Diplomarbeit

Netzwerk

Intrusion Detection Systeme

Ausgeführt zum Zweck der Erlangung des akademischen Grades eines
Dipl.-Ing. (FH) für Telekommunikation und Medien
am Fachhochschul-Diplomstudiengang Telekommunikation und Medien St. Pölten
Vertiefungsrichtung Telekommunikation

ausgeführt von:
Christoph Reikerstorfer
tm0110038093

unter der Leitung von:
Dipl.-HTL-Ing. Andreas Schaupp MSc

Zweitbegutachter:
Dipl.-Ing. Georg Barta

St.Pölten, am 1. Juli 2005 Unterschrift:

Ehrenwörtliche Erklärung

Ich versichere, dass

- ich diese Diplomarbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfe bedient habe.

- ich dieses Diplomarbeitsthema bisher weder im Inland noch im Ausland einem Begutachter/einer Begutachterin zur Beurteilung oder in irgendeiner Form als Prüfungsarbeit vorgelegt habe.

Diese Arbeit stimmt mit der vom Begutachter beurteilten Arbeit überein.

Ybbs, 28. Juni 2005 .....

Unterschrift

Kurzfassung

Unterschiedliche Statistiken zeigen, dass viele Verletzungen der IT-Sicherheit nicht von außen, sondern von innen kommen. Die meisten Netzwerke entsprechen aber derzeit dem Prinzip „Harte Schale, weicher Kern“. Die Außengrenzen werden durch Firewalls strikt überwacht, während im Inneren nahezu Blindheit herrscht. Interne Schwachstellen können jedoch schnell zu den schlimmsten Albträumen eines Unternehmens führen, wenn die interne Sicherheit stiefmütterlich behandelt wird.

Diese Diplomarbeit widmet sich der Problematik von fehlender interner Sicherheit in Netzwerken und zeigt einen Lösungsansatz mit Network Intrusion Detection Systems. Die Firewall ist in Sicherheitsbelangen kein Allheilmittel gegen jegliche Bedrohung. Ein probates Mittel zur Einhaltung der Sicherheit in Netzwerken stellen Intrusion Detection Systeme (ID-Systeme) dar. Diese wirken als Wächter im Netzwerk, analog zu einer Alarmanlage in Gebäuden, und sind in der Lage die Einhaltung der Unternehmenssicherheitsrichtlinien bezüglich Netzwerk-Datenverkehr zu überprüfen.

Der allgemeine Aufbau, die Komponenten, und die Funktionsweise von ID-Systemen, gefolgt von verschiedenen ID-System Typen, werden in den ersten Abschnitten behandelt. Nachstehend werden gängige Netzwerkangriffstechniken geschildert, welche Netzwerk ID-Systeme erkennen müssen. Ein ID-System sammelt eine Fülle von Informationen, auch personenbezogene Daten, daher werden rechtliche Aspekte in diesem Zusammenhang ebenfalls betrachtet. Im Mittelpunkt dieser Arbeit steht die Planung und praktische Umsetzung eines Netzwerk Intrusion Detection Systems mit dem Open Source ID-System Snort. Trotz der vielen positiven Eigenschaften und Verbesserungen der Sicherheitsstruktur durch den Einsatz solcher Systeme bestehen natürlich auch Schwächen und Probleme, die am Ende der Arbeit beleuchtet werden.

Diese Diplomarbeit zeigt, dass in Zukunft Intrusion Detection Systeme eine vernünftige und notwendige Ergänzung zum bestehenden Sicherheitskonzept eines Unternehmens darstellen werden.

Abstract

Statistics have shown that many problems of IT security do not come from outside but from inside. External borders are strictly protected by firewalls whereas there is nearly no internal monitoring although internal vulnerabilities can – if ignored - quickly turn into a company's worst nightmare.

This thesis deals with the problems of internal security in networks and tries to point at solution approaches by means of Network Intrusion Detection Systems. Concerning security, a firewall is by no means a universal remedy against all kinds of threats. Intrusion Detection Systems (ID systems) are an appropriate measure for maintaining security in networks; they act as guardians in networks, comparable to alarm devices in buildings, and are also able to check adherence to a company's security guidelines concerning network data traffic.

General design, components, mode of operation of ID systems as well as different ID system types are presented in the first part. After this, current network attack techniques which a network ID system must be able to recognize are illustrated. An ID system collects an abundance of information, even personal data; for this reason legal restrictions have also been taken care of. This paper focuses on the planning and practical implementation of a Network Intrusion System using the open source ID system Snort.

Although security structure can be enhanced by the many positive features and improvements of such systems, there still remain shortcomings and problems which will be dealt with in the end of this paper.

This thesis shows that Intrusion Detection Systems will present a sensible and inevitable complement to existing security concepts of companies in future.

Inhaltsverzeichnis

1 Einleitung

Der Wandel der Zeit, im Bezug auf die Informationstechnologie (IT), hat es mit sich gebracht, dass Unternehmensnetzwerke eine immer größere Abhängigkeit zum Internet beziehungsweise anderen Netzwerken aufweisen. Man denke dabei an die täglich notwendigen Geschäftsprozesse wie Mailverkehr und Bestellungen, um nur wenige zu nennen. Diese zunehmende Interaktion mit fremden Netzen und Systemen erhöht die Gefahr von Angriffen auf das eigene Netzwerk. Daher hat sich über die letzten Jahre hinweg der Einsatz von Firewalls als Standard etabliert. Die Firewall sorgt dafür, dass nur berechtigte Verbindungen stattfinden. Eine Firewall sichert jedoch nur den Datenverkehr an einem Punkt (Gateway), am Übergang eines Netzes zu einem anderen. Dieser Schutz ist aufgrund des heutigen Erkenntnisstandes unzureichend. Es passiert des Öfteren, dass ein Angreifer die Firewall passiert oder ein verärgerter Mitarbeiter versucht Systeme von innen zu komprimittieren. Ein normaler Rechnerverbund bietet standardmäßig wenig bis gar keinen Schutz vor solchen unerwünschten Aktivitäten. Der Verlust, Missbrauch oder die Verfälschung von Datenbeständen kann in einem Unternehmen den Untergang bedeuten.

Um solchen Risiken vorzubeugen setzt man gerade in jüngster Zeit vermehrt auf so genannte Intrusion Detection Systeme (ID-Systeme). Diese wirken wie eine Art Alarmanlage oder Detektive im Netzwerk und überwachen Systeme auf Missbrauch und Angriffe. Sicherheitskritische Vorfälle sollen möglichst in Echtzeit erkannt beziehungsweise verhindert werden. Die meisten älteren Sicherheitsmethoden orientieren sich an prophylaktischen Methoden, indem sie Zugriffe einschränken. Dadurch können zwar fremde Benutzer ausgeschlossen werden, jedoch versagen diese Systeme bei erratenen Passwörtern oder Spoofing-Angriffen. Das ID-System muss in der Lage sein zwischen, normalem Datenverkehr beziehungsweise Verhalten der User und einem tatsächlichen Angriff zu unterscheiden.

Durch die rapide Vernetzung, die immer größer werdenden Bandbreiten und permanenten Verbindungen zu anderen Netzen ist die Sicherheit in IT-Systemen

heute wichtiger denn je, daher bedarf es auch eines Intrusion Detection Systems in einer modernen Sicherheitsarchitektur.

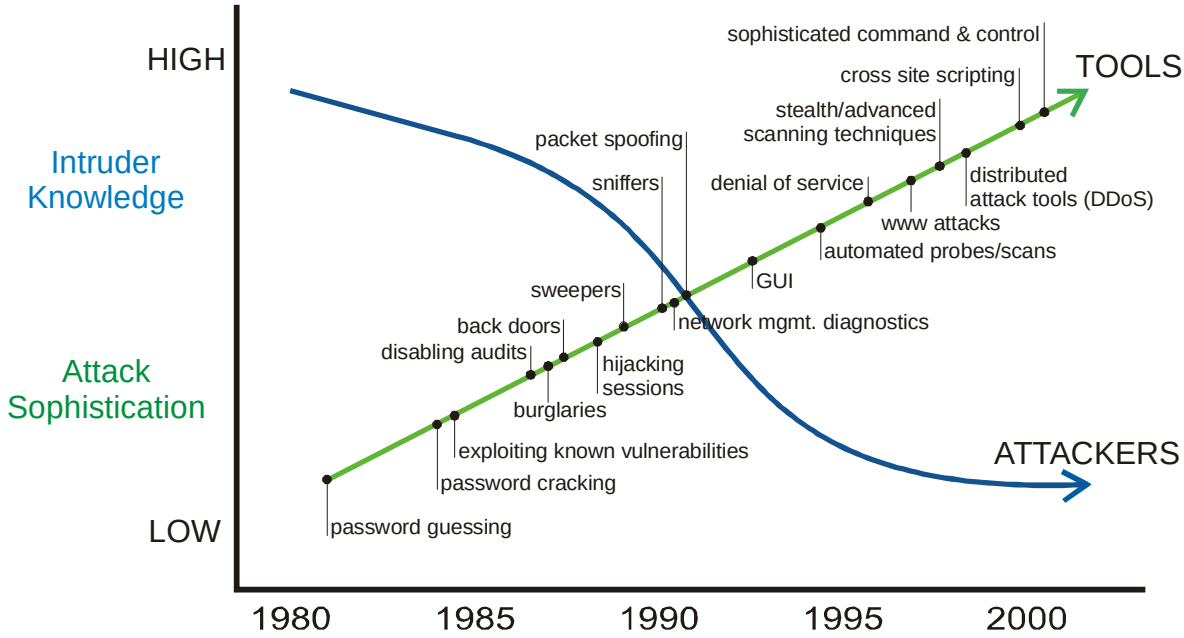


Abbildung 1-1: Raffinesse der Attacken vs. technisches Wissen der Angreifer

[vgl. CERT 02]

Die Abbildung 1-1 zeigt, dass trotz zunehmender Raffinesse der Angriffe das technische Wissen des durchschnittlichen Angreifers sinkt. Nur wenige kluge Köpfe erfinden neue Angriffstechniken oder entdecken Systemschwächen. Die breite Masse der selbsternannten Hacker bedient sich fertiger Software wie Script-Kiddies oder Trojaner, die im Internet an vielen Orten zu bekommen sind. Oftmals wissen diese neugierigen User oder verärgerten Mitarbeiter gar nicht, welchen enormen Schaden sie durch diese Toolkits anrichten. Dieser Trend spricht auch eindeutig für den Einsatz eines ID-Systems zur vernünftigen Absicherung einer IT-Infrastruktur. Viele dieser Tools würden bei einem bestehenden ID-System bereits im Keim ersticken.

1.1 Warum benötigt man ein Intrusion Detection System?

Nicht der böse Hacker von außen stellt die größte Bedrohung für ein Firmennetzwerk dar. Prozentuell betrachtet finden Angriffe viel häufiger von innen (66 Prozent) [vgl. CSI 04], zum Beispiel durch neugierige, technisch versierte Benutzer, oder verärgerte Mitarbeiter statt. Als Angriff definiert man Handlungen,

deren Ziel es ist, die Integrität, Verfügbarkeit oder Vertraulichkeit eines Systems zu verletzen. Ein nicht zu unterschätzender Vorteil sind die Vorkenntnisse eines internen Angreifers. Er muss nicht erst seine Ziele erkunden, wodurch er vielleicht entdeckt wird, sondern kennt den Netzwerkaufbau bereits. Auch wenn der beleidigte Mitarbeiter keine lokalen administrativen Rechte besitzt, kann er sich Live-CDs[1] (z.B. Knoppix oder Win PE) oder bootfähiger USB-Sticks bedienen und geheime Daten mit einem Netzwerk-Sniffer ergaunern. In vielen Fällen wird der Angriff gar nicht entdeckt.

Notebooks, PDAs (Personal Digital Assistants) und Smartphones besitzen oft ein nicht ausreichendes Sicherheitsniveau, da sie zeitweise in nicht vertrauenswürdigen Netzen betrieben werden. Am Firmennetz angeschlossen können sich Viren und Würmer unter Umgehung aller Sicherheitsmaßnahmen ungehindert verbreiten.

Gegen viele dieser Problematiken bietet eine übliche Firmenfirewall keinen Schutz. Diese schaffen ohnehin oft nur ein falsches Gefühl von absoluter Sicherheit statt tatsächlicher Sicherheit. Firewalls sind kein Allheilmittel und lösen nicht alle Sicherheitsprobleme, sie sind Teile eines Sicherheitsmodells, aber nicht das Sicherheitsmodell! In vielen Unternehmen besteht die Sicherheit nur durch eine Firewall, welche das interne Netzwerk (LAN) von einem externen Netz (z.B. Internet) trennt. Dabei werden die Verbindungsanfragen zwischen den Netzen nach bestimmten Regeln überprüft und aufgrund derer zugelassen oder nicht. Zugriffe von außen nach innen sind meist nicht, oder nur in bestimmten Fällen erlaubt.

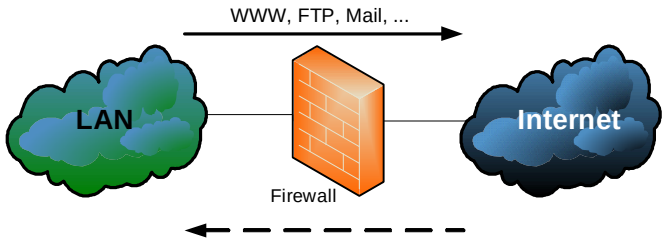


Abbildung 1-2: Sicherheit durch Firewall

[1] Eine bootfähige CD auf der ein vollfunktionsfähiges Betriebssystem ohne Festplatte läuft.

Die Abbildung 1-2 soll veranschaulichen, dass Datenabläufe in den Wolken (externes, internes Netz) für die Firewall unsichtbar sind und deswegen auch nicht kontrollierbar sind.

Schwächen von Firewalls sind:

- *Angreifer ist schon im LAN; Firewall ist wirkungslos*
- *Angriff erfolgt über eine Hintertür (Modemverbindung,VPN)*
- *gefährliche Daten werden durch erlaubte Verbindungen (HTTP) getunnelt*
- *schützt nicht vor neuen Gefahren*
- *kann sich nicht selbst optimieren*
- *überprüfen Datenpakete oft nur begrenzt (einfacher Paketfilter)*

Gegen viele dieser Probleme schafft ein ID-System Abhilfe. Weiters ist es für den Sicherheitsbeauftragten wichtig zu wissen, ob die Sicherheitsrichtlinien im Unternehmen eingehalten werden (z.B. Verbot von Messenger wie ICQ). Mithilfe eines ID-Systems kann das unbefugte Verändern einer Systemkonfiguration, das unerlaubte Installieren von gefährlicher Software oder Netzwerkprotokollen überwacht werden. Es kann aber wiederum nicht die Funktionalität einer Firewall ersetzen. Vielmehr soll das ID-System eine sinnvolle Ergänzung zur bestehenden Sicherheitsstruktur sein. Abbildung 1-3 zeigt einen typischen Zyklus bei der Umsetzung einer Sicherheitsstruktur. [vgl. Spenneberg 03]

Das Sicherheitskonzept muss von realen Personen umgesetzt, geleitet und überprüft werden. Es ist unbedingt schriftlich festzuhalten, wer für welche Aufgaben zuständig ist, sonst entstehen Grauzonen, für die sich niemand verantwortlich fühlt [vgl. Leitold 03].

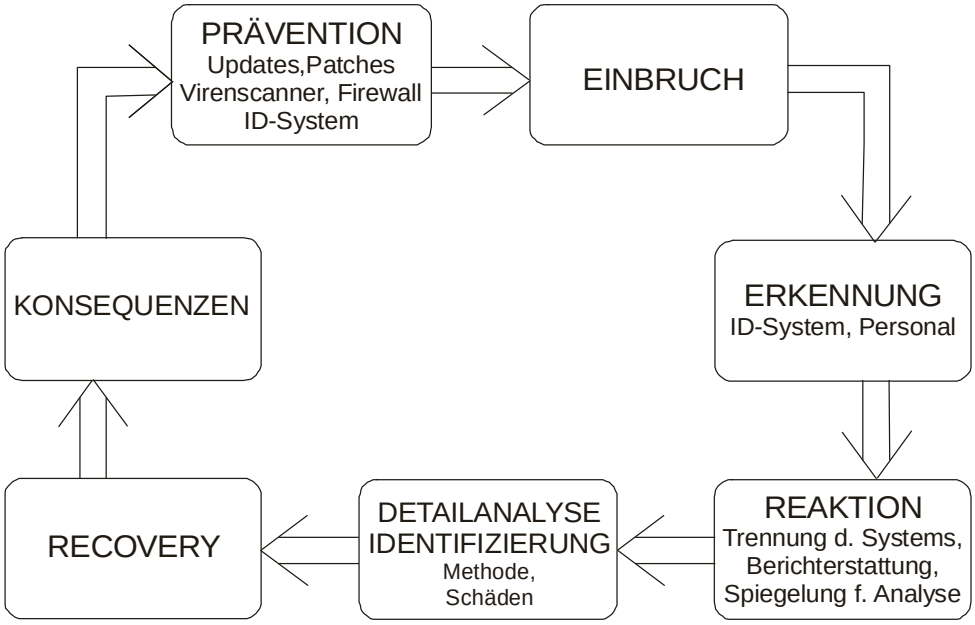


Abbildung 1-3: Zyklus einer Sicherheitsstruktur

ID-Systeme sollten nicht nur für Unternehmen ein Thema sein. Oder wollen Sie, dass ihre drei Computer hinter dem ADSL-Anschluss zu Zombies werden? Zombies sind Computer die ohne Wissen des Besitzers einen Hacker bei Großangriffen gegen ein Unternehmen (z.b. Februar 2000 DDoS-Angriff gegen eBay) unterstützen. Daraus folgt: kein Netzwerk ist klein genug, um ungeschützt zu bleiben! [vgl. Anonymous 03]

1.2 Definition, Grundprinzip

Intrusion Detection Systems heißen übersetzt „einbruchserkennende Systeme“, sie dienen also primär der Feststellung des unerlaubten Eindringens in einen Rechner oder das Netzwerk. Eine „Intrusion“ bezeichnet eine absichtliche Verletzung der Sicherheitsmaßnahmen durch unerlaubte Zugriffe oder Aktivitäten. Um dies zu verhindern, wird der Datenverkehr analysiert und anhand von Regeln bewertet. Bei einem stattfindenden Angriff können die ID-Systeme diesen erkennen und je nach Produkt auch unterbinden (siehe 2.1.4 Intrusion Response Systeme). Das ID-System versucht also aus seinem Überwachungsgebiet jene Ereignisse zu filtern, die auf Angriffe oder Missbräuche hinweisen. Der Sicherheitsadministrator erhält Einblick in das Innere des Netzwerks, ähnlich wie ein Mediziner beim Bluttest Einblick in das Innere des Körpers bekommt, um gefährliche Viren oder Stoffe festzustellen. Wird anhand der analysierten Daten

ein Vergehen gegen die Sicherheitsrichtlinien festgestellt meldet das System an den Sicherheitsbeauftragten per Mail, PopUp-Fenster, SMS, Pager etc.

Ein ID-System ist nicht für sich alleine zu betrachten, vielmehr besteht es aus einer Vielzahl von Komponenten und Werkzeugen, welche als Zusammenstellung das Intrusion Detection System ergeben.

2 Aufbau eines ID-Systems

2.1 Arbeitsweise

Prinzipiell arbeitet ein ID-System in vier Schritten [Hildebrandt 01]. Sicherheitsadministratoren setzen Software- oder Hardware-Sensoren (Agenten) an zentralen Punkten des Netzwerks (Gateways) ein, um Daten zu sammeln. Die Management-Station analysiert die gewonnenen Informationen und stellt sie entsprechend dar. Bei Verletzung bestimmter Regeln können Gegenmaßnahmen erfolgen.

- *Sammeln von Daten*
- *Analyse der gesammelten Daten*
- *Visualisierung der Ergebnisse*
- *Gegenmaßnahmen*

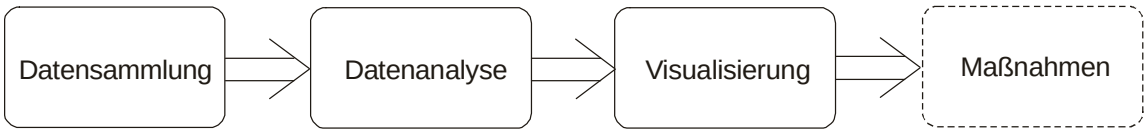


Abbildung 2-1: Arbeitsschritte eines IDS

Der letzte Block des Diagramms ist grundsätzlich nicht Bestandteil des ID-Systems. Man spricht dann bereits von einem Intrusion Response System (siehe Kapitel 2.1.4) oder einem zusätzlichen Feature des ID-Systems.

2.1.1 Datensammlung

Damit das ID-System gute Arbeit leisten kann, muss es Auditdaten aus verschiedenen, vor allem verlässlichen Quellen beziehen. Folgende Daten werden je nach Art des ID-Systems üblicherweise benutzt:

- *Netzwerkdaten*

Beispiele: Durchsatz, IP-Adressen, Portnummern, Flags

- *Daten des Betriebssystem*

Beispiele: CPU-Last, virtueller Speicher, Dateizugriffe, Netzdienste, Anmeldeversuche

Insbesondere die Zugriffe auf das Dateisystem sind interessant, da protokolliert wird, wie oft eine Datei geöffnet wird und wer sie öffnet oder versucht zu öffnen.

- *Programmdaten*

Beispiele: Log-Daten diverser Applikationen für ein Application IDS (vgl. Kapitel 4.5)

Falls die Daten bereits von einer höheren Ebene bezogen werden (z.B. Proxy-Server), so können unter Umständen schon wichtige Informationen auf einer niedrigeren Ebene verloren gehen, um einen Angriff zu erkennen.

Entscheidend ist, dass die gesammelten Daten für die spätere Analyse nicht zu groß werden, womit das System stark belastet wird, aber auch nicht zu klein werden, wodurch vielleicht wichtige Informationen zur Erkennung verloren gehen.

2.1.2 Analyse der gesammelten Daten

Im diesem Arbeitsschritt werden die zuvor gesammelten Datenbestände auf bösartige Inhalte überprüft und bewertet. Es findet ein Erkennungsprozess statt. Dabei unterscheidet man zwischen zwei Techniken. Die Missbrauchserkennung anhand von Signaturen (**misuse detection**) und die Anomalieerkennung (**anomaly detection**), die auch unbekannte, neuartige Angriffsversuche aufdecken soll.

- *Erkennung durch Signaturen (misuse detection)*

Meist benutzen Angreifer bekannte Techniken oder Lücken in Systemen, die bekannt sind, und in Form von Signaturen, Mustern in einer Datenbank

gespeichert sind. Das ID-System weiß also, wonach es suchen muss. Exploits[2] [Anonymous 03] weisen meist spezielle Charakteristika auf, die mit der Referenzdatenbank verglichen werden. Jedermann kann sich auf http://online.securityfocus.com über die neuesten Exploits und Sicherheitslücken informieren. Die SANS-Gruppe (SysAdmin, Audit, Network, Security) bietet eine durch Sicherheitsexperten erstellte Top 20 Liste [www 11] der häufigsten Schwachstellen in Unix- und Windows-Systemen.

Bei Übereinstimmen einer Signatur mit einem bestimmten Exploit schlägt das System Alarm. Die Signatur ist wie ein Fingerabdruck eines Angriffs. Als Beispiel lassen sich hier ungewöhnliche TCP-Flag Kombinationen nennen. Dies wäre ein Hinweis für einen möglichen Portscan. Generell ist dieses Verfahren der Signaturerkennung ähnlich dem von Viren-Scannern und erkennt eben nur bekannte Missbräuche, wie TCP/UDP Portscans, Fingers, Bufferoverflow Attacks, Denial-of-Service Attacks und so weiter; dazu mehr im Kapitel 3 (Angriffsformen). Daher ist es auch sehr wichtig, die Signaturen regelmäßig zu aktualisieren. Nachteilig ist, dass es erstens für den Angriff eine entsprechende Signatur geben muss, und zweitens die Methode etwas zeitraubend ist. Im Falle einer Angriffserkennung ist die misuse detection aber ziemlich genau, womit die Falschalarm-Rate niedrig ist.

- *Anomalieerkennung (anomaly detection)*

Hierbei verwendet man einen statistischen Ansatz: besondere Abweichungen vom „normalen“ Verhalten lösen Alarm aus. Dafür muss das ID-System über einen gewissen Zeitraum entsprechend „trainiert“ werden. Unbekannte, noch nie da gewesene Angriffstechniken, können so theoretisch aufgespürt werden. Die Abweichung vom Normalverhalten bietet also die Entscheidungsgrundlage für das System. Um diese Abweichung festzustellen gib es verschiedene Methoden:

Statistische Methode:
Das System stützt sich auf statistische Auswertungen (z.B. die CPU-Last liegt „immer“ zwischen 30 - 40% auf jenem Server). Bei einer starken Abweichung

[2] Applikation, die Schwachstellen eines Systems mit destruktiver Absicht nutzt.

dieser Werte über einen gewissen Schwellwert (thresh-hold) vermutet das ID-System ein atypisches Verhalten und verständigt je nach Konfiguration den Administrator oder leitet weitere Aktionen ein. Diese Methode benötigt eine gewisse Anlaufphase, während dieser die Normalprofile für User und Netzwerk etc. in der Datenbank erstellt werden. Die Profile werden periodisch aktualisiert. Ist ein Angreifer während der Trainingsphase des ID-Systems bereits im eigenen System eingebunden, wird er mit diesem Verfahren nicht entdeckt, sein Verhalten wurde ja mitgelernt, ist daher als „normal“ bekannt!

Regelbasierte Methode:
Dieses Verfahren entspricht eigentlich der statistischen Methode, mit dem Unterschied, dass der Sicherheitsadministrator zusätzlich Regeln, Parameter nach seinen Vorstellungen einbringen kann. Schwierig ist es allerdings, die Parameter wirklich gut zu setzen.

Künstliche Intelligenz:
Bei dieser Methode erkennt das System Anomalie durch adaptives Lernen. Es ist daher keine Parametereingabe durch den Benutzer erforderlich. Nachteilig ist der Umstand, dass nicht nachvollzogen werden kann, warum ein Alarm gemeldet wird.

Im Gegensatz zur Signaturanalyse ist die Anomalieerkennung schwieriger und benötigt großen Administrationsaufwand. Fehlalarme (*false positives*) bei anomaly detection sind relativ häufig, da auch Softwarefehler oder Bedienungsfehler Alarme schnell auslösen. Noch ärgerlicher sind *false negatives*, dies sind Angriffe die nicht erkannt werden. Das System hätte Alarm auslösen sollen.

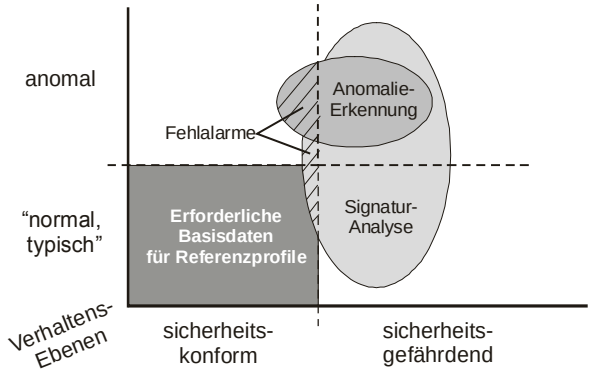


Abbildung 2-2: Verhaltensebenen der Verfahren [vgl. Sobirey 99]

In Abbildung 2-2 zeigt die Verhaltensebenen beider Methoden auf. Die Fläche der Anomalie-Erkennung, welche in den sicherheitskonformen Bereich ragt, ist

deutlich größer als die der Signatur-Analyse und verdeutlicht die höhere Fehlerrate.

Sowohl die Signaturerkennung, als auch die Anomalieerkennung haben ihre Vor- und Nachteile. Darum verwenden viele Hersteller eine Hybridtechnik an, wo die Vorteile beider Techniken vereint werden. Leider erhöht sich auch der administrative Aufwand stark.

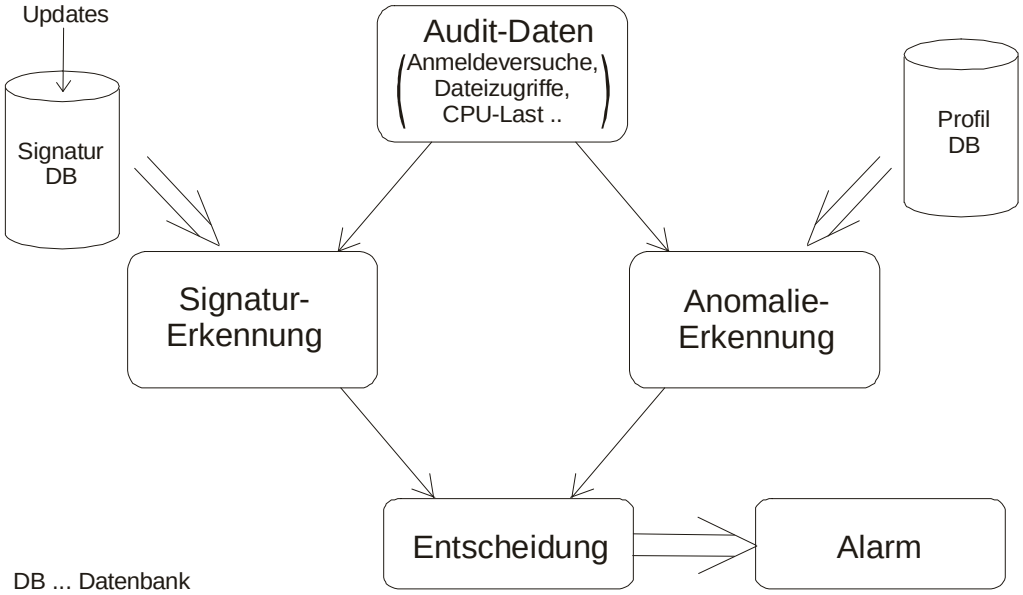


Abbildung 2-3: Hybrides IDS

2.1.3 Visualisierung der Ergebnisse

Wer einmal die Logfiles einer größeren Firewall durchforstet hat, kennt das Dilemma: tausende Zeilen, die unüberschaubar wirken. Um die Ergebnisse für den Administrator brauchbar darzustellen ist eine Aufbereitung nötig. Simple Formen bieten nur eine einfache JA/NEIN Anzeige, andere bieten ein ausgeklügeltes Webinterface mit vielen Details zum Angriff. Die anomaly detection liefert meist nur eine Verdachtsentscheidung, wo angezeigt wird, welcher Parameter überschritten wurde. Genauere Aufschlüsse über den Angriff lässt die signaturbasierte Erkennung zu, da sie ja genau für eine bestimmte Angriffsmethode zutrifft.

Die Benachrichtigung über einen Vorfall im System kann über mehrere Wege - wie zum Beispiel SMS, Mail, Pager, Alarmglocke, PopUp-Fenster etc. - erfolgen. Man sollte sich jedoch im Klaren darüber sein, dass ein geschickter Angreifer

womöglich auch den Datenverkehr mitlauscht und diverse Mails des ID-Systems sieht. Daher empfiehlt es sich Meldungen über sichere Wege, eventuell kryptografisch gesichert (verschlüsselt), zu senden.

2.1.4 Gegenmaßnahmen

Im Falle eines Angriffs ist schnelles Handeln notwendig, trotzdem lautet die oberste Regel: „Keine Panik“, sonst passieren grobe Fehler. Angenommen ein überforderter Administrator entscheidet sich voreilig für einen Shutdown, und vernichtet damit wichtige Aufzeichnungen! Ratsamer ist, das betroffene System vom Netzwerk zu trennen, somit ist der Angreifer mit Sicherheit ausgesperrt.

Einige ID-Systeme bieten ein integriertes Intrusion Response System (IR-System) [vgl. Kapitel 4.4], welches nahezu in Echtzeit auf Bedrohungen aktiv reagieren kann. Die Betonung liegt auf aktiv, denn passiv reagiert auch ein normales ID-System mit Meldungen. Oberstes Ziel ist es, durch schnelle Reaktion Schäden zu vermeiden und vielleicht den Angreifer zu identifizieren. Die Umsetzung des IR-Systems kann manuell, halbautomatisch oder vollautomatisch erfolgen. Halbautomatisch bedeutet, dass es vor der Handlung noch eine Interaktion mit dem Systembetreuer gibt, was natürlich auch zu einer Zeitverzögerung führt. Die Möglichkeiten können je nach Konfiguration und Produkt vielfältig und umfangreich sein. Hosts können vom lokalen Netz abgeschottet werden (sperren der IP-Adresse), Verbindungen können getrennt werden oder die Firewall wird vom Response-System rekonfiguriert. Die Maßnahmen in solch einem Fall müssen in den Sicherheitsrichtlinien stehen. Ein mögliches Szenario wäre sogar, einen Gegenangriff zu starten, was jedoch aus rechtlicher Sicht nicht unbedenklich ist. Mögliche aktive Reaktionen sind:

- *Rekonfiguration der Firewall*
- *Rekonfiguration der Router*
- *Sperren von IP-Adressen*
- *Abschalten von Diensten (schließt Ports)*

Die automatische Reaktion auf Einbrüche sollte seitens des Herstellers so implementiert sein, dass der Angreifer diese Funktionalität nicht für ein Denial-of - Service nutzen kann.

Nach jedem relevanten Vorfall muss eine Abschätzung und spätere Detail-Analyse erfolgen:

- *Welche Systeme wurden kompromittiert?*
- *Wann wurde der Einbruch erkannt?*
- *Wie viele Netze sind betroffen?*
- *Konnte der Angreifer administrative Rechte erlangen?*
- *Kam der Angriff von innen oder außen?*
- *Welche Methode hat der Angreifer verwendet?*

Nicht immer können alle Fragen beantwortet werden. In Abhängigkeit von den Fähigkeiten des Eindringenden kann die genaue Analyse mehr oder weniger Erkenntnisse liefern.

Anhand eines bestehenden und aktuellen Notfallplans sollten die Zuständigen Personen im Unternehmen umgehend informiert werden. Nach einem Einbruch sollten die Präventionsmaßnahmen überarbeitet beziehungsweise neu entwickelt werden, um vor neuen Gefahren geschützt zu sein. Eine Gruppe von Experten sollte evaluieren [vgl. Leitold 03]:

- *Welche Systeme haben versagt, oder wurden überwunden?*
- *Warum kam es zu diesem Versagen?*
- *Gibt es alternative Produkte oder Verbesserungen zum bestehenden System?*

Eine Wiederherstellung der Systeme durch Backups und ein Übergang zur normalen Tagesordnung genügt nicht! Anhand der forensischen Analyse müssen Verbesserungen an der Sicherheitsstruktur stattfinden.

2.2 Komponenten eines ID-Systems

Um diese Arbeitsschritte durchführen zu können besteht ein übliches ID-System aus mehreren Komponenten [vgl. BSI 02a]:

2.2.1 Sensoren

Die Sensoren überwachen Teile eines Netzes, oder dedizierte Rechner. Sie liefern die wesentlichen Informationen für das ID-System. Man unterscheidet hierbei zwischen netzbasierten Sensoren, welche den Verkehr in Segmenten überwachen, oder hostbasierten Sensoren, die den Datenfluss auf einem Rechner überwachen, aber auch Prozesse und Dateizugriffe am System. Hostsensoren liefern mehr verwertbare Information, belasten den Rechner aber auch zusätzlich. Netzsensoren erkennen Angriffe auf mehrere Systeme gut und können außerdem für den Angreifer unsichtbar sein (haben keine IP-Adresse), was hostbasierte nicht sind. Je nach eingesetztem System kann der Netzsensor ein Computer im „promiscuous mode“ oder ein spezielles Hardwaredevice sein. Im promiscuous mode liest die Netzwerkkarte auch Pakete, die nicht für den eigenen Rechner bestimmt sind. Normalerweise werden diese Datenpakete verworfen.

2.2.2 Datenbank

Um sinnvolle Aussagen zu liefern braucht das ID-System viel Information. Dadurch entstehen riesige Datenmengen. Eine Verwaltung auf Basis von Dateien ist also nicht möglich. Daher verwendet man Datenbanken, welche die großen Datenmengen komfortabel organisieren und Zugriffszeiten minimieren. Die Datenbank kann sich auf der Managementstation selbst befinden oder auf einem zentralen Datenbankserver abgelegt sein. Die Daten sind natürlich vor unbefugten Zugriffen streng zu schützen. Wie lange sollen die Daten aufbewahrt werden? Die Antwort auf diese Frage hängt vom verwendeten System und von der Menge der aufgenommenen Information ab. Als Richtwert kann man cirka 6 Monate nennen.

2.2.3 Management-/Auswertungsstation

Von dieser Station erfolgt die Konfiguration und Wartung des ID-Systems und der Sensoren. Dies stellt die zentrale Komponente dar. Meist sind Management-/Auswertungsstation und Datenbank auf einem Computer vereint. Hier werden die

Daten zentral gesammelt und bewertet. Die Komponenten des ID-System - wie Sensoren und Datenbanken - werden zusammengeführt, Konfigurationsparameter (IP-Adressen, Verschlüsselung etc.) werden eingestellt. Dieser Rechner wird auch als IDS-Server bezeichnet.

Die Kommunikation zwischen den Sensoren und der Managementstation sollte unbedingt gesichert oder über ein eigenes physikalisches Netzwerk erfolgen. Die meisten Hersteller setzen hierzu Verschlüsselungsmethoden ein. Schließlich soll der Eindringling die Meldungen des ID-Systems nicht mitlesen oder gar verfälschen können.

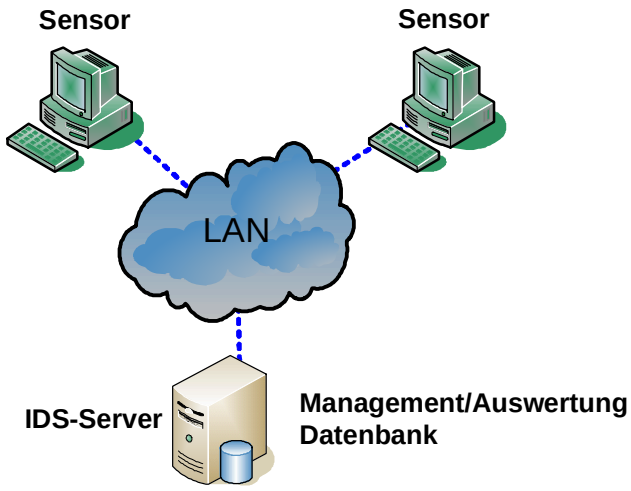


Abbildung 2-4: Komponenten eines ID-Systems

Große Aufmerksamkeit sollte man auch der Absicherung der Managementstation selbst schenken. Nicht jeder Administrator darf Netzsensoren deaktivieren bzw. konfigurieren können. Ein entsprechendes Rollenkonzept für Benutzer (verschiedene Administratoren-Levels) empfiehlt sich an dieser Stelle.

2.3 Wie wird das ID-System geschützt?

Ist die Sicherheit des ID-Systems selbst nicht mehr gegeben, kann es zu vielen Fehlalarmen oder zum Entfall von Meldungen führen, weil es selbst zum Opfer wurde. Eventuell gelingt es dem Angreifer das ID-System zu deaktivieren, bevor er den tatsächlichen Angriff startet. Zur Vermeidung solcher Ereignisse sollte man sich an folgende Empfehlungen halten:

- *Keine zusätzlichen Dienste auf einer Maschine des ID-Systems*

Viele Administratoren benennen diesen Vorgang als System-Hardening.

- *Regelmäßig die aktuellen Security-Patches des Betriebssystems einspielen*

Wenn dies nicht erfolgt, könnte das System womöglich durch Schwächen des Betriebssystems ausgehebelt werden.

- *Rechner sollte unsichtbar sein*

Auf ICMP echo requests erfolgt keine Antwort (stealth), der Angreifer weiß nicht, ob das ID-System wirklich existiert.

- *Unerwünschten Datenverkehr blocken*

Mit Hilfe einer Firewall (z.B. iptables auf Linux) wird unnötiger Verkehr geblockt.

- *Nicht mehr User-Accounts als unbedingt notwendig für ID-System*

Manche Betriebssysteme wie Linux erlauben sogar den Sensor-Betrieb ohne Zuweisung einer IP-Adresse am Interface. Somit ist der Rechner über das Netzwerk nicht zugänglich!

Insertion – Evasion, Angriff auf das ID-System

Insertion beschreibt das „Einflechten“ von Datenpaketen in einen Datenstrom. Das Prinzip beruht darauf, dass das ID-System ja wirklich jedes Paket mitlauscht und beurteilt. Das bedeutet aber auch, dass man dem Sensor Pakete schicken kann, die ein normaler Zielhost gar nicht akzeptiert (z.B. fehlerhafte Prüfsumme), kein anderer Host kümmert sich um diese Daten. In Folge dessen kann also der Datenstrom, den das netzbasierte ID-System für den Signaturvergleich heranzieht, ein anderer sein, als der Zielrechner annimmt. Das ID-System kann keine bösartigen Inhalte erkennen, trotzdem findet auf dem Zielrechner ein Angriff statt. Insertion ermöglicht somit die Tarnung eines Angriffs. Viele ID-Systeme sind anfällig auf diese schwer durchzuführende Methode. Abhilfe schafft nur eine geeignete Feinabstimmung des Regelwerks.

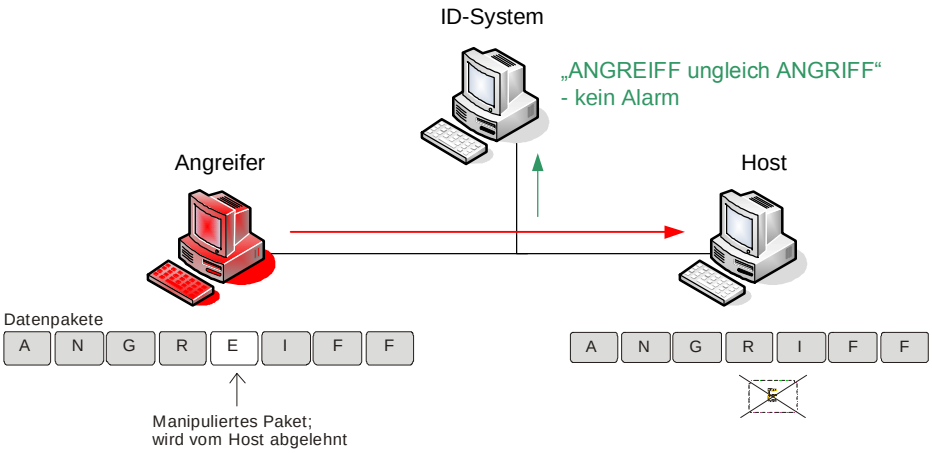


Abbildung 2-5: Insertion

Evasion ist das Gegenstück zu Insertion. Ein Host im Netzwerk bekommt Pakete, die das ID-System nicht akzeptiert. Das Grundprinzip beider Methoden ist gleich: das System bewertet einen anderen Datenstrom als der Rechner empfängt. In der Praxis sind beide Varianten nur sehr schwer und mit viel Wissen zu realisieren, aber dennoch möglich.

[vgl. Brutschy 03] [vgl. Hildebrandt 01]

3 Angriffsformen

Die folgenden Beschreibungen zu Angriffen sind keinesfalls neu oder eine vollständige Auflistung. Dies würde den Rahmen einer Diplomarbeit sprengen. Mehr Detailinformationen zu Angriffsmethoden und Schwachstellen in Netzwerken bietet die Diplomarbeit „Maßnahmen gegen Sicherheitslücken in Netzwerk-Infrastruktur-Komponenten“ von Bernd Hahnbauer (FH St. Pölten 2002). Es soll eine kurze Einführung über die Methoden von Angreifern sein und welche Schwachstellen sie ausnutzen. Ein wesentliches Problem ist, dass die meisten Verbindungen ohne Verschlüsselung stattfinden, und so ein großes Potential für Angriffe bieten. Die meisten Protokolle, die heute eingesetzt werden, sind lange bewährt (TCP/IP Anfang der 70er Jahre) und erfüllen ihren Zweck, lassen jedoch vom Standpunkt der Sicherheit einiges zu wünschen offen. Zu deren Entwicklungszeit konnte man diese Probleme auch nicht voraussehen. Nur langsam vollzieht sich ein Wechsel zu sichereren Protokollen, wie HTTPS, SCP oder SSH.

Nicht nur Designfehler in Protokollen werden genutzt, falsche Konfiguration der Zugriffsrechte, Implementierungsfehler oder falsches Benutzerverhalten stellen genauso eine breite Angriffsfläche dar.

Damit der Angreifer einen erfolgreichen Zug gegen ein System machen kann, versucht er im Vorfeld gewisse Informationen zu erlangen. Ein beliebtes Mittel dazu ist das „social engineering“, wobei zum Beispiel ein Telefonanruf bei der Sekretärin genügt: Der Angreifer ist freundlich und verwirrt die Dame mit technischen Details und sagt, er wäre der neue IT-Mitarbeiter und bräuchte ihr Passwort um neue Software für sie zu installieren. In einer etwas größeren Firma führt dies schnell zu einem gültigen Usernamen und Passwort. Eine weitere vorbereitende Maßnahme ist ein TCP- oder UDP-Portscan, der Aufschluss über offene Ports und verwendetes Betriebssystem und User-Infos bringen kann. Meist vollzieht sich ein Angriff in drei Phasen:

- *Erkunden – Eindringen - Modifizieren*

3.1 Kategorien

Angriffe lassen sich meist kategorisieren. Eine strikte Trennung ist hier nicht immer möglich, daher kann ein Angriff oft mehreren Kategorien zugeordnet werden. Viele lassen sich in folgende einteilen [Helden 98]:

- *Fehlverhalten von Benutzern*

Verwendet ein User entgegen der Sicherheitspolitik sehr schwache Passwörter oder macht er sie anderen leicht zugänglich (Zettel unter Tastatur), so stellt er ein Risiko dar.

- *Konfigurationsfehler*

Meist Fehler, die durch mangelndes Wissen oder Unachtsamkeit des Administrators entstehen (z.B. Zugriffsrechte oder offene Test-Ports).

- *Designfehler*

Es liegen Schwächen im Protokoll oder Dienst vor, die aufgrund fehlender Sicherheitsmechanismen ausgenützt werden können. Bekannte Vertreter hierfür wären das IP-Spoofing und DNS-Spoofing. Zur Unterstützung für die folgenden Angriffsbeispiele im Abschnitt 3.3 sei hier die Einordnung der wichtigsten TCP/IP-Protokolle im OSI-Referenzmodell dargestellt [vgl. Müller 03].

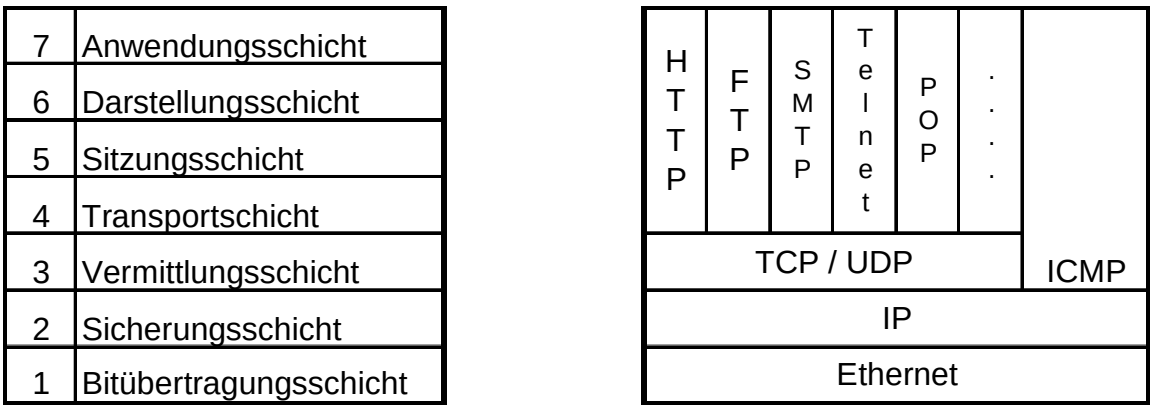


Abbildung 3-1: Vergleich des OSI-Referenzmodells mit dem TCP/IP-Stack

- *Implementierungsfehler*

Eine fehlerhafte Implementierung des TCP/IP-Stacks ins Betriebssystem wäre ein Beispiel dieser Gruppe.

3.2 Auswirkungen

Die Auswirkungen von erfolgreichen Angriffen gegen die eigene Umgebung können weit reichende Folgen haben.

- *Verlust von Vertraulichkeit der Daten*

Dem Eindringling ist es möglich Informationen mitzulesen und zu seinem Vorteil zu nutzen (Bankinformationen, Betriebsgeheimnisse, Entwicklungen).

Frage: Was passiert, wenn die Konkurrenz interne Daten kennt?

- *Verlust von Datenintegrität*

Die Echtheit der Daten ist nicht mehr gewährleistet. Womöglich wurden Pakete im Netzwerk oder Daten auf Servern bewusst zum Schaden der Firma verfälscht.

Frage: Was geschieht, wenn falsche Daten weiterverwendet werden?

- *Verlust von Verfügbarkeit der Daten*

Der Täter löscht bzw. zerstört wichtige Datenbestände.

Frage: Wie lange kann eine Firma ohne die Kunden-Datenbank existieren?

Das Mitbenutzen von Netzwerk- oder Rechner-Ressourcen aus dem internen Netzwerk zum Zwecke weiterer Aktivitäten (z.B. Distributed-Denial-of-Service) kann auch eine unangenehme Auswirkung sein.

Sämtliche Folgen können für ein Unternehmen einen enormen wirtschaftlichen Verlust darstellen oder sogar den finanziellen Untergang bedeuten. Man sollte sich dessen bei der Investition in neue Sicherheitsstrukturen bewusst sein.

3.3 Beispiele

3.3.1 Scans

Port-Scans selbst sind nicht unbedingt als Angriff zu werten, sind aber oft eine Vorbereitung dazu. Als Ergebnis erhält man eine Liste offener Ports auf einem Rechner. Diese Information ist schon sehr viel wert. Erstens gibt es den Rechner unter dieser IP-Adresse, und zweitens kann je nach offenen Ports ein bestimmter Angriff gestartet werden. Einige Produkte wie der LANGuard können sogar noch weitere Informationen (NetBIOS) erlangen.

- *Normaler Port Scan (Connect Scan)*

Entsprechend der nachfolgenden Abbildung wird eine TCP-Verbindung über einen 3-Way-Handshake aufgebaut. Erst nach einem erfolgreichen Aufbau (Connect) werden die eigentlichen Daten ausgetauscht. Das Transmission Control Protocol ist daher ein verbindungsorientiertes Protokoll.

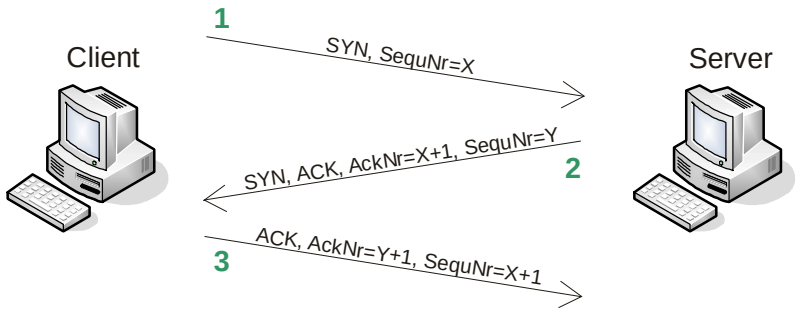


Abbildung 3-2: TCP-Verbindungsaufbau

Dabei werden die Flags im TCP-Header (SYN, ACK siehe Abbildung 3-3) zur Signalisierung benutzt. Die Sequenznumber (siehe Abbildung 3-2) ist für jedes Paket eindeutig. Im Gegensatz zu TCP ist das User Datagram Protocol (UDP) ein einfaches, verbindungsloses Protokoll mit wenig Overhead. Es benötigt daher keine Paketnummerierung.

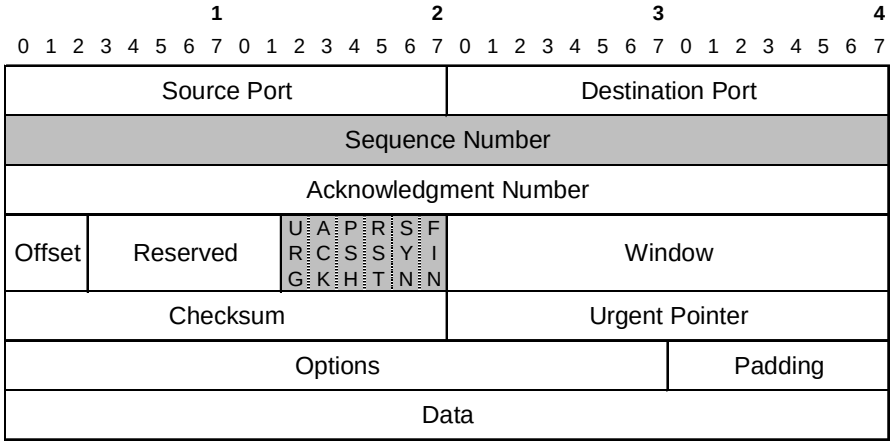


Abbildung 3-3: TCP-Header [vgl. www 01]

Ein Portscanner führt nun diesen TCP-Verbindungsaufbau für jeden Port in kurzer Zeit durch. Kann mit dem Zielsystem ein erfolgreicher Handshake auf einem bestimmten Port durchgeführt werden, so ist dieser offen. Offen bedeutet, dass der Server auf eine Verbindung wartet. Meistens sind nur die so genannten „Wellknown-Ports“ (0 bis 1024 siehe Tabelle 3-1) interessant, wo sich die typischen Dienste befinden, die reservierte Ports verwenden. Ein aktiver Portscan dieser Art wird allerdings auch leicht erkannt. Es wird eine vollständige Verbindung aufgebaut und somit auch vom Rechner protokolliert.

Port	Protokoll	Dienst	Beschreibung
20,21	tcp	FTP	Datentransfer
22	tcp	SSH	Verschlüsselter Remote-Zugriff
23	tcp	Telnet	Remote-Zugriff auf Konsole
25	tcp	SMTP	E-Mail Versand
53	tcp/udp	DNS	Löst Namen in IP-Adressen auf
69	udp	TFTP	Datentransfer bei Devices
79	udp	Finger	Liefert Informationen über Rechner
80	tcp	HTTP	Liefert Webseiten
110	tcp	POP3	Empfang von E-Mails
119	tcp	NNTP	Network News
137-139	tcp/udp	NetBIOS	Datei-/Druckerfreigaben
143	tcp	IMAP	Empfang von E-Mails
443	tcp	SSL	Sichere Verbindungen über Web
445	tcp	ADS	Active Directory Service
3306	tcp	MySQL	Datenbankkommunikation

Tabelle 3-1: Typische Wellknown-Ports [vgl. www 02]

Die weiteren Scan-Methoden versuchen über die Antwort des Zielhosts auf ungültige Pakete Informationen über aktive Dienste zu erhalten.

- *Half Open Scan (TCP SYN Scan)*

Besser, im Sinne des Angreifers, ist der Half Open Scan. Dabei wird nicht der vollständige Verbindungsaufbau vollzogen. Der letzte Schritt (ACK des Clients) bleibt aus. Nach dem 2. Schritt ist bereits bestätigt, ob ein offener Port besteht oder nicht. Unangenehm ist dieser Scan seitens des Servers, da eine unvollständige Verbindung meist nicht protokolliert wird. In weiterer Folge kann sich aus vielen Half Open Scans eine Denial-of-Service Attacke entwickeln (Kapitel 3.3.3). [vgl. Müller 03]

- *Stealth Scan*

Sendet der Client (Angreifer) ein Paket mit gesetztem FIN-Flag (Finish), und der Server antwortet mit einem RST-Flag (Reset) so ist ebenfalls ein aktiver Port vorhanden (FIN Scan). Ohne Zusatzsoftware kann dieser Scan nicht entdeckt werden. Allerdings funktioniert diese Methode auch nicht bei allen Systemen. Windows-Server reagieren darauf in der Regel nicht.

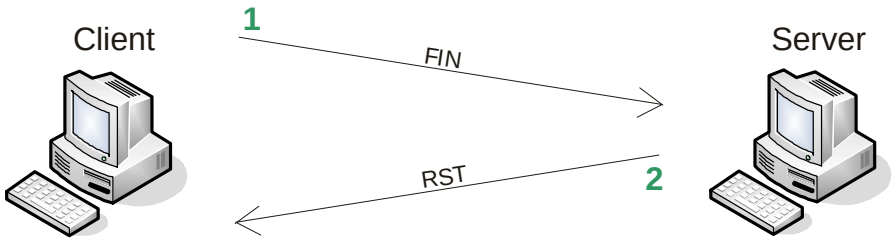


Abbildung 3-4: Stealth Scan

Es gibt verschiedene Arten von Stealth-Scans:

FIN Scan	sendet Pakete mit FIN Flag an jeweiligen Port
XMAS Scan	alle Flags sind gesetzt (alle Kerzen leuchten)
NULL Scan	Gegenstück zu XMAS Scan; kein Flag gesetzt
SYN/ACK Scan	verwendet SYN/ACK Pakete [vgl. www 04]

Eine Herausforderung für das ID-System stellen distributed Scans und zeitlich ausgedehnte Scans dar. Beim distributed Scan wird ein System von unterschiedlichen Rechnern gescannt. Ein Server der Dienste bereitstellt, verändert sich eher selten. Daher sind über längeren Zeitraum die offenen Ports

gleich. Es genügt zum Beispiel pro Tag nur einen Port abzufragen. Dies hätte den Vorteil, dass es dem ID-System vielleicht nicht abnormal vorkommt. Nach einigen Tagen erhält der Angreifer eine brauchbare Auswertung und bleibt eventuell unentdeckt.

3.3.2 Spoofing

Spoofing bedeutet, dass Pakete oder Frames von einem Angreifer gefälscht werden. Beispielsweise wird die Absenderadresse eines IP-Pakets auf einem für das Zielsystem bekannten, vertrauenswürdigen Rechner eingesetzt. Das anzugreifende System wird also absichtlich durch eine falsche Information getäuscht. Spoofing-Techniken gibt es praktisch für jedes Protokoll, das Adressen oder Bezeichner verwendet. Die häufigsten Spoofing-Techniken sind:

[Anonymous 03]

- *IP-Spoofing*

Die wohl am meisten verbreitete Technik. Der Angreifer weist sich gegenüber einem Client oder Server als ein „trusted Host“ aus, ist durch seine gefälschte Absender-IP-Adresse möglicherweise schon authentifiziert (einfache Access-Lists oder Paketfilter) und erhält bereits Zugriff auf Ressourcen.

- *MAC-Spoofing*

Viele Netzwerke mittlerer Größe werden oftmals durch statische MAC-Adress-Tabellen abgesichert (Medium-Access-Control). Somit darf sich nur ein Host mit gültiger MAC-Adresse im Netzwerk anmelden. Gerade bei den beliebten drahtlosen Netzwerken (Wireless LANs) kann eine gültige Adresse leicht mit Hilfe eines Sniffertools wie AeroPeek ermittelt werden. Moderne Betriebssysteme, wie Windows XP oder SuSE Linux 9.1, erlauben in den erweiterten Settings der Netzwerkkarte das Verändern der MAC-Adresse. In kürzester Zeit ist es dem Angreifer also möglich sich in ein Netz zu hängen, wenn es keine weiteren Sicherheitsmaßnahmen gibt.

- *DNS-Spoofing*

Das Domain-Name-System verknüpft Namen zu IP-Adressen. Gelingt es einem Angreifer diese Zuordnung auf einem DNS-Server zu manipulieren, so kann dadurch großer Schaden entstehen. Man stelle sich vor, es ruft jemand die Seite des ORF (www.orf.at) auf und erhält stattdessen eine Seite mit pornografischen Inhalten. Eine weitere Variante ist, dass der Angreifer einen DNS-Server betreibt der schneller auf Anfragen von Clients reagiert.

[vgl. www 12]

- *ARP-Spoofing*

In den meisten LANs wird die Ethernet-Technologie auf OSI (Open Systems Interconnection) Layer 2 und IP-Technik auf OSI Layer 3 verwendet. Für beide Layer existieren zwei Adressierungsschemata. Für die Punkt zu Punkt Verbindung auf Layer 2 verwendet man die eindeutigen MAC-Adressen. Diese bestehen aus 6 Byte, wobei 3 Byte den Hersteller identifizieren und der Rest eine fortlaufende Nummer ist. Auf Layer 3 kommt die virtuelle Adressenvergabe durch IP zum Tragen.

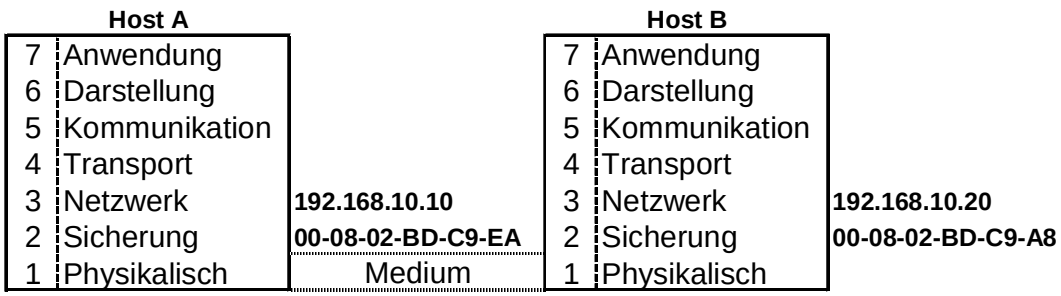


Abbildung 3-5: OSI-Schichtenmodell

Das Address-Resolution-Protocol liefert zu einer bekannten IP-Adresse die dazu gehörige MAC-Adresse. Das Ergebnis speichert jeder Rechner für sich in seinem ARP-Cache. Gelingt es dem Angreifer diesen Cache zu „vergiften“ (ARP-Poisoning), so stimmt diese Zuordnung nicht mehr, und der Angreifer erhält Pakete, die eigentlich an einen anderen Rechner gerichtet sind. Dazu muss er weder seine MAC- noch IP-Adresse ändern. Diese Technik ist sehr mächtig und lässt viele Varianten, wie Datenumleitungen, in geswitchten Netzwerken zu.

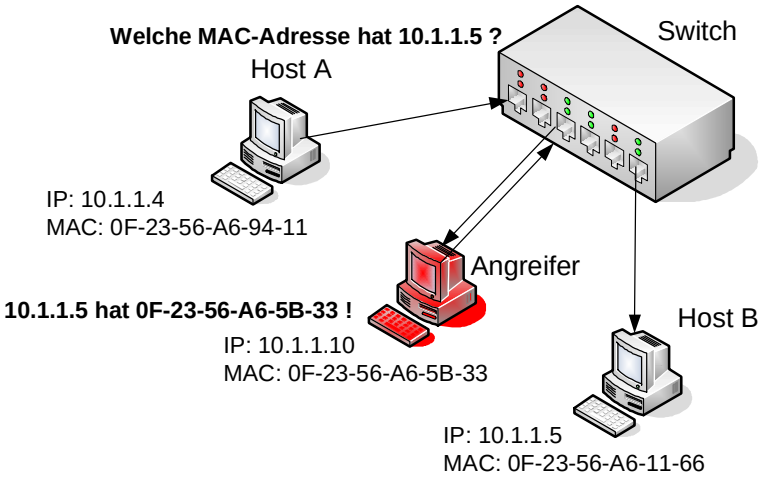


Abbildung 3-6: ARP-Spoofing

- *Mail-Spoofing*

Immer wieder treffen Mails von Microsoft oder einer Bank ein, bei der man gar kein Konto besitzt. Die Bank fordert zur Eingabe der Kontodaten und Passwort auf (social engineering, Phishing-Mails). Dies ist der klassische Fall von Mail-Spoofing. Hierbei wird mit geringstem Aufwand der Absendername im Mailheader verfälscht. Als die Protokolle für Mail (SMTP, IMAP, POP) entwickelt wurden, war man sich dieser Gefahren einfach noch nicht bewusst. Seitens der großen Konzerne gibt es aber intensive Bestrebungen für ein neues System (z.B. mit digitalen Unterschriften).

3.3.3 DoS Attacken

Diese bewirken das Lahmlegen eines Zielsystems. Man versucht die kompletten Ressourcen des Rechners oder Netzwerkes aufzubrauchen, damit sie anderen nicht mehr zur Verfügung stehen und das System daher nicht mehr erreichbar ist. Die Ausfallszeiten und Wiederherstellung können große finanzielle Verluste für ein Unternehmen bedeuten. Bekanntes Beispiel: im Februar 2004 kam die Website der Firma SCO zum Erliegen. Es wird nicht versucht einzudringen, sondern abzuschotten. Bei erfolgreichem DoS kann sich der Angreifer auch als der lahm gelegte Rechner ausgeben (Spoofing). DoS Attacken können auch als Ablenkungsmanöver für die Administratoren dienen. Groß angelegte Aktionen der Hackergemeinde gegen bestimmte Konzerne können nur mehr durch sinnvoll gesetzte Gegenmaßnahmen von Providern begrenzt werden.

- *ICMP-Flooding*

Die normale TCP/IP-Implementierung sieht vor, dass ein Computer auf ein ICMP echo request mit einem echo reply antwortet. Dies macht auch Sinn um die Erreichbarkeit von Systemen zu testen. Bekommt ein Server nun tausende ICMP requests (ping of death), so ist er ständig mit dem Antworten beschäftigt und kann sich keinen normalen Netzanfragen widmen. Die Absenderadresse ist meist noch gespoofed, wodurch der Angreifer entlastet ist.

- *SYN-Flooding*

Werden an einen Server mehrere tausend Pakete mit gesetztem SYN-Flag gesendet, hält der diese Verbindungen halb offen. Der dritte Schritt des TCP-Handshake, die Bestätigung vom Client (ACK-Bit gesetzt), bleibt aus. Für jede Verbindung reserviert der Server Arbeitsspeicher der nur begrenzt vorhanden ist.

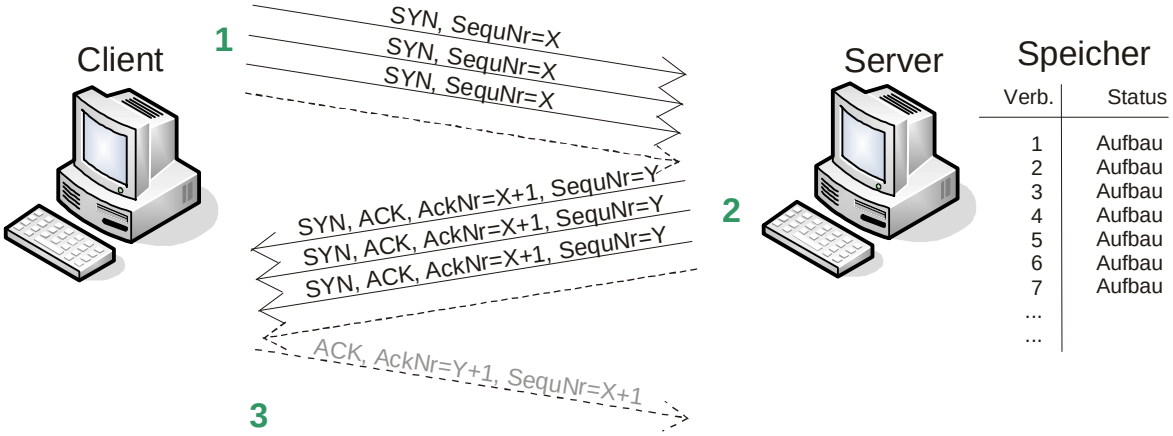


Abbildung 3-7: SYN-Flooding

In Folge dessen kann es zu einem Absturz (Buffer Overflow) kommen oder der Server kann keine Verbindungen von anderen Clients akzeptieren.

[vgl. Northcutt 04]

- *Smurf-Attacke*

Das ist eine beliebte und effektive Form einer DoS-Attacke. Dabei werden ICMP echo requests mit der IP-Adresse des zu schädigenden Systems als Quelladresse an eine Netzwerk-Broadcast-Adresse gesendet. In Folge dessen kommt es aufgrund von Betriebssystemdesign-Fehlern zu vielen Antworten.

Primitive DoS-Formen stellen einfach sehr viele HTTP-Anfragen an einen Webserver oder senden viele Mails an einen Mailserver, damit diese blockiert sind und reguläre Anfragen nicht bedienen können. Für eine Firma, die auf den Online-Handel angewiesen ist, bedeutet dies erhebliche Einbußen. In den letzten Jahren wurden vermehrt Erpressungsversuche im Zusammenhang mit DoS Attacken bekannt. Starten viele verteilte Rechner einen DoS Angriff zur selben Zeit, so bezeichnet man diese Form als Distributed-Denial-of-Service Attacke (DDoS).

3.3.4 Session-Hijacking

Eine bestehende IP-Verbindung wird durch einen Entführer übernommen. Dieser kann die Daten einsehen, umleiten, verfälschen oder einen ausführbaren Code einspielen. Normalerweise ist die Integrität einer TCP/IP-Verbindung durch die Initial Sequence Number (ISN) gegeben. Hat der Hacker schon einige Pakete mitgelesen, kann er die ISN vorhersagen. Die IP-Adresse des echten Clients wird gespoofed und der Client selbst wird durch SYN-Flooding abgeschottet. Der Hacker kann ohne Wissen des zweiten Kommunikationspartners eine bestehende Verbindung übernehmen. Möglich ist diese Attacke nur, weil TCP im Klartext sendet und der Hacker die ISN nach einigen Datenpaketen vorhersagen kann. Diese Angriffsmethode wurde durch den legendären Hacker Kevin Mitnick entdeckt und daher später als Mitnick-Angriff bezeichnet. Durch Authentifizierung und Verschlüsselung (IPSec) sind Verbindungen vor Session-Hijacking sicher.

3.3.5 Man-in-the-middle Attack

Dies bezeichnet allgemein jene Angriffsformen, wo der Angreifer zwischen beiden Kommunikationspartnern steht. Im Gegensatz zu Session-Hijacking wird der ursprüngliche Client nicht ausgeschlossen. Der Hacker bildet einen Knoten, der Verbindung und volle Kontrolle über die Verkehrsdaten hat. Mögliche Techniken sind ARP-Spoofing oder Ändern von Routingeinträgen.

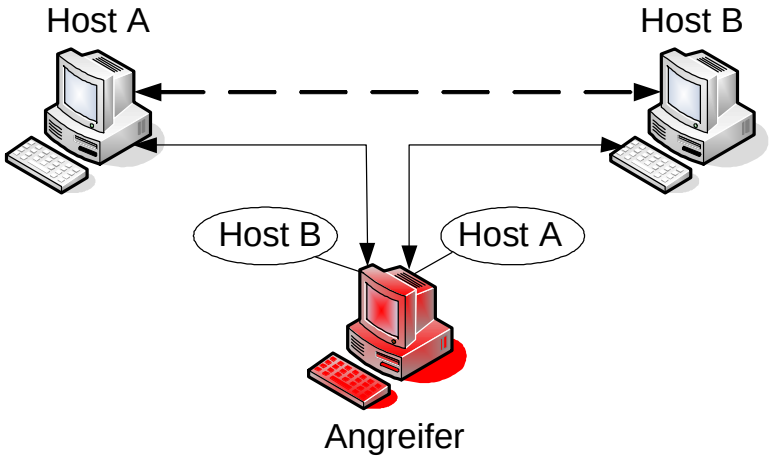


Abbildung 3-8: Man in the middle attack

3.3.6 SYN/FIN Angriff

Ein Paket mit gesetztem SYN-Flag bedeutet einen Verbindungsaufbau. Einige Hersteller von Firewallprodukten vergleichen nur dieses eine Bit, um den Verbindungsaufbau von außen zu sperren. Listige Angreifer entdeckten, dass ein Paket mit gesetztem SYN- und FIN-Flag die Firewall passieren kann. Beim Host wird dies aufgrund der Interpretation des Betriebssystems als Verbindungsaufbau akzeptiert.

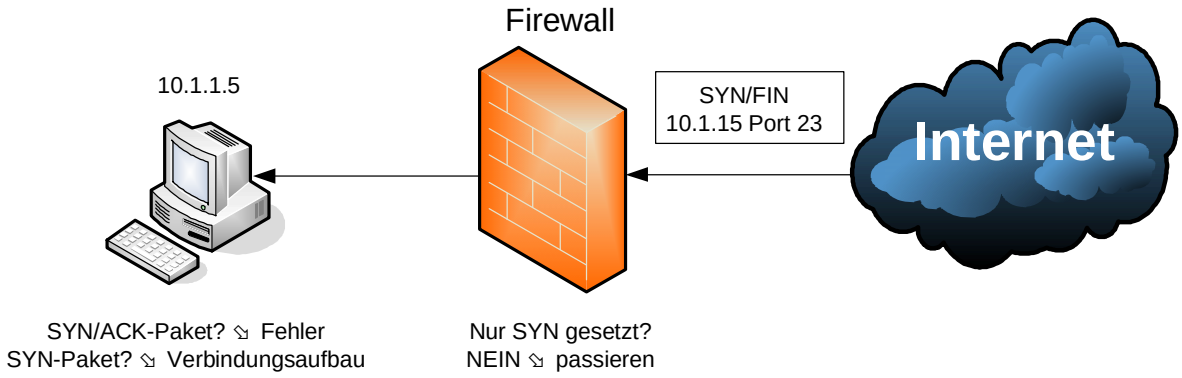


Abbildung 3-9: SYN/FIN Angriff

Jüngere Produkte weisen diese Schwäche in der Regel nicht mehr auf.

3.3.7 Ping of Death

Die normale TCP/IP-Implementierung sieht vor, dass ein Computer im Netzwerk auf einen 32 Byte langen ICMP echo request mit einem echo reply reagiert. Sendet man sehr große echo request Datenpakete andauernd (flooding), so kann

es beim Zielsystem zum Absturz oder einer Ausprägung von DoS kommen. Diese Schwäche ist mittlerweile von allen modernen Betriebssystemen behoben. Ältere Systeme z.B. mit Windows 95 konnten damit regelrecht „abgeschossen“ werden.

3.3.8 Buffer Overflow

Ein Schlagwort in der IT-Branche. Es handelt sich um einen Programmierfehler, wobei durch den Programmierer für gewisse Eingaben zu wenig Speicherplatz reserviert wird. Der zur Verfügung stehende Puffer (kleiner Speicherbereich) am Stack (Stapelspeicher) ist zu klein für die unerwartete Menge an Daten. Es kommt in Folge zu einem Überlauf, wodurch Speicherbereiche mit anderen gültigen Daten überschrieben werden. Im harmloseren Fall stürzt das Programm ab. Im schlechtesten Fall kann der Angreifer Root-Privilegien auf dem System erlangen.

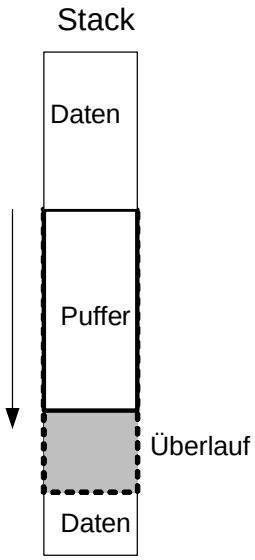


Abbildung 3-10: Buffer Overflow

4 Typen von ID-Systemen

Generell unterscheidet man zwischen hostbasierten ID-Systemen und netzwerkbasierten ID-Systemen. Während sich HIDS auf den jeweiligen Host mit seinen Logfiles und Datenaufkommen fokussieren, betrachten NIDS das Netzwerk mit seinem Datenverkehr.

4.1 Network IDS

Wie der Name schon vermuten lässt, überwacht diese Form von ID-Systemen den Verkehr in einer bestimmten Netzwerkumgebung. Ähnlich einem Sniffer-Tool wird jedes Paket mitgelesen. Der Sensor liefert Informationen über Verkehrslast und der Art plus Zustand (z.B. SYN-Flag) von Paketen. Damit ist es möglich, Portscans schnell zu erkennen, die oft als Vorbereitung für den tatsächlichen Angriff dienen. Bekannte Angriffsmuster können aufgrund von Schwächen in diversen Protokollen gut und sicher erkannt werden.

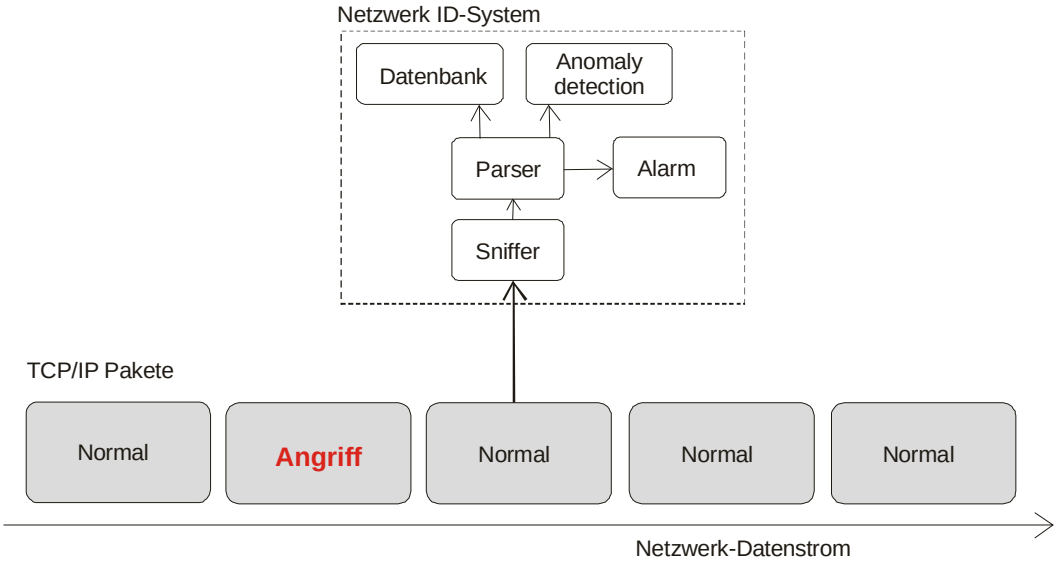


Abbildung 4-1: Funktionsweise eines NIDS [vgl. Spenneberg 03]

Abbildung 4-1 stellt die Funktion dar. Jedes Paket wird mitgelesen und auf verdächtige Merkmale im Header oder bezüglich Inhalt überprüft. Ein einfacher Paketanalyser erfüllt den Zweck nicht, dieser könnte durch Fragmentierung leicht umgangen werden. Viele Angriffe können nur dann detektiert werden, wenn der Kontext der gesamten Verbindung betrachtet wird. Ein großer Anteil der Angriffe

richtet sich gegen Applikationen. Zum Beispiel: wird die URL-Kodierung (Uniform Resource Locator) teilweise durch ASCII-Darstellung ersetzt. Kann das Netzwerk ID-System das Applikationsprotokoll nicht dekodieren, gelingt der Angriff.

- *Beispiele: Snort, ISS Real Secure, NetRanger, Dragon*

Typischerweise setzt man mehrere Netzsensoren in verschiedenen Segmenten, sowie vor und hinter einer Firewall ein, oder der Sensor verfügt über Anschlüsse für mehrere Netzwerksegmente (typische Enterprise-Appliance-Lösung).

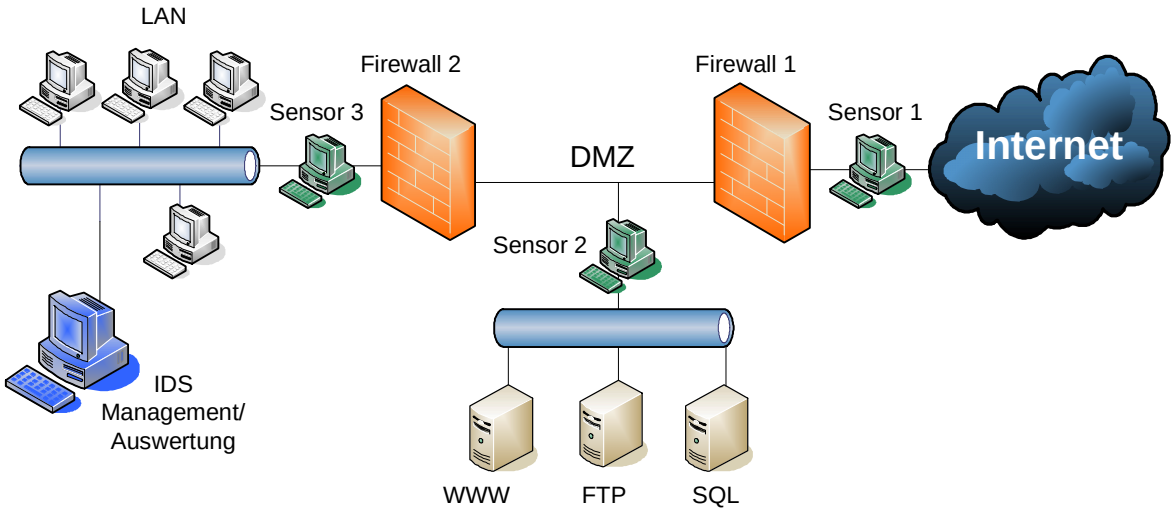


Abbildung 4-2: Anordnung eines Netzwerkbasierten ID-Systems

Missbräuche auf dedizierten Maschinen, wie häufige unerlaubte Dateizugriffe etc., können natürlich nicht detektiert werden. Auch verschlüsselte Datenströme (IPSec, SSL) bereiten dem NIDS bezüglich der Inhalte Probleme. Ein Einsatz netzbasierter ID-Systeme in Segmenten, wo verschlüsselt kommuniziert wird, kann trotzdem sinnvoll sein. Erste Angriffe oder Vorbereitungen dazu erfolgen meist unverschlüsselt.

Bei der Verwendung von mehreren Sensoren und einer zentralen Management-Station wird in der Fachliteratur oft auch von Distributed Intrusion Detection Systems (DIDS) gesprochen.

Performance

Die Performance des Netzwerkes muss zuvor kontrolliert werden. Der Sensor ist für den erwarteten Traffic, am Einsatzort, zu dimensionieren. Ist er nicht entsprechend ausgelegt, kann es zu einer Beeinträchtigung (Performance-problemen) im Netz kommen und Pakete gehen verloren. In einem moderneren

LAN (Local Area Network) wo man Switches benutzt, muss der Sensor bei einem Network Intrusion Detection System auf einem Monitor-Port angeschlossen sein, oder an einem Hub, der die Daten nur verteilt, sonst erhält der Sensor nur die Pakete, welche an ihn gerichtet sind. Auf diesem SPAN-Port (Switch Port Analyser) werden alle Pakete, die durch den Switch gelangen, kopiert. Allerdings muss genügend Bandbreite am so genannten Mirrorport vorhanden sein, denn 23 mal 100 Mbit/s bei einem 24-Port Switch können z.B. nicht an einem 100 Mbit/s Port kopiert werden! Viele Hersteller implementieren daher den SPAN-Port mit höherem Datendurchsatz (Gbit).

4.2 Host IDS

Dieser Typ eines ID-Systems läuft direkt auf einem bestimmten Host. Es werden die lokalen Ressourcen verwendet. Somit können nicht nur der Netzwerkverkehr des Hosts überwacht werden, sondern auch Dateizugriffe, CPU-Last oder Anmeldeversuche. Besonders wichtig ist die Überwachung der lokalen Logfiles (z.B. syslog) sowie die Kontrolle der Administrator-Aktivitäten. Angreifer versuchen gerne ihre Spuren in diversen Logfiles zu verwischen. Durch kryptografische Methoden und Prüfsummen wird das Dateisystem auf solche Änderungen überwacht. Jene Anwendung, die auf dem Rechner läuft, bezeichnet man auch als Agent. Bekanntestes Beispiel dafür ist das Programm „tripwire“. Einfache Port-Scans werden von HIDS oft nicht erkannt.

- *Beispiele: Tripwire, Swatch, Tiger, Security Manager*

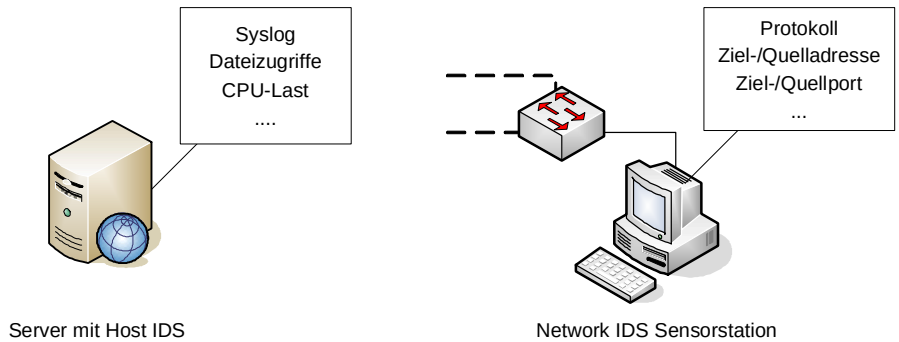


Abbildung 4-3: Eigenschaften HIDS und NIDS

Die

Tabelle 4-1 fasst die wesentlichen Merkmale beziehungsweise Unterschiede beider Systeme zusammen.

Network IDS	***Host IDS***
Überwacht Teile eines Netzwerks	Läuft auf zu überwachendem System
Überprüft Inhalte der Pakete (Snifferfunktion)	Benutzt Logfiles vom Betriebssystem
Ist unabhängig von anderen Systemen	Überwacht lokales Dateisystem, User, Prozesse
Erkennt viele protokollbasierte Angriffe	Erkennt Trojaner und Rootkits

Tabelle 4-1: Unterschiede NIDS und HIDS

Um ein hohes Maß an Sicherheit zu erlangen werden diese Formen oft in Kombination eingesetzt. Man spricht von einem Hybridsystem. Der Unterschied wird in Zukunft immer geringer, die Grenzen beider Varianten verschmelzen allmählich.

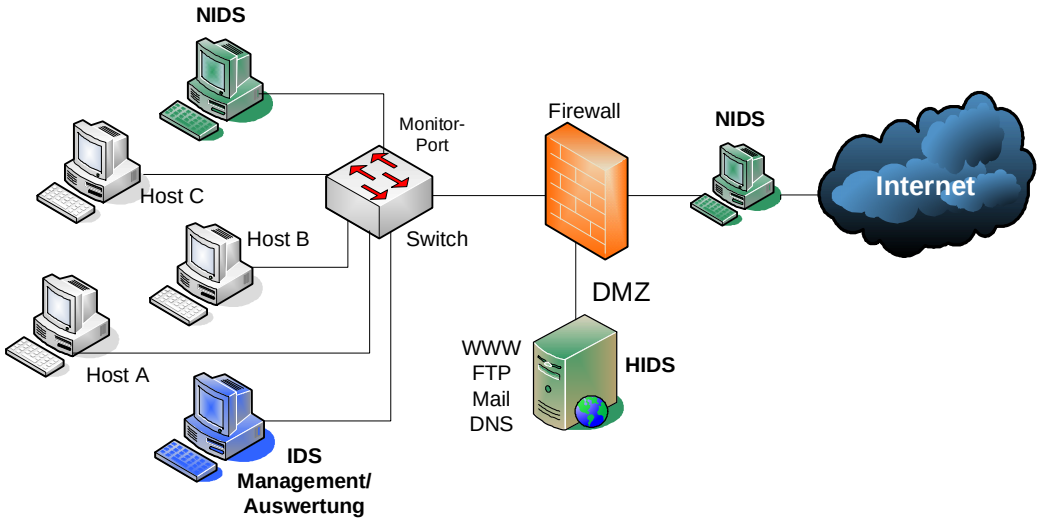


Abbildung 4-4: Host-/Netwerkbasierte IDS

Die geografische Aufteilung der Netz- oder Host-Sensoren spielt eine entscheidende Rolle und ist nicht trivial. Der Sensor soll ja möglichst viel und vernünftige Informationen aufnehmen. Ein Hostsensor wird meist an einem oder mehreren wichtigen Servern der IT-Infrastruktur platziert. Man sollte eine zusätzliche CPU-Last von 5% berücksichtigen. Netzsensoren setzt man vor und

hinter der Firewall und in wichtigen internen Netzsegmenten. Vor der Firewall erhält man einen guten Überblick über diverse Aktivitäten von außen, allerdings auch viele Fehlalarme, wobei das ID-System hinterher als eine gewisse Kontrollinstanz arbeiten kann und Attacken von innen erkennt. Eine allgemein gültige Regel für die Platzierung eines Network ID-Systems gibt es nicht. Je nach Interessen und Gegebenheiten können die Orte variieren.

Die zu überwachende IT-Umgebung und die Funktion des ID-Systems stehen in direkter Beziehung zueinander. Bei Änderungen im Gesamtsystem (z.B. neue Applikationen, Änderungen an der Netzwerkinfrastruktur) ist zu prüfen, welche Auswirkungen sich auf das ID-System ergeben.

4.3 Intrusion Prevention System (IPS)

Intrusion Prevention ist der nächste Schritt in der Evolution der Intrusion Detection. Ein Intrusion Prevention System (IP-System) soll einen Angriff nicht nur erkennen, sondern in Echtzeit abhalten. Das sind Strukturen die Vorgänge nicht nur erfassen und protokollieren, sondern aktiv versuchen, potentielle Intrusions zu verhindern und zu blockieren. Auf Basis der aufgenommenen Daten können schnelle automatische Reaktionen erfolgen. Dazu benötigt man eine gute Produktauswahl und viel Wissen des Administrators. Manch Intrusion Prevention System ist sogar in der Lage, Schädlinge wie den Wurm Blaster, aktiv zu dämmen. Der Wurm wird aus dem Datenverkehr gefiltert, gleichzeitig kann aber „zugelassener Traffic“, wie Active Directory, ungehindert passieren um den Betrieb aufrecht zu erhalten. Gefährliche Verbindungen oder Angriffe können also in Echtzeit blockiert werden. Man spricht auch von einem Inline Intrusion Detection System oder Intrusion Protection System.

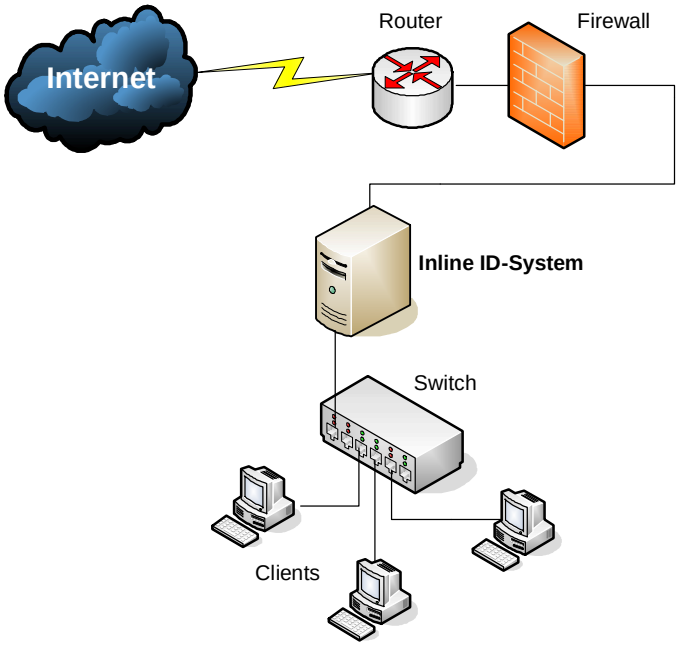


Abbildung 4-5: Inline Intrusion Detection System

Der gesamte Datenverkehr muss natürlich über das Inline ID-System verlaufen. Somit wird ein Schutz geboten, ohne den normalen Geschäftsablauf zu beeinträchtigen. Vorausgesetzt wird eine genaue Regelanpassung an die herrschende Systemlandschaft. IP-Systeme werden auch als IDP (Intrusion Detection and Prevention) bezeichnet. Viele Hersteller von Firewalls wie Checkpoint und Fortigate implementieren bereits IPS Funktionen in ihre Geräte.

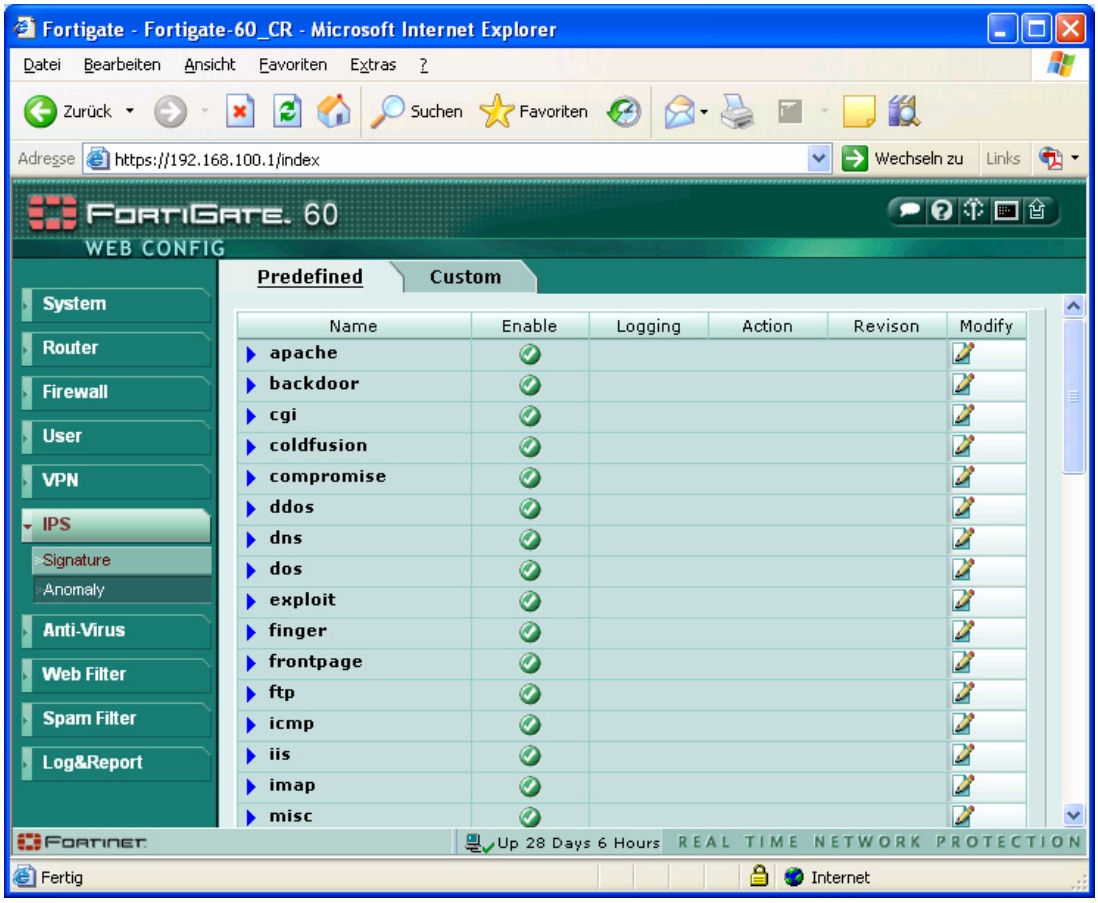


Abbildung 4-6: IPS Funktionalität in Firewalls

4.4 Intrusion Response System (IRS)

Diese Art von Sicherheitssystemen verfolgt die Ziele den Angreifer zu identifizieren und die lokale IT-Umgebung aktiv vor Schäden zu bewahren. Bei einem erkannten Angriff werden automatisch Gegenmaßnahmen, wie das Schließen von Verbindungen, Niederfahren von Rechnern oder Sperren von Zugriffsrechten eingeleitet. Man spricht in diesem Zusammenhang auch oft von Intrusion Detection and Response Systemen kurz IDR-Systemen. Der Einsatz von Intrusion Response Systemen ist sehr heikel und erfordert eine sehr gute Kalibrierung auf die herrschende Umgebung, sonst wird z.B. durch einen falschen Netzzugriff der Sekretärin womöglich gleich der Server abgeschaltet! Auch aus rechtlicher Sicht sind IR-Systeme als kritisch anzusehen, da erkannte Angriffe nicht immer echte Angriffe sind. Intrusion Response Systeme sind eine aggressivere Form von IP-Systemen, sie bieten weit mehr Funktionen als nur das Blockieren von bestimmten Datenverbindungen. Rekonfigurationen von Routern

oder das Sperren von Usern im Active Directory sind mögliche Folgen von IR-Systemen.

4.5 Application IDS

Application Intrusion Detection Systems sollen den gefährlichen oder fehlerhaften Einsatz von Software erkennen. Ereignisse der Applikationsebene werden überwacht. Ein AID-System gehört zur Gruppe von HIDS mit dem Unterschied, dass es sich nur um einen kleinen Teil des Rechners (eine bestimmte Applikation wie HTTP) kümmert und versucht, deren Bedeutung zu verstehen. Der Aufwand ist nur für sicherheitskritische Server zu rechtfertigen.

4.6 Honeypots

Der übersetzte „Honigtopf“ soll den Angreifer gezielt anlocken, ähnlich wie der Honig den Bär. Es handelt sich also um ein System, welches offensichtlich große Sicherheitslücken aufweist und wie ein Köder für Angreifer wirkt. Der Eindringling stürzt sich auf die leichte Beute und zeigt dem Sicherheitsadministrator unbewusst seine Methoden und kann vielleicht identifiziert werden. Honeypots können auch von kritischen Bereichen gezielt ablenken [vgl. Spenneberg 03].

Auf dem System kann im Prinzip nichts relevantes zerstört werden, trotzdem sollte man den Honeypot in einem isolierten Bereich, wie der demilitarisierten Zone (DMZ) platzieren. Honeypots können natürlich auch im LAN sitzen, um interne Angreifer aufzudecken. Eine ordentliche Überwachung des Honeypots ist Voraussetzung. Traurig ist, dass viele Administratoren ihre Server aufgrund fehlender Kenntnisse versehentlich als Honeypots betreiben (keine Security-Patches, viele offene Ports, Standardinstallation).

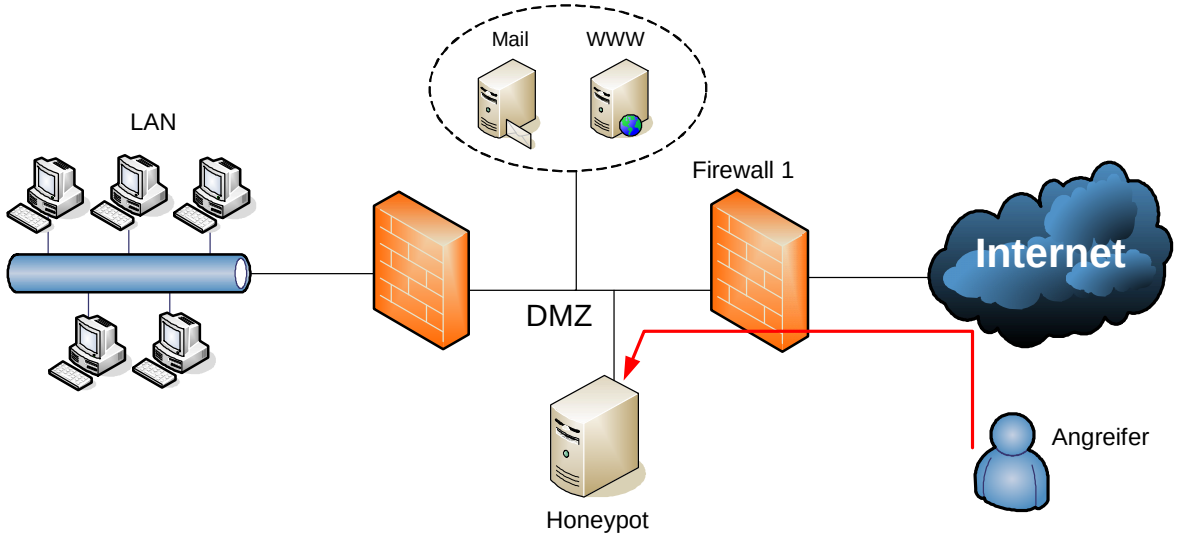


Abbildung 4-7: Honeypot

Die Anwendung von Honeypots ist umstritten. Denn im österreichischen und deutschen Gesetz ist ja die Beihilfe zu einer Straftat ebenfalls gesetzeswidrig. Wenn vorsätzlich die notwendigen Patches nicht eingespielt werden, sodass ein Eindringen möglich wäre, besteht sogar in Extremfällen die Möglichkeit dies als Beihilfe zum Einbruch auszulegen.

5 IDS Lösungen am Markt

Die Anschaffung eines ID-Systems für die Sicherheitsinfrastruktur stellt meist eine große Investition dar. Je nach Einsatzgebiet und Sicherheitszielen sind unterschiedliche Kriterien mehr oder weniger wichtig. Bereits bei der Planung und Evaluierung der Lösung sollte dies beachtet werden um einen maximalen Erfolg zu erzielen.

5.1 Anforderungen

„Das Intrusion Detection System soll jeden Angriff erkennen und bei der Verteidigung der Systeme helfen“. Das wäre eine einfach formulierte Anforderung. Für größere Projekte ist es jedoch hilfreich die relevanten Auswahlkriterien etwas detaillierter zu formulieren. Typische Anforderungen für ein ID-System sind:

- *leichte Implementierung*
- *laufender Betrieb ohne wesentliche Administration*
- *Fehlertoleranz*
- *Integrität und Verfügbarkeit der bereitgestellten Daten*
- *minimaler Overhead im Netzwerk durch das System selbst*
- *eigene Intelligenz*
- *wenig Fehlalarme*
- *keine Beeinträchtigung der bestehenden Applikationen*
- *gezielte Reaktion auf Angriffe*
- *Skalierbarkeit der Lösung*
- *Anonymisierung der personenbezogenen Daten*

5.2 Kommerzielle Produkte

Für einen aussagekräftigen Vergleich von ID-Systemen müsste man vernünftige Kriterien und Parameter aufstellen und diese dann in mehreren Teststellungen verifizieren. Aus Zeit- und Kostengründen war das im Rahmen dieser Diplomarbeit nicht möglich. Die folgenden Zeilen sollen nur einen kleinen theoretischen Einblick in kommerzielle Produktlösungen, und vor allem deren Kosten geben. Es soll an dieser Stelle keinesfalls eine qualitative Aussage über die Systeme getroffen werden. Das perfekte ID-System muss erst entwickelt werden. Sowohl freie, wie kommerzielle Systeme haben ihre Stärken und Schwächen. Nachstehend wurden einige Lösungen namhafter Hersteller betrachtet. Sämtliche Preisangaben und Produktdaten (Stand: Mai 2005) dienen nur für einen groben Überblick, und sind keinesfalls exakte Angaben.

5.2.1 ISS Proventia Intrusion Detection

Wurde von ISS (Internet Security Systems), dem Marktführer auf diesem Segment (laut Infonetics Research, Studie 2002), entwickelt. ISS bietet ein komplettes Sicherheitskonzept (Antivirus, Mail Security, Intrusion Prevention, VPN, etc.) namens Proventia ESP (Enterprise Security Plattform). Intrusion Prevention ist also nur ein Teil des ganzen Sicherheitsmodells [siehe Abbildung 5-1], was die Erfahrung und Kompetenz dieses Herstellers beweist.

Abbildung 5-1: Internet Security Systems ESP-Konzept [www 06]

Proventia bietet ein netzbasiertes ID-System nach dem Client/Server-Prinzip, das an verschiedenen Positionen im Netzwerk verankert wird. Die Appliance-Sensoren werden mit Proventia (A,G,M-Serie) bei ISS bezeichnet und können als Intrusion Detection oder Prevention System (inline) fungieren.

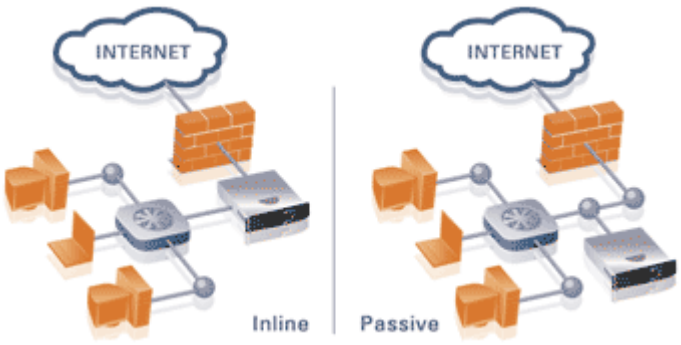


Abbildung 5-2: Unterschied IPS- (Inline-Modus) oder IDS-Funktion [www 14]

Im so genannten Inline-Modus fließt der gesamte Datenverkehr direkt durch das Intrusion Prevention System. Somit kann unerwünschter oder gefährlicher Verkehr aktiv gefiltert oder blockiert werden. Im passiven Modus verhält sich das Intrusion Detection System wie ein stiller Beobachter und meldet nur Ereignisse ohne direkte Eingriffe in den Datenstrom.

Zusätzlich zu den Hardware-Netzsensoren ist auch eine hostbasierte Software (hostbasiertes ID-System) namens Realsecure Server für verschiedene Serverplattformen erhältlich. Alle Sensoren (NIDS- und HIDS-Sensoren) übermitteln ihre Daten an den SiteProtector, die Management-Zentrale [siehe Abbildung 5-3]. Jeder Sensor ist durch den SiteProtector zentral verwaltbar (grafische Auswertung, Konfiguration, etc.). Die Kosten für die Sensoren richten sich nach dem möglichen Datendurchsatz (200 Mbps bis 1200 Mbps).

[www 06]

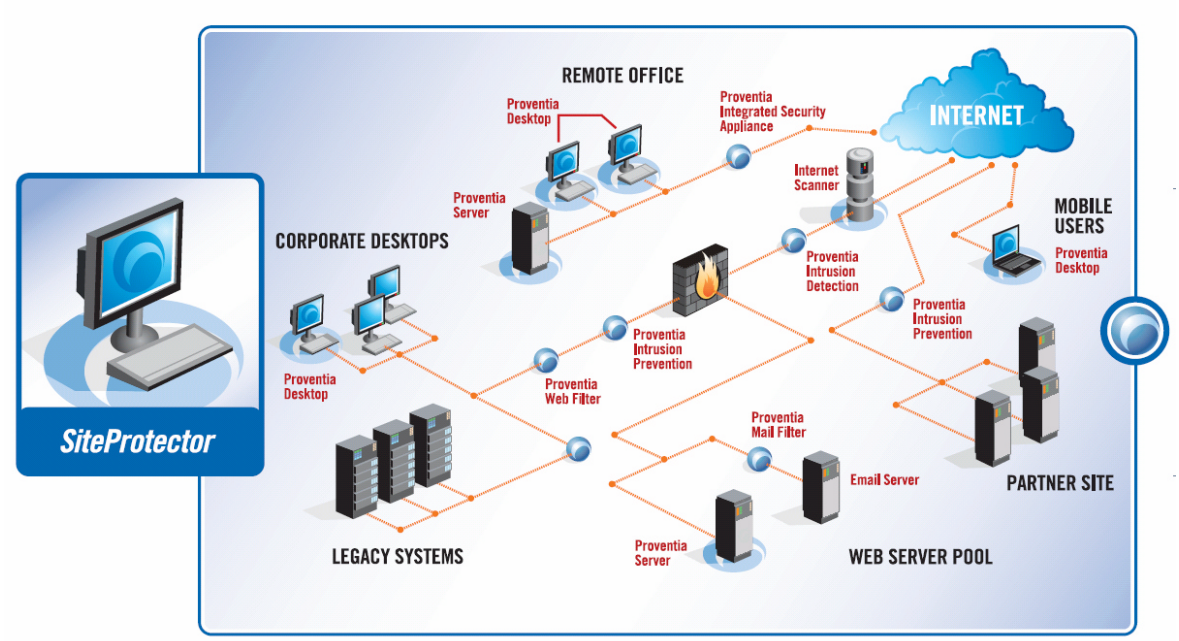


Abbildung 5-3: ISS Einsatzszenario Proventia [www 06]

5.2.2 Cisco Secure IPS 4200 Series

Die Sensoren dieser Serie sind die Schlüsselkomponenten des „Cisco Self Defending Network“ – Konzepts. Ziel ist es, Attacken, Würmer und Viren zu stoppen, bevor sie Daten beschädigen. Durch mehrere Interfaces kann ein Hardware-Sensor mehrere Subnetze (bis zu 8) gleichzeitig schützen. Inline- und Promiscuous-Modus sind mit den Geräten bis zu einer Geschwindigkeit von 1 Gbit/s möglich. Eine web-basierende Managementlösung ist bereits integriert.

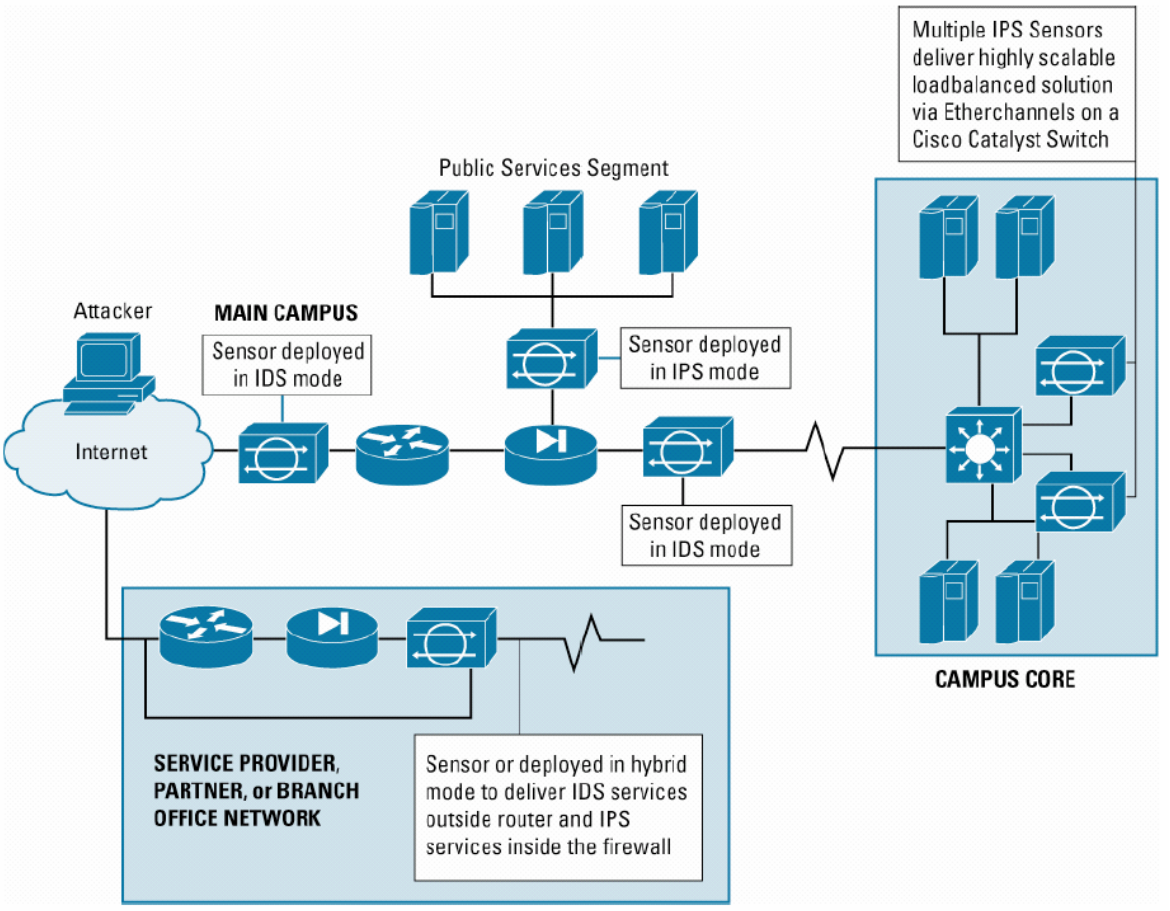


Abbildung 5-4: Cisco Einsatzszenario für IPS 4200 Sensoren

Cisco empfiehlt bei einer Platzierung der Sensoren vor der Firewall den Promiscuous-Modus. Der Sensor arbeitet also nur passiv und protokolliert Verletzungen, greift aber nicht aktiv in Verbindungen ein. Erst im Inneren (DMZ und Client-Bereich [siehe Abbildung 5-4]) arbeitet der Sensor inline als Intrusion Prevention System und kann somit Angriffe aktiv verhindern. Cisco Switches, Router und Firewalls können von dem Intrusion System als Reaktion auf Gefahren direkt modifiziert werden. Würmer und Viren können aus den laufenden Datenströmen absorbiert werden, ohne den normalen Datenverkehr zu beeinflussen.

Cisco bietet weiters für viele aktive Netzwerk-Komponenten IPS- beziehungsweise IDS-Funktionalität durch ein IOS[3]-Upgrade an (z.B. Switch Catalyst 6500).

[3] Internetwork Operating System; Ciso-eigenes Betriebssystem für Router, Switches, etc.

5.2.3 Enterasys Dragon IDS

Die Dragon Serie von Enterasys bietet Network Intrusion Defense in Soft- oder Hardware-Form (Appliance) an. Eingesetzt werden Protokoll-Analysen, Signaturvergleiche und Anomalie-Detection. Bei Angriffserkennung können Verbindungen beendet werden oder andere aktive Komponenten rekonfiguriert werden. Die mitgelieferten Signaturen liegen in XML (Extensible Markup Language) vor und können selbst adaptiert werden. Eine Besonderheit ist die Möglichkeit, einen physikalischen Sensor auf mehrere virtuelle Sensoren aufzuteilen. Jeder virtuelle Sensor kann in einem eigenen LAN oder VLAN (Virtuell Local Area Network) mit eigenen Regelsätzen arbeiten. Ein Dragon Network Sensor wird durch den Dragon Enterprise Management Server verwaltet (Konfiguration, Updates, Reports)

[www 08]

5.2.4 McAfee IntruShield Network IPS Sensor, Entercept

McAfee hat gleich zwei Produkte in seinem Portfolio: IntruShield, eine hardwarebasierte NIDS-Lösung, und Entercept, ein HIDS-Softwareprodukt.

McAfee IntruShield IPS bietet Echtzeit-Schutz gegen bekannte und unbekannte Angriffe (Zero-Day[4]), verschlüsselte Attacken (SSL), Spyware und Denial-of-Service. Die Hardware-Lösungen decken Übertragungsraten von einigen Mbit/s (Randbereiche des Netzwerks) bis zu Multi-Gbit/s (Core-Bereich) ab. Auch McAfee verwendet eine Hybridform mit Signaturerkennung, Anomlaie-Detection und Protokollanalysen. ISM (IntruShield Security Management) nennt sich die dazu gehörige Enterprise-Management-Lösung. Immer wieder wird die Erkennung von unverschlüsselten und verschlüsselten Angriffen durch den Hersteller betont. Für die Erkennung von Angriffen, die über das SSL (Secure Socket Layer) Protokoll stattfinden, muss der Private-Key auf dem Sensor gespeichert werden. Die Sensoren können ebenfalls inline als IPS oder nur als Beobachter (IDS) arbeiten. Die Updates werden automatisch per Internetverbindung vom Managementsystem eingespielt. Ein Sensor-Reboot ist nicht notwendig. Mit Hilfe

[4] Angriffe auf neue, bisher unbekannte Sicherheitslücken.

der grafischen Oberfläche am ISM können auch eigene Regeln kreiert werden. Die IntruShield-Geräte können auch als interne Firewalls zum Schutz von Serverfarmen eingesetzt werden. Ein einzelner Sensor kann in bis zu 1000 virtuelle Sensoren mit jeweils eigenen Policies segmentiert werden.

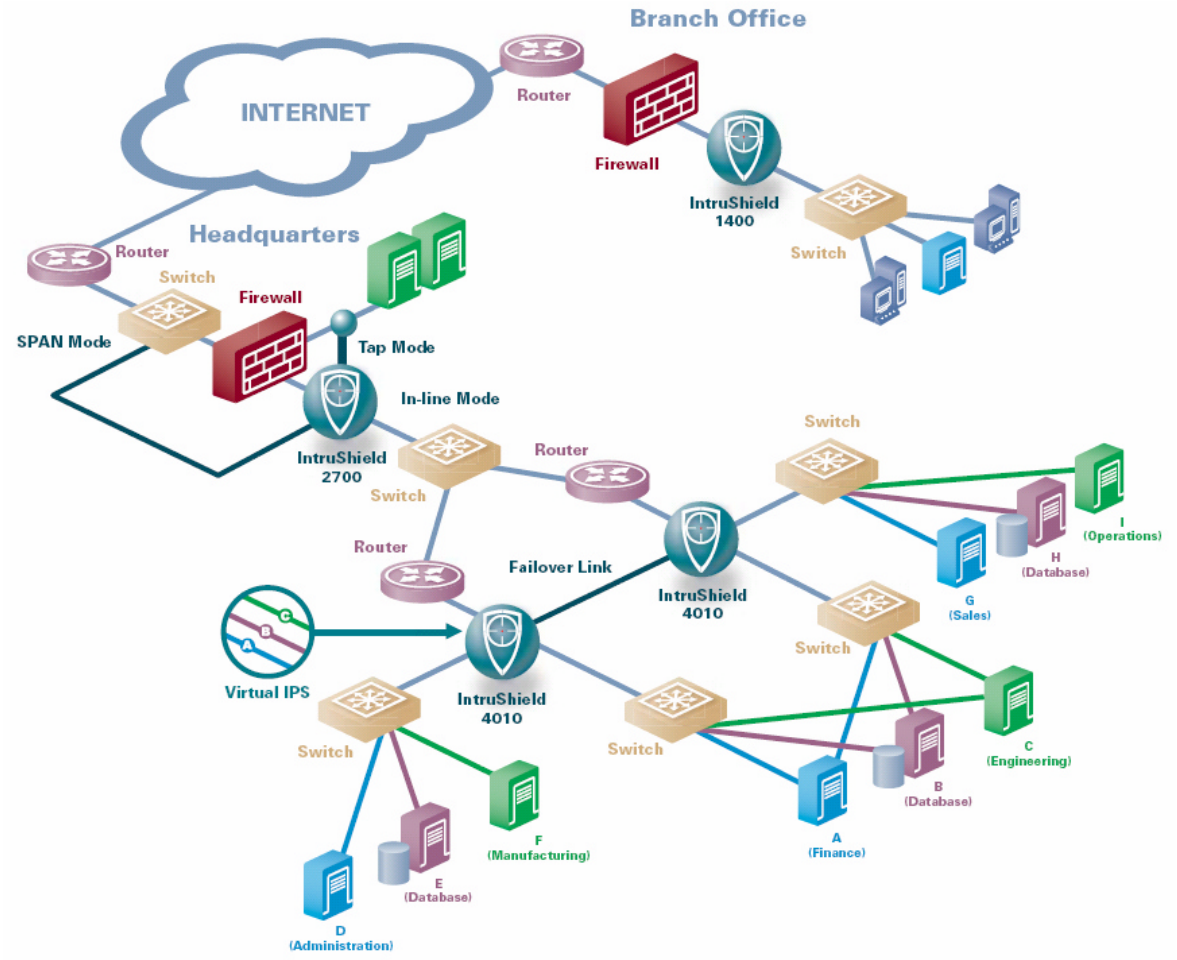


Abbildung 5-5: McAfee Einsatzszenario für IntruShield [www 13]

Abbildung 5-5 zeigt die Einsatzmöglichkeiten der IntruShield Sensoren. In den Randbereichen des Netzwerks (kleine Niederlassungen bis zu 20 Mitarbeiter) werden die kleinen IntruShield 1400 als IPS eingesetzt, in der Zentrale natürlich die stärkeren Appliances (IntruShield 2700) im Edge-Bereich bis hin zu den leistungsfähigsten IntruShield 4010 redundant im Core-Bereich. Ohne zusätzliche Hardware bietet McAfee zwischen 2 Sensoren „stateful failover“. Fällt einer der beiden aus, kann der zweite sofort im laufenden Betrieb (hot standby) übernehmen.

[www 13]

5.2.5 CA E-Trust Intrusion Detection

Basiert auf Windows-Servern und ist daher eine reine Software-Lösung. Die Installation und Konfiguration gestaltet sich laut Hersteller besonders einfach. Ein gelungenes Feature ist die Aufzeichnung von bestimmten Sessions, wie HTTP oder TELNET, und die gezielte Sperrung von unerwünschten URLs. In Unternehmen mit sehr strengen Sicherheitsbestimmungen kann sogar ein bestimmter Datenverkehr gezielt blockiert werden. Eine Antivirus-Engine und automatische Update-Funktionen sind ebenfalls integriert. Der Preis von ca. 2.100 € für 125 Sessions ist - verglichen mit anderen Herstellern - sehr günstig.

[Tett 04]

5.2.6 Juniper Networks - Netscreen IDP 100

Die Juniper Netscreen Intrusion Detection und Prevention wird inline ins Netzwerk gesteckt um Schadensfälle durch Attacken im Vorfeld zu vermeiden. Es ist eine reine Hardware-Lösung (Appliance), die Zeit und Kosten im Zusammenhang mit Intrusions (Verletzungen) minimieren soll. Das Policy Management, Log Viewing und Incident Management erfolgt ebenfalls zentral. Der Management Server läuft nur auf Red Hat Linux oder Sun Solaris Plattformen. Die grafische Client-Software arbeitet java-basierend von verschiedensten Systemen aus. Signatur-Updates sind wöchentlich verfügbar und werden an die Sensoren verteilt.

[www 15]

5.2.7 Sourcefire 3D System

Mit den Schlagwörtern „Discover. Determine. Defend.“ vertreibt Sourcefire sein Sicherheitskonzept [siehe Abbildung 5-6].

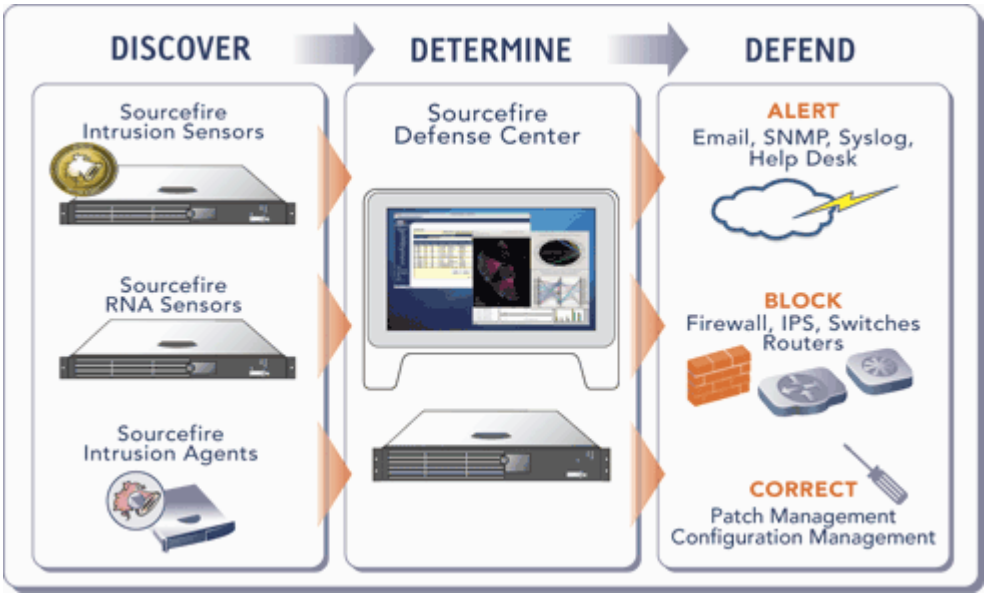


Abbildung 5-6: Sourcefire 3D System [www 14]

RNA (Real-Time Network Awareness) Sensoren sind nicht die Sensoren für die Intrusion Detection Funktionalität, sondern liefern Vulnerability-Informationen zu Rechnern (Betriebssystem, IP-Adresse, Spyware), und können als Appliance in Netzwerk-Segmente integriert werden. RNA erkennt sofort, wenn ein Server zum Beispiel einen neuen Dienst startet, beziehungsweise Ports öffnet. Das RNA-System arbeitet aber im Netzwerk und ist nicht mit einem HIDS zu verwechseln!

Die Sourcefire Intrusion Sensoren sind die Agenten des IP-Systems und basieren auf der Open Source Snort-Entwicklung von Marty Roesch. Dieser ist auch der Gründer von Sourcefire. Snort selbst wird auch von Sourcefire weiterentwickelt und bleibt Open Source. Die Sensoren können inline aktiv als Intrusion Prevention System arbeiten, oder passiv als Intrusion Detection System. Die Hardware-Sensoren kombinieren Signatur-, Protokoll- und Anomalie-Erkennung für eine maximale Erkennungsrate. Benutzer können selbst neue Regeln erstellen, bzw. bestehende Regeln modifizieren. Die Steuerung und Verwaltung erfolgt zentral über das Sourcefire Defense Center, was gleichzeitig den Mittelpunkt des 3D Systems bildet. Am Defense Center erfolgt das Event Monitoring und die

Darstellung diverser Analysen. Die Konsole (Defense Monitor) kann in bestehende Management-Software, wie HP OpenView, integriert werden.

5.2.8 Zusammenfassung Appliance-Lösungen

Produkte	ISS Proventia A	McAfee IntruShield IPS	Cisco IPS 4200	Enterasys Dragon	Sourcefire Inrusion Sensor	Juniper Netscreen
Geschwindigkeiten	200 - 1.200 Mbps	100 - 2.000 Mbps	80 - 1.000 Mbps	20 - 1.000 Mbps	5 - 8.000 Mbps	20 - 1.000 Mbps
Appliance-Bauform	✓	✓	✓	✓	✓	✓
schützbare Segmente	max. 12	2 bis 12	max. 8	max. 6	max. 4	max. 4
Stealth-Mode	✓	✓	?	?	✓	✓
Protokollanalyse	> 100	> 100	✓	✓	✓	60
Signaturen	> 1.700	> 3.000	✓	✓	✓	> 500
Anomalie-Detection	✓	✓	?	✓	✓	✓
Management	web-based + zentral (Site-Protector)	zentral (IntruShield-Manager)	direkt (web-based) + zentral	zentral (Dragon Server)	zentral (Defense Center)	zentral (Management Server)
virtuelle Sensoren	x	max. 1.000	x	✓	x	x
Erkennt Verschlüsselte Angriffe (SSL)	x	✓	x	x	x	x
IPS-Mode	✓	✓	✓	✓	✓	✓
IDS-Mode	?	✓	✓	?	✓	✓
Signatur-Update	~ 20 Updates/Jahr	sofort durch Pull Technik	?	✓	sofort	wöchentlich
Eigene Signaturerstellung	snort-based rule support	durch graf. Oberfläche	?	✓	✓	✓
DoS Protection	✓	✓	✓	✓	✓	✓
Failover	✓	✓	✓	?	✓	✓
Key Features	SNMP Traps; Drop Packets/ Connections; Blocks traffic;	Stop Connections; Drop Packets, Sessions; Reconfigure FW; TCP Reset; ICMP unreachable; Logging;	ACL Modification (Router, FW, SW); Session Termination; Drop Packets, Sessions; Loggin and Replay;	Terminate Sessions; Reconfigure Routers, FW, SW; TCP Reset; ICMP unreachable;	Drop traffic; TCP reset; ICMP unreachable; Terminate sessions; Reconfigure Router, FW, SW;	Drop Packets/ Connections; Close Connection; TCP Reset; E-Mail; SNMP-Traps; Syslog;
Preis	Sensor ~10.300 €	Sensor ~12.500 €	IDS 4215 ~6.000 €	Sensor ~6.000 €	Sensor ~6.400 €	IDP 10 ~9.400 €

Tabelle 5-1: Tabellarische Übersicht

Es ist sehr schwierig Kriterien für einen direkten Vergleich heranzuziehen. Die Tabelle 5-1 bietet nur einen groben Überblick der am Markt erhältlichen, bekannteren Systeme. Die Preise sind nur Richtwerte, da die effektiven Kosten je nach dem maximalen Datendurchsatz, Anzahl der Interfaces und CPUs, Serviceverträge etc. stark variieren. Die angeführten Preise beziehen sich immer auf das kleinste Produkt einer Serie, mit den kleinsten Leistungswerten (z.B. Cisco

IDS 4215 unterstützt nur 80 Mbit/s). Diese kommen meist am Rand von Enterprise-Netzwerken (oder Filialen) zum Einsatz.

Alle Hersteller verfolgen eine ähnliche Strategie. Intrusion Detection und Prevention ist nur ein Teil des gesamten Sicherheitskonzepts. Alle Security-Komponenten (z.B. Antivirus-Appliances, IDS-Sensoren, etc.) werden von einem zentralen Management-Server konfiguriert, verwaltet und grafisch betrachtet. Nur die Geräte von Cisco und ISS bieten überhaupt direkten grafischen Zugriff ohne zentrales Management. Im Bereich dieser Systeme erkennt man, dass Cisco kein reiner Security-Spezialist ist, und seine Wurzeln bei Router- und Switch-Technologien hat. Man bekommt den Eindruck, das Cisco hier auf den Zug (Security-Trend) schnell mitaufspringen wollte. Die Lösungen hinken denen der führenden Hersteller in diesem Segment (ISS, McAfee) eindeutig nach. Auch das Marketing ist hier mit dürftigen Datenblättern etwas schlecht. Ausgezeichnete Produktdatenblätter mit viel detaillierten Informationen liefern McAffee und ISS, die führenden Anbieter. McAfee bietet sehr umfangreiche Funktionen und als einziger Hersteller Schutz gegen verschlüsselte Angriffe über SSL. Dazu muss der Private-Key ebenfalls am Sensor abgelegt werden. Ein besonderes Feature ist das Virtualisierungskonzept, das McAfee und Enterasys bei ihren Produkten bieten. Dabei kann ein physikalischer Sensor in viele virtuelle Sensoren mit eigenen Konfigurationen unterteilt werden. Jeder virtuelle Sensor kann einem eigenen VLAN, Interface oder IP-Adresskreis zugeordnet werden.

Alle angeführten Systeme wenden eine Hybridtechnik mit Signaturerkennung, Anomalieerkennung und Protokollanalyse an. McAfee punktet in diesen Bereichen mit einer großen Anzahl von Signaturen (> 3.000) und ca. hundert unterstützen Protokollen. In den Datenblättern von Cisco findet man zu den Erkennungsmethoden keinerlei relevante Informationen, deshalb die Fragezeichen in der Übersichtstabelle.

Punkto Datendurchsatz (bis zu 8 Gbit/s) und Aktualisierung bildet Sourcefire, die kommerzielle Distribution von Snort, die Spitze. Durch ein eigenes Vulnerability Research Team (VRT) liefert Sourcefire umgehend Signaturen zu neuen Angriffen.

Ohne praktische Erfahrungen mit diesen Systemen ist es schwer, Empfehlungen abzugeben. McAfee ist aber aufgrund der Informationen aus den Datenblättern und den vielen Produkten in dieser Sparte sicher ein interessanter Partner in Sachen Enterprise-Security Lösungen, was auch seinen Preis hat. Für Klein- und Mittelbetriebe bietet die Juniper Netscreen IDP Serie attraktive Modelle mit gutem Preis- Leistungsverhältnis.

5.2.9 Fazit

Die obige Tabelle zeigt, dass kommerzielle ID-Systeme für Unternehmensnetzwerke einen sehr hohen Preis haben. In den Produkten steckt viel Entwicklungszeit und Erfahrung der Firmen, wodurch der Preis auch gerechtfertigt ist. Für jene, denen diese Systeme zu teuer sind, stehen immer noch Open Source Lösungen zur Auswahl. Fällt die Wahl auf ein Open Source Produkt, so sollte ausreichendes Fachwissen durch qualifizierte Mitarbeiter im Unternehmen vorhanden sein. Auffallend bei den Recherchen war die Tatsache, dass alle großen Hersteller bereits mehrere Produkte im Portfolio haben. Der Trend geht auch eindeutig in Richtung Intrusion-Prevention-Appliances mit zentralem Management. Zusätzlich bieten viele Hersteller hostbasierte Intrusion-Software für Serversysteme an, die ebenfalls ins zentrale Management integriert werden. Das zentrale Management kann eine Vielzahl von NIDS- und HIDS-Sensoren im Enterprise-Netzwerk verwalten und updaten.

5.3 Open Source Systeme

Im Bereich der Open Source Community wird immer wieder beziehungsweise noch an ID-Systemen (Logwatch, Logsurfer etc.) gebastelt. Wirklich hervorstechen konnten bis jetzt nur Snort von Martin Roesch [siehe Kapitel 7] als Netzwerk Intrusion Detection System und Tripwire als Host ID-System. Beide werden auch in kommerziellen Umgebungen von Sicherheitsexperten eingesetzt. Tripwire erkennt Veränderungen an zu schützenden Systemen durch Integritätstests (Prüfsummen). Es ist kommerziell für viele verschiedene Systemumgebungen erhältlich. Für Linux-Systeme ist auch eine Open Source Version verfügbar. [www 10]

Die Internet-Seite der Technischen Universität Cottbus bietet eine umfangreiche Auflistung nahezu aller ID-Systeme im Open Source und kommerziellen Umfeld:

http://www-rnks.informatik.tu-cottbus.de/en/security/ids.html

6 Rechtliche Aspekte

Wie schon erwähnt müssen ID-Systeme eine Vielzahl von Daten aufnehmen, um ihre Funktion zu erfüllen. In einem Mini-Heimarbeitsnetz mit 3 Computern ist der Einsatz unbedenklich. Im Wesentlichen werden ja nur die eigenen Daten protokolliert. In Unternehmen können die Daten den Personen direkt zugeordnet werden aufgrund von IP-Adressen, Rechnernamen oder Anmeldeversuchen. Genau aus diesem Grund gibt es auch gewisse rechtliche Bedenken in Richtung Datenschutz und Anonymität. Besonders bei einer internen Überwachung fallen personenbezogene Daten an, wodurch Verstöße gegen interne Unternehmensrichtlinien angezeigt werden könnten. Möglich wäre natürlich auch die Beobachtung von Verhaltensweisen der Mitarbeiter. Daher sind datenschutzrechtliche Aspekte, wie auch Punkte der Arbeitnehmer-Mitbestimmung zu beachten. Der Datenschutz sollte physische und juristische Personen davor schützen, dass sensible, persönliche Daten in die falschen Hände gelangen. [vgl. österr. DSG 2000 §1]. Manche ID-Systeme erlauben daher eine Pseudonymisierung (siehe Abbildung 6-1), wodurch ein direkter Bezug vermieden wird. Erst bei einem begründeten Verdacht erfolgt eine Reidentifikation.

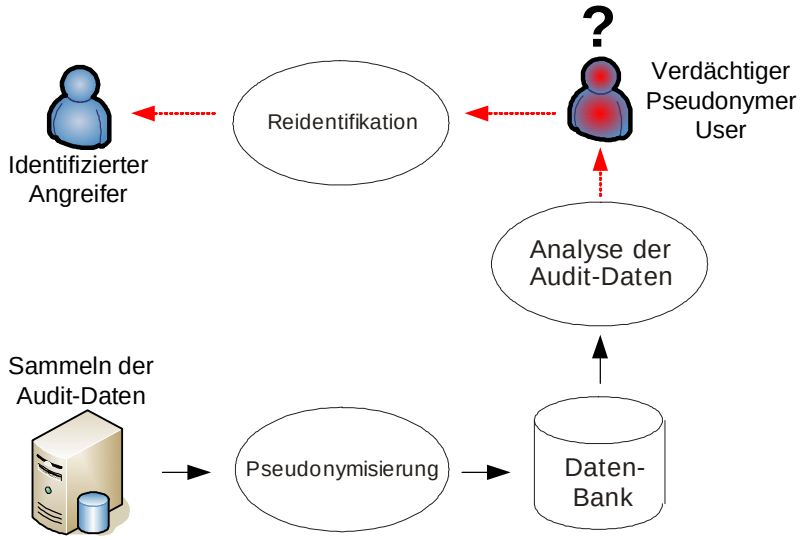


Abbildung 6-1: Pseudonymisierung

Prinzipiell sollten möglichst wenig personenbezogene Daten gesammelt werden. Es ist auch verboten, diese Daten an Dritte weiterzugeben. Die personen-

bezogenen Daten dürfen nur zur Sicherstellung des ordnungsgemäßen Betriebs oder zur Verfolgung von Widrigkeiten genutzt werden. Im Normalfall sind die Auditdaten eines ID-Systems kein gerichtlich anerkanntes Beweismaterial, wie etwa ein notariell beglaubigtes Schriftstück. Die Beurteilung liegt im Ermessen des Gerichts. Um hier Abhilfe zu schaffen, müssen die Daten des ID-Systems mit einer digitalen Signatur versehen werden, um die Integrität, Echtheit der Daten zu sichern. Im Rechtsfall muss ein Nachweis erfolgen, dass die digitale Signatur vertrauenswürdig und der private Schlüssel sicher ist.

Der Einsatz eines ID-Systems ist datenschutzkonform, wenn die Daten nicht zur Leistungs- oder Verhaltenskontrolle herangezogen werden. Allgemein ist es ratsam, vor dem Einsatz des Systems Rücksprache mit einem Juristen und dem Betriebsrat zu halten, da dieser ein Mitbestimmungsrecht hat. Der Einsatz eines ID-Systems muss sensibel geplant und an den Datenschutz angepasst werden. Die einfachste Methode, die Anwender zu informieren ist es, die entsprechenden Punkte in einer Benutzerordnung schriftlich festzuhalten. Dies könnte auch eine Ergänzung zum Arbeitsvertrag sein.

[BSI 02c]

Näheres zu personenbezogenen Daten und deren Verarbeitung findet man im österreichischen Datenschutzgesetz (DSG 2000) §1 und §9.

7 Umsetzung eines NIDS mit Open Source Mitteln

7.1 Einleitung

Zur praktischen Umsetzung eines netzbasierten Intrusion Detection Systems wurde vom Autor Snort gewählt. Einer von vielen Gründen war die weite Verbreitung von Snort - auch in kommerziellen Umgebungen. Deshalb wird dieses von vielen Entwicklern ständig aktualisiert und weiterentwickelt. Ursprünglich wurde es von Martin Roesch 1998 als ein „Light Weight Intrusion Detection System“ entwickelt, ist aber heutzutage mit allen wesentlichen Funktionen vergleichbarer kommerzieller Produkte ausgestattet, und weit davon entfernt als ein „Light-System“ zu gelten. Roesch gründete 2001 das Unternehmen Sourcefire, welches eine kommerzielle grafische Managementkonsole und professionellen Support bietet. Trotzdem bleibt Snort ein Open Source Produkt und wird weiterentwickelt. [www 09]

Snort selbst basiert auf einem Paket-Sniffer, der die Berkeley-Paketfilterbibliothek libpcap verwendet. Eine Regelsprache, die sehr mächtig, zugleich aber auch einfach und schnell zu erlernen ist. Es ist ein Open Source Produkt (steht unter der GNU GPL - General Public License) und ist daher für jedermann ohne Kosten verfügbar. Diverse Schnittstellen zur Alarmierung (syslog, mail, winpopup, database, snmp-traps etc.) sind ebenfalls gegeben. Sollte eine erfolgreiche Installation in Unternehmen nicht gelingen, gibt es sogar kommerziellen Support.

Obwohl Snort für nahezu jede Plattform verfügbar ist, sollte ein Unix-/Linux-System die erste Wahl für das NIDS sein. Für einen unter Unix/Linux orientierten Administrator wird dies nicht schwer sein, da er mit der Kommandozeile vertraut ist.

Vorteile von Snort sind:

- *ist für nahezu jede Plattformen erhältlich (Win, BSD, HP-UX, Linux, Solaris),*
- *Snort ist kostenlos verfügbar,*

- *wird durch die* „Community“ immer aktualisiert,
- *einfache Regelerstellung,*
- *hohe Flexibilität,*
- *Quellcode kann eingesehen werden,*
- *kommerzieller Support ist erhältlich.*

Natürlich hat Snort auch **Nachteile**, die jedoch für alle ID-Systeme gleichermaßen zutreffen:

- *Performanceprobleme*

aufgrund hoher Netzwerkübertragungsgeschwindigkeiten oder schlechten TCP/IP-Implementierungen im Betriebssystem.

- *False positives*

Irrtümliche Bekanntgabe eines Angriffs aufgrund zu strenger Regeln. Wird Snort mit dem Standardregelsatz ohne weitere Anpassungen betrieben, kommt es zu sehr vielen Fehlalarmen.

- *False negatives*

Missbräuche bleiben unentdeckt, da zu wenige Regeln definiert sind.

- *Kein multithreaded Programm*

Bis jetzt ist Snort nicht multithread-fähig. Es kann daher nur ein Paket nach dem anderen abarbeiten. Besonders komplizierte Output-Plug-Ins (z.B. Protokollierung in eine Datenbank) können die Geschwindigkeit von Snort erheblich bremsen. Programme wie „Barnyard“ können dabei Abhilfe schaffen, indem sie den Snort-Prozess bei der Protokollierung, z.B. in eine MySQL-Datenbank entlasten.

Typischerweise erfolgt eine Snort-Installation nicht alleine, sondern mit Remotezugriff SSH, einer Datenbank MySQL und einem Webserver Apache. Dadurch wird das System gesamt betrachtet zu einer Anwendung, wie jede andere, mit mehreren offenen Ports. Snort selbst gilt als sehr sicher, jedoch haben

in den letzten Jahren die anderen Anwendungen (SSH, MySQL, Apache etc.) immer wieder Sicherheitsprobleme gezeigt. Grundlegende Systemadministrations-praktiken, wie das regelmäßige Einspielen von Sicherheitspatches, dürfen daher keinesfalls vernachlässigt werden!

7.2 Arbeitsweise, Aufbau, Architektur von Snort

Die folgende Abbildung zeigt die Arbeitsweise und die Komponenten von Snort [vgl. Rafeeq 03].

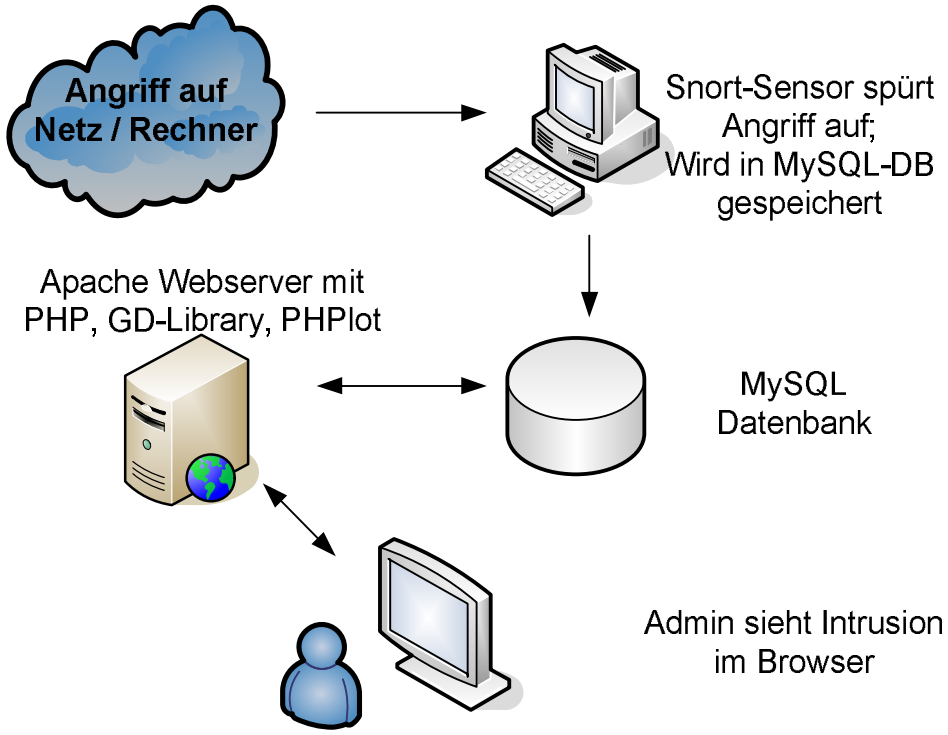


Abbildung 7-1: Prinzip eines NIDS mit Snort

Snort ist im Grunde nur ein Produkt, welches mit Hilfe der libpcap-Bibliothek Pakete vom Interface aufnimmt. Die libpcap Bibliothek stellt das Grundgerüst - die Basis von Snort - dar und ist daher unbedingt notwendig. Die Gesamtheit der Module ergibt ein Netzwerk ID-System.

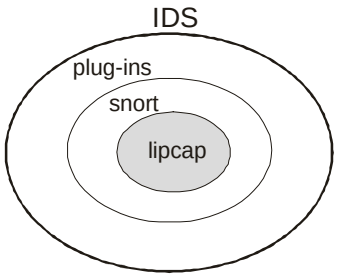


Abbildung 7-2: Modularer Aufbau von Snort

Snort kann als Paket-Sniffer, Netzwerkprotokollant oder vollständiges Network Intrusion Detection System eingesetzt werden. Die Inspektion der Pakete auf schädliche Inhalte ist nicht Sache des Snortkerns. Dies wird durch Plug-Ins und deren Regelsätze implementiert. Somit ist ein modularer und erweiterbarer Aufbau gegeben, an dem sich viele Programmierer weltweit beteiligen.

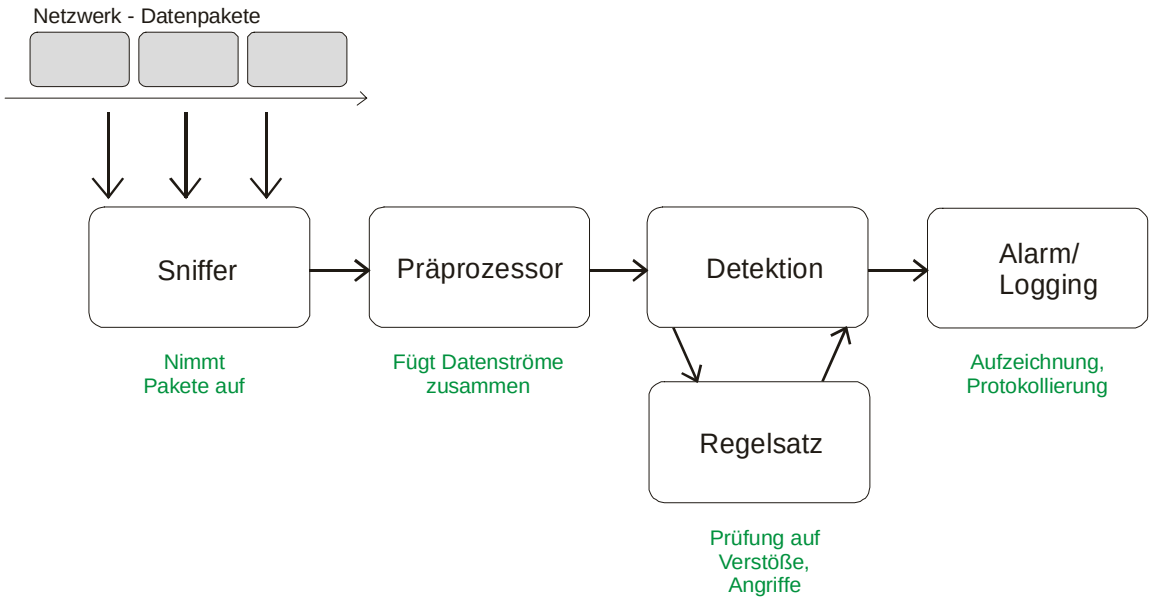


Abbildung 7-3: Architektur von Snort [Beale 03]

Man unterscheidet bei der Architektur von Snort zwischen **vier Basiskomponenten:**

7.2.1 Sniffer (Packet-Capture)

Die Netzwerkkarte nimmt im promiscuous mode alle Pakete des eigenen Netzsegments auf.

7.2.2 Präprozessoren

Fragmentierung ist eine notwendige Funktion der TCP/IP-Suite, da verschiedene Netzwerktechnologien (Ethernet, Frame Relay, ATM) unterschiedliche Paketgrößen (MTU, Maximum Transmission Unit) einsetzen. Leider ist Fragmentierung auch ein beliebtes Mittel von Hackern, um signaturbasierte ID-Systeme zu umgehen [vgl. Northcutt 04]. Wurden Pakete auf ihrer Reise fragmentiert, so müssen sie vor einer Analyse durch entsprechende Präprozessoren (z.B. frag2) zusammengesetzt werden. Die Aufgabe der Präprozessoren liegt also in der Normalisierung der Pakete, bevor diese an die Detection-Engine zur Analyse weitergereicht werden. Die Fragmentierung kann in hohem Ausmaß sogar zu einem erfolgreichen DoS-Angriff gegen das ID-System selbst führen. Dekodierung, Reassembling und Defragmentierung von Datenströmen sind die Hauptaufgaben der Präprozessoren. Bei der Verwendung von TCP als Protokoll ist es möglich Informationen Zeichen für Zeichen in jeweils einem TCP-Segment und damit auch in einem IP-Paket zu übertragen. Geeignete Präprozessoren wie Stream4, reassemblieren solche TCP-Sessions und erkennen somit Angriffe. Präprozessoren können bei verdächtigen Aktionen auch direkt Alarme auslösen. Viele Portscans zum Beispiel werden schon auf niederer Ebene durch Präprozessoren erkannt.

Die Dekodierung von Protokollen wie telnet oder HTTP wird ebenfalls durch Präprozessoren übernommen. Beispielsweise erlaubt der IIS-Webserver von Microsoft auch Backslash-Zeichen „\“, Unicode-Kodierung oder Hexadezimal-Kodierung in den URL-Angaben. Somit kann der Angreifer seinen maliziösen Code auf beliebige Weise mischen.

Beispiel:

http://www.opfer.at\ %75ser\hack.html

Für eine signaturbasierte Angriffserkennung ist das ohne Aufbereitung unmöglich zu erkennen, daher erfolgt vor dem Mustervergleich eine HTTP-Normalisierung durch den Präprozessor (http_decode):

http://www.opfer.at/user/hack.html

Für die Protokolle telnet und rpc (remote procedure call) gibt es ebenfalls geeignete Präprozessoren. Portscans und ARP-Spoofing [vgl. Kapitel 3.3.1, Kapitel 3.3.2] können auch durch Präprozessoren erkannt bzw. verhindert werden (portscan, arpspoof).

7.2.3 Detection Plug-Ins

Diese untersuchen Inhalte der Pakete (z.B. icmp_code_check) anhand definierter Regeln nach Bedrohlichem. Sie stehen in textbasierten Dateien und sind nach logischen Kategorien organisiert (z.B. ftp.rules, icmp.rules, dns.rules, etc.).

7.2.4 Output Plug-Ins

Unterstützen die unterschiedlichen Ausgabeformate (z.B. alert_syslog, alert_database).

Der Ablauf bei Snort ist in logische Einheiten eingeteilt. Die Netzwerkkarte nimmt sämtliche Datenpakete mit Hilfe der Paketfilterbibliothek libpcap im Promiscuous-Modus auf und übergibt sie zum Signaturvergleich. Bis hierher ist der Weg für alle Pakete gleich. Nun wird anhand des Regelwerks entschieden, ob ein Angriff vorliegt oder nicht. Wenn ja, dann kann eine Aufzeichnung, eine Verständigung an den Sicherheitsbeauftragten, oder eine automatische Reaktion durch ein Intrusion Response System erfolgen.

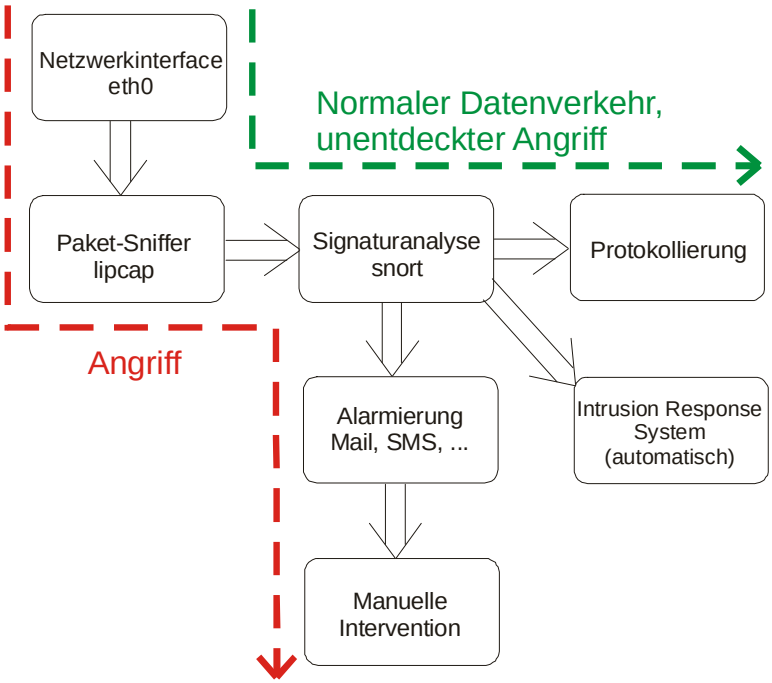


Abbildung 7-4: Ablaufdiagramm Snort [Faulhaber 02]

Anhand der Grafik ist erkennbar, dass eine nicht entdeckte Verletzung trotzdem aufgezeichnet wird und für spätere Beweisführungen verwendet werden kann.

Den Vergleich mit teuren herstellerspezifischen Lösungen braucht Snort nicht zu scheuen. Es wurde für einen hohen Datendurchsatz und eine gute Leistung optimiert. Zwar bietet es in seiner Grundform keine schönen Oberflächen, dem wurde aber durch ACID (steht für Analysis Console for Intrusion Databases) Abhilfe geschaffen. ACID ist eine Applikation, die aus den Informationen der MySQL-Datenbank PHP-Seiten generiert und ist am CERT Coordination Center entwickelt worden. Durch schnell verfügbare und kostenlose Signaturen der großen Snort-Community kann Snort besonders punkten. Prinzipiell arbeitet es nach dem Prinzip der misuse detection. Mittlerweile gibt es aber auch schon Plugin-Möglichkeiten, wie SPADE (Statistical Packet Anomaly Detection Engine) für anomaly detection, die sich noch in der Testphase befinden.

7.3 Überlegungen bezüglich Hardware und Betriebssystem

Bei der Auswahl der Hardware sind einige Faktoren zu beachten. Nicht jedes Betriebssystem mit beliebiger Hardware eignet sich als Snort-Sensor. Oberstes Ziel, aus der Sicht der Hardware ist es, keine Pakete zu verlieren. Die Leistung des Sensors wird maßgeblich durch

- *Prozessor,*
- *Arbeitsspeicher (RAM),*
- *Massenspeicher (Festplatte) und*
- *Netzwerkkarte*

bestimmt.

Die Prozessorarchitektur bestimmt, wie schnell die Daten analysiert und kategorisiert werden können. Für eine effiziente Ausführung muss genügend RAM-Speicher vorhanden sein. Bevor die Daten auf die Festplatte geschrieben werden, stehen diese ja im Systemspeicher. Um die anfallenden Verbindungsdaten und Alarme auch entsprechend aufbewahren zu können, ist ausreichender Festplatten-Speicherplatz notwendig. Die wichtigste Hardwarekomponente stellt sicherlich die Netzwerkkarte dar. Diese muss in der Lage sein, eine Vielzahl von Datenpaketen in kürzester Zeit vom Medium aufzunehmen. Je nach Größe des Netzwerkes und Rahmen des Budgets sollte mindestens eine 100 Mbit- oder besser eine Gbit-Karte eingesetzt werden. Setzt man eine Gbit-Netzwerkkarte ein, so sollte der Rest der Rechnerhardware auch entsprechend leistungsfähig sein, sonst können die Daten vom schnellen Netzwerkinterface nicht verarbeitet werden, und die Karte wird somit ausgebremst. Wenn möglich sollte wegen besserer Datendurchsatzraten die Karte eines Markenherstellers gewählt werden (z.B. 3Com, Intel).

Die beste Hardware ist jene, die keinen Paketverlust hinnehmen muss! In der Praxis kommt es sehr wohl zu Verlusten aufgrund der Hardware und Konfiguration des ID-Systems. Man muss zwischen Preis und Leistung abwägen. Vor der Anschaffung der Hardware muss die Wahl des Betriebssystems erfolgen, denn ein Windows-System stellt zum Beispiel höhere Anforderungen an die Hardware-komponenten, als ein Unix-System. Während für Unix/Linux-Systeme ohne grafische Oberfläche wie KDE oder Gnome 128 MB RAM-Speicher ausreichend sind, sind für Windows 256 MB RAM erforderlich. Bei der Zusammenstellung des Systems sollte man auch auf externe Medien wie CD, DVD oder Bandlaufwerk zur Sicherung der Files nicht vergessen.

Tabelle 7-1: Hardwareanforderungen

Prozessor	2,4 GHz aufwärts
RAM	128 MB Unix/Linux, 256MB Windows
Festplattenspeicher	60 GB aufwärts
Netzwerkkarte	100 Mbit oder Gbit

Eine sehr praktikable Methode für die Fernwartung des Sensors beziehungsweise der Managementstation stellt die Verwendung einer zweiten Netzwerkkarte da. Es kann eine zweite gesicherte Verbindung hergestellt werden, ohne dass die Snifferfunktion der ersten Karte beeinträchtigt wird. Jene Netzwerkkarte die für das ID-System die Pakete aufnimmt wird „stealth“ geschaltet, das heißt sie wird von den restlichen Netzwerkteilnehmern nicht gesehen, und kann auch nicht „gepingt“ (ICMP echo request) werden (nur unter Unix/Linux-Systemen möglich).

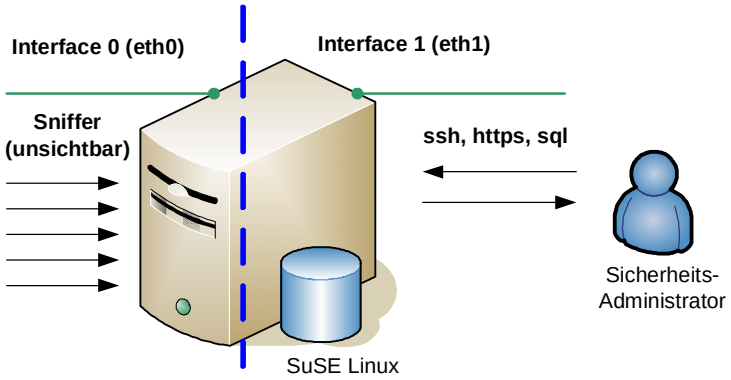


Abbildung 7-5: Hardwarekonfiguration

Performance, Benutzerfreundlichkeit und Vertrautheit sind wichtige Punkte bei der Wahl des Betriebssystems. Unix/Linux-Distributionen kommen meist mit geringeren Hardwareanforderungen aus. Mit Windows sind jedoch die meisten User und Administratoren besser vertraut. Ist man mit unix-ähnlichen Systemen vertraut, sollte man zu Gunsten der Stabilität und Performance ein Betriebssystem aus dieser Gruppe wählen, auch wenn die Installation nicht immer benutzerfreundlich ist. Snort selbst ist kein einfaches Programm und erfordert einiges an Zeit. Gleichzeitig ein Selbststudium für ein bestimmtes Betriebssystem durchzuführen, ist nicht sinnvoll. In diesem Fall sollte man einfach zur Windowsvariante mit guter Hardwareausstattung greifen. Die

Tabelle 7-2 zeigt eine Gegenüberstellung der Vor- und Nachteile der unterschiedlichen Systeme.

Unix/Linux		**Windows**	
Pro	Contra	Pro	Contra
hardware-effizienter	Installation	einfache Installation	schlechtere Performance
(gratis)	Lernaufwand	grafische Administration	keine native Snort-Plattform
viele zusätzliche Tools			Lizenzkosten
automatisierte Filter (Scripts)			

Tabelle 7-2: Betriebssystemwahl

Die beste Auswahl trifft der Administrator dann, wenn er das Betriebssystem wählt, mit dem er selbst am besten vertraut ist.

7.4 Format von Snort

Anhand eines Ping-Befehls (ICMP echo request) wird hier die Darstellung der Netzwerkpakete durch Snort kurz aufgezeigt:

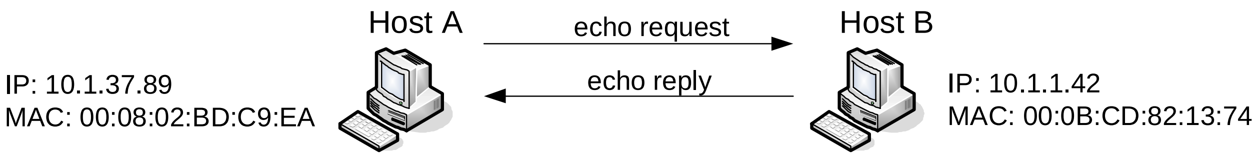


Abbildung 7-6: ICMP echo request

ICMP echo request:

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
+

Datum    Zeit                MAC Absender        MAC Empfänger    Type          Länge

11/04-08:07:21.465782   0:8:2:BD:C9:EA   ->   0:B:CD:82:13:74   type:0x800
len:0x4A
```

IP Absender IP Empf. Prot. TTL TOS ID Hdr-Länge Paket-Länge

```
10.1.37.89  ->  10.1.1.42  ICMP  TTL:128  TOS:0x0  ID:24547  IpLen:20
DgmLen:60
```

Prot-Type Code ID Sequ-Nr Nachrichtentyp

```
Type:8    Code:0  ID:512   Seq:13568       ECHO
```

DATEN

```
abcdefghijklmnopqrstuvwabcdefghi
```

ICMP echo reply:

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
+

11/04-08:07:21.465991  0:B:CD:82:13:74  ->  0:8:2:BD:C9:EA  type:0x800
len:0x4A

10.1.1.42 -> 10.1.37.89 ICMP TTL:128 TOS:0x0 ID:16117 IpLen:20 DgmLen:60

Type:0  Code:0  ID:512  Seq:13568  ECHO REPLY

abcdefghijklmnopqrstuvwabcdefghi
```

Zur Hilfestellung für weitere Beispiele und Analysen ist hier der IPv4-Header abgebildet.

1 2 3 4

0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7

<table>
<tr><td colspan="4">Version</td><td colspan="4">IHL</td><td colspan="8">Type of Service</td><td colspan="16">Total Length</td></tr>
<tr><td colspan="16">Sequenznumber</td><td colspan="3">Flags</td><td colspan="13">Fragment Offset</td></tr>
<tr><td colspan="8">TTL</td><td colspan="8">Protocol</td><td colspan="16">Header Checksum</td></tr>
<tr><td colspan="32">Source Address</td></tr>
<tr><td colspan="32">Destination Address</td></tr>
<tr><td colspan="24">Options</td><td colspan="8">Padding</td></tr>
</table>

Abbildung 7-7: IP-Header Version 4 [vgl. www 03]

7.5 Modi von Snort

Snort kann in drei unterschiedlichen Betriebsarten laufen:

[Beale 03] [Spenneberg 03]

7.5.1 Sniffer

Die Pakete und deren Header Information (IP/TCP/UDP/ICMP) werden laufend auf der Konsole ausgegeben. Das Verhalten ist ähnlich dem von tcpdump[5].

```
snort –v

11/04-10:11:02.868568 10.1.37.35:138 -> 10.1.255.255:138
UDP TTL:128 TOS:0x0 ID:10326 IpLen:20 DgmLen:229
Len: 201
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
+

11/04-10:11:03.255641 ARP who-has 10.1.13.13 tell 10.1.1.221

11/04-10:11:03.376811 10.1.3.147 -> 224.0.0.1
PROTO002 TTL:1 TOS:0xC0 ID:0 IpLen:20 DgmLen:28
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
+
```

Ohne weitere Optionen werden nur die IP-Informationen angezeigt. Mit weiteren Optionen (*-de*) werden auch Ethernet-Daten (Layer 2) und Paketinhalte dargestellt:

```
snort –vde
11/04-10:14:27.506411   0:8:2:BD:C9:EA   ->   0:3:BA:F:39:CB   type:0x800
len:0x1EE
10.1.37.89:2389  ->  192.168.1.12:8008  TCP  TTL:128  TOS:0x0  ID:46521
IpLen:20 DgmLen:480 DF
***AP*** Seq: 0xC979A63F  Ack: 0x5316E836  Win: 0x4470  TcpLen: 20
47 45 54 20 68 74 74 70 3A 2F 2F 77 77 77 2E 67  GET http://www.g
6D 78 2E 61 74 2F 20 48 54 54 50 2F 31 2E 31 0D  mx.at/ HTTP/1.1.
0A 48 6F 73 74 3A 20 77 77 77 2E 67 6D 78 2E 61  .Host: www.gmx.a
74 0D 0A 55 73 65 72 2D 41 67 65 6E 74 3A 20 4D  t..User-Agent: M
6F 7A 69 6C 6C 61 2F 35 2E 30 20 28 57 69 6E 64  ozilla/5.0 (Wind
6F 77 73 3B 20 55 3B 20 57 69 6E 64 6F 77 73 20  ows; U; Windows
4E 54 20 35 2E 30 3B 20 64 65 2D 44 45 3B 20 72  NT 5.0; de-DE; r
76 3A 31 2E 37 29 20 47 65 63 6B 6F 2F 32 30 30  v:1.7) Gecko/200
34 30 38 30 33 20 46 69 72 65 66 6F 78 2F 30 2E  40803 Firefox/0.
39 2E 33 0D 0A 41 63 63 65 70 74 3A 20 74 65 78  9.3..Accept: tex
74 2F 78 6D 6C 2C 61 70 70 6C 69 63 61 74 69 6F  t/xml,applicatio
6E 2F 78 6D 6C 2C 61 70 70 6C 69 63 61 74 69 6F  n/xml,applicatio
6E 2F 78 68 74 6D 6C 2B 78 6D 6C 2C 74 65 78 74  n/xhtml+xml,text
2F 68 74 6D 6C 3B 71 3D 30 2E 39 2C 74 65 78 74  /html;q=0.9,text
2F 70 6C 61 69 6E 3B 71 3D 30 2E 38 2C 69 6D 61  /plain;q=0.8,ima
67 65 2F 70 6E 67 2C 2A 2F 2A 3B 71 3D 30 2E 35  ge/png,*/*;q=0.5
0D 0A 41 63 63 65 70 74 2D 4C 61 6E 67 75 61 67  ..Accept-Languag
65 3A 20 64 65 2D 64 65 2C 64 65 3B 71 3D 30 2E  e: de-de,de;q=0.
```

[5] Standard Protokollanalysator unter Linux Systemen

Das Beispiel zeigt eine typische HTTP-Abfrage einer Webseite (www.gmx.at). Sehr schön erkennt man hier die zusätzlichen Informationen im HTTP-Header wie User-Agent (Mozilla) oder Betriebssystem (Windows), die mit übertragen werden. Nach Beendigung der Anwendung stellt Snort eine Statistik dar:

```
===============================================================================
Snort received 1255 packets
    Analyzed: 1255(100.000%)
    Dropped: 0(0.000%)
===============================================================================
Breakdown by protocol:
    TCP: 130        (10.359%)
    UDP: 73         (5.817%)
   ICMP: 1          (0.080%)
    ARP: 8          (0.637%)
  EAPOL: 0          (0.000%)
   IPv6: 0          (0.000%)
    IPX: 4          (0.319%)
  OTHER: 161        (12.829%)
DISCARD: 0          (0.000%)
===============================================================================
Action Stats:
ALERTS: 0
LOGGED: 376
PASSED: 0
```

7.5.2 Netzwerkprotokollant (Packet Logger)

Dieser unterscheidet sich nur geringfügig vom Sniffer-Modus, in dem die protokollierten Daten in ein Verzeichnis (muss schon vorher existieren) gespeichert werden. Dabei legt Snort selbst Unterverzeichnisse mit den IP-Adressen von Kommunikationspartnern an. Es erzeugt pro Protokoll und verwendetem Port eine Datei. Entscheidend ist daher, was aufgezeichnet wird, andernfalls erzeugt ein simpler Portscan (65.536 TCP + 65.536 UDP) schon 131.072 unbrauchbare Dateien!

```
snort –vdel c:\snort\log\
```

Wird mit der Option –v (verbose) die Protokollierung auf die Standardausgabe geschaltet, so wird der Bildschirm wie ein serielles Gerät mit 38400 Baud angesprochen, dies wäre wesentlich langsamer, als eine 10 MBit-Netzwerkkarte leisten könnte.

Zum Zwecke der schnelleren Dokumentation wurden einige Beispiele und Versuche auf einem System mit Windows XP durchgeführt. Das eigentliche Network Intrusion Detection System wurde auf einem SuSE Linux-System konfiguriert [vgl. Kapitel 9.2].

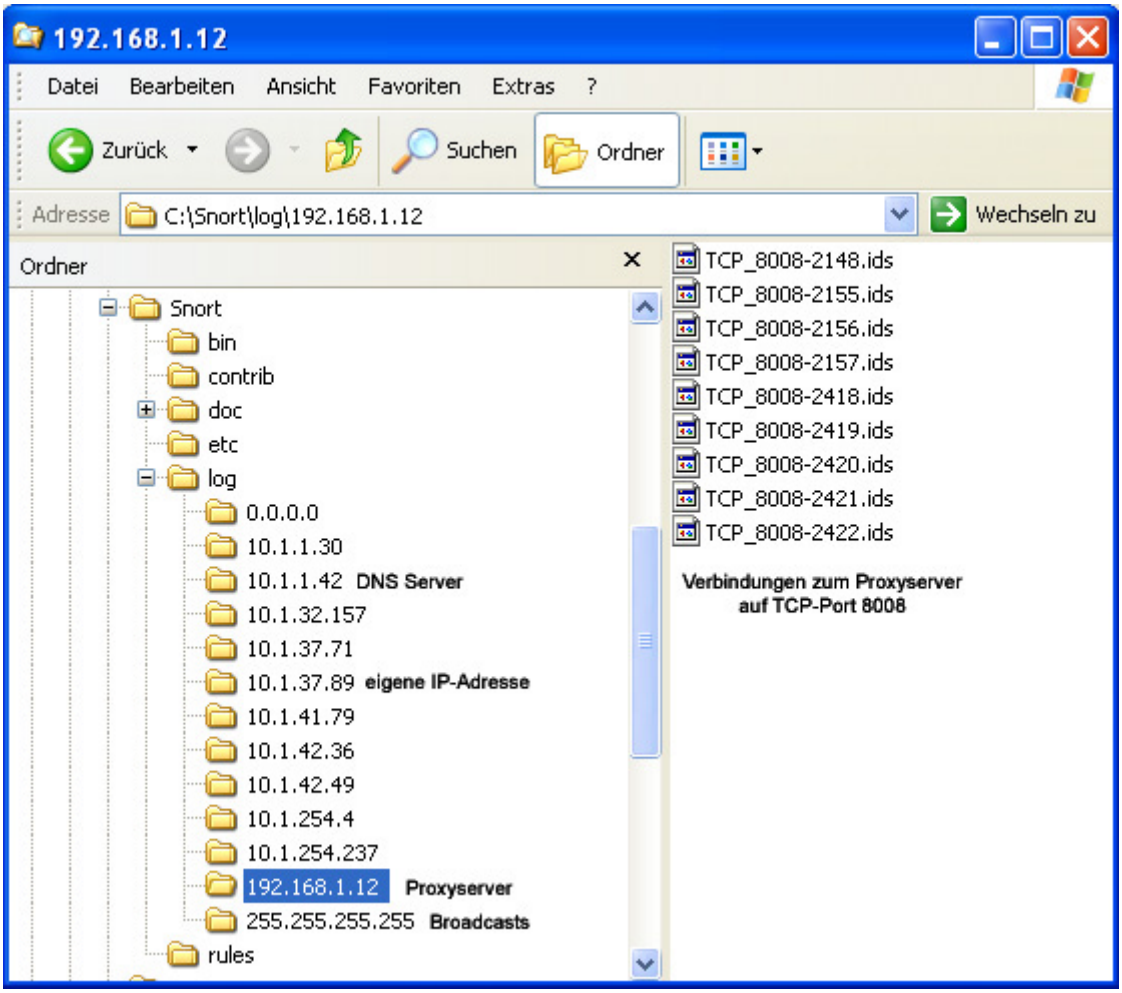


Abbildung 7-8: Protokoll-Verzeichnisstruktur von Snort

Besser ist eine Aufzeichnung nicht im Klartext-Modus (ASCII), wie eben beschrieben, sondern den binären Protokollmodus (Option *-b*) dafür einzusetzen. Der Rechner braucht die Daten während der Aufzeichnung nicht ins lesbare Format umwandeln, und schreibt in eine Datei. Dieses Format kann auch von anderen Tools wie tcpdump oder ethereal (weit verbreitete Sniffer) gelesen werden.

```
snort –vbdel c:\snort\log\
```

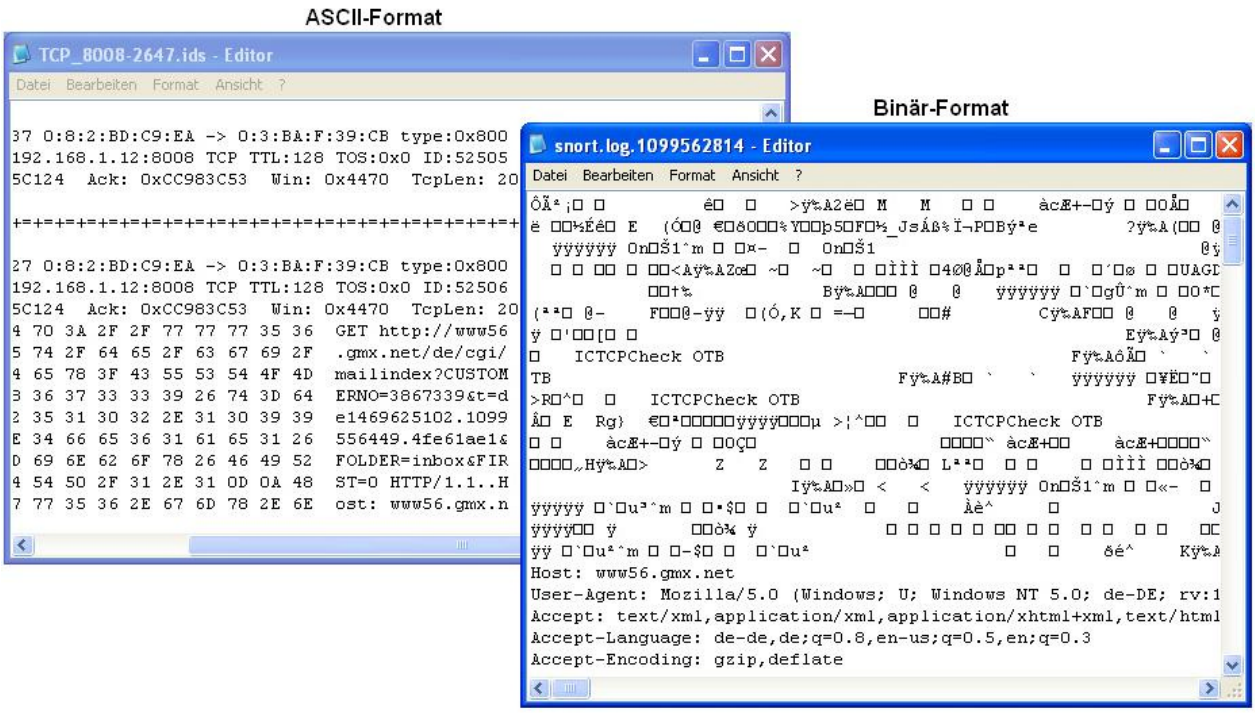


Abbildung 7-9: Vergleich ASCII-/Binäraufzeichnung

Die Headerinformationen (MAC-/IP-Adressen) werden zur Laufzeit nicht analysiert sondern binär in einer Datei gespeichert.

7.5.3 Netzwerk ID-System

In dieser komplexeren Betriebsart spielt Snort seine Vorteile voll aus. Die Pakete werden anhand von Regeln überprüft und nur die zutreffenden Pakete werden protokolliert. Maliziöse Datenströme werden anhand von Signaturen aufgedeckt. Ein riesiger Regelsatz wird bereits mitgeliefert. Für einen sinnvollen Einsatz ist dieser aber nicht zu empfehlen, da ohne Anpassung sehr viele Fehlalarme entstehen.

Typischer Aufruf im Windows NIDS-Modus mit Regeldatei:

```
snort -de -l c:\Snort\log\test -i \Device\NPF_{F6B48100-C7A5-4523-9ACC-
CF30EAA15E9D} -c testrule.conf
```

7.6 Regelwerk

Im NIDS-Mode untersucht Snort den Datenverkehr anhand vordefinierter Regeln (Signaturen). Somit ist es möglich, unterschiedlich auf den speziellen Netzwerkverkehr zu reagieren. Die Regelsätze sind nach verschiedenen Kategorien (Buffer Overflows, Backdoors, Trojaner, etc.) gruppiert. Im Gegensatz zum Protokoll-Modus werden nur zutreffende Pakete protokolliert. Die Regeln gestalten sich ähnlich, wie man es bei Access-Control-Lists von Cisco kennt.

Der **Regelheader** beschreibt die Kommunikationsverbindung, also die Verbindungspartner (IP-Adressen), Protokoll, Ports und die Aktion die eingeleitet wird.

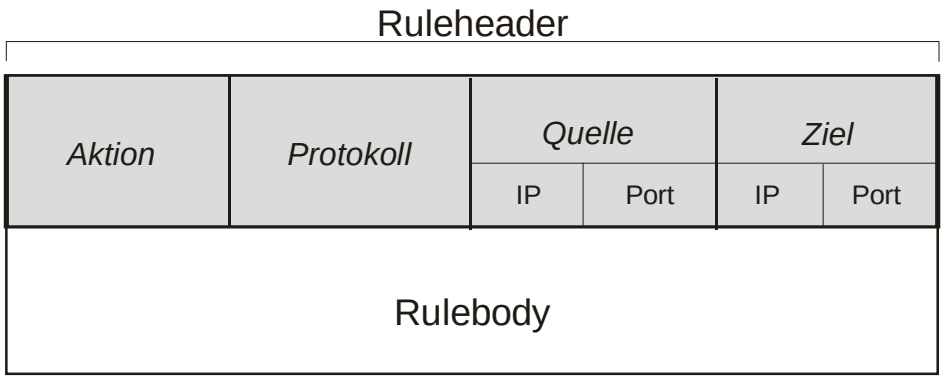


Abbildung 7-10: Aufbau des Ruleheader [vgl. Beale 03]

Der zusätzliche **Regelbody** (mit diversen Optionen) erlaubt die Analyse nach bestimmten Inhalten und verfeinerte Reaktionen darauf. Eine Regel muss nicht immer Optionen (Body) enthalten [vgl. Northcutt 04].

Snortregel = Header + (Body)

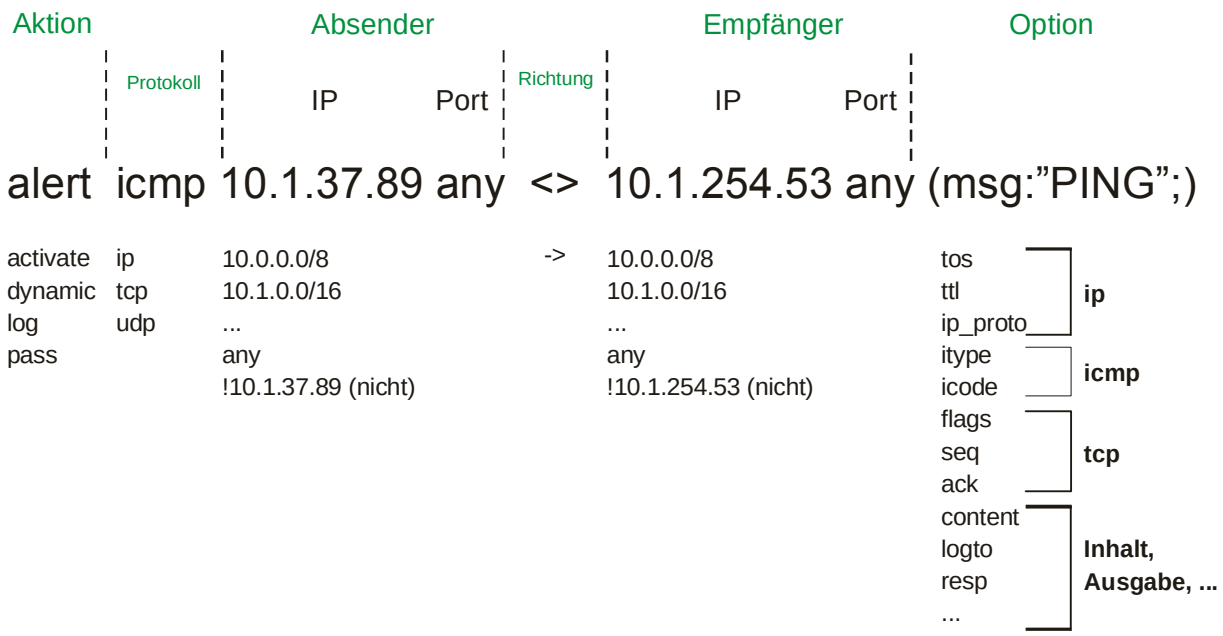


Abbildung 7-11: Kurzübersicht Regelsyntax

Die Abbildung 7-11 soll nur eine kleine Übersicht darstellen, und Hilfe für das Verständnis der Regelsyntax von Snort geben. Es gibt eine Vielzahl weiterer Optionen. Hier sind nur ein paar wichtige davon aufgelistet. Das Beispiel in Abbildung 7-11 würde bei jeglichem ICMP-Traffic (nicht nur ping) einen Eintrag mit der Überschrift „PING“ im Alert-File von Snort generieren:

```
[**] [1:0:0] PING [**]
[Priority: 0]
11/05-10:33:51.603717  0:8:2:BD:C9:EA  ->  0:E0:1E:58:6D:39  type:0x800
len:0x4A
10.1.37.89 -> 10.1.1.204 ICMP TTL:128 TOS:0x0 ID:32285 IpLen:20 DgmLen:60
Type:8  Code:0  ID:512  Seq:28416  ECHO

[**] [1:0:0] PING [**]
[Priority: 0]
11/05-10:33:51.604510  0:E0:1E:58:6D:39  ->  0:8:2:BD:C9:EA  type:0x800
len:0x4A
10.1.1.204 -> 10.1.37.89 ICMP TTL:255 TOS:0x0 ID:32285 IpLen:20 DgmLen:60
Type:0  Code:0  ID:512  Seq:28416  ECHO REPLY
```

Beispiele

Nachstehend werden einfache Beispiele erklärt, die nur das Grundprinzip der Regelerstellung vermitteln sollen.

- *Ping*

Wie schon oben erwähnt, wurde beim vorherigen Beispiel jedes ICMP-Paket mit der PING Überschrift protokolliert, was so nicht korrekt ist, denn Fehlermeldungen (destination unreachable) werden ja auch per ICMP versendet. Eine Unterscheidung ist durch das ICMP-Typfeld möglich. Eine ping-Anfrage verwendet ICMP-Typ 8, wobei mit ICMP-Typ 0 geantwortet wird.

Obwohl das ICMP-Protokoll keine Ports, sondern message-types verwendet, muss laut Snort-Syntax „any“ an dieser Stelle stehen.

Regeln:
```
alert icmp any any <> any any (msg:"PING Anfrage";itype:8;)
alert icmp any any <> any any (msg:"PING Antwort";itype:0;)
```

Würde folgende Einträge im Alert-File erzeugen:
```
[**] [1:0:0] PING Anfrage [**]
[Priority: 0]
11/05-10:49:13.081743  0:8:2:BD:C9:EA  ->  0:E0:1E:58:6D:39  type:0x800
len:0x4A
10.1.37.89 -> 10.1.1.204 ICMP TTL:128 TOS:0x0 ID:33915 IpLen:20 DgmLen:60
Type:8  Code:0  ID:512  Seq:30720  ECHO

[**] [1:0:0] PING Antwort [**]
[Priority: 0]
11/05-10:49:13.082511  0:E0:1E:58:6D:39  ->  0:8:2:BD:C9:EA  type:0x800
len:0x4A
10.1.1.204 -> 10.1.37.89 ICMP TTL:255 TOS:0x0 ID:33915 IpLen:20 DgmLen:60
Type:0  Code:0  ID:512  Seq:30720  ECHO REPLY
```

- *Content-Filterung*

Natürlich können auch Dateninhalte gefiltert werden, um darauf gezielte Aktionen zu setzen. Nachstehend werden Pakete mit dem Inhalt www.gmx.at gefiltert und eine Alarmmeldung erzeugt. Die Verbindung wird - wie in Firmen üblich - durch einen Proxy-Server auf TCP Port 8008 aufgebaut.

Regel:
```
alert tcp any any <> any 8008 (content:"www.gmx.at";msg:"GMX Aufruf ueber
Proxy";)
```

Protokolliert:
```
[**] GMX Aufruf ueber Proxy [**]
11/05-11:00:13.810282  0:8:2:BD:C9:EA  ->  0:3:BA:F:39:CB  type:0x800
len:0x170
10.1.37.89:2111  ->  192.168.1.12:8008  TCP  TTL:128  TOS:0x0  ID:35873
IpLen:20 DgmLen:354 DF
```

```
***AP*** Seq: 0x91B6648F  Ack: 0xD4328A33  Win: 0x4470  TcpLen: 20
GET http://www.gmx.at/ HTTP/1.0..Accept: image/gif, image/x-xbit
map, image/jpeg, image/pjpeg, application/vnd.ms-excel, applicat
ion/msword, application/x-shockwave-flash, */*..Accept-Language:
 de-at..User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows N
T 5.0)..Host: www.gmx.at..Proxy-Connection: Keep-Alive....
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
+
```

Sollen raffinierte Angriffe oder Portscans entdeckt werden, sind wesentlich komplexere und umfangreichere Regeldefinitionen notwendig, da ja der gesamte Zusammenhang der Datenverbindung vom ID-System verstanden werden muss. Es genügt nicht, jedes Paket einzeln zu bewerten. Der Kontext einer Verbindung ist entscheidend. Die meisten Regelsätze zum Erkennen von Angriffen werden von vielen freiwilligen Helfern (Snort-Community) entwickelt. Trotzdem ist es essentiell wichtig, anhand dieser einfachen Beispiele das Prinzip zu verstehen. Kein ID-System kann „out-of-the-box“ (Standardinstallation) verwendet werden, daher sind Anpassungen immer ein Muss! Wird beispielsweise ein Microsoft Internet Information Server (IIS Webserver) in einer DMZ überwacht, so genügt es, jene Regeln zu aktivieren, die sich auf diesen Servertyp beziehen.

Extrem wichtig ist, dass eigens erstellte Regeln anhand von Teststellungen mit Packet-Sniffern (z.B. Ethereal) getestet werden. Fehlerhafte Regelsätze können die Systemleistung erheblich beeinträchtigen und Alarm- bzw. Log-Dateien schnell überfluten.

7.7 Verschlüsselung

Beim Einsatz mehrerer Sensoren und einer zentralen Management-Station sollten die Verbindungen kryptografisch gesichert sein (Verschlüsselung). Snort selbst kümmert sich um keine Verschlüsselung, dies erreicht man durch den Einsatz des **stunnel** Tools. Diese Anwendung erlaubt es, jede TCP-Verbindung durch das sichere Übertragungsprotokoll SSL (Secure Socket Layer) zu schützen.

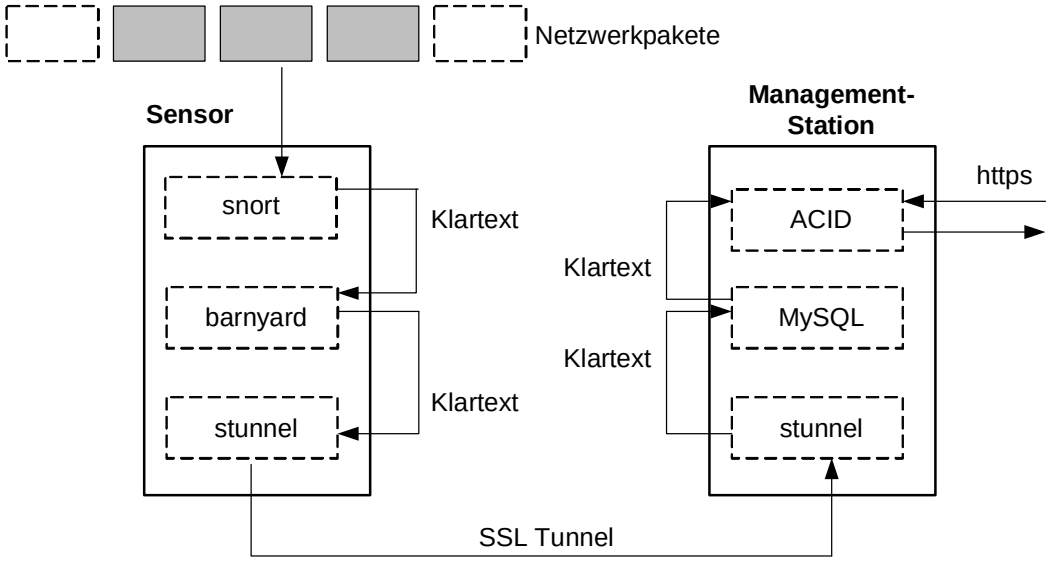


Abbildung 7-12: Verschlüsselung [vgl. Spenneberg 03]

Die lokale Anwendung **Barnyard** [vgl. Kapitel 8.2] am Sensor ist zuständig für die schnelle Weiterverarbeitung der durch Snort aufgesammelten Daten, damit der Snort-Prozess entlastet wird. Barnyard übergibt die Daten im Klartext an stunnel. Das stunnel-Tool sorgt für eine verschlüsselte Verbindung zur Managementstation, wo eine Entschlüsselung und anschließende Klartext-Übergabe an die lokale MySQL-Datenbank erfolgt. ACID ermöglicht eine grafische Auswertung der Ergebnisse, die dann per Browser abrufbar sind.

Natürlich belastet eine Verschlüsselung auch das System und erzeugt zusätzlich Overhead. Eine gängige Methode ist es, die Sensoren und die Managementstation separat in ein eigenes logisches (Adressierung) und physikalisches (Verkabelung) Netzwerk auszulagern. Nur das unsichtbare Interface der Sensoren lauscht im lokalen Netz der User. Damit wird auch eine zusätzliche Verkehrslast durch das NIDS im Anwender-LAN vermieden.

7.8 Installation von Snort

Wie schon kurz erwähnt, ist Snort auch für Windows, sogar als exe-File, erhältlich. Die bessere Wahl für einen dauerhaften Einsatz ist jedoch, Snort unter einer Linux/Unix Distribution zu verwenden (Sensor kann unter Linux „unsichtbar“ betrieben werden). Viele Linux-Distributionen, wie RedHat oder SuSE integrieren Snort schon in ihre Distributionspakete. Die mitgelieferten Softwarepakete sind oft

nicht am aktuellen Stand oder besitzen nicht den vollen Funktionsumfang, daher können auch die vorkompilierten Binaries oder RPM-Pakete (RedHat Packet Manager) von http://www.snort.org zur Installation verwendet werden. Unter Verwendung der binären Version sind mehrere Konfigurationseinstellungen bei der Installation und eine bessere Anpassung an die jeweilige Hardware möglich. Vor der Installation sollte man sich der Abhängigkeiten zu anderen Applikationen wie Webserver (Apache), Datenbank (MySQL), grafische Auswertung (ACID) etc. vor allem bei der Managementstation bewusst sein. Besonders wichtig ist viel lokaler Speicherplatz auf dem Rechner. Die Snort-Community empfiehlt für einen Sensor ca. 9GB freien Speicherplatz, am besten auf einer eigenen Festplatte oder Partition. Das bringt Vorteile bei der Sicherung, und entlastet das root-System.

Unnötige Dienste wie FTP, NIS oder NFS bringen nur zusätzliche Sicherheitsrisiken und sollten auf dem System zu Gunsten von Performance und Sicherheit nicht laufen!

Eine detaillierte schrittweise Installation wird hier nicht abgehandelt, dazu gibt es eine Vielzahl von Online-Dokumentationen, How-To´s und Bücher. Wichtige Punkte für die Installation werden aber auch hier erklärt.

- *Aufteilung der Festplatte*

Die Systempartition wird mit 700 MByte für einen Sensor definiert. Da keine grafischen Oberflächen, wie Gnome oder KDE zum Einsatz kommen, ist diese Größe für ein Minimalsystem ausreichend.

Eine Swap-Partition hat die gleiche Aufgabe wie die Auslagerungsdatei von Windows. Wenn zu wenig physikalischer Arbeitsspeicher (RAM) vorliegt, werden Teile des RAM-Inhalts auf diese Partition ausgelagert. In der Regel wählt man diese Partition in der doppelten Größe des physikalischen RAM.

Beim Analysieren der Netzwerkdaten kann ein Sensor erhebliche Datenmengen erzeugen. Um eine Beeinflussung des Betriebssystems zu vermeiden, verwendet man eine zusätzliche Datenpartition (/sensor). Diese kann auch besser separat auf externe Medien gesichert werden.

/swap	512 MByte (256 MByte phys. RAM)
/	700 MByte
/sensor	Rest (Minimum 5 GByte)

Abbildung 7-13: Partitionierungsempfehlung für Sensorinstallation

Die Installation eines Minimalsystems bei SuSE 9.1 professionell benötigt mit den notwendigen Tools rund 400MByte.

/swap	512 MByte (256 MByte phys. RAM)
/	1 GByte
/var	Rest (Minimum 10 GByte)

Abbildung 7-14: Partitionierungsempfehlung für Management-Station

Auf der Management-Station sind mehrere Pakete, wie z.B. Apache, MySQL, ACID etc. zu installieren, daher muss auch die Systempartition größer gewählt werden. In die Datenpartition /var/lib/mysql werden die Daten der Sensoren in eine MySQL-Datenbank zusammen abgelegt, deshalb sollte sie nicht zu klein gewählt werden.

- *Paketabhängigkeiten*

snort libpcap barnyard ssh mysql_client	Kernkomponente zum sniffen der Pakete Bibliothek, auf der Snort basiert Prozess zum Protokollieren in die Datenbank Für einen gesicherten Remotezugriff Zur Kommunikation mit dem DB-Server	***Notwendig für Sensor***
mysql apache2 php4_mod php4_mysql mod_ssl acid phpmyadmin adodb	Multithreaded Datenbank Webserver PHP-Modul für Webserver Datenbankfunktionen Gesicherter Webzugriff Grafische Managementkonsole für Snortalerts Grafische Steuerung für MySQL-Datenbank Datenbank Plug-In für ACID	***Notwendig für Management-Station***
phplot gd libpng jpgraph zlib	Grafische Bibliotheken für ACID	
make gcc automake autoconf	Tools für die Installation aus Quellcode	

Abbildung 7-15: Paketabhängigkeiten bei der Installation

Für den Sensor ist nur eine kleine Auswahl von Nöten. Die Kompilierung der Sensor-Pakete erfolgt meist auf einer anderen Maschine, da auch ein Compiler wie „gcc“ auf dem Sensor ein Sicherheitsrisiko birgt. Natürlich kann eine Managementstation auch gleichzeitig als Sensor arbeiten, sicherer ist es allerdings diese Prozesse auch physikalisch auf getrennten Rechnern laufen zu lassen.

- *Installation aus dem Quellcode*

Die Installation aus dem Quellcode ist immer besser als die Verwendung von vorkompilierter Software, da der Quellcode für das eigene System (Hardware) optimal angepasst wird. Für diese Art der Installation müssen jedoch zusätzlich Pakete, wie der GNU-Compiler gcc [siehe Abbildung 7-15] installiert werden.

Der Großteil der für Linux entwickelten Software liegt im tar-Format vor (z.B. acid-0.9.6b.tar.gz oder snort-2.3.2.tar.gz) und wird zuerst lokal im temporären Verzeichnis /tmp extrahiert. Danach werden die nachstehenden Befehle der Reihe nach ausgeführt:

./configure
Das Skript überprüft das System, Umgebungsvariablen, Abhängigkeiten zu weiterer Software. Beim Aufruf können diverse Optionen und Einstellungen angegeben werden. Verläuft die Ausführung ohne Fehler, wird das makefile erstellt.

Eine typische Anweisung bei der Sourcecode-Installation des Barnyard-Tools mit MySQL-Unterstützung sieht wie folgt aus:

```
./configure --enable-mysql --with-mysql-libraries=/usr/lib/mysql
```

make
Make ist ein Dienstprogramm, welches das vom Skript configure erstellte makefile benutzt um den Quellcode zu kompilieren.

make install
Dies bildet den Abschluss der Installation. Die durch den Befehl make erstellten Dateien werden an die entsprechenden Stellen in der Verzeichnisstruktur verteilt.

Wurden alle drei Schritte ohne Fehler ausgeführt, ist die Software erfolgreich installiert und einsatzbereit.

- *Einrichten der Datenbank*

Die Softwarepakete von Snort und ACID enthalten Skriptdateien für das Einrichten der Datenbank mit den entsprechenden Tabellen. Die nachstehende Abbildung zeigt das Datenbankschema für Snort und ACID.

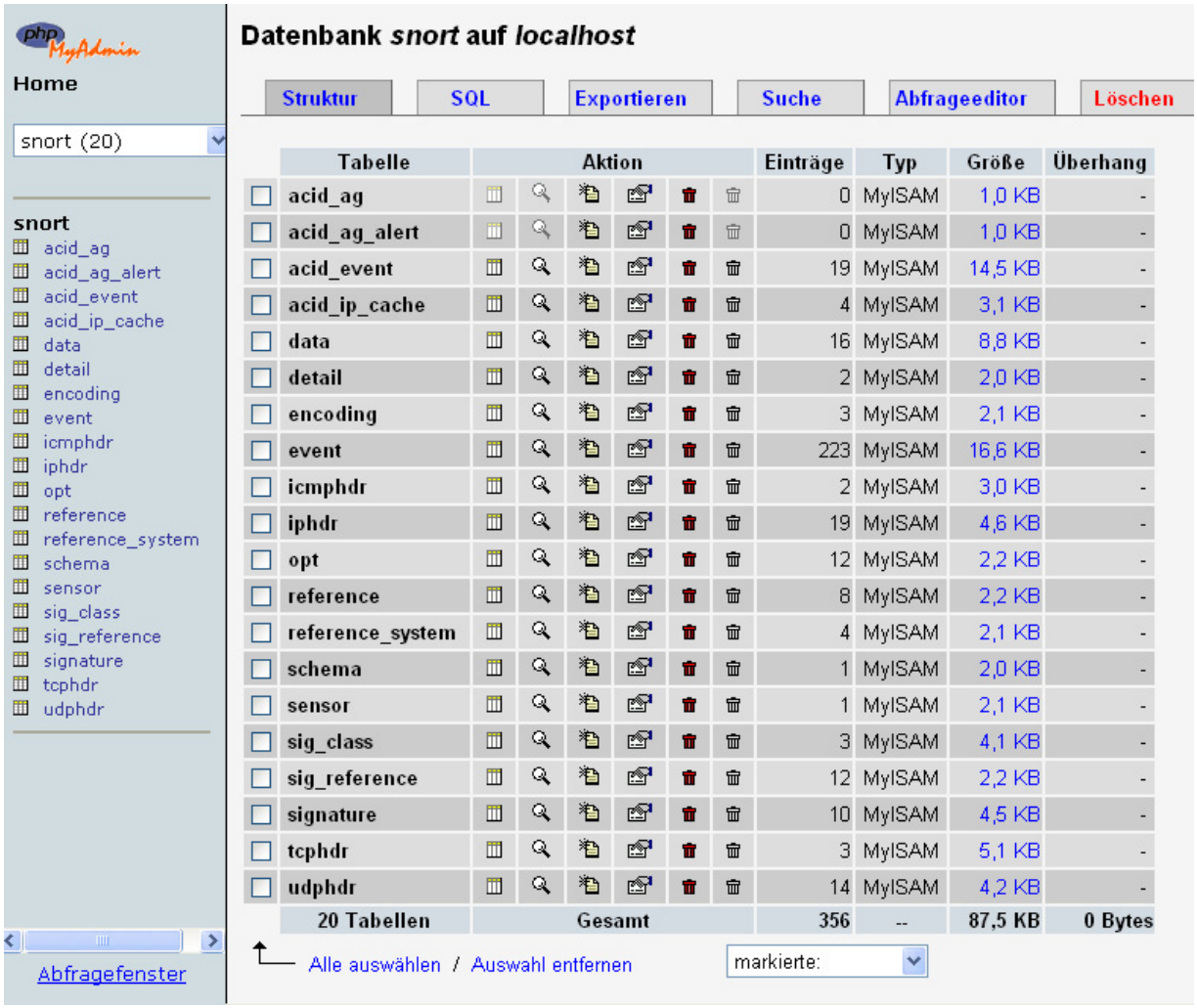


Abbildung 7-16: Einrichten der Datenbank

Jeder Sensor erhält einen eigenen Datenbankuser, der nur Insert-Rechte hat, also nur Daten eintragen, aber aus Sicherheitsgründen nicht löschen darf. Die ACID-Konsole braucht umfangreichere Rechte, weil über dieses Auswertungstool Einträge verwaltet werden können (SELECT, INSERT, UPDATE und DELETE).

In der Standardeinstellung vom MySQL-Server sind Zugriffe nur vom localhost erlaubt, für die Datenbank snort muss der Zugriff vom entfernten Sensor ebenfalls eingerichtet werden.

Bei einem dauerhaften Einsatz eines Netzwerk Intrusion Detection Systems wächst die Datenbank auf eine beachtliche Größe mit tausenden Einträgen an, was die Performance bei Such- und Bearbeitungsfunktionen des Managementsystems negativ beeinflusst. Damit die Alarmdatenbank auf einer verwaltbaren Größe bleibt, gibt es zwei Möglichkeiten. Die einfachste Technik

besteht darin, uninteressante oder ältere Einträge mit Hilfe von Abfragen zu löschen, auch als **trimming** bezeichnet. Die andere Methode ist **archiving**, wobei ältere oder unwichtigere Einträge (z.B. false positives) in eine zweite Archivdatenbank ausgelagert werden. Somit bleibt die Alarmdatenbank schlank und schnell, trotzdem können ältere Vorfälle untersucht werden.

7.9 Konfiguration

Die Datei snort.conf bildet das Zentrum der Konfiguration. Die Konfigurationsdatei teilt sich in vier wichtige Abschnitte:

- *Zuweisung der Variablen*
- *Auswahl der Präprozessoren*
- *Konfiguration der Output Plugins*
- *Auswahl der Regelsätze*

und kann mit jedem Editor beliebig manipuliert werden [siehe Anhang F] .

Für größere NIDS-Installationen mit vielen Sensoren ist es besser die Konfiguration auf mehrere Dateien logisch zu verteilen. Die Datei master.conf bildet die Hauptkonfiguration. Mittels der include-Anweisung werden die Datei local.conf mit den lokalen Variablen (z.B. Interface, Netzwerk, Server) und die Datei rules.conf mit den ausgewählten Regeln eingebunden. Dies ermöglicht eine einfache Anpassung, die eingebundenen Dateien können je nach Standort des Sensors verändert werden.

Abbildung 7-17: Snort Konfigurationsdateien [Spenneberg 03]

Der Snort-Prozess läuft auf Unix/Linux-Umgebungen unter den Rechten des neu angelegten User **snort** in der Usergroup **snort**. Sollte es einem Einbrecher gelingen die Rechte von Snort zu erlangen, ist er trotzdem eingeschränkt und hat somit keine root-Befugnisse (Administratorrechte).

7.10 Wartung

Das Snort-Projekt unterliegt einer ständigen Änderung und Weiterentwicklung wie es im Open Source Bereich üblich ist. In regelmäßigen Abständen sollte ein Update durch den Administrator erfolgen. Die Versionen sind abwärtskompatibel und machen normalerweise keine Probleme. Lediglich der Source-Code wird neu kompiliert, die Konfigurationsdateien behalten ihre Gültigkeit.

Viel wichtiger als das Update von Snort selbst ist die Aktualisierung der Regeln, diese reagieren auf neue Gefahren in der IT. Die wichtigsten Quellen für Regel-Updates sind www.snort.org und www.whitehats.com, da diese Seiten nahezu täglich aktualisiert werden.

Von einer manuellen Update-Methode durch einzelne Kopiervorgänge wird abgeraten, da man bei umfangreichen Regelsätzen schnell die Übersicht verliert. Besser ist die Automatisierung dieses Prozesses durch den **Oinkmaster,** der jedoch Perl-Unterstützung benötigt. Das Perl-Skript vereinfacht das Downloaden und Zusammenführen der Snort-Regeln erheblich. Der Anwender bekommt eine Auswahlübersicht der Regeln, die sich seit dem letzten Update geändert haben.

Die sehr schnelle Aktualisierung der Regeln innerhalb weniger Stunden durch die Open Source Community ist ein besonderer Pluspunkt für Snort.

8 Snort und seine Helfer ACID und Barnyard

Snort alleine bietet bei weitem noch kein vollständiges Netzwerk Intrusion Detection System. Erst durch das Zusammenspiel der drei Hauptkomponenten wie dem Sniffer Snort, der grafischen Konsole ACID und dem Ausgabe-Utility Barnyard entsteht ein NIDS.

8.1 ACID (Analysis Console for Intrusion Databases)

Jeder Administrator kennt riesige Log-Files mit tausenden Meldungen, die unüberschaubar wirken. Niemand hat die Zeit, endlose Seiten peinlichst genau zu prüfen.

Gegen dieses Problem schafft ACID, dass vom CERT-Team entwickelt wurde, Abhilfe. Es ist ein grafisches Managementsystem und basiert auf einigen PHP-Skripts, die auf eine entsprechende Datenbank (z.B. PostgreSQL oder MySQL) mit den Alarmevents zugreifen.

Voraussetzungen für die Verwendung sind ein funktionsfähiger Webserver (z.B. Apache2) mit PHP-Modul und eine lauffähige SQL-Datenbank mit entsprechenden Usern und Berechtigungen. ACID erlaubt eine übersichtliche Verwaltung und detaillierte Darstellung der Ereignisse über einen gewöhnlichen Internetbrowser. Erst durch den Einsatz dieses grafischen Werkzeugs wird ein Open Source IDS brauchbar.ADOdb und PHPlot sind zwei wichtige Bibliotheken, die ACID für den Datenbankzugriff und die Erstellung von Diagrammen braucht (siehe Abbildung 7-15).

[Beale 03]

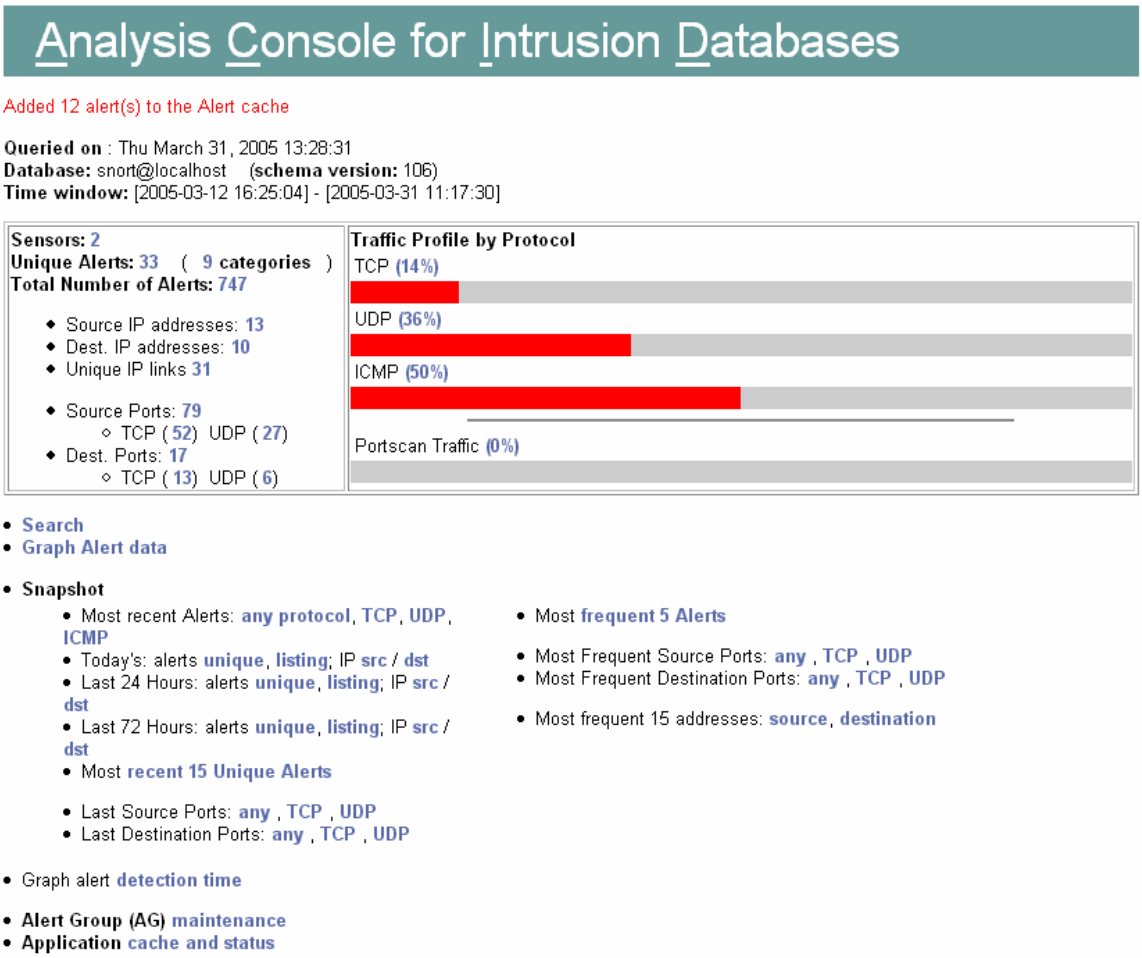


Abbildung 8-1: Übersicht ACID

Die Installation dieser Managementkonsole gestaltet sich relativ einfach. Das Paket wird im Wurzelverzeichnis des Webservers extrahiert. Anschließend wird die Datei acid_conf.php entsprechend der Systemgegebenheiten (Datenbankkonfiguration: User, Passwort, Port etc.) mittels Editor eingerichtet.

Über die eingerichtete Konsole können sämtliche Alarme der Sensoren betrachtet und verwaltet werden. Gezielte Abfragen nach Ports, Zeit, IP-Adressen, Empfänger, Absender usw. stehen dem Sicherheitsadministrator zur Verfügung. Jeder angezeigte Vorfall kann detailliert mit Beschreibung und weiterführenden Links zu Securityspezialisten betrachtet werden.

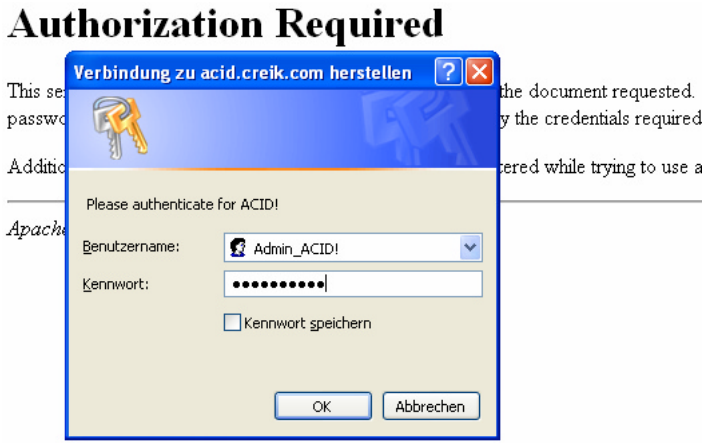


Abbildung 8-2: Authentifizierung für die Konsole

Der Zugriff auf das Managementsystem via Browser wird serverseitig durch Basic Authentication (.htaccess Datei) kontrolliert (Abbildung 8-2).

Die Abbildung 8-3 zeigt ein typisches Event Listing von ACID. Jeder Alarm wird in einer Zeile beginnend mit der zutreffenden Signatur (Fingerabdruck des Angriffs) dargestellt. Besonders wichtig ist die Information welcher Sensor den möglichen Angriff detektierte, da diese ja unterschiedliche Standorte im Netzwerk besitzen.

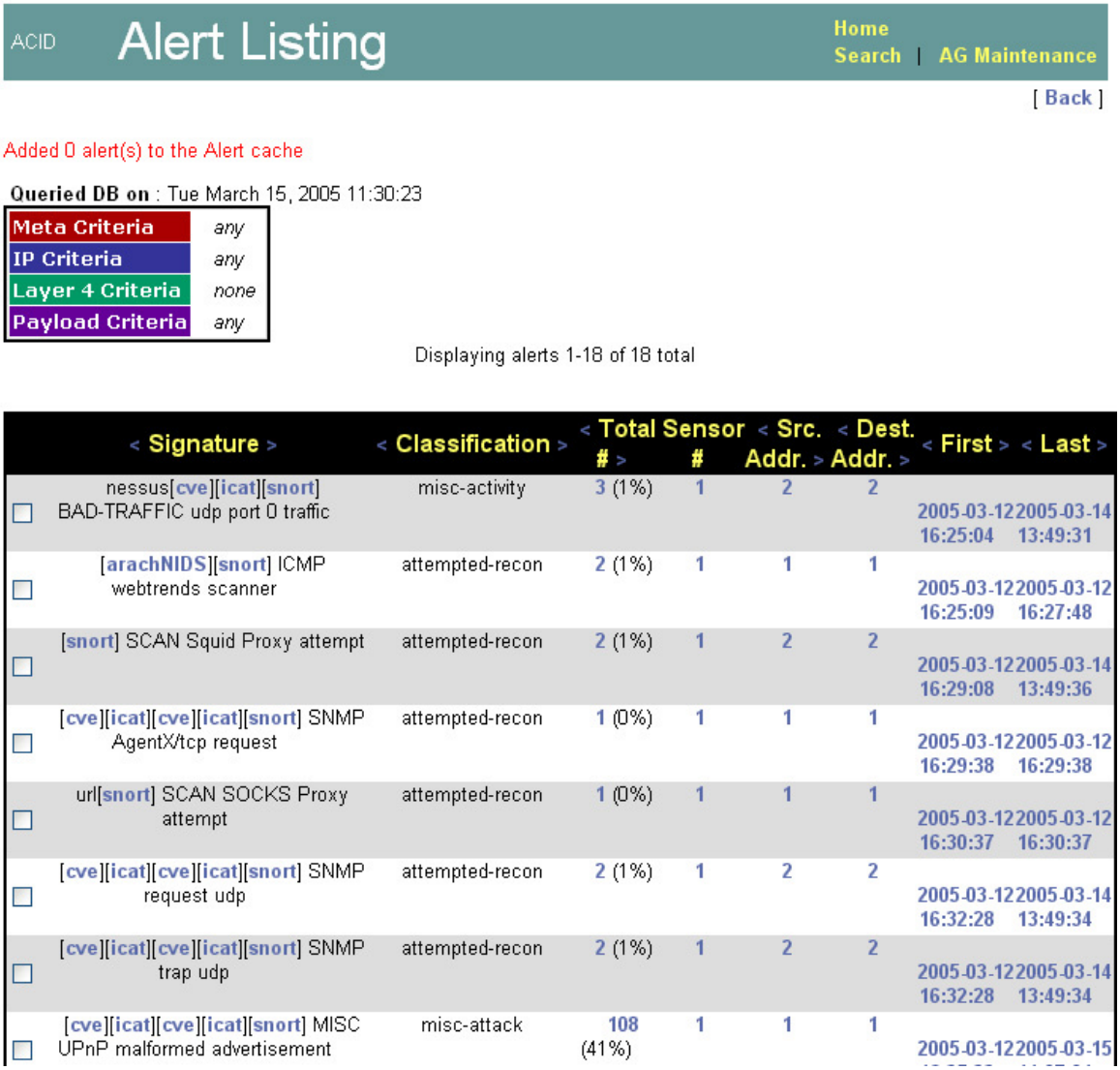
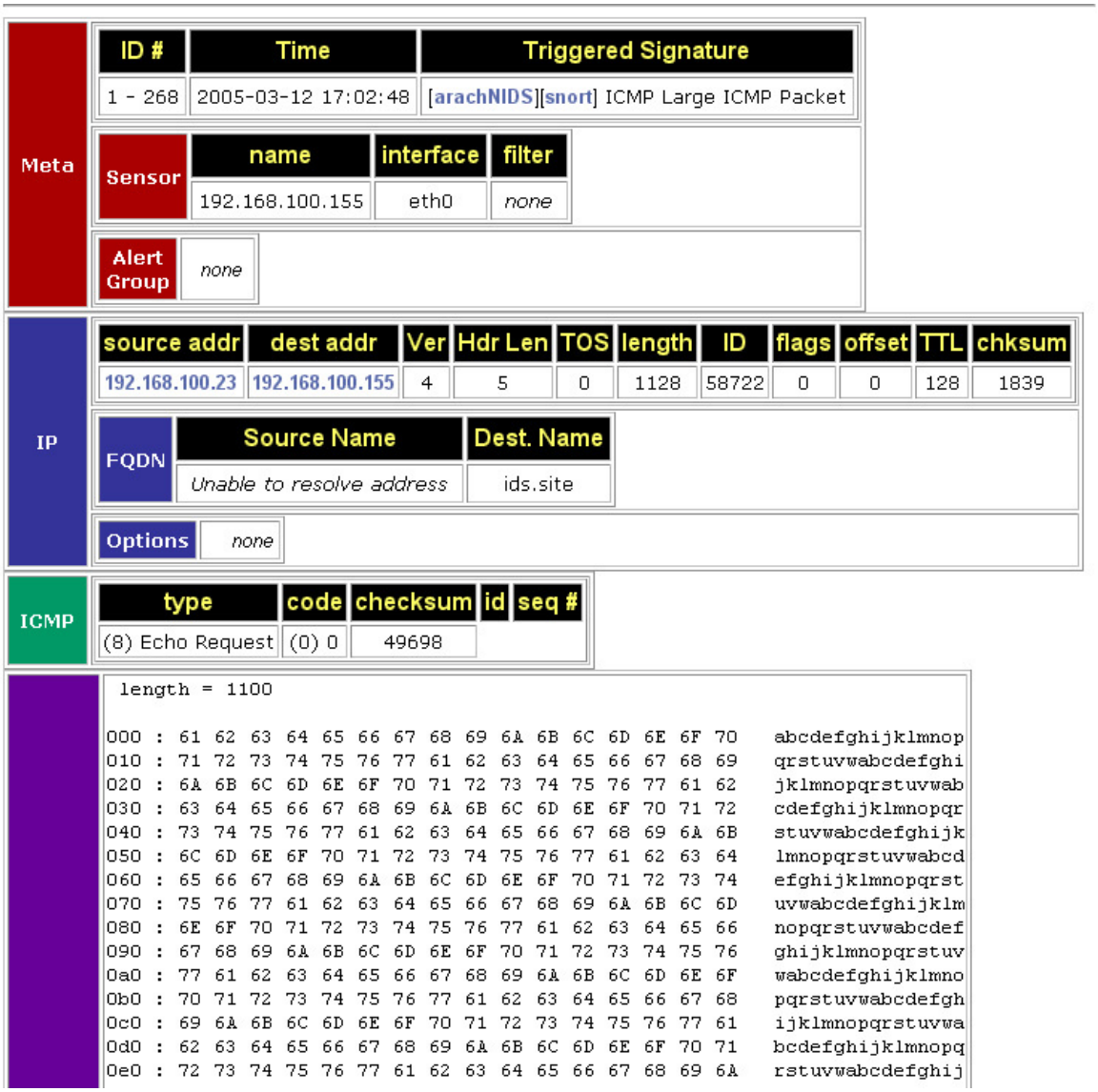


Abbildung 8-3: Alert Listing von ACID

Ein wirklich gelungenes Feature ist die Detailansicht von Paketen, wie sie in der Abbildung 8-4 gezeigt wird. Beginnend mit den Meta-Daten des Intrusion Detection Systems (ID, Zeitpunkt, zutreffende Signatur, Sensor etc.) werden die Paketschichten und der Paketinhalt (Payload) ausführlich aufgeschlüsselt. Das folgende Beispiel zeigt einen rudimentären DoS-Angriff mit sehr großen ICMP Echo-Requests (1100 Byte), der von einer normalen Windows-Workstation initiiert wurde.



Meta

ID #	Time	Triggered Signature
1 - 268	2005-03-12 17:02:48	[arachNIDS][snort] ICMP Large ICMP Packet

Sensor	name	interface	filter
	192.168.100.155	eth0	none

Alert Group	none

IP

source addr	dest addr	Ver	Hdr Len	TOS	length	ID	flags	offset	TTL	chksum
192.168.100.23	192.168.100.155	4	5	0	1128	58722	0	0	128	1839

FQDN	Source Name	Dest. Name
	Unable to resolve address	ids.site

Options	none

ICMP

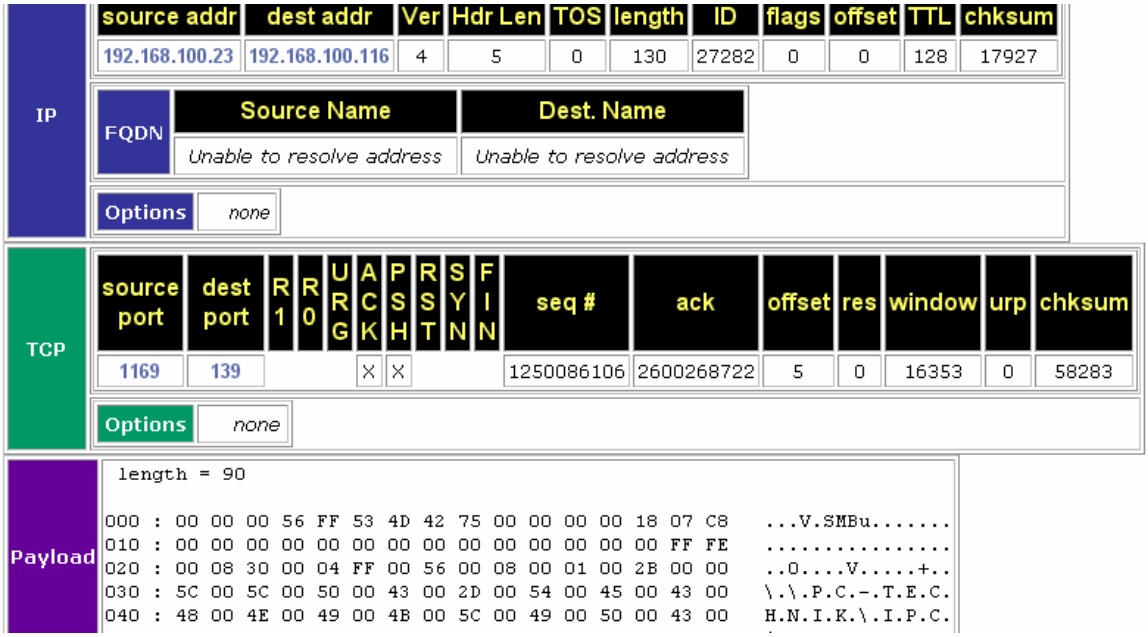
type	code	checksum	id	seq #
(8) Echo Request	(0) 0	49698		

```
 length = 1100

000 : 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70   abcdefghijklmnop
010 : 71 72 73 74 75 76 77 61 62 63 64 65 66 67 68 69   qrstuvwabcdefghi
020 : 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76 77 61 62   jklmnopqrstuvwab
030 : 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70 71 72   cdefghijklmnopqr
040 : 73 74 75 76 77 61 62 63 64 65 66 67 68 69 6A 6B   stuvwabcdefghijk
050 : 6C 6D 6E 6F 70 71 72 73 74 75 76 77 61 62 63 64   lmnopqrstuvwabcd
060 : 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74   efghijklmnopqrst
070 : 75 76 77 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D   uvwabcdefghijklm
080 : 6E 6F 70 71 72 73 74 75 76 77 61 62 63 64 65 66   nopqrstuvwabcdef
090 : 67 68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76   ghijklmnopqrstuv
0a0 : 77 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F   wabcdefghijklmno
0b0 : 70 71 72 73 74 75 76 77 61 62 63 64 65 66 67 68   pqrstuvwabcdefgh
0c0 : 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76 77 61   ijklmnopqrstuvwa
0d0 : 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70 71   bcdefghijklmnopq
0e0 : 72 73 74 75 76 77 61 62 63 64 65 66 67 68 69 6A   rstuvwabcdefghij
```

Abbildung 8-4: Detailansicht von Datenpaketen

Bei Datenübertragungen wo das TCP- oder UDP-Protokoll zum Einsatz kommen werden auch diese Schichten fein aufgegliedert und wie in Abbildung 8-5 dargestellt.



IP

source addr	dest addr	Ver	Hdr Len	TOS	length	ID	flags	offset	TTL	chksum
192.168.100.23	192.168.100.116	4	5	0	130	27282	0	0	128	17927

FQDN	Source Name	Dest. Name
	Unable to resolve address	Unable to resolve address

Options: none

TCP

source port	dest port	R 1	R 0	URG	ACK	PSH	RST	SYN	FIN	seq #	ack	offset	res	window	urp	chksum
1169	139				X	X				1250086106	2600268722	5	0	16353	0	58283

Options: none

Payload

```
length = 90

000 : 00 00 00 56 FF 53 4D 42 75 00 00 00 00 18 07 C8   ...V.SMBu.......
010 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF FE   ................
020 : 00 08 30 00 04 FF 00 56 00 08 00 01 00 2B 00 00   ..0....V.....+..
030 : 5C 00 5C 00 50 00 43 00 2D 00 54 00 45 00 43 00   \.\.P.C.-.T.E.C.
040 : 48 00 4E 00 49 00 4B 00 5C 00 49 00 50 00 43 00   H.N.I.K.\.I.P.C.
```

Abbildung 8-5: Detailansicht der Protokollschichten

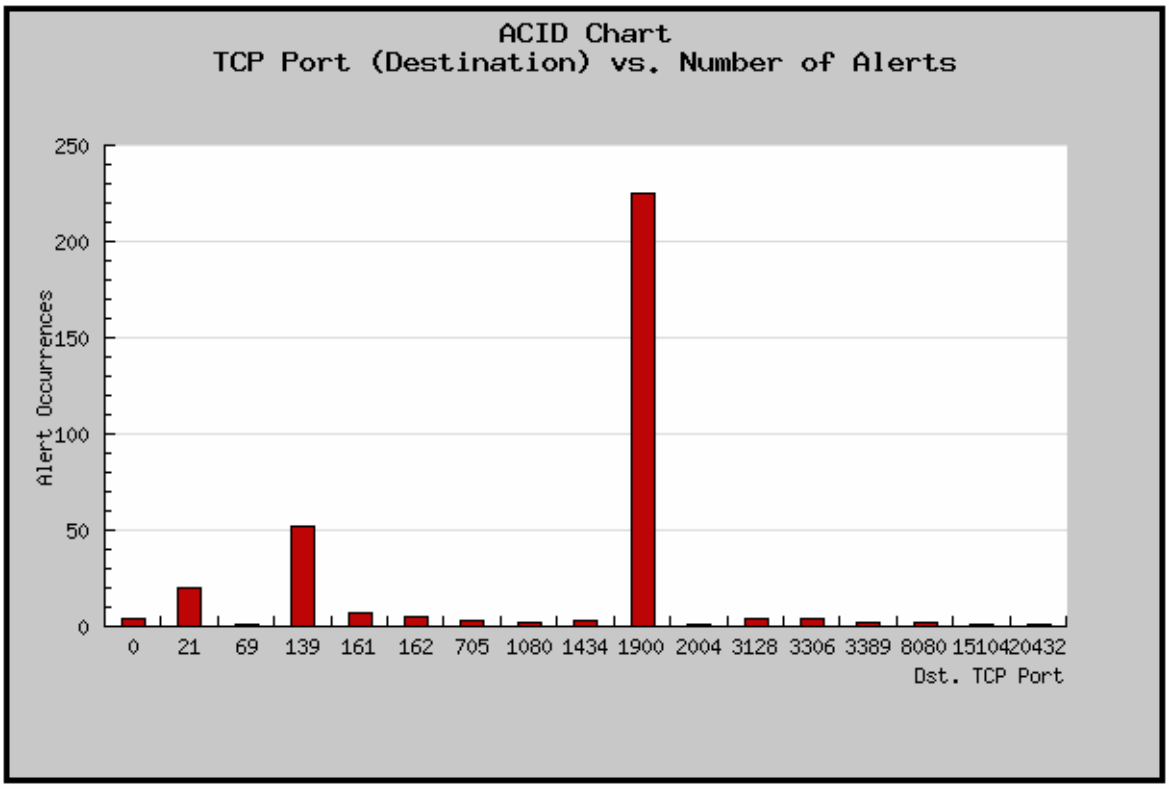


Abbildung 8-6: Grafikfunktion von ACID

Einen sehr guten Überblick bzw. statistische Aussage lässt die Grafikfunktion von ACID zu. Die Daten aus diversen Angriffen, wie zum Beispiel Zeit oder Ziel-Ports, können über die Zeit oder Anzahl der Angriffe in grafischer Form als Balken- oder

Tortendiagramme ausgewertet werden. Der Administrator erkennt auf einen Blick, auf welche Ports oftmals Angriffe abzielen (z.B. FTP, netbios, snmp) oder zu welchen Zeiten die meisten Angriffe stattfinden.

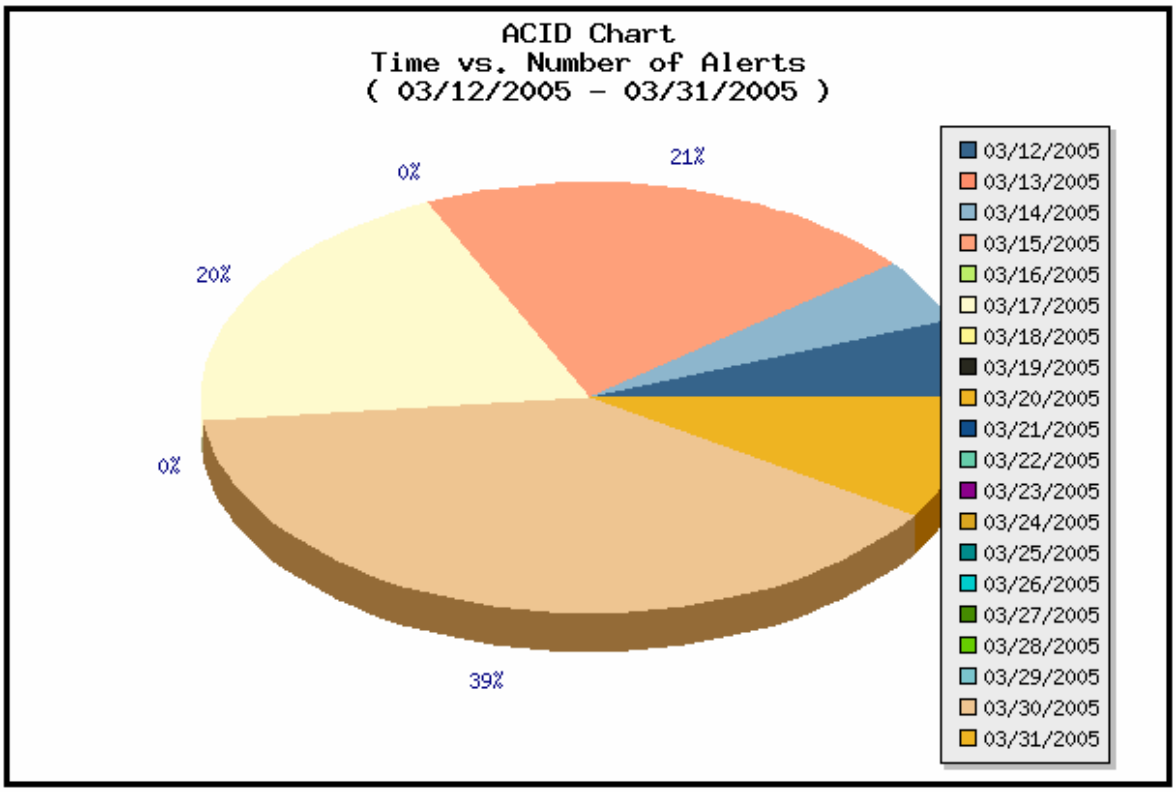


Abbildung 8-7: Zeitliche Darstellung der Angriffe

ACID ist ein unverzichtbares Tool zum Durchsuchen und Analysieren der Alarmevents von Snort, weiters unterstützt es die Verwendung mehrerer Sensoren. Eine Benützung unabhängig von Snort ist möglich, es kann im Zusammenhang mit Linux-Firewalls oder Cisco Access Lists ebenfalls eingesetzt werden.

Ein weiterer Vorteil bei der Verwendung von ACID ist das Erstellen von Alarmgruppen in der Datenbank. Verschiedene Alarme können vom Administrator selbst logisch gruppiert werden. Wird ein bestimmter Einbruchsvorfall untersucht, können alle zugehörigen Events zusammengefasst und gemeinsam ausgewertet werden.

8.2 Barnyard

Snort ist ein gutes und schnelles Werkzeug zur Erkennung von gefährlichen Paketen auf Netzwerkebene. Die Aufbereitung der Pakete in ein für den Menschen lesbares Format ist ein sehr ressourcen-intensiver Prozess, wobei möglicherweise Pakete gedropped werden müssen, denn das Netzwerk wartet nicht auf Snort. Um dieses Problem zu lösen, wurde das Tool Barnyard entwickelt. Es entlastet Snort von den aufwendigen Datenbankschreibvorgängen. Der Snortprozess protokolliert sehr schnell auf einem lokalen Speicherbereich im binären Unified-Format. Barnyard übernimmt quasi die Last, liest die binären Ausgabedateien von Snort und kümmert sich um die vergleichsweise langsame Übertragung zur entfernten Datenbank. In einem Netzwerk mit hohem Verkehrsaufkommen ist Barnyard ein Muss.

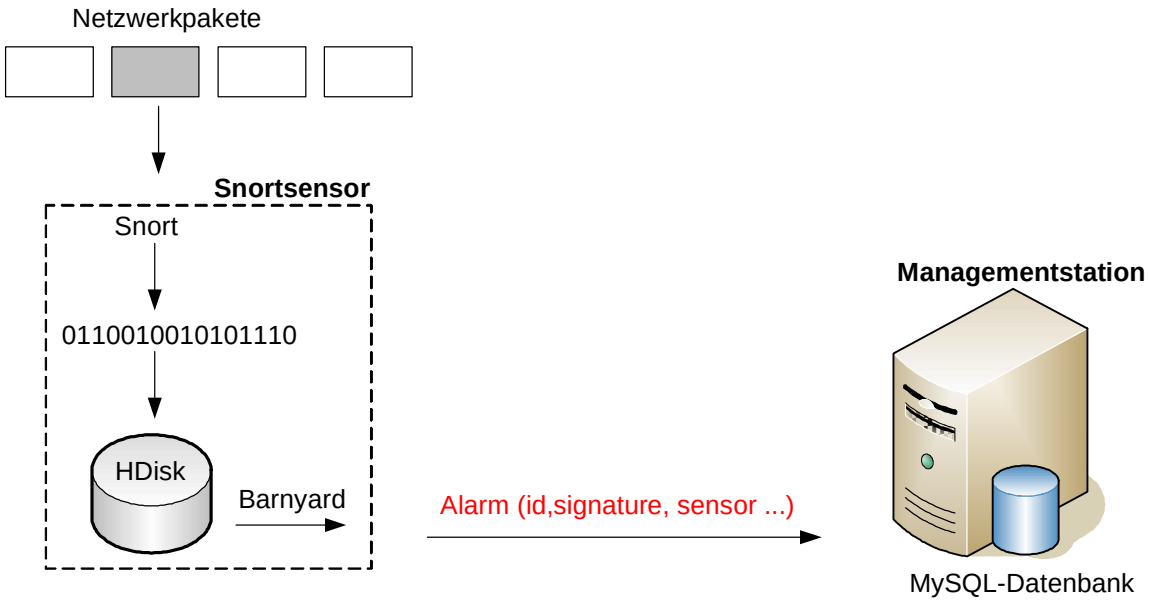


Abbildung 8-8: Zusammenspiel Snort und Barnyard

Die Konfigurationsdatei barnyard.conf [siehe Anhang G] muss im Konfigurationsverzeichnis von Snort platziert sein, da es Abhängigkeiten zu anderen Dateien von Snort gibt (Signaturzuordnung etc.).

9 Praktischer Einsatz eines NIDS mit Snort & Co

Auf den folgenden Seiten wird der Aufbau und reale Einsatz eines Network Intrusion Detection Systems mit zwei Sensoren dokumentiert.

9.1 Beschreibung der Testumgebung

Die folgende Abbildung zeigt ein typisches Netzwerk für ein mittelständisches Unternehmen. Im Netzwerk gibt es einen Windows Small-Business-Server (SBS) der die ca. 10 Clients mit IP-Adressen per DHCP versorgt und gleichzeitig als File- und Mail-Server fungiert. Die Internetanbindung erfolgt über einen regionalen Kabel-ISP mit 2 MBit/s, zum Schutz vor böswilligen „Gesellen“ wird eine Hardware-Firewall (Fortigate 60) eingesetzt.

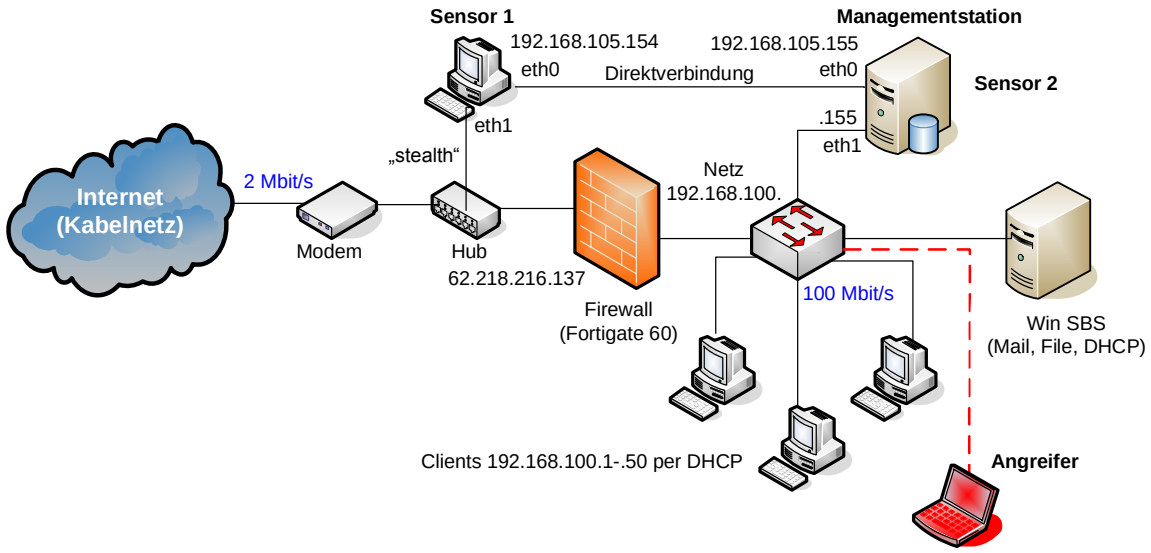


Abbildung 9-1: Teststellung NIDS

Zur Kontrolle von Angriffen und maliziösem Datentraffic kommen zwei Sensoren zum Einsatz. Die Positionierung erfolgt einmal vor der Firewall, um Angriffe von außen zu erkennen, und ein zweites Mal im LAN am Monitorport des Switches. Der Sensor 2 innerhalb des Netzes arbeitet zugleich als Managementstation, bietet also grafischen Zugriff auf die Alarmdatenbank für den Sicherheitsadministrator. In größeren Netzwerken, bzw. bei größerem Budget

sollten Sensor- und Managementfunktion zu Gunsten der Sicherheit und Performance getrennt werden. In kleineren Umgebungen ist es aber aufgrund kleinerer Datenraten und geringeren finanziellen Mitteln durchaus üblich, auf diese Trennung zu verzichten.

Die Kommunikation vom Sensor 1 zur Datenbank der Managementstation erfolgt über eine Direktverbindung mit einem ausgekreuzten Patchkabel in einem eigenen logischen Netz (192.168.105/24), womit eine hohe Sicherheit erreicht wird. Die „sniffende“ Schnittstelle des Sensors 1 ist natürlich stealth geschaltet. Sie ist somit nicht ansprechbar und arbeitet nur passiv. Eine andere Möglichkeit wäre die Übertragung durch die Firewall mittels Portforwarding (z.B. Port 3306 für MySQL), was aber wiederum Bandbreite verschwendet und die Sicherheit keinesfalls verbessert.

Vor einer Implementierung bzw. Anschaffung eines NID-Systems in einem produktiven Netzwerk sollte eine möglichst detaillierte Ist-Analyse der technischen Infrastruktur und Protokolle erfolgen. Dies erleichtert die Planung (Standorte der Sensoren, Managemenstation) und Konfiguration – speziell die Anpassung der Regeln des Intrusion Detection Systems - erheblich [vgl. BSI 02b].

9.2 Eingesetzte Systeme

Auf beiden Rechnern wurde wegen der geringeren Hardwareanforderungen ein SuSE 9.1 Linux-System aufgesetzt. Bereits bei der Installation hat man auf ein „Hardening“ geachtet. Nur die essentiell notwendigen Pakete bzw. Dienste wurden ausgewählt, damit die Systeme keine zusätzlichen Sicherheitslücken aufweisen. Auf beiden Rechnern wurde ein entsprechendes Minimalsystem (ohne grafische Oberfläche, wie Gnome oder KDE) eingerichtet, was die Performance wesentlich steigert. Selbstverständlich wurden im Zuge der Installation alle verfügbaren Sicherheitsupdates eingespielt.

9.2.1 Softwareauswahl

Die Installation der Sensoren im Zusammenhang mit dem Protokollierungstool Barnyard und der MySQL-Unterstützung stellte sich in der Folge als sehr schwierig und zeitraubend heraus. Nur eine bestimmte Kombination der Versionen

[vgl. Tabelle 9-1] und ein manueller Eingriff in das „configure-Skript“ von Barnyard ermöglichten eine fehlerfreie Installation aus dem Quellcode mit MySQL-Unterstützung.

Software	Version	Beschreibung
SuSE	9.1 / Kernel 2.6.4-52	Linux-Betriebssystem
mysql-client	4.0.18	für Datenbankzugriff
mysql-shared	4.0.18	für Datenbankzugriff
mysql-devel	4.0.18	für Datenbankzugriff
libpcap	0.8.1	Paketfilterbibliothek
pcre	5.0	Perlbibliothek f. Snort
snort	2.3.2	Sniffer für NIDS
barnyard	0.2.0	Protollierungstool

Tabelle 9-1: Softwareauswahl Sensor 1

Software	Version	Beschreibung
SuSE	9.1 / Kernel 2.6.4-52	Linux-Betriebssystem
mysql		für Datenbankserver
mysql-client	4.0.18	für Datenbankzugriff
mysql-shared	4.0.18	für Datenbankzugriff
mysql-devel	4.0.18	für Datenbankzugriff
apache2	2.0.49	Webserver
mod_php4	4.3.4	PHP-Unterstützung
acid	0.9.6b23	grafische Konsole
adodb	4.6.1	Datenbankplugin
phplot	5.0	grafische Bibliothek
jpgraph	1.17	grafische Bibliothek
libpng	1.2.8	grafische Bibliothek
gd	2.0.33	grafische Bibliothek
libpcap	0.8.1	Paketfilterbibliothek
pcre	5.0	Perlbibliothek f. Snort
snort	2.3.2	Sniffer für NIDS
barnyard	0.2.0	Protollierungstool

Tabelle 9-2: Softwareauswahl Sensor 2 / Managementstation

Für beide Systeme wurde aufgrund der geringeren Hardwareanforderungen und der Stealth-Konfiguration (unsichtbare Netzwerkinterfaces) SuSE Linux 9.1 als Betriebssystem gewählt.

9.2.2 Verwendete Hardware

Die Internetanspeisung des lokalen Netzwerks erfolgt mit 2 Mbit/s (siehe Abbildung 9-1). Sensor 1 wurde auf einem Compaq Presario Pentium II 700 MHz mit 256 MB RAM realisiert. Die Surecom 100 Mbit/s Netzwerkkarte (eth1) reicht für die Datensammlung an diesem neuralgischen Punkt mit max. 2 MBit/s Datendurchsatz aus.

Der zweite Rechner muss wesentlich mehr leisten, da er den Sensor 2 und die Managementstation darstellt. Außerdem herrscht im LAN ein höherer Datentraffic mit 100 Mbit/s. Hierfür wurde ein MaxData Pentium IV 1,4 GHz mit 512 MB RAM gewählt.

Der Sensor 2 wird an den SPAN-Port des zentralen Switches angeschlossen, damit der gesamte Datenverkehr überwacht werden kann. Leider wird am Monitorport des Switches nur eine Geschwindigkeit von 100 MBit/s unterstützt. Das heißt bei großer Netzwerklast werden Pakete verworfen! Für die Tests im Rahmen der Diplomarbeit ist dieser Punkt kein Problem. Beim Einsatz eines Network Intrusion Detection Systems im Produktionsbetrieb eines Unternehmens muss die Hardware sicherlich entsprechend dimensioniert werden, um seriöse Analysen machen zu können.

- „Unsichtbare“ Sensoren

Die zweite Netzwerkkarte (eth1) des Sensors 1 wird ohne IP-Adresse aktiviert, und ist somit nicht ansprechbar, also unsichtbar im Netzwerk. Man spricht auch von einer „Stealth“-Konfiguration (engl. getarnt, versteckt). Das Interface kann nur Pakete vom Netzwerk aufnehmen, aber keine Daten versenden.

```
ids-sensor:/ # ifconfig eth1 up
ids-sensor:/# ifconfig eth1 –arp up
```

Beim Aufruf von Snort wird die unsichtbare Netzwerkkarte eth1 angegeben:

```
ids-sensor:/usr/local/snort-2.3.2/etc # snort -i eth1 -c snort.conf -D
Running in IDS mode

Initializing Network Interface eth1
OpenPcap() device eth1 network lookup:
        eth1: no IPv4 address assigned

        --== Initializing Snort ==--
Initializing Output Plugins!
Decoding Ethernet on interface eth1
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file snort.conf

+++++++++++++++++++++++++++++++++++++++++++++++++
Initializing rule chains...
,-----------[Flow Config]----------------------
| Stats Interval:  0
| Hash Method:     2
| Memcap:          10485760
| Rows  :          4099
| Overhead Bytes:  16400(%0.16)
`-------------------------------------------
```

Diese Art der Konfiguration ist ein großer Pluspunkt für die Verwendung von Unix/Linux als Plattform. Auf Windows-Systemen kann die Netzwerkkarte nicht unsichtbar konfiguriert werden.

9.3 Testangriffe im lokalen Netz

Nachstehend wurden einige bekannte Angriffs- und Spionagetools im lokalen Netz ausgeführt, um die grundlegende Erkennung des ID-Systems zu testen. Die Ergebnisse sind tabellarisch zusammengefasst. Es wurde ein typischer Angriffszug eines internen Users simuliert. Ein beliebiger Rechner (privater Notebook) wird mit einer Live-CD (Betriebssystem, das direkt von CD bootet; BSD Linux) gestartet, der „böse“ Mitarbeiter erhält per DHCP eine gültige IP-Konfiguration und kann sofort mit Angriffen gegen das IT-System loslegen. Die meisten Testtools wurden aus der Auditor Security Collection[6] verwendet. Die Tests sind kategorisiert nach der Vorgehensweise eines Angriffs. Zuerst wird das Ziel erkundet, danach folgen gezielte Angriffe.

[6] Eine frei verfügbare Sammlung von Security-Tools: http://www.remote-exploit.org

9.3.1 Ping-Versuche

Die Workstation mit der IP-Adresse 192.168.100.25 wird von verschiedenen Plattformen aus gepingt. Alle getesteten Systeme (Windows, SuSE Linux, BSD Linux) wurden anhand der TCP/IP-Implementierung des Betriebssystems erkannt. Beispielmeldung von Snort:

```
ICMP PING BSDtype
```

9.3.2 Scans

- *Nmap mit Front End*

Verwendet wurde Nmap Version 3.7, der wohl bekannteste Scanner in Unix/Linux Umgebungen.

Scanmethode	erkannt	nicht erkannt
Connect Scan	x	
FIN Stealth Scan	x	
SYN Stealth Scan	x	
ACK Stealth Scan	x	
FIN|ACK Stealth Scan	x	
NULL Stealth Scan	x	
XMAS Stealth Scan	x	
TCP Window Scan	x	
UDP Port Scan	x	
IP Protocol Scan	x	
Idle Scan	x	

Abbildung 9-2: Erkannte Scans durch das NIDS

Alle Scan-Methoden gegen Server und Workstations wurden vom ID-System erkannt, allerdings hatte ein Scan oft viele verschiedene Alarmmeldungen zur Folge, aufgrund derer man nicht eindeutig oder nur schwer schließen konnte, dass es sich um einen Portscan handelte.

Folgende Abbildung zeigt das Listing nach einem durchgeführten XMAS Scan mit Nmap [vgl. Kapitel 3.3.1]. Beim XMAS Scan sind alle TCP Flags gesetzt, alle „Kerzen“ leuchten.

< Signature >	< Timestamp >	< Source Address >	< Dest. Address >	< Layer 4 Proto >
[cve][icat][cve][icat][snort] SNMP AgentX/tcp request	2005-04-07 15:08:56	192.168.100.27:33539	192.168.100.25:705	TCP
[cve][icat][cve][icat][snort] SNMP trap tcp	2005-04-07 15:09:02	192.168.100.27:34351	192.168.100.25:162	TCP
[cve][icat][cve][icat][snort] SNMP trap tcp	2005-04-07 15:09:02	192.168.100.27:34367	192.168.100.25:162	TCP
[snort] SCAN Squid Proxy attempt	2005-04-07 15:09:03	192.168.100.27:34635	192.168.100.25:3128	TCP
[cve][icat][cve][icat][snort] SNMP request tcp	2005-04-07 15:09:10	192.168.100.27:35867	192.168.100.25:161	TCP
[cve][icat][cve][icat][snort] SNMP request tcp	2005-04-07 15:09:10	192.168.100.27:35887	192.168.100.25:161	TCP
[arachNIDS][snort] SCAN nmap TCP	2005-04-07 15:09:21	192.168.100.27:58575	192.168.100.25:139	TCP
[arachNIDS][snort] SCAN nmap XMAS	2005-04-07 15:09:21	192.168.100.27:58578	192.168.100.25:40739	TCP
[arachNIDS][snort] SCAN nmap fingerprint attempt	2005-04-07 15:09:22	192.168.100.27:58574	192.168.100.25:139	TCP
[arachNIDS][snort] SCAN nmap TCP	2005-04-07 15:09:22	192.168.100.27:58577	192.168.100.25:40739	TCP

Abbildung 9-3: Alarmlisting bei XMAS Scan

Das System konnte den Versuch von Nmap und die Methode XMAS eindeutig identifizieren. Auch der schwer zu entdeckende Scan-Versuch mit gesetztem FIN Flag wurde ebenfalls gut erkannt.

< Signature >	< Timestamp >	< Source Address >	< Dest. Address >
[arachNIDS][snort] SCAN FIN	2005-04-07 15:19:44	192.168.100.27:52662	192.168.100.25:1466
[arachNIDS][snort] SCAN FIN	2005-04-07 15:19:44	192.168.100.27:52662	192.168.100.25:532
[arachNIDS][snort] SCAN FIN	2005-04-07 15:19:44	192.168.100.27:52662	192.168.100.25:1504
[arachNIDS][snort] SCAN FIN	2005-04-07 15:19:18	192.168.100.27:52661	192.168.100.25:21
[arachNIDS][snort] SCAN FIN	2005-04-07 15:19:18	192.168.100.27:52661	192.168.100.25:53
[arachNIDS][snort] SCAN FIN	2005-04-07 15:19:18	192.168.100.27:52661	192.168.100.25:443
[arachNIDS][snort] SCAN FIN	2005-04-07 15:19:18	192.168.100.27:52661	192.168.100.25:25
[arachNIDS][snort] SCAN FIN	2005-04-07 15:19:18	192.168.100.27:52661	192.168.100.25:23
[arachNIDS][snort] SCAN FIN	2005-04-07 15:19:20	192.168.100.27:52662	192.168.100.25:636
[arachNIDS][snort] SCAN FIN	2005-04-07 15:19:20	192.168.100.27:52662	192.168.100.25:80
[arachNIDS][snort] SCAN FIN	2005-04-07 15:19:20	192.168.100.27:52662	192.168.100.25:554
[arachNIDS][snort] SCAN FIN	2005-04-07 15:19:20	192.168.100.27:52662	192.168.100.25:256
[arachNIDS][snort] SCAN FIN	2005-04-07 15:19:20	192.168.100.27:52662	192.168.100.25:3389
[arachNIDS][snort] SCAN FIN	2005-04-07 15:19:21	192.168.100.27:52661	192.168.100.25:389
[arachNIDS][snort] SCAN FIN	2005-04-07 15:19:21	192.168.100.27:52661	192.168.100.25:113
[arachNIDS][snort] SCAN FIN	2005-04-07 15:19:21	192.168.100.27:52661	192.168.100.25:178
[arachNIDS][snort] SCAN FIN	2005-04-07 15:19:21	192.168.100.27:52661	192.168.100.25:468

Abbildung 9-4: Alarmlisting bei FIN Scan

- *Webserver Scanner Whisker*

Auch spezielle Application-Scanner, wie der Webscanner Whisker bleiben vor dem ID-System nicht verborgen:

```
[snort] ATTACK-RESPONSES 403 Forbidden
```

In der Regel werden Scan-Versuche von Snort sehr gut erkannt, auch wenn die Signaturen oft schwierig zu deuten sind.

9.3.3 Spoofing

- *Cain&Abel*

Für diverse Spoofing Versuche wurde das Allround-Tool Cain&Abel[7] eingesetzt. Mit der Software können vor allem Verbindungen mittels ARP-Spoofing im geswitchten LAN umgeleitet werden, dadurch ist es möglich plain-text Passwörter oder unverschlüsselte Nachrichten (Mails) zu lesen. Brute-Force und Dictionary Attacken auf gesammelte Hash-Werte sind ebenfalls im Programm enthalten.

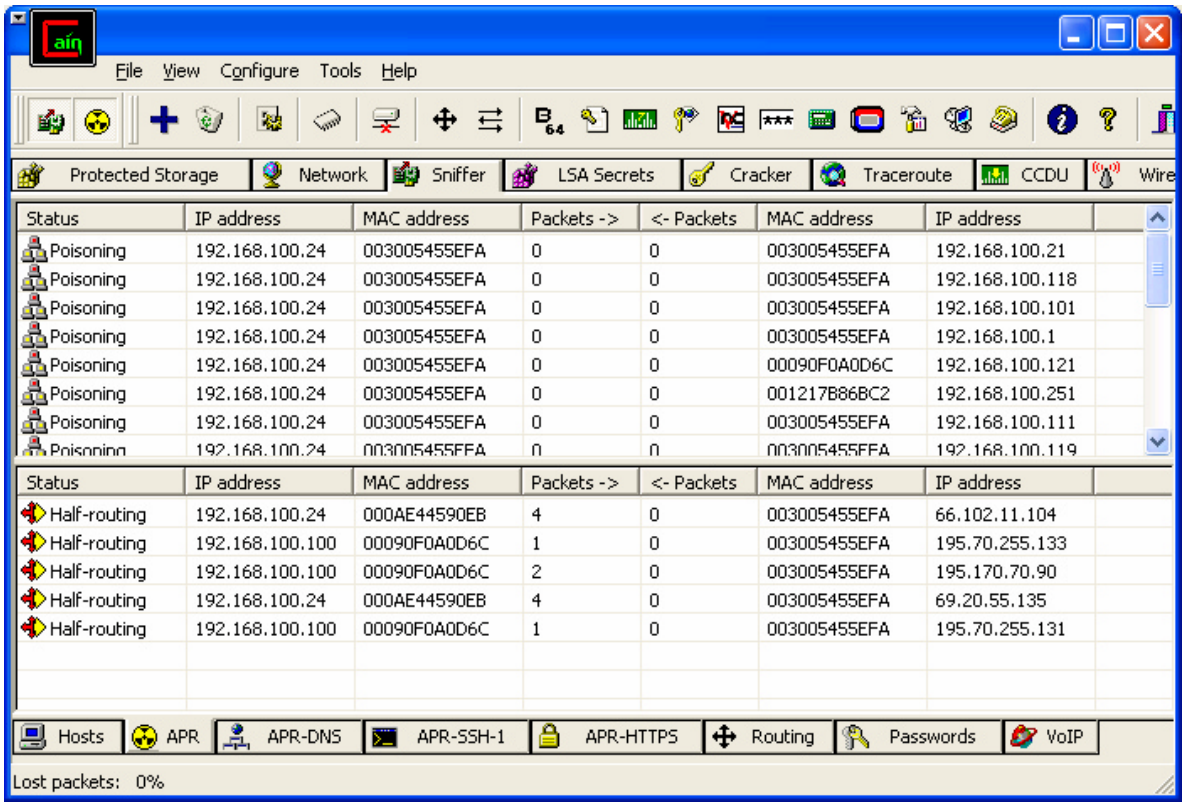


Abbildung 9-5: ARP-Spoofing

Nur wenn der entsprechende **S**nort **P**räp**r**ozessor (spp_arpspoof) in der Konfigurationsdatei snort.conf [siehe Anhang F] aktiviert ist, kann diese gefährliche Angriffsform in geswitchten Netzwerken entdeckt werden [siehe Abbildung 9-6]:

```
preprocessor arpspoof
```

[7] Quelle:http://www.oxid.it

< Signature >	< Timestamp >	< Source Address >	< Dest. Address >
[snort] spp_arpspoof: Etherframe ARP Mismatch SRC	2005-04-18 11:54:36	0.0.0.0	0.0.0.0
[snort] spp_arpspoof: Etherframe ARP Mismatch SRC	2005-04-18 11:54:36	0.0.0.0	0.0.0.0
[snort] spp_arpspoof: Etherframe ARP Mismatch SRC	2005-04-18 11:54:36	0.0.0.0	0.0.0.0

Abbildung 9-6: Erkanntes ARP-Spoofing

9.3.4 Sniffer

- *Dsniff*

Dsniff ist Bestandteil der Auditor Security Collection und sammelt unverschlüsselte Passwörter von Mail oder Webanwendungen am Netzwerkinterface. Diese Art an Passwörter heranzukommen erfolgt passiv und kann daher von einem Snort-Sensor nicht festgestellt werden. Die Methode allein führt jedoch in modernen Netzwerken mit Switches zu keinerlei Erfolg. Um Passwörter zu ergattern, bedarf es zusätzlicher Vorbereitungen, wie dem ARP-Spoofing, das aber dann entdeckt wird.

9.3.5 Überprüfungen und Angriffe mit Nessus

Nessus ist ein leistungsfähiger, aktueller Security-Scanner, der frei verfügbar ist und unter der GNU General Public License steht. Mit dem Scanner können Sicherheitslücken und Schwächen in Computernetzwerken aufgedeckt und auch ausgenutzt werden. Nessus ist ein intelligenter Scanner, der sich bei den Überprüfungen nicht nur blind an den Ports orientiert. Wird zum Beispiel ein Webserver auf dem Port 1410 statt wie üblich auf Port 80 betrieben, so wird dieser trotzdem als Webserver erkannt. Die Client/Server Architektur von Nessus erlaubt eine flexible Installation mit verschiedenen Usern und Rechten. Der Scanner (Server) und die Benutzeroberfläche (Klient) müssen nicht zwingend auf derselben physikalischen Maschine laufen. Außerdem kann der Server von mehreren Benutzern verwendet werden. Jeder Penetrationstest kann durch ausgewählte Plugins modular gestaltet werden. Mit Hilfe von NASL