versteckt hinter eine IP mehrere physikalische Server. Das Verteilen der Anfragen und der Antworten kann über drei verschiedene Wege gelöst werden, abhängig von den Anforderungen an die Konfiguration.

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 Content Filtering für Mail und HTTP/FTP Proxies

> Charon, in Greek mythology, is the ferryman of the dead. The souls of the deceased are brought to him by Hermes, and Charon ferries them across the river Acheron. He only accepts the dead which are buried or burned with the proper rites, and if they pay him an obolus (coin) for their passage. For that reason a corpse had always an obolus[12] placed under the tongue.

– *Encyclopedia Mythica*[13]

[12] Occasionally, a danace – an ancient Persian coin which is worth rather more than the Greek obolus – was placed in the mouth of the dead.

[13] http://www.pantheon.org/articles/c/charon.html

2.1 Content Filtering mit Squid Proxy

- **Squid[14]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

[14]http://www.squid-cache.org/

2.2 Squid als Reverse Proxy

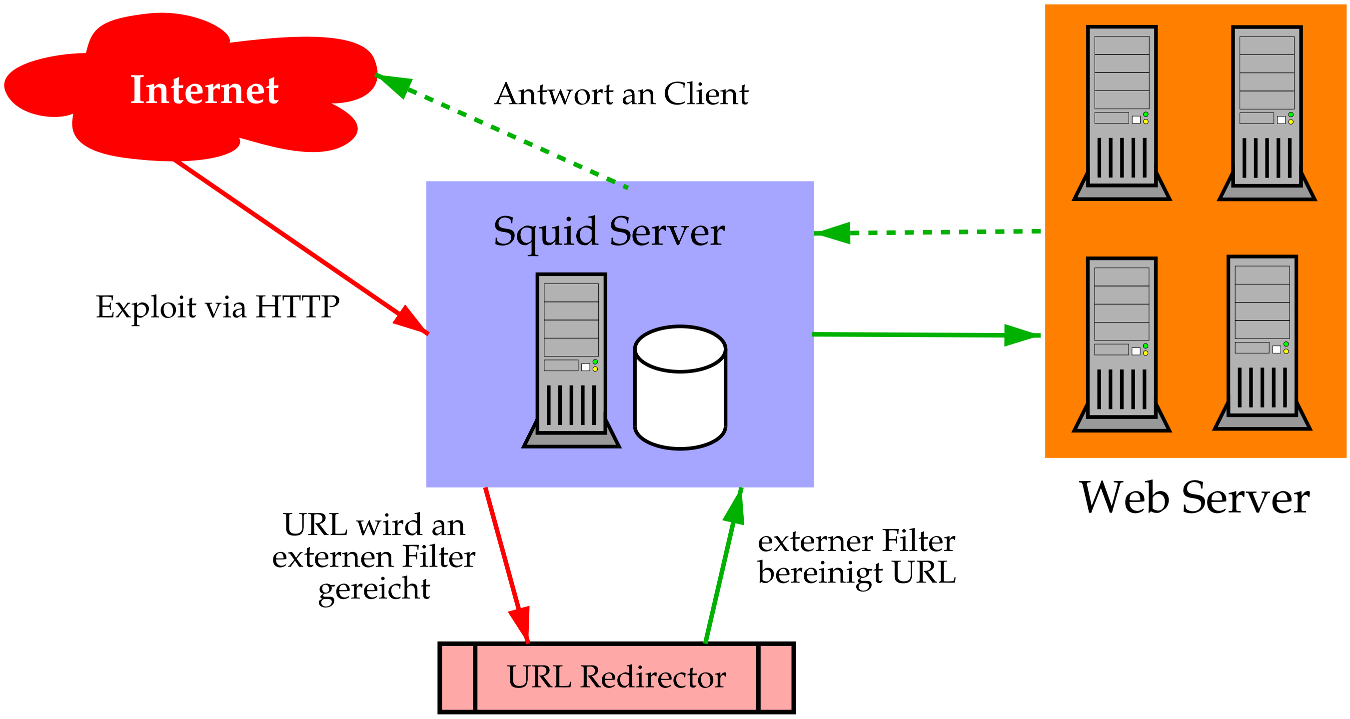

Abbildung 4: Ein Squid als Reverse Proxy mit URL Filtern vor einer Reihe von Web Servern. Das externe Filterprogramm am Squid prüft URLs auf Attacken und leitet diese um bzw. blockt sie. Zusätzlich stehen Loginformationen zur Verfügung, die abgeführt werden können.

2.3 Sendmail & Mail Filter API (Milter)

- **Sendmail[15] besitzt API um Mails in-transit zu prüfen**
  - Mail Filter API[16] (Milter) erlaubt Plugins
  - Sendmail MTA[17] 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

[15] http://www.sendmail.com/
[16] API = Application Programmer's Interface, Schnittstelle für Programmierer
[17] MTA = Mail Transport Agent, Begriff für Mailtransportapplikation

2.4 MIMEDefang MTA Plugin

- **MIMEDefang[18] ist ein milter-fähiges Plugin**
- **implementiert in Perl und C**
- **erlaubt Verbindung zu anderen Filtern**
  - Spamassassin[19]
  - 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[20]
  - Attachments werden entfernt und durch Hinweis ersetzt
- **sehr gut konfigurierbar**

