Scientia est potentia

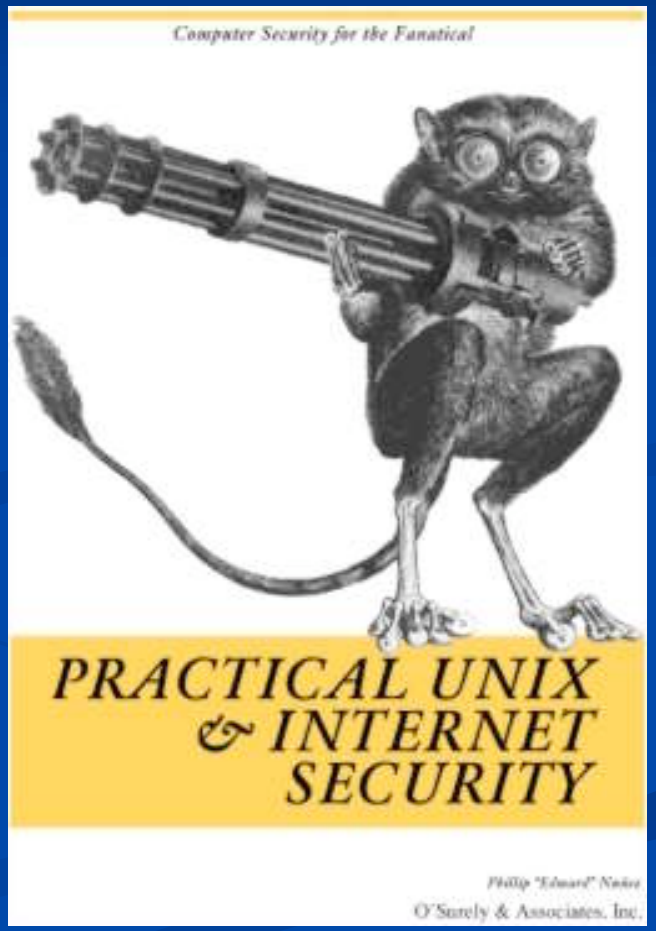
Intrusion Detection (und Response) am Beispiel Snort

Andree Linke

Was ist IDS

Software oder Gerät,

- das im Idealfall Angriffe oder Eindringen in ein Netzwerk oder einen Host erkennt,
- das Anomalien im Traffic meldet,
- das durch Manipulation oder Gegenangriff reagiert,
- oder durch genaues Loggen des Traffics Beweise sammeln kann und
- das den Administrator bei Angriffen alarmiert.



Warum IDS?

- “Alarmanlage” im Netzwerk
- Möglichkeit, automatisiert zu reagieren
- Beweissichernde Komponente
- Packetlogger mit Regelbasis, daher speicherplatzschonend

Bedrohungen

- Würmer (Code Red, Nimda)
- Script Kiddies
- Trojaner (mit automatisch ablaufenden Scripten)
- Regierungen, Geheimdienste
- Cracker
- Spyware

Notwendigkeit für Dynamik

- Anpassung von Firewallregeln
- Sammeln von Traffic (oder Hostparametern) als Beweise
- Gegenschlagmöglichkeit (rechtlich unklar)
- Aktives Sammeln von Informationen über den Angreifer (Whois, Tracert, Finger usw…)

Subkategorien des IDS

Host IDS (System Integrity Verifier): monitort kontinuierlich bestimmte Systemparameter (crond, registry,...) und speichert sie. Bei Parameteränderung ist ein Alarm möglich.

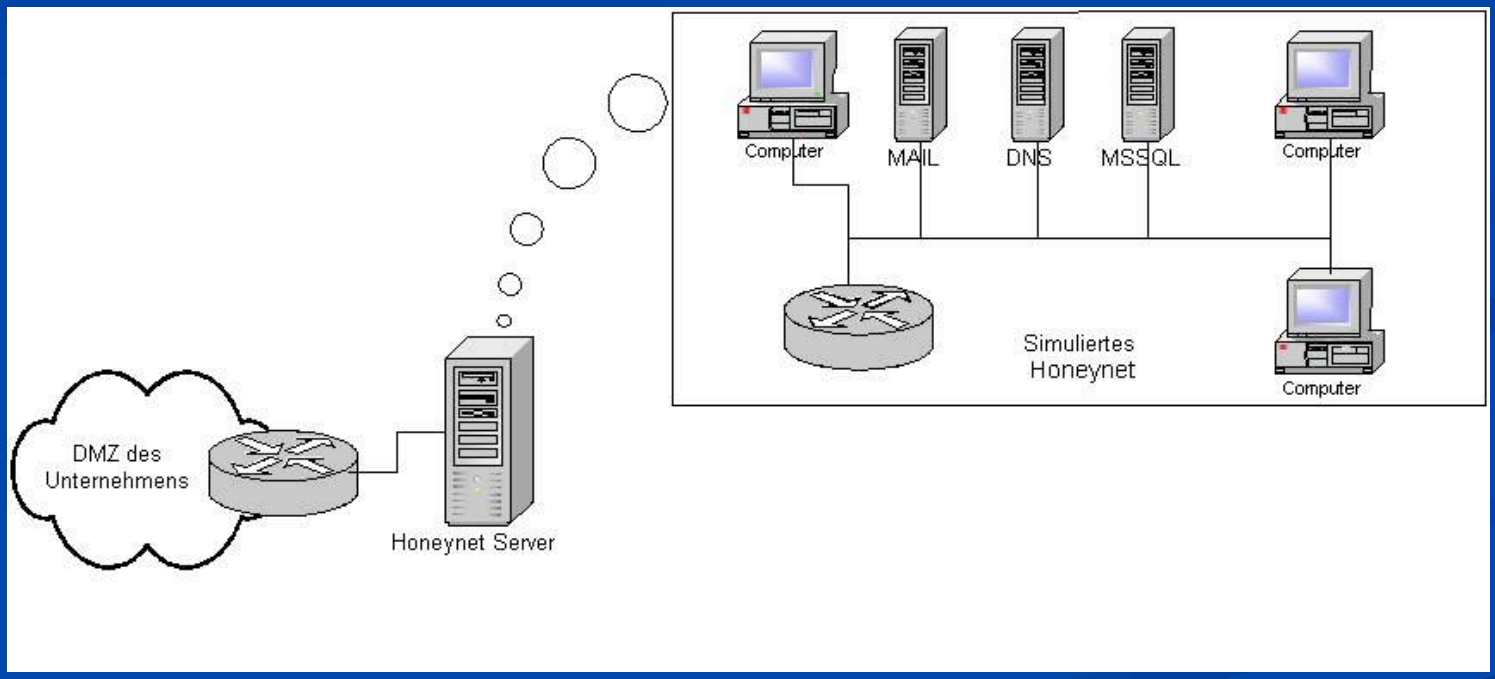
Network IDS: Analysiert den Traffic auf dem Netz und stellt mittels Anomaly Detection oder Pattern Matching fest, ob ein Angriff stattfindet.

Verteiltes NIDS: wie NIDS, sammelt und speichert die Daten verteilt.

Log File Monitor: Analysiert die Logfiles verschiedener Hosts, ähnlich NIDS.

Honeynets

Decoys (Honeypot/Honeynet): System als Köder für Einbrecher, zur Ablenkung oder Beobachtung der Vorgehensweise geeignet. Honeynets können in Hardware realisiert werden oder kostengünstig simuliert werden.



Das Honeynet wird von diversen Sensoren (u.a. IDS) genauestens überwacht.

Funktion von NIDS

Beispiel: Snort

- **Netzwerksniffer**
  - Sammlung von Rohdaten auf dem Netzwerk
- **Packetdecoder**
  - Organisation der Rohdaten in Datenstrukturen
  - Strukturierung nach zugrundeliegendem Modell (TCP/IP)
- **Detection Engine**
  - Regelbasis nach Pattern Matching oder Anomaly Detection
  - Zusammenführung von Fragmenten
  - Erkennen von Zuständen in einer Verbindung
  - Entscheidungsfindung über Alarm, Logeintrag, Reaktion
- **Reaktives System**
  - Ausführung der Entscheidung

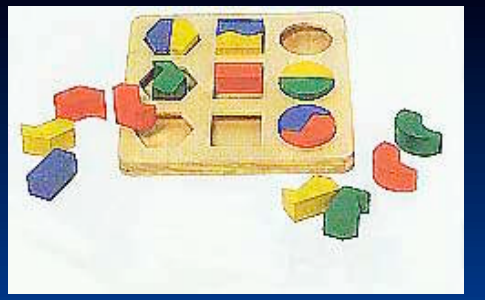
Anomaly Detection

Netzwerktraffic ist im allgemeinen gleichmässig. Wenn man den normalen Traffic eines Netzes zugrunde legt, kann jede Abweichung ein Angriff sein. So führt jede Anomalie im Netz zu einer Reaktion.

- Unbekannte Angriffe können erkannt werden
- Ungewöhnliches Verhalten von Usern wird erkannt
- Keine riesige Angriffsdatenbank notwendig

- Grosser Aufwand beim Installieren, Lernphase
- Viele Fehlalarme
- Nur in gleichmässigen Umgebungen möglich
- Resourcenfressend

Pattern Matching



Viele Angriffe sind in ihrer Signatur bekannt. Daher kann man eine Datenbank bekannter Angriffe mit dem fliessenden Traffic vergleichen und reagieren, wenn ein Vergleich positiv ist.

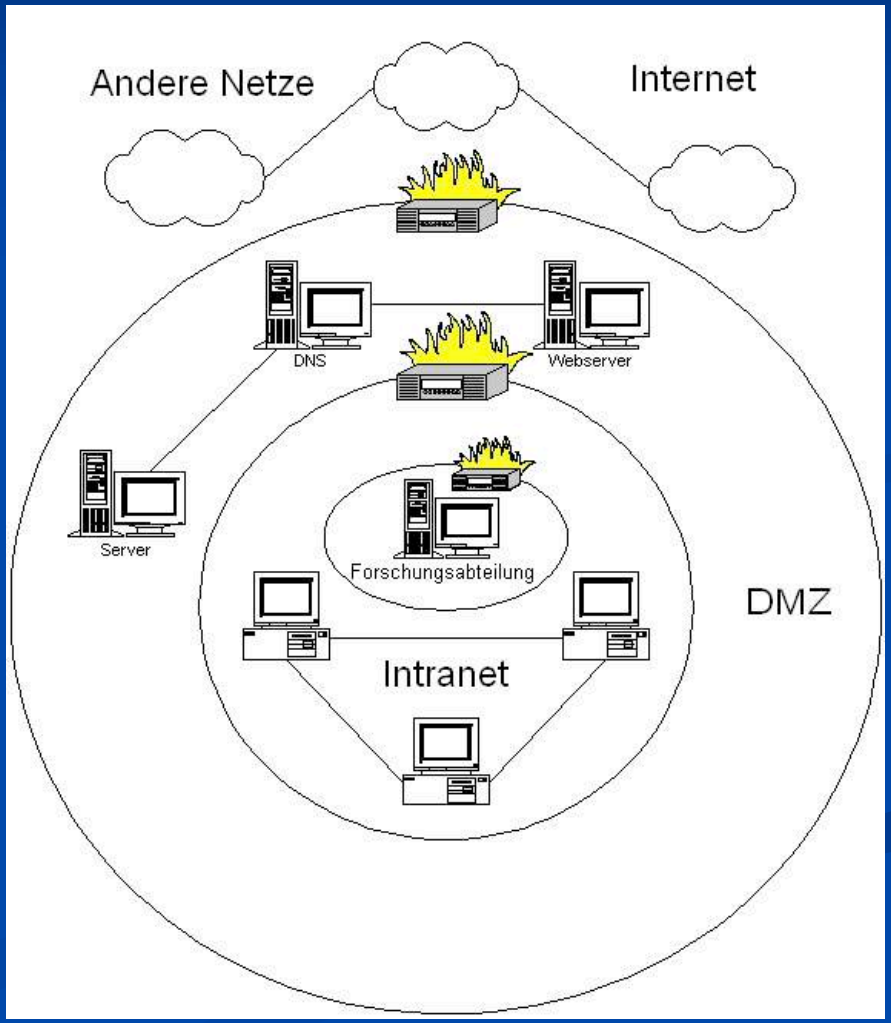
- Deutlich weniger Fehlalarme
- Zuverlässiges erkennen bekannter Angriffe
- Schnellere Installation und Einarbeitung

- Einspielen neuer Signaturen notwendig
- Erkennt keine neuen Angriffe
- Leichte Unterschiede in den Angriffspacketen können zur Nichterkennung führen

Netzwerkarchitektur

Da ein IDS allein keine Sicherheit bietet, benötigt man ein Sicherheitskonzept für das Netzwerk

- Router
- Firewalls
- Proxies
- Verschiedene Sicherheitsstufen
- Patches gegen Sicherheitslücken
- Den richtigen Standort für IDS, Logging Hosts und Datenbanken

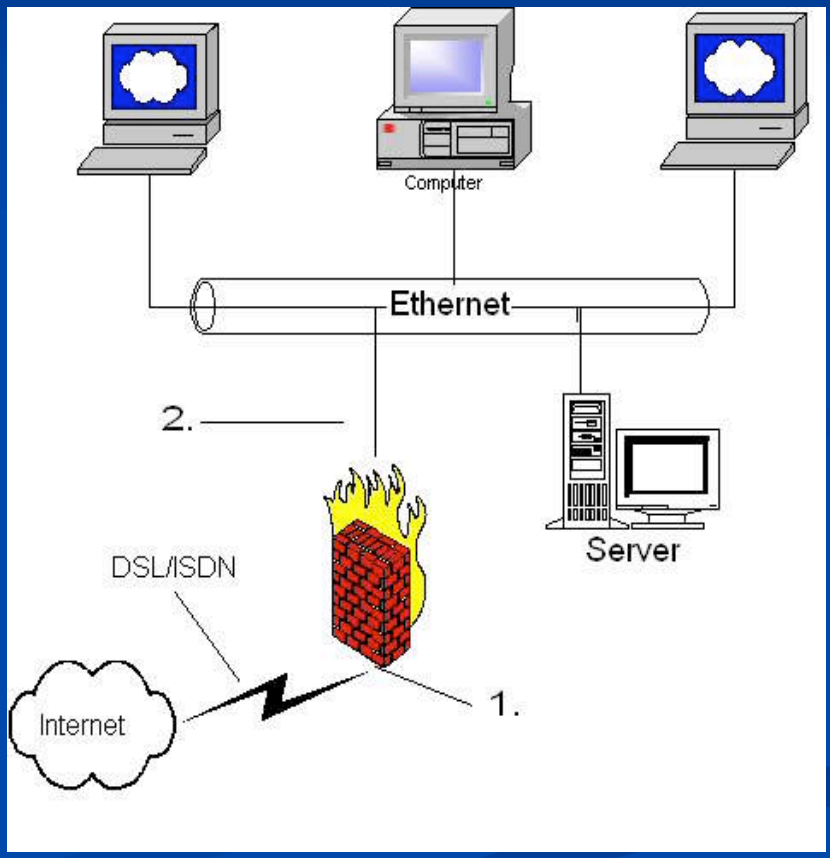


Sensorplatzierung

Was will man mit dem IDS sehen?

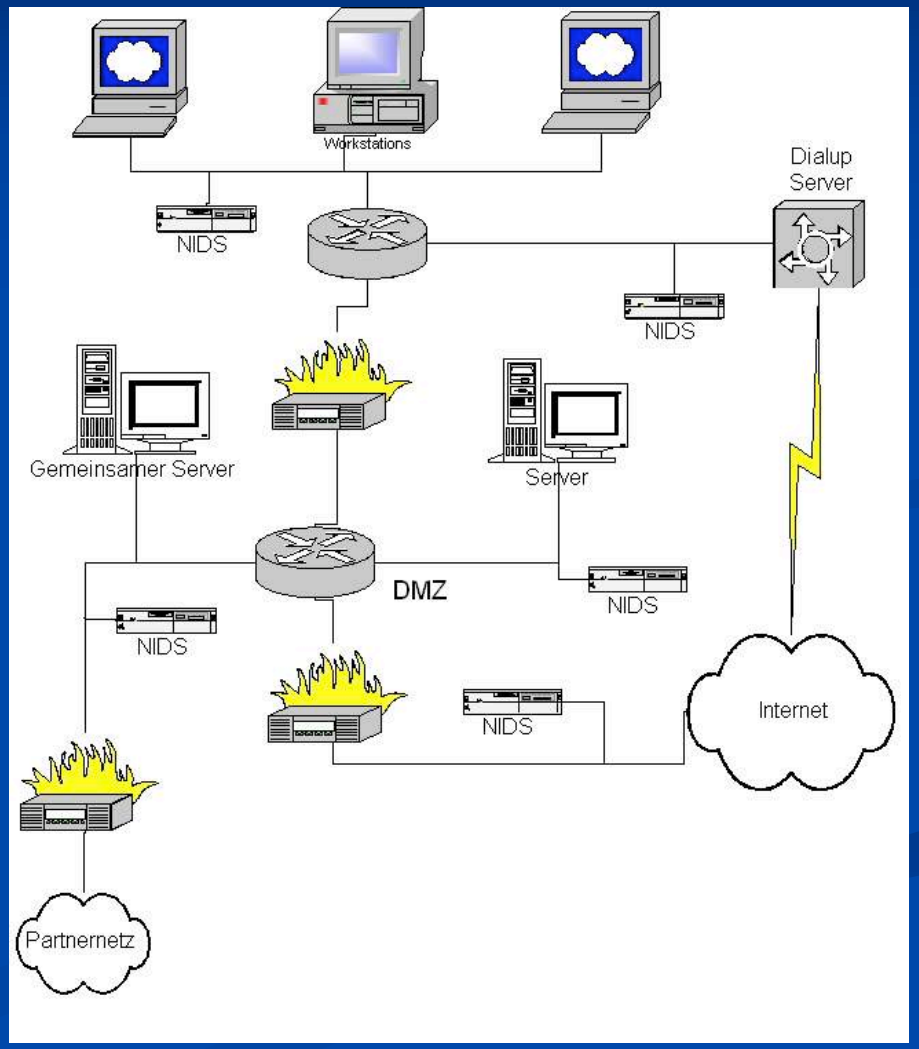
- Alle Angriffe von Aussen
- Durch die Firewall gedrungene Angriffe
- Angriffe von Innen

- Alle Angriffe
- Angriffe auf bestimmte Rechner (Server)
- Angriffe auf bestimmte Services (Ports, Programme)

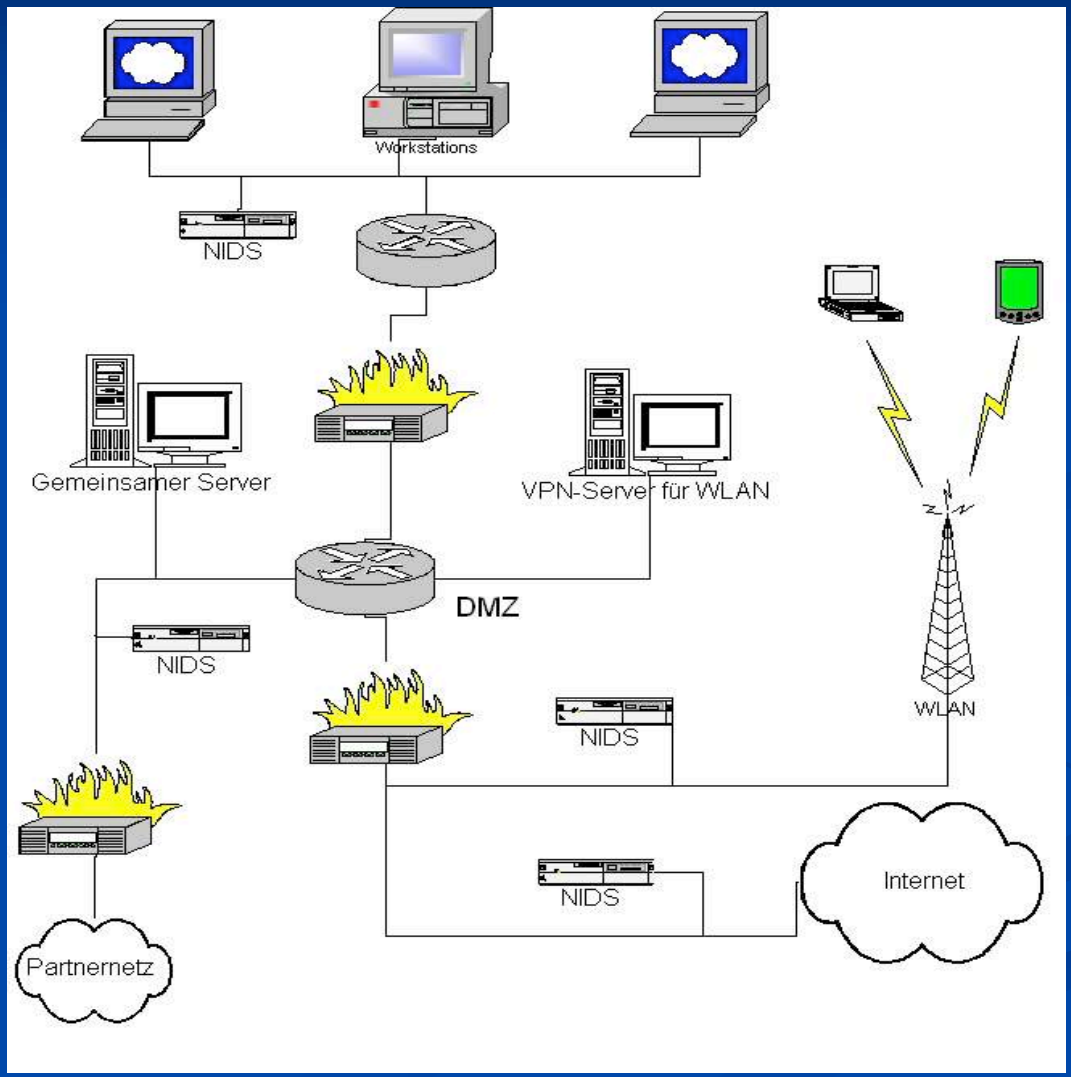


Sensorplatzierung

- In allen Netzbereichen
- An allen Übergangspunkten
- An sicherheitskritischen Systemen

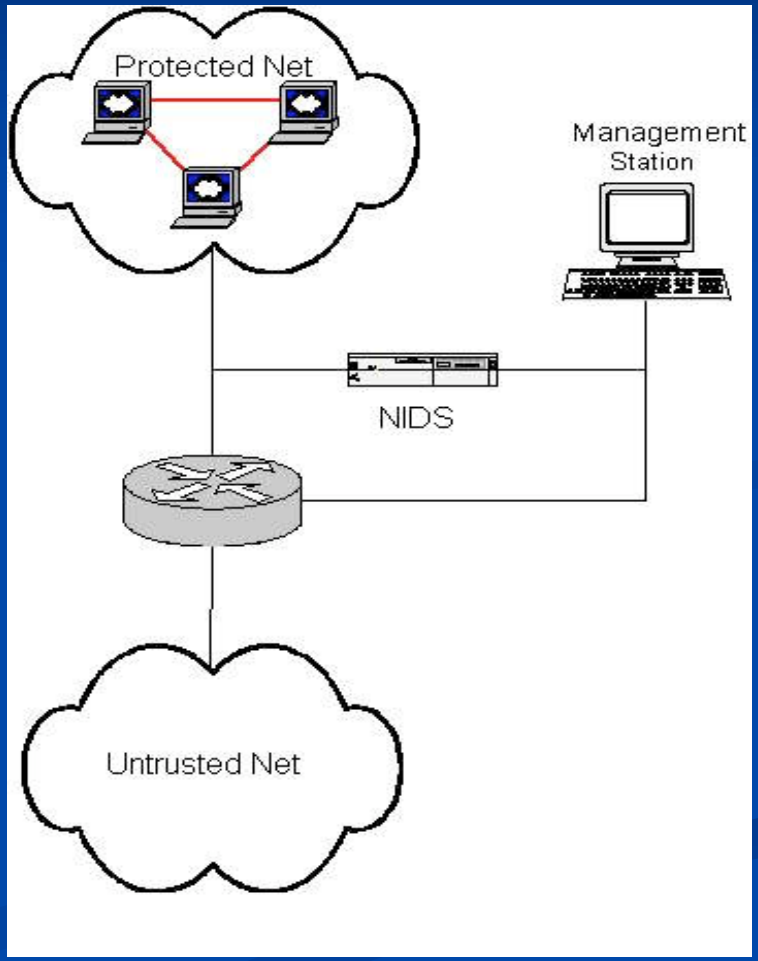


WLAN-Sicherung



Management Net

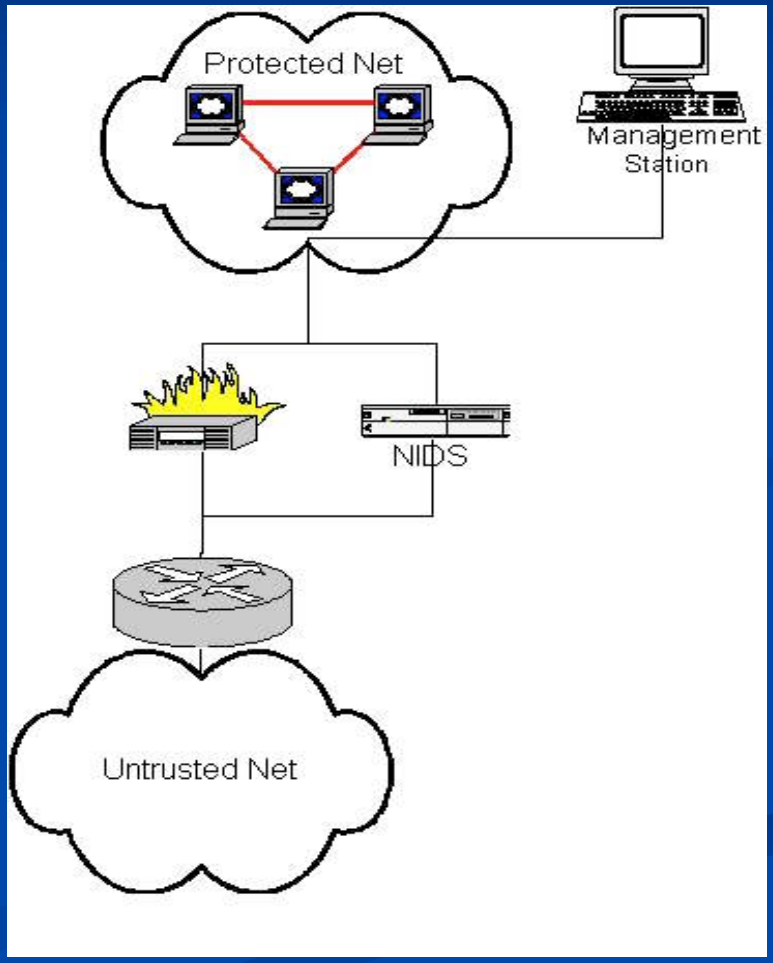
- Separates Netz ohne Traffic
- Keine Verbindung zum produktiven Netz
- Von Aussen unsichtbar
- Keine versehentliche Manipulation möglich
- Zusätzliche Hardware erforderlich
- Zusätzlicher administrativer Aufwand



„Sandwich“-Installation

- Management der Geräte übers interne Netz ohne zusätzliche Verkabelung
- Zentrale Loggingstellen möglich

- Angriff von Innen betrifft auch das Management
- Versehentliche Manipulation möglich



Installation

- Apache Webserver
- Mysql Datenbank
- PHP
- Diverse grafische Bibliotheken
- ACID oder andere Auswertungstools
- Snort ab Version 2.0 (wg. Buffer Overflow)
- Zur Kontrolle der Datenbank phpMyAdmin

Allgemeines

- Software, die auf einem Rechner unter z.B. Linux, *nix, NT5 in ISO/OSI Schicht 3 läuft (DoD Network Layer)
- Pattern Matching (Lightweight-) NIDS mit Möglichkeit zur verteilten Datensammlung
- Frontend ACID (Analysis Console for Intrusion Databases) oder snort-stat.pl stellt Daten für Menschen lesbar dar
- Gute Integration durch Skriptbarkeit und logging in MySQL oder Syslog

./configure

- --with-mysql (oder anderer Datenbank)
- --enable-flexresp

  flexible response zur Terminierung von Verbindungen
- --enable-perfmonitor

  performance monitor im Betastadium
- --enable-smbalerts

  Eine weitere Output-Möglichkeit

snort.conf

Wichtigste Variablen:

- $HOME_NET
- $EXTERNAL_NET
- $RULE_PATH
- output database

SUID oder eigener User?

Schreiben von Regeln

- Über **include** werden Skripte und Regeln in die snort.conf eingebunden
- Die Regelskripte benutzen sh-Sprachkonstrukte und Funktionen (if, !, $x = 8 …)
- Man kann über Variablen, die über andere Skripte gesetzt werden, die Regelbasis (mit einigem Aufwand) portabel und wartbar machen
- Vorsicht bei in Distris enthaltenen Snorts (zB. bei SuSE die Start- und Konfigurationsskripte gleich deaktivieren, nutzlos)

Regelaufbau

alert tcp ![130.83.0.0/16,$HOME_NET] any -> $HOME_NET 135:139

Aktion | Protokoll | Quelle | Quellport | Ziel | Zielport

(msg:“Samba file info“;flow:to_server, established; content „|32|“; content:“|05 00|“;

Beschreibung | Flussrichtung | Status | Inhalt in Hex (Mehrfach möglich)

classtype:not-suspicious; logto:“/var/log/MyFile“rev:4;)

Klassenidentifikation nach classification.config

Optional: anderes Logfile für spezielle Alarme

Revisionsnummer der Regel

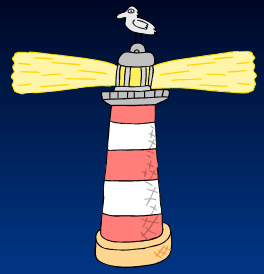
Beispielregeln

192.168.1.254 - PuTTY

```
# Other ICMP rules are included in icmp-info.rules
# altered

alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP ISS Pinger"; content:"|495353504e475251|";itype:8;depth:32; re
ference:arachnids,158; classtype:attempted-recon; sid:465; rev:1;)
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP L3retriever Ping"; content: "ABCDEFGHIJKLMNOPQRSTUVWABCDEFGHI"
; itype: 8; icode: 0; depth: 32; reference:arachnids,311; classtype:attempted-recon; sid:466; rev:1;)
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP Nemesis v1.1 Echo"; dsize: 20; itype: 8; icmp_id: 0; icmp_seq:
 0; content: "|00000000000000000000000000000000000000|"; reference:arachnids,449; classtype:attempted-recon; sid:467;
 rev:1;)
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP PING NMAP"; dsize: 0; itype: 8; reference:arachnids,162; class
type:attempted-recon; sid:469; rev:1;)
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP icmpenum v1.1.1"; id: 666; dsize: 0; itype: 8; icmp_id: 666 ;
icmp_seq: 0; reference:arachnids,450; classtype:attempted-recon; sid:471; rev:1;)
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP redirect host";itype:5;icode:1; reference:arachnids,135; refer
ence:cve,CVE-1999-0265; classtype:bad-unknown; sid:472; rev:1;)
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP redirect net";itype:5;icode:0; reference:arachnids,199; refere
nce:cve,CVE-1999-0265; classtype:bad-unknown; sid:473; rev:1;)
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP superscan echo"; content:"|0000000000000000|"; itype: 8; dsize
:8; classtype:attempted-recon; sid:474; rev:1;)
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP traceroute ipopts"; ipopts: rr; itype: 0; reference:arachnids,
238; classtype:attempted-recon; sid:475; rev:1;)
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP webtrends scanner"; content: "|00 00 00 00 45 45 45 45 45 45 4
5 45 45 45 45 45|"; itype: 8; icode: 0; reference:arachnids,307; classtype:attempted-recon; sid:476; rev:1;)
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP Source Quench"; itype: 4; icode: 0; classtype:bad-unknown; sid
:477; rev:1;)
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP Broadscan Smurf Scanner"; itype: 8; icmp_id: 0; icmp_seq: 0; d
size:4;  classtype:attempted-recon; sid:478; rev:1;)
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP PING speedera"; content: "|3839 3a3b 3c3d 3e3f|"; depth: 100;
itype: 8;  sid:480;  classtype:misc-activity; rev:2;)
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP TJPingPro1.1Build 2 Windows"; content:"|544a 5069 6e67 5072 6f
20 6279 204a 696d|";itype:8;depth:32; reference:arachnids,167; sid:481;  classtype:misc-activity; rev:2;)
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP PING WhatsupGold Windows"; content:"|5768 6174 7355 7020 2d20
4120 4e65 7477|";itype:8;depth:32; reference:arachnids,168; sid:482;  classtype:misc-activity; rev:2;)
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP PING CyberKit 2.2 Windows"; content:"|aaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaa|";itype:8;depth:32; reference:arachnids,154; sid:483;  classtype:misc-activity; rev:2;)
@
                                                                                17,9          64%
```

Regelbaum



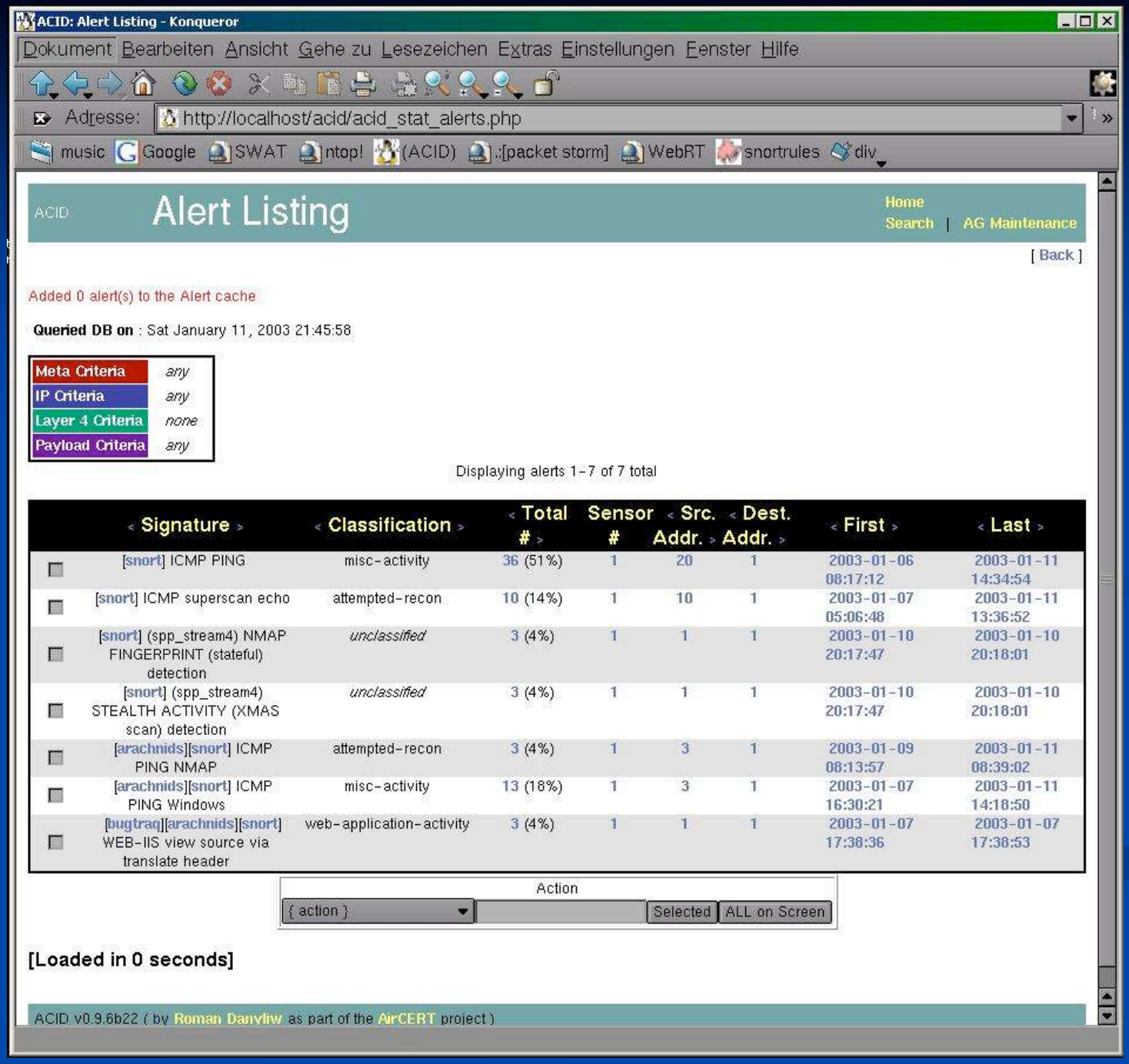
- Wurzel (SrcAdr/M:SrcPRange->DstAdr/M:DstPRange)
  - Unterregel (Inhalt des Packets, Flags etc)
  - Unterregel (Andere Signaturen)
- Nächste Wurzel
  - Unterregeln…

Erfahrungen

- Sehr viele False Positives gerade durch Samba- und SNMP-Regeln
- In der Praxis viele unterschiedliche Arten des selben Traffics (zB Ping, RPC)
- Programme oder OSes nutzen Formate, die mit bekannten Angriffen übereinstimmen
- Viel Arbeit ist das Anpassen aufs Unternehmen

ACID

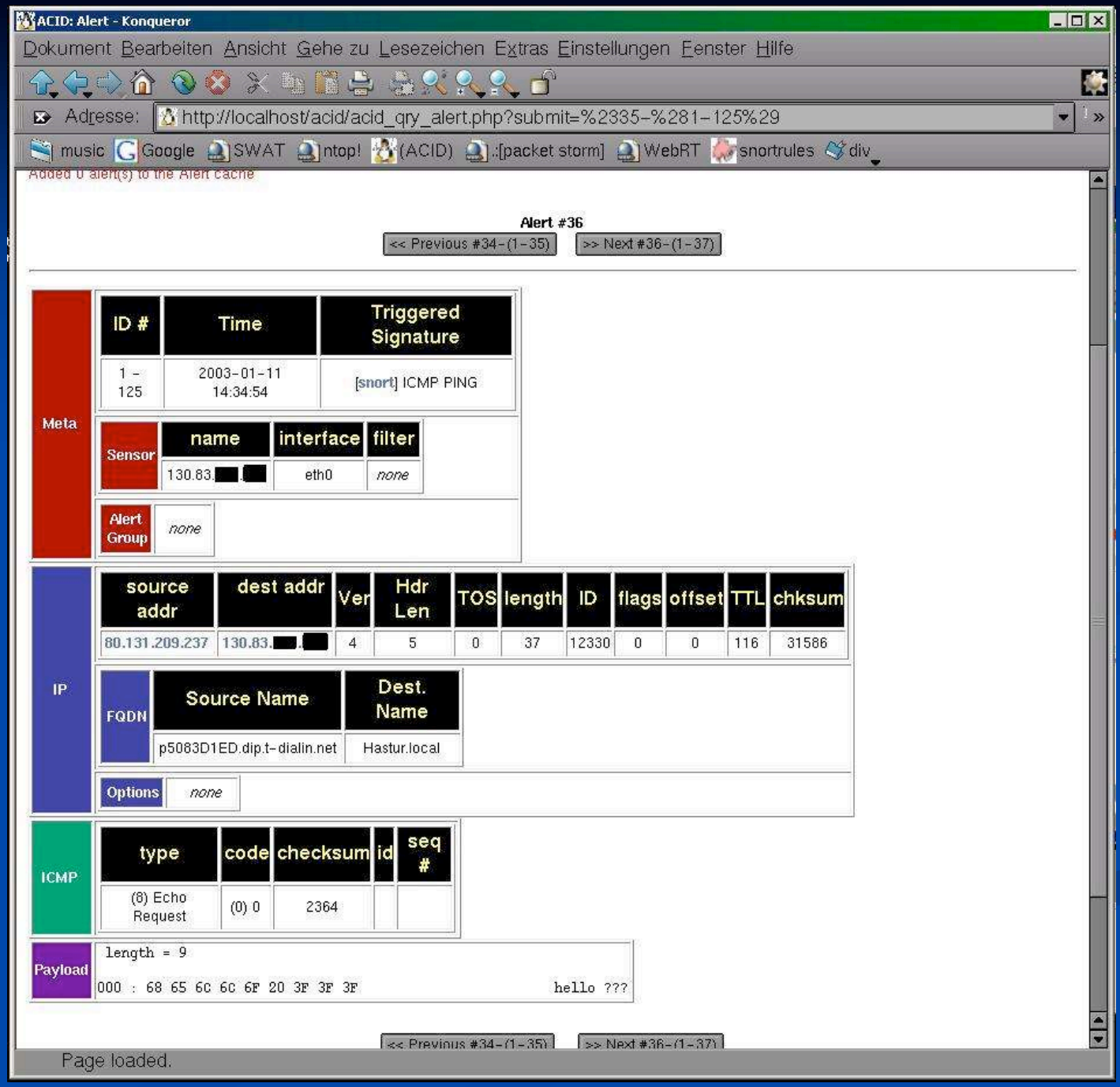
Alertliste



ACID

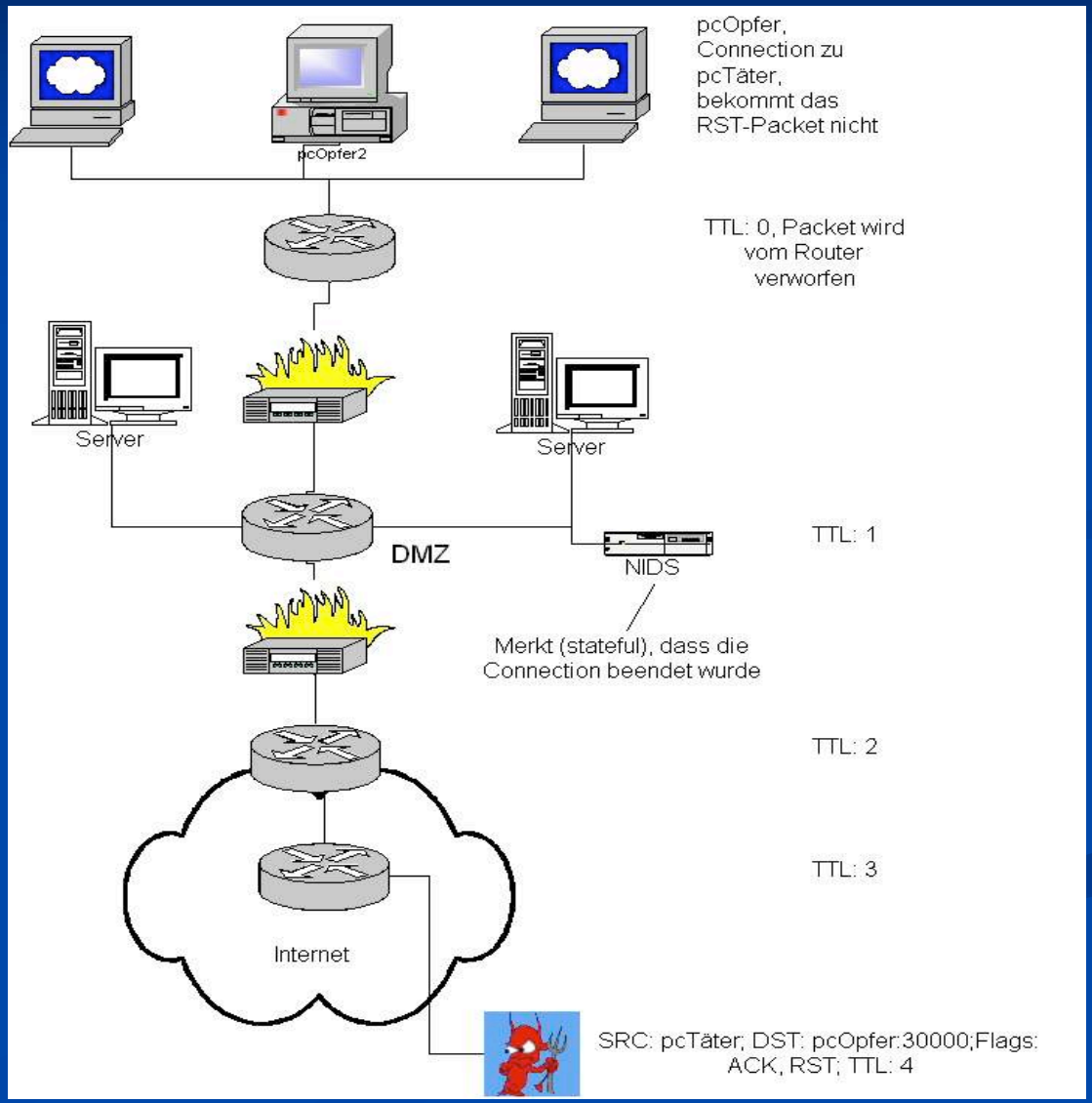
Paketansicht

Gut aufgeschlüsselt und schnell lesbar



Fake Reset

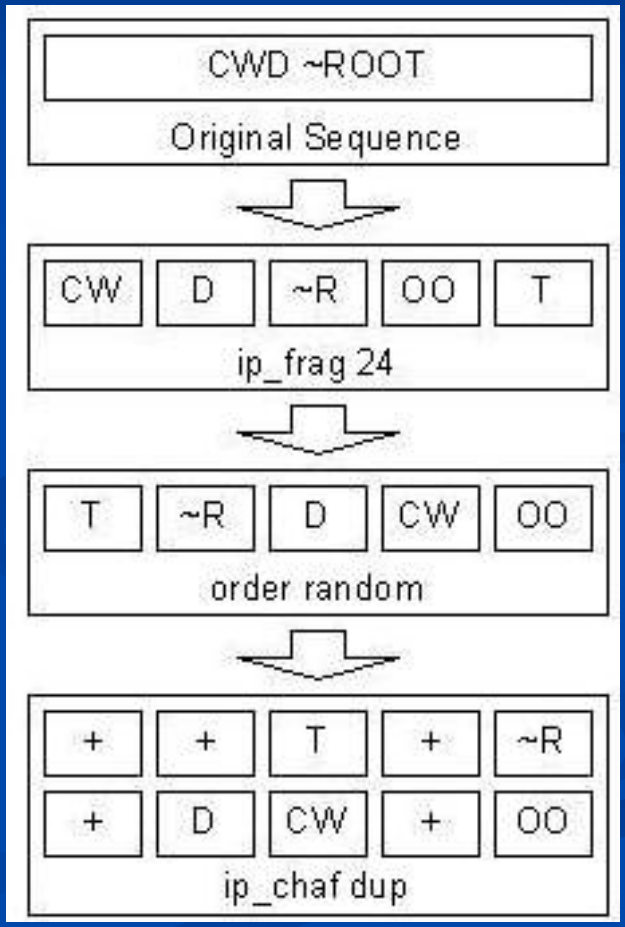
Schicken eines Pakets mit TTL<Hops zum Ziel aber TTL >=Hops zum IDS



Fragmentierung

z.B. mit Fragrouter

- Originalpakete werden in kleinere Stücke fragmentiert (Grösse kann variiert werden um den Signaturraum zu vergrössern)
- Reihenfolge wird verändert um im Zustandsautomaten ein Timeout zu provozieren und Speicher zu belegen
- Leere Füllpakete werden zwecks Speicherverbrauch eingefügt



Session Splicing

- Ähnlich wie Fragmentierungsangriff
- Befehle (Exploit) wird über viele kleine Pakete verteilt
- Pakete sind besonders klein um als simples ACK-Paket zu wirken
- Zwischen den Paketen kann beliebig viel Zeit verstreichen (bis zum Timeout der Verbindung)

Lohnt es sich?

Ja, wenn…

- Man bereits eine Sicherheitsinfrastruktur hat
- Man ein beliebtes Ziel für Angriffe ist
- Man bezüglich Beweissicherung auf der sicheren Seite sein will
- Man etwas Zeit dafür aufwenden will
- Man dynamische Anpassung von ACLs braucht (nicht nur gegen Angriffe)
- Man durch Downtime vielstellige Summen verliert (DoS) oder Server unbedingt schützen muss

Möglichkeiten

- Als Ergänzung zu bestehender Infrastruktur
- Um „mal schnell“ über bestimmte Dinge informiert zu werden
- Um gezielt nach Vorgängen zu schauen
- Für ‚intelligenteres‘ Logging des Traffics
- Auf gar keinen Fall als Ersatz für herkömmliche Sicherheitsmassnahmen geeignet

Scientia est potentia

Ende

Quellen:

- http://www.snort.org
- SANS Institute
- http://www.robertgraham.com/pubs/network-intrusion-detection.html
- TCP/IP (Data Becker)
- Das Usenet
- Cisco Secure IDS Coursebook