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

2.5 Filternder Mailproxy

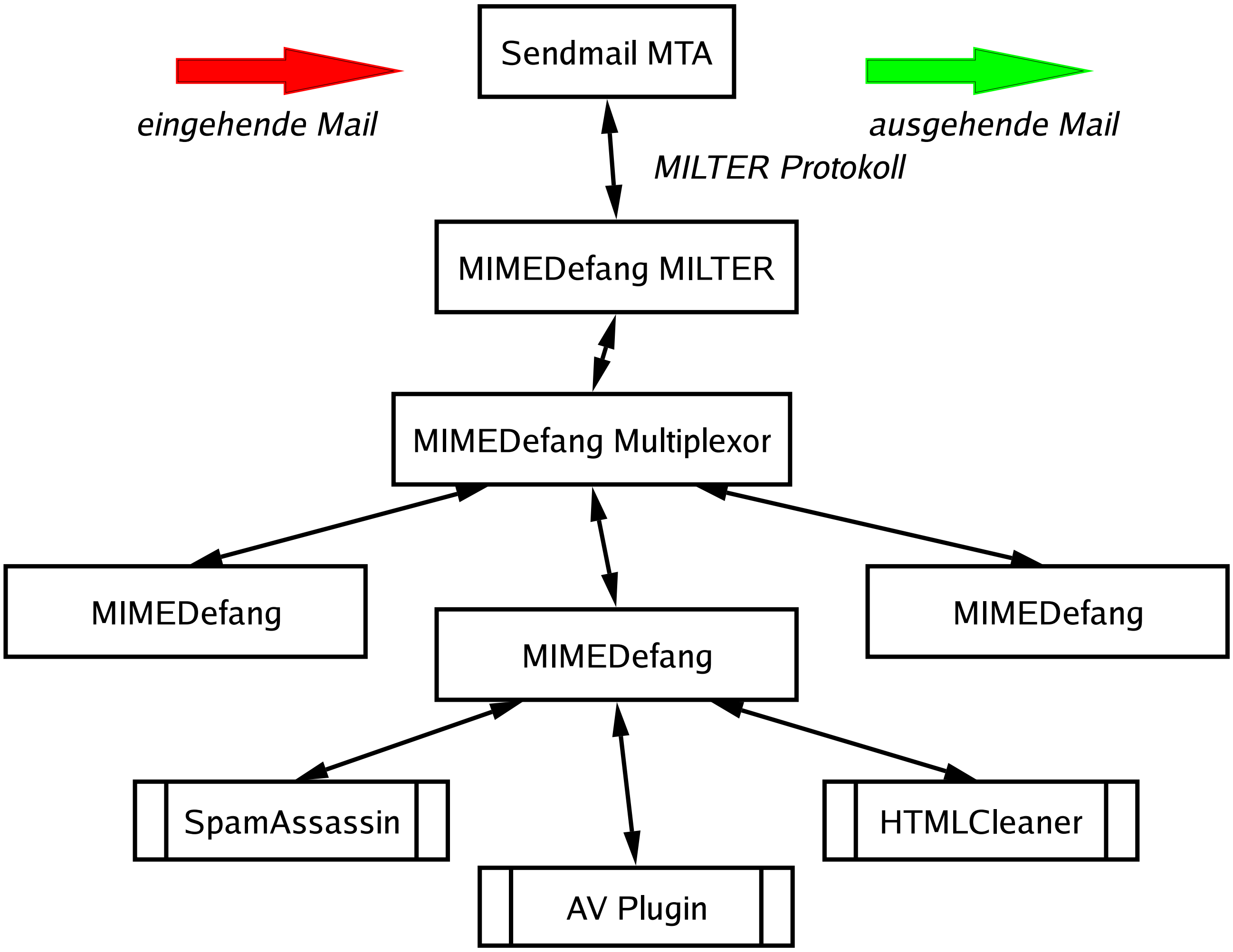

Abbildung 5: Schema eines Mailproxy mit Filterfunktionen. Die Emails werden über die Milter API des MTA weitergereicht. Der daran angeschlossene MIMEDefang Multiplexor verteilt die durchgeschleusten Nachrichten an einzelne MIMEDefang Instanzen, die eine Prüfung des Inhalts durchführen. Jede einzelne Instanz kann weitere Plugins einbinden. Damit ist eine leichte Anbindung an gängige Antivirus-Programme möglich.

2.6 Postfix MTA mit Zusätzen

- **filternder Mailproxy mit Postfix[21] ebenso möglich**
- **Anbindung der Filter über Local Mail Transfer Protocol (LMTP)[22]**
- **Möglichkeiten mit Regular Expressions zu Filtern**
  - Postfix hat eingebaute Kopf-/Inhaltsprüfungen[23]
  - Dateiendungen und andere Muster können so entfernt oder abgelehnt werden
- **Filtermöglichkeiten ähnlich reichhaltig wie bei Sendmail**

[21] http://www.postfix.org/
[22] http://www.ietf.org/rfc/rfc2033.txt
[23] http://www.postfix.org/header_checks.5.html

3 Intrusion Detection Systeme

The Cat only grinned when it saw Alice. It looked good-natured, she thought: still it had *very* long claws and a great many teeth, so she felt that it ought to be treated with respect.

'Cheshire Puss,' she began, rather timidly, as she did not at all know whether it would like the name: however, it only grinned a little wider. 'Come, it's pleased so far,' thought Alice, and she went on. 'Would you tell me, please, which way I ought to go from here?'

'That depends a good deal on where you want to get to,' said the Cat.

'I don't much care where–' said Alice.

'Then it doesn't matter which way you go,' said the Cat.

'–so long as I get *somewhere*,' Alice added as an explanation.

– *„Alice's Adventures in Wonderland"*, *Lewis Carroll*

3.1 Intrusion Detection Vielfalt

- **Network Intrusion Detection Systems (NIDS)**
  beobachten Pakete, die über das Netzwerk gesendet und empfangen wurden
  - Nachvollziehbarkeit von Vorfällen
  - Archivierung von Netzwerkaktivität
  - erfordert unter Umständen ein eigenes Netzwerk von Sonden
  - Logging alleine reicht nicht
    $\longrightarrow$ *Auswertung und Monitoring ist kritische Punkt*
- **System Integrity Verifiers (SIV)**[24]
  überwachen kritische Bereiche eines laufenden Systems (Benutzerrechte, Binaries, Konfigurationen, ...)
  - Systeme werden bei Installation mit Signatur versehen
  - „Fingerabdrucks" von Binaries durch Checksummen
  - Vergleich mit archivierten Signaturen
- **Log File Monitore (LFM)**
  durchsuchen Logfiles in regelmäßigen Abständen nach Anomalien
  - Koordinieren der Logs kann ein Problem werden
    $\longrightarrow$ *Time Stamps*
    $\longrightarrow$ *Zusammenführen der Logs von verschiedenen Orten*
  - Größe der Logs
    $\longrightarrow$ *Einsatz von Datenbanken*
    $\longrightarrow$ *Einsatz von Methoden des Data Minings*
- **Täuschen - Deceptions Systems**
  Vereiteln von Portscans, Ausgabe von Falschinformationen, Umschreiben der Mailheader

[24] Der Begriff Host Intrusion Detection System (HIDS) ist auch gebräuchlich.

3.2 IDS Sinnfragen

- **Gegenprüfen der Paketfilter - Watching the Watchmen**
  - Portscanner und Packet Shaper als „Aggressoren“
  - „vergessene“ Ports
  - Logging erzeugt Protokoll, welches den Zustand wiedergibt
- **Aufspüren von Attacken durch Kanäle, die die Firewall passieren**
- **Festhalten von fehlgeschlagenen Attacken**
  $\longrightarrow$ *Ermittlung von Trends bei den Eindringversuchen*
- **Sonden in interne Netzwerke**
  interne Netze bergen oft versteckte Gefahrenquellen
  - skriptfähige Mail User Agents
  - makrofähige Programme mit Schnittstellen zu anderer Software oder dem System
  - skriptfähige Web-Browser
  - eingeschleuste trojanische Pferde & Viren

  LANs sind in der Regel nicht mehr als sichere Netzwerke zu betrachten.
- **Qualitätskontrolle des Sicherheitskonzepts**

3.3 Network Intrusion Detection mit Snort

Die Fähigkeiten von Snort[25] 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 Wege die Rate zu verbessern und schnellere Netzwerke zu beobachten.
- **Stealth Modus**
  Snort benötigt *keinen TCP/IP Stack* auf dem System

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

3.4 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

3.5 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

3.6 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

3.7 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;)
```

3.8 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**[26]
  - 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.

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

3.9 Entkopplung von Detektieren und Auswerten

- **Snort beschränkt sich auf Detektieren**
  - Format *unified output*
  - Aufzeichnung aller Ereignisse und Pakete binär
- **Barnyard[27] wertet aus**
  - Transferieren der Alerts in mySQL/Postresql DB
    - weitere Auswertung
    - Verwendung von Frontends wie ACID[28] oder SGUIL[29]
  - Extrahieren der Pakete in libpcap Format
- **Barnyard kann als Dämon parallel laufen**

[27] http://www.snort.org/dl/barnyard/
[28] http://www.andrew.cmu.edu/user/rdanyliw/snort/snortacid.html
[29] http://sguil.sourceforge.net/

3.10 Snort Sensoren im Einsatz

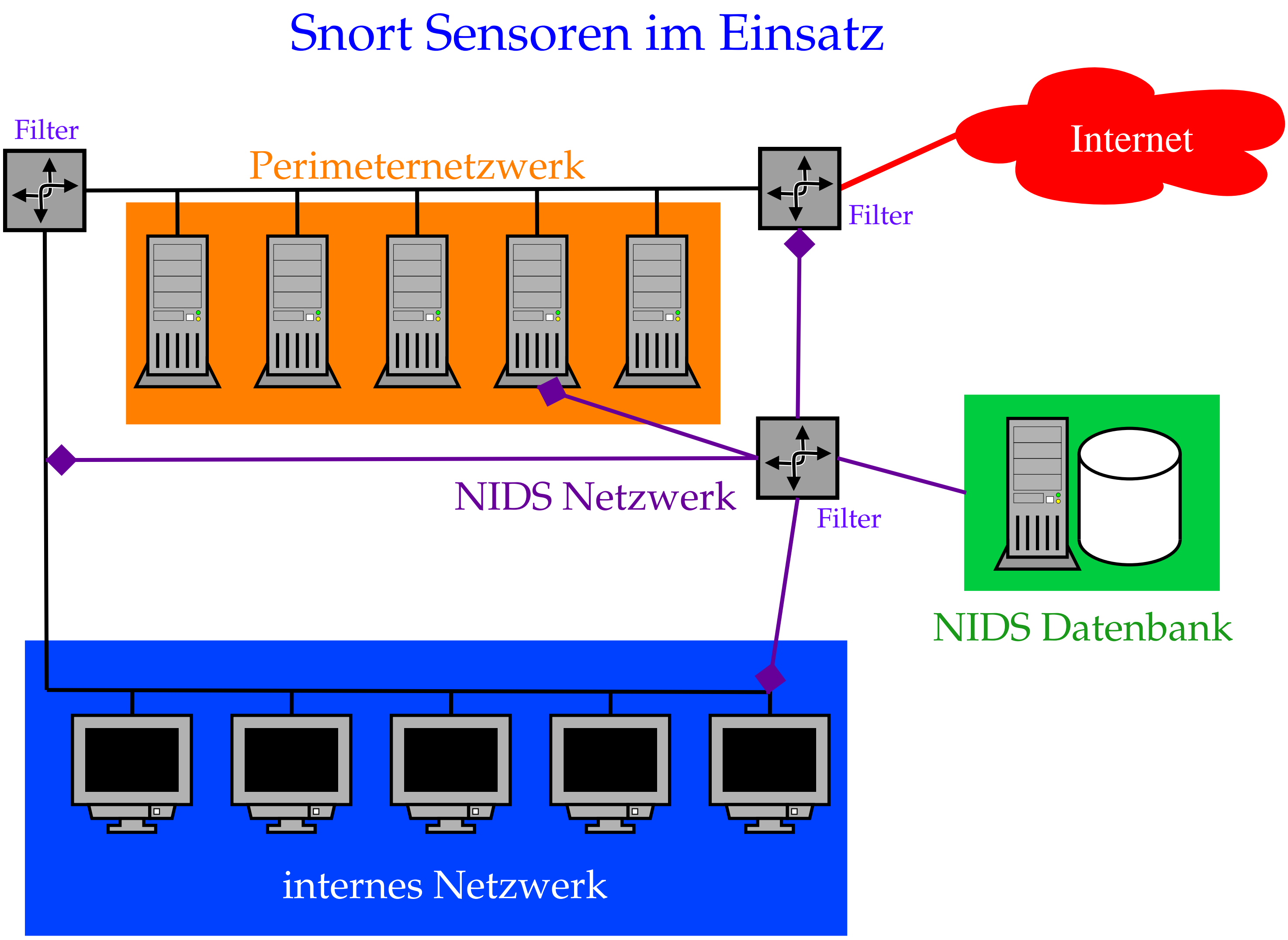

Abbildung 6: Schematischer Einsatz von Snort Sensoren in einem Netzwerk. Man kann Sensoren auf Paketfiltern, Servern oder auf speziellen Logservern unterbringen. Der sinnvollste Einsatz geschieht auf einer eigenen Maschine, die ausgewählten Netzwerkverkehr über einen dafür abgestellten Port an einem Switch erhält.

3.11 NIDS Zonen in Netzwerken

- **unsichere Netzwerke**
  - hohe Rate von Alarmen und Fehlalarmen
  - High Risk Zone
  - Sensitivität von NIDS muß gering sein
- **Perimeternetzwerk / DMZ**
  - mittlere Rate von Alarmen und Fehlalarmen
  - Netzwerkverkehr ist vorgefiltert
  - Sensitivität von NIDS muß mit Umgebung abgestimmt sein
- **lokale/vertrauenswürdige Netzwerke**
  - geringe bis mittlere Rate von Alarmen und Fehlalarmen
  - in der Regel keine Paketfilter zwischen den Clients
  - NIDS kann nach eingeschleusten Risiken schauen
  - höchster Anspruch an Performance

3.12 Attacken gegen NIDS Implementationen

- **Sensor mit Paketflut blenden**
  - IP Pakete mit gefälschten Quelladressen
  - fragmentierte Pakete
  - Versuch den Sensor durch Überlastung auszuschalten (CPU, Storage, Alarmrate)
  - Mischen von Attacken und Paketflut zur Täuschung
- **langsame Proben und Portscans**
  - Proben können sich über Tage und Wochen hinziehen
  - Detektierung durch manuelle Auswertung unmöglich
- **koordinierte Attacken von verschiedenen Ausgangspunkten**
- **wechselnde Muster bei Proben und Attacken**
  $\longrightarrow$ *manche Attackvektoren erlauben Variationen*
  $\longrightarrow$ *gute Definition und Wartung von Signaturen notwendig*
- **Verwenden von nicht-standard Ports**
  oft wird ein Protokoll nur nach Portnummer identifiziert

Quelle: `http://www.robertgraham.com/pubs/network-intrusion-detection.html` **Snort übersteht solche Attacken mittlerweile sehr gut ohne seine Funktionalität einzubüßen!**

4 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*

4.1 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

4.2 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?*

4.3 Logging - Aufspüren von Unregelmäßigkeiten

- **Wie schauen die 10 häufigsten Logmeldungen im Normalbetrieb aus?**

  ```
  cat /var/log/{messages,maillog} \
  | sed -e "s/^... .. ........ $HOSTNAME //" -e "s/\[[0-9]*\]:/:/" \
  | sort | uniq -c | sort -nr > /tmp/uniq.sorted
  ```

- **Wie schauen die 10 häufigsten Logmeldungen im „Ernstfall“ aus?**
- **Was bedeuten diese Meldungen?**
- **Gibt es Beispieldaten, die man für eine Analyse heranziehen kann?**
- **Kann man den Betrieb eines Servers/Netzwerks in Statistiken abbilden?**
  - Mails pro Zeiteinheit
  - Transaktionen bzw. E/A Zugriffe pro Zeiteinheit
  - Pakethistogramme, Datendurchsatz
- **Wie werden Daten abgelegt?**
  $\longrightarrow$ *XML, SQL, Rohform*

4.4 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*
  $\longrightarrow$ *Administratorn 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*

4.5 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*

4.6 Methoden zur Netzwerküberwachung

- **Einsetzen von HUBs in Segmente**
  - *leicht zu implementieren, keine besondere Konfiguration*
  - *Paketkollisionen steigen, Performanceverlust*
- **Switch Port Analyzer (SPAN) Port**
  - *keine weitere Hardware nötig, Infrastruktur bleibt unverändert*
  - *begrenzte Anzahl von Ports pro Switch, NIDS nur passiv*
  - *Paketverlust bei großem Netzwerkverkehr*
- **Network Test Access Ports (TAPs)**
  - *kein Performanceverlust, keine Störung am Netzwerk*
  - *NIDS muß im Stealth Mode arbeiten*
  - *zusätzliche Kosten durch TAP Hardware*
- **Abfrage von Kenndaten der Applikation selbst**

4.7 SPAN Port Implementation

- **meist gibt es nur einen SPAN Port pro Switch**
- **Überwachen mehrerer Ports problematisch**
  - mehrere Ports meist nur duch Erstellung von VLANs möglich
  - Überlastung des SPAN Ports möglich
    insbesondere im full-duplex Modus
- **Performanceprobleme bei hoher Last am Switch**
- **manche SPAN Ports sind nicht bidirektional**
  $\longrightarrow$ *aktives Terminieren von Sessions nicht möglich*
- **Verlust von fehlerbehafteten Paketen**
  - überlange oder zu kurze Pakete
  - Pakete mit Checksummenfehler

  Switchlogik kann daher durch Korrekturen Bild des Netzwerks verfälschen

4.8 Überlegungen zur Performance

- **Abstimmen der Signaturen**
  ⟶ *je weniger Checks, desto schneller der Sensor*
  ⟶ *NIDS Regeln unbedingt auf Einsatzort abstimmen*

- **Verteilung der Sensoren auf das Netzwerk**

- **Verteilung der Aufgaben auf mehrere Sensoren**
  ⟶ *Aufteilung von Protokollen (Load Balancer, Layer 7 Switch)*

- **Ändern der Default Timeouts für verschiedene Protokolle**
  ⟶ *geringeres Timeout für High Volume Protokolle (z.B. HTTP)*
  ⟶ *Anpassen der Session Timeouts für TCP*
  ⟶ *sauberes Terminieren von Verbindungen (TCP RST, ICMP)*

- **Filtern von unerwünschtem Netzwerkverkehr**
  ⟶ *an Routern (ACLs)*
  ⟶ *an Sensoren und Logservern*

4.9 Mehrstufige Architektur für Hochleistungs IDS

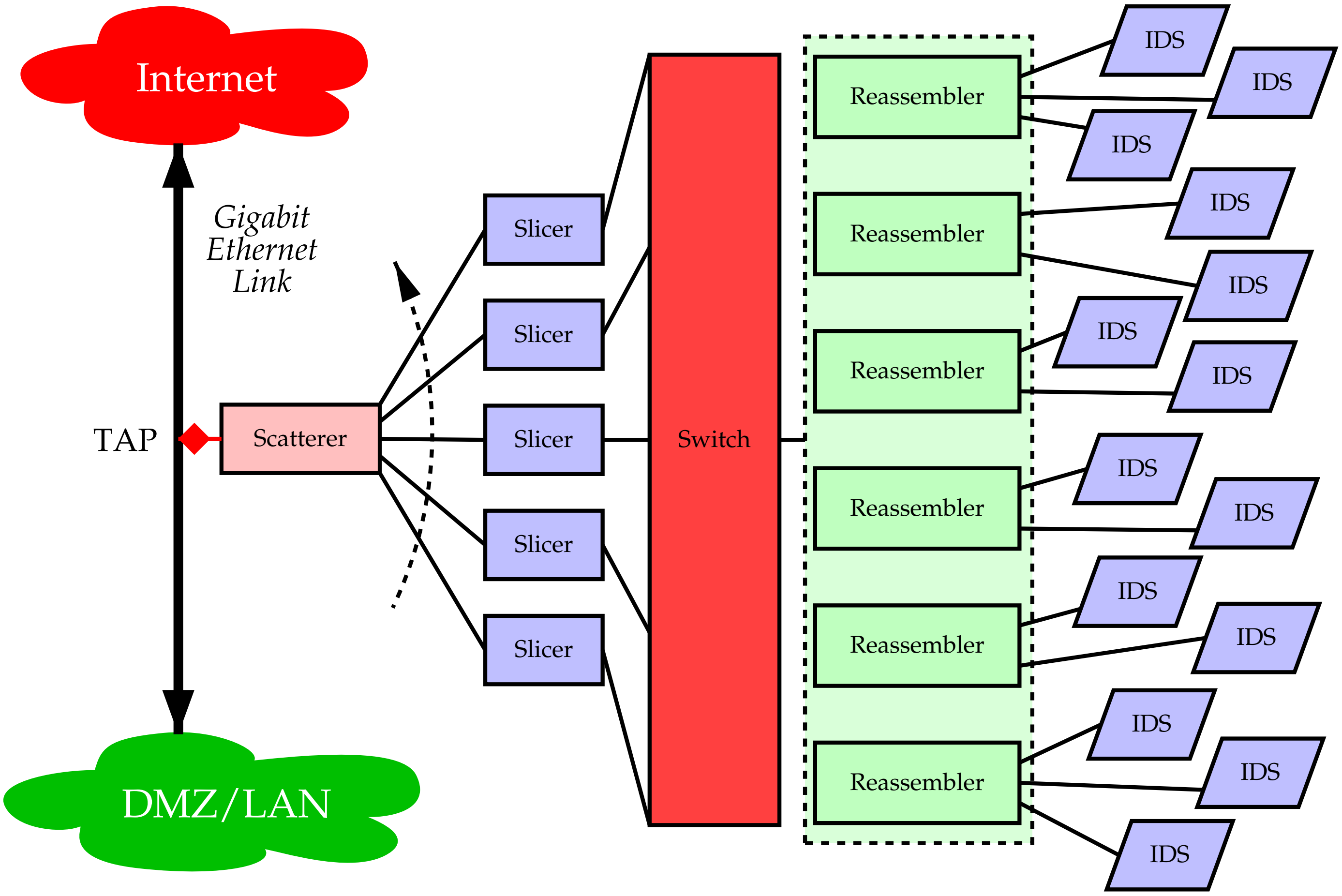

Abbildung 7: Schema eines verteilten Intrusion Detection Systems für Hochleistungsnetze. Ein Scatterer greift alle Daten eines Zeitabschnitts ab und verteilt sie nach einem Splitting Algorithmus auf verschiedne Slicer. Jeder Slicer sortiert nun die einzelnen Ethernet Frames an einen oder mehrere Sensorkanäle. Die hinter dem Switch liegenden Reassembler sorgen für die richtige Reihenfolge der einzelnen Frames. Jeder Sensor bzw. jede Sensorgruppe analysiert nur bestimmte Angriffszenarien, um die Last zu verteilen. (*Quelle:* Stateful Intrusion Detection for High-Speed, University of California, `http://www.computer.org/proceedings/sp/1543/15430285abs.htm`)

4.10 Zentralisieren der Konfiguration

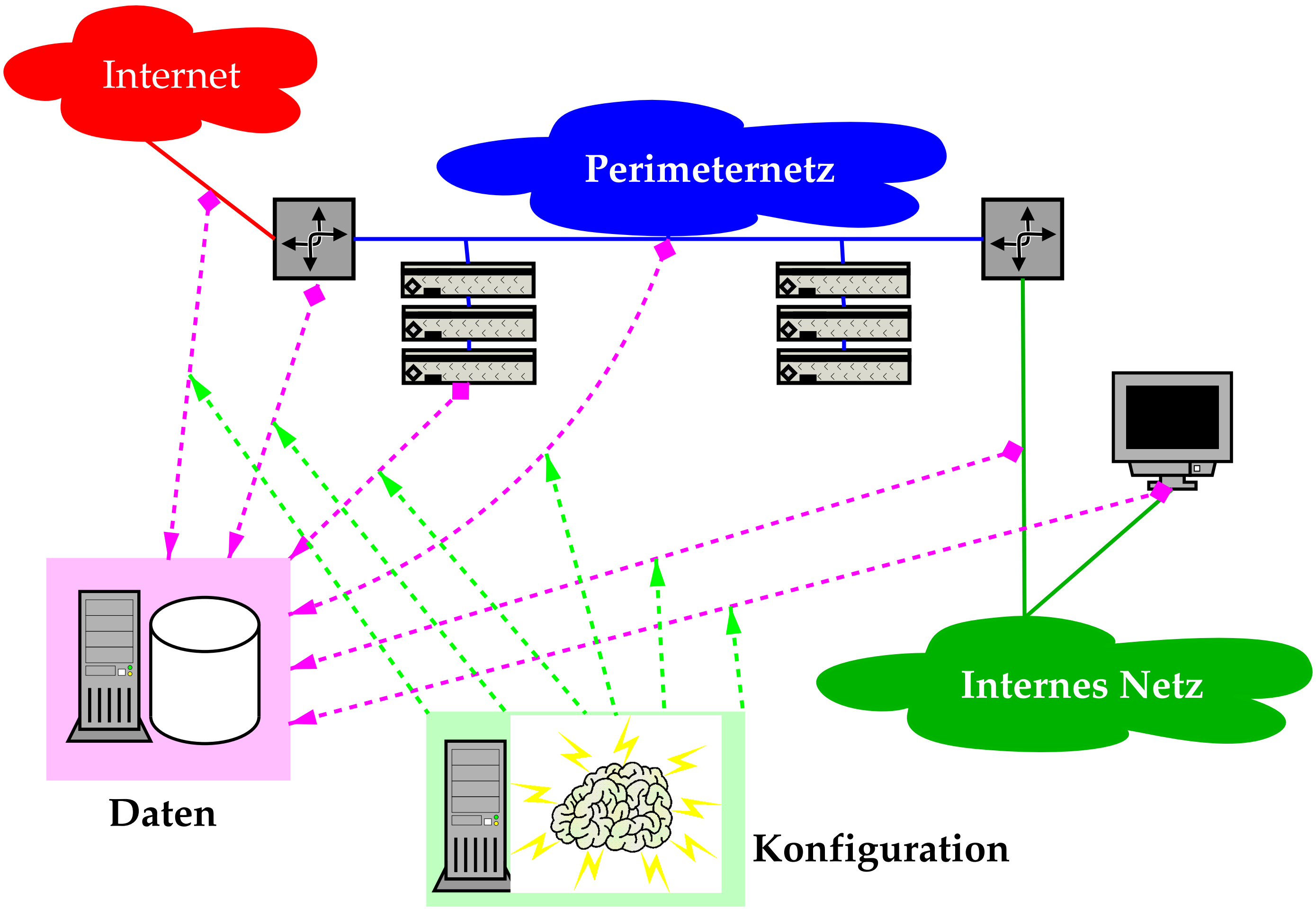

Abbildung 8: Ausgedehnter Einsatz von IDS Maßnahmen erfordert ein gewisses Maß an zentralisiertem Sammeln der Daten. Dies gilt ebenso für die Verteilung der Konfigurationen an die verschiedenen Sensoren.

4.11 Auswahl von IDS-Konsolen für Snort

- **Analysis Console for Intrusion Databases (ACID)**[30]
  - PHP, mySQL
  - UN*X, MS Windows
- **Demarc® PureSecure**[31]
  - Linux, FreeBSD, OpenBSD, Solaris, NetBSD, WindowsNT, Windows2000, WindowsXP
- **Henwen**[32]
  - Mac OS X
- **IDScenter**[33]
- **IDS Policy Manager**[34]
  - Windows2000, WindowsXP
- **Snortcenter**[35]
- **Snortreport**[36]
  - Linux, FreeBSD, OpenBSD, Mac OS X, MS Windows
- **Webmin**[37] **Plugin**[38]

[30] http://www.andrew.cmu.edu/user/rdanyliw/snort/snortacid.html
[31] http://www.demarc.com/products/puresecure/
[32] http://seiryu.home.comcast.net/henwen.html
[33] http://www.engagesecurity.com/products/idscenter/
[34] http://www.activeworx.org/programs/idspm/index.htm
[35] http://www.snort.org/dl/contrib/front_ends/snortcenter/
[36] http://www.circuitsmaximus.com/download.html
[37] http://www.webmin.com/
[38] http://www.snort.org/dl/contrib/front_ends/webmin_plugin/

4.12 Konsolidierung

- **IDS werden in bestehenden Netzen meist nachgerüstet**
  ⟶ *Nutzung bestehender Log Infrastruktur*
  ⟶ *Platzierung von HIDS auf exponierte Systeme*
  ⟶ *NIDS hinter Paketfiltern*
- **Wahl der eingesetzten Systeme gut planen**
  ⟶ *Interoperabilität zwischen Herstellern*
  ⟶ *Kompatibilität der generierten Alarme & Daten*
- **Verzicht auf High-Speed IDS**
  - ähnlicher Ansatz wie bei Paketfiltern
  - High-Speed IDS Infrastruktur derzeit komplex
- **Wahl von Lösungen, die plattformunabhängig sind**
  ⟶ *leichtere Migration auf vereinheitlichte Hardware*
  ⟶ *Beibehaltung der Konfiguration*

4.13 Performance-schonende Lösungen

- **Real Time Auswertung stellt höchste Ansprüche an IDS**
  $\longrightarrow$ *Pattern Matching beansprucht CPU*
  $\longrightarrow$ *Abgreifen der Pakete muß zuverlässig sein*
- **Einsatz von TAPs und performanten Sensoren**
  $\longrightarrow$ *gezielt an „Hot Spots“*
- **Teilen der zu überwachenden Ereignisse**
  - Festlegen von kritischen Überwachungsbereichen
  - Sammlung der anderen Daten in größeren periodischen Abständen
- **Data Warehouse Ansatz - bestehende Informationen nutzen**
  - oft existieren schon Informationsresourcen
  - statistische Analyse und Zusammenführen der Daten

  Performance geht in diesem Falle an Datenbanken und Extraktionslogik

4.14 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**

4.15 Automationsmöglichkeiten - Übersicht

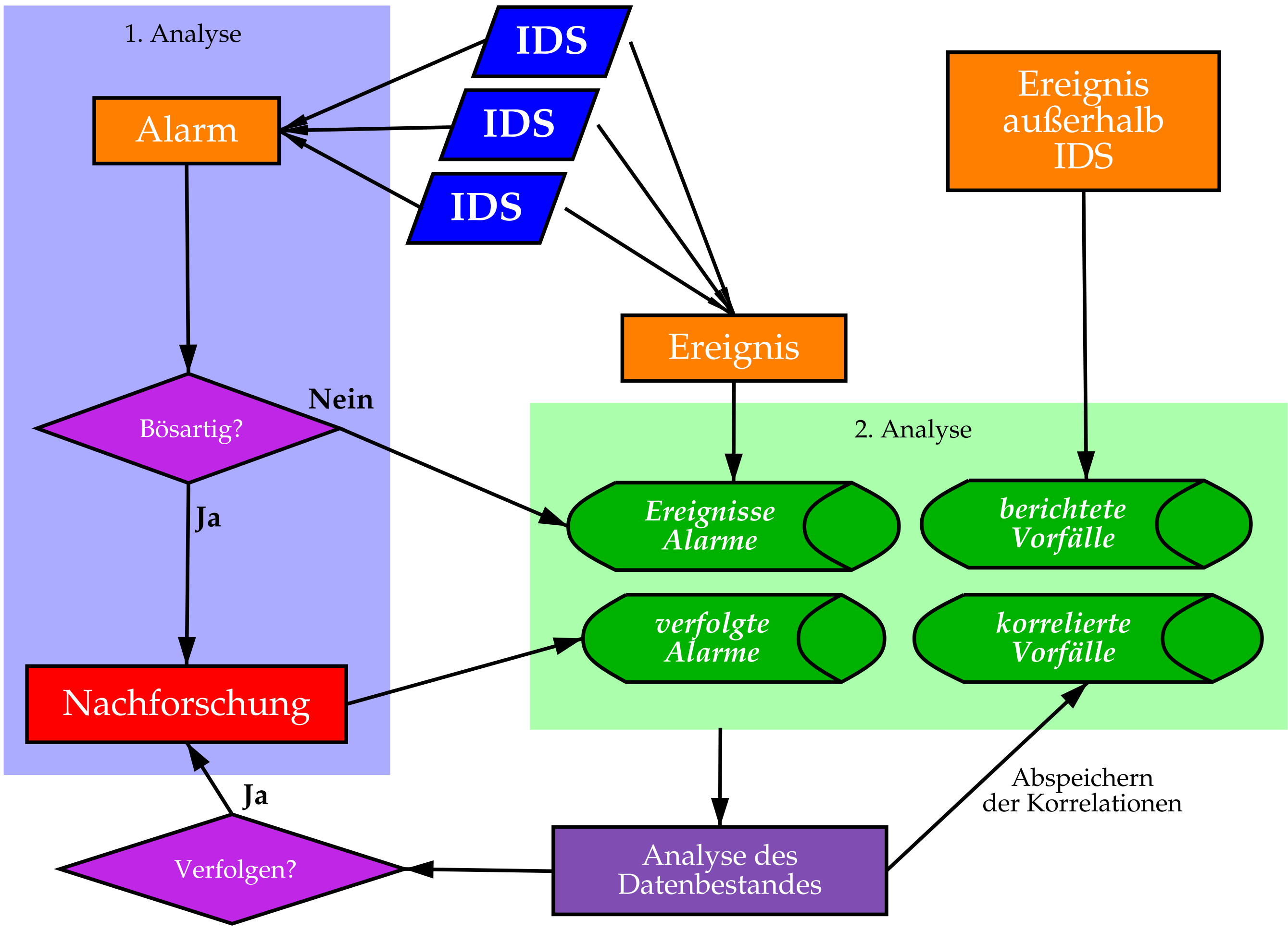

Abbildung 9: Die Abbildung zeigt die mehrstufige Auswertung von IDS Daten. In der ersten Stufe wird unmittelbar auf einen Alarm reagiert. Die Daten werden in Datenbanken gesammelt und durch weitere Prozesse analysiert. Dadurch können sich weitere Alarme ergeben, die behandelt werden müssen. (`http://www.securityfocus.com/infocus/1201`)

5 Noch Fragen?

Bitte kontaktieren Sie mich bei noch offenen Fragen oder Bemerkungen:

René Pfeiffer
pfeiffer@technikum-wien.at
rene.pfeiffer@paradigma.net
pfeiffer@luchs.at

Vielen Dank für Ihre Aufmerksamkeit!

A Linux 2.4.x/2.6.x

A.1 Netfilter Kernel Konfiguration

Der Linux Netfilter befindet sich im Linux Kern und besteht aus einer Reihe von Modulen. Die Konfiguration wird dem Kern über ein Shell-Kommando namens `iptables` mitgeteilt. Alle gängigen GNU/Linux Distributionen bringen beides schon von Haus aus mit. Je nach Anwendung bietet es sich jedoch an einen eigenen Kern zu konfigurieren. Abbildung 10 zeigt einen Ausschnitt aus der Kernel Konfiguration kurz vor Start des C Compilers.

A.2 QoS und Traffic Shaping

Traffic Shaping muß man mit einem Linux Kern durchführen, der explizit auf Routing spezialisiert ist (Menüpunkt *Advanced Router* in der Kernel Konfiguration). Linux Router können verschiedene Routing Queues für die einzelnen Netzwerkgeräte verwenden und so den Paket■uß sehr genau steuern. Abbildung 11 zeigt alle vorhandenen Möglichkeiten bei QoS und Traffic Shaping.

A.3 Traffic Shaping Beispielskript

```
#!/bin/sh
#
# shaping - traffic shaping for gw.wherever.net
# Copyright (C) 2004  Rene Pfeiffer <lynx@luchs.at>
#
# This program is free software; you can redistribute it and/or
# modify it under the terms of the GNU General Public License
# as published by the Free Software Foundation; either version 2
# of the License, or (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.

# Thu Mar 11 17:54:03 CET 2004 lynx@luchs.at

# Get include file
. /etc/rc.fireconf

# Speeds
EXT_SPEED="2320kbit"
EXT_DEV="eth0"

HTTPS_SPEED="464kbit"
HTTP_SPEED="594kbit"
DNS_SSH_SPEED="47kbit"
IMAPPOP_SPEED="348kbit"
SMTP_SPEED="403kbit"
REST_SPEED="464kbit"

HTTPS_CLASS="1:2"
HTTPS_HANDLE="120"

HTTP_CLASS="1:3"
HTTP_HANDLE="130"
```

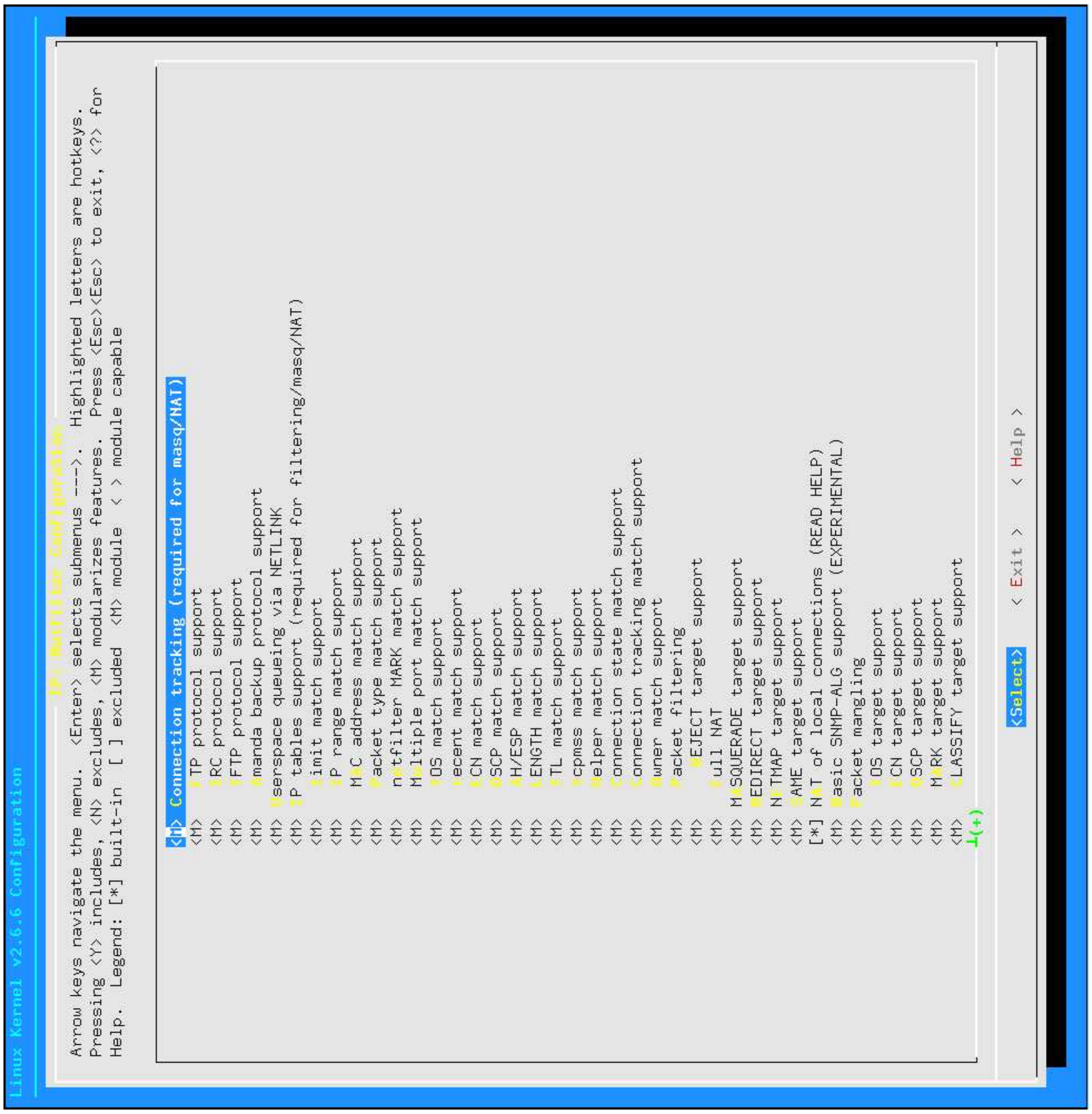

Abbildung 10: Das Bild zeigt einen Screenshot der Netfilter Konfiguration im Kern 2.6.6. Man sieht einen Teil der ganzen Module, die dem Netfilter hinzugefügt werden können. Das Filtern geschieht im Kern, konfiguriert wird alles über das Userland Tool `iptables`.

Abbildung 11: Das Bild zeigt einen Screenshot der QoS und Traffic Shaping Konfiguration im Kern 2.6.6. Die Menüpunkte sind die einzelnen Routing Queues, die nur zur Verfügung stehen, wenn man den Kern explizit als Router konfiguriert. Das Traffic Shaping wird mit den Kommandos `tc` und `ip` aus der `iproute2` Package konfiguriert.

```
DNS_SSH_CLASS="1:4"
DNS_SSH_HANDLE="140"

IMAPPOP_CLASS="1:5"
IMAPPOP_HANDLE="150"

SMTP_CLASS="1:6"
SMTP_HANDLE="160"

REST_CLASS="1:7"
REST_HANDLE="170"

# HTB root
$TC qdisc add dev $EXT_DEV root handle 1:0 htb default 20
$TC class add dev $EXT_DEV parent 1:0 classid 1:1 htb rate $EXT_SPEED
echo "HTB root class set"

# Classes
$TC class add dev $EXT_DEV parent 1:1 classid $HTTPS_CLASS   htb rate $HTTPS_SPEED   ceil $EXT_SPEED
$TC class add dev $EXT_DEV parent 1:1 classid $HTTP_CLASS    htb rate $HTTP_SPEED    ceil $EXT_SPEED
$TC class add dev $EXT_DEV parent 1:1 classid $DNS_SSH_CLASS htb rate $DNS_SSH_SPEED ceil $EXT_SPEED
$TC class add dev $EXT_DEV parent 1:1 classid $IMAPPOP_CLASS htb rate $IMAPPOP_SPEED ceil $EXT_SPEED
$TC class add dev $EXT_DEV parent 1:1 classid $SMTP_CLASS    htb rate $SMTP_SPEED    ceil $EXT_SPEED
$TC class add dev $EXT_DEV parent 1:1 classid $REST_CLASS    htb rate $REST_SPEED    ceil $EXT_SPEED
echo "Classes were defined"

# Queues
# RED queues are calculated with 2320 kbit/s and 500 ms latency
$TC qdisc add dev $EXT_DEV parent $HTTPS_CLASS   handle $HTTPS_HANDLE sfq perturb 10
$TC qdisc add dev $EXT_DEV parent $HTTP_CLASS    handle $HTTP_HANDLE sfq perturb 10
$TC qdisc add dev $EXT_DEV parent $DNS_SSH_CLASS handle $DNS_SSH_HANDLE sfq perturb 10
$TC qdisc add dev $EXT_DEV parent $IMAPPOP_CLASS handle $IMAPPOP_HANDLE sfq perturb 10
$TC qdisc add dev $EXT_DEV parent $SMTP_CLASS    handle $SMTP_HANDLE \
    red limit 1160000 min 49000 max 145000 avpkt 1000 burst 81 probability 0.02 bandwidth $EXT_SPEED ecn
$TC qdisc add dev $EXT_DEV parent $REST_CLASS    handle $REST_HANDLE sfq perturb 10
echo "Queues were done"

# Filters
#
# HTTPS
$TC filter add dev $EXT_DEV parent 1:0 protocol ip prio 1 u32 \
    match ip src $SRV_NET \
    match ip sport $HTTPS \
    0xfff flowid $HTTPS_CLASS
# HTTP
$TC filter add dev $EXT_DEV parent 1:0 protocol ip prio 1 u32 \
    match ip src $SRV_NET \
    match ip sport $HTTP \
    0xfff flowid $HTTP_CLASS
# DNS & SSH
$TC filter add dev $EXT_DEV parent 1:0 protocol ip prio 1 u32 \
    match ip src $SRV_NET \
    match ip sport $DNS \
    0xfff flowid $DNS_SSH_CLASS
$TC filter add dev $EXT_DEV parent 1:0 protocol ip prio 1 u32 \
    match ip src $SRV_NET \
    match ip dport $DNS \
    0xfff flowid $DNS_SSH_CLASS
$TC filter add dev $EXT_DEV parent 1:0 protocol ip prio 1 u32 \
    match ip src $SRV_NET \
    match ip sport $SSH \
    0xfff flowid $DNS_SSH_CLASS
$TC filter add dev $EXT_DEV parent 1:0 protocol ip prio 1 u32 \
    match ip src $SRV_NET \
    match ip dport $SSH \
    0xfff flowid $DNS_SSH_CLASS
# IMAP & POP3
$TC filter add dev $EXT_DEV parent 1:0 protocol ip prio 1 u32 \
    match ip src $SRV_NET \
    match ip sport $POP3 \
```

```
    0xfff flowid $IMAPPOP_CLASS
$TC filter add dev $EXT_DEV parent 1:0 protocol ip prio 1 u32 \
    match ip src $SRV_NET \
    match ip sport $IMAP \
    0xfff flowid $IMAPPOP_CLASS
# SMTP
$TC filter add dev $EXT_DEV parent 1:0 protocol ip prio 1 u32 \
    match ip src $SRV_NET \
    match ip sport $SMTP \
    0xfff flowid $SMTP_CLASS
$TC filter add dev $EXT_DEV parent 1:0 protocol ip prio 1 u32 \
    match ip src $SRV_NET \
    match ip dport $SMTP \
    0xfff flowid $SMTP_CLASS
# Rest
$TC filter add dev $EXT_DEV parent 1:0 protocol ip prio 1 u32 \
    match ip src $SRV_NET \
    flowid $REST_CLASS
$TC filter add dev $EXT_DEV parent 1:0 protocol ip prio 1 u32 \
    match ip src $LAN_NET \
    flowid $REST_CLASS
echo "Filters were set"

# SMTP Ingress filtering
$IPTABLES --append PREROUTING \
          --in-interface $EXT_DEV -t mangle -p tcp \
  --source $EVERYWHERE --destination $SRV_NET --destination-port $SMTP \
  --jump MARK --set-mark 23
$IPTABLES --append PREROUTING \
          --in-interface $EXT_DEV -t mangle -p tcp \
  --source $EVERYWHERE --destination $FW_VBS --destination-port $SMTP \
  --jump MARK --set-mark 23
$TC qdisc add dev $EXT_DEV handle ffff: ingress
$TC filter add dev $EXT_DEV parent ffff: protocol ip prio 50 handle 23 fw \
    police rate $SMTP_SPEED burst 65535 mpu 0 drop flowid :1
echo "SMTP Ingress filtering active"
```

B Intrusion Detection

B.1 Network TAPs

- **TAP Elektronik wird transparent, wenn Strom ausfällt**
  $\longrightarrow$ *verringerte Fehleranfälligkeit*
- **Verfügbar für 10/100/1000 Mbit/s Netzwerke**
- **keine Beeinflussung des Netzverkehrs**
  $\longrightarrow$ *kein Performanceverlust*
- **IDS kann Paketanomalien untersuchen**
  $\longrightarrow$ *überlange oder zu kurze Pakete*
  $\longrightarrow$ *Checksummenfehler*

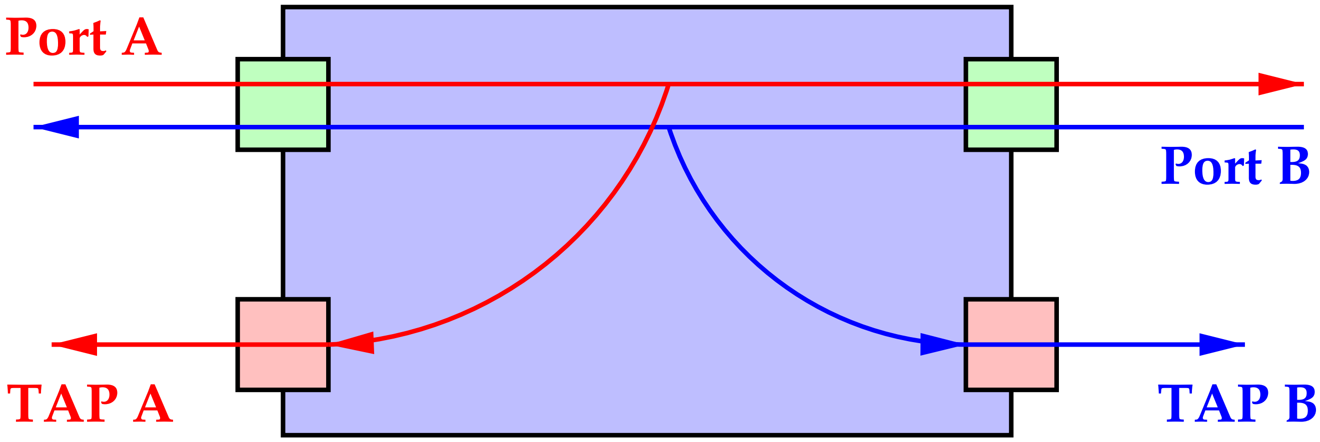

Abbildung 12: Das Diagramm zeigt die schematische Ansicht eines Network TAPs. TAP A fängt Pakete von an Port A angeschlossenen Geräten ab, TAP B welche von an Port B angeschlossenen Geräten. Eine Kombination von TAP A und TAP B ergibt den gesamten Netzwerkverkehr. (Quelle: Internet Security Systems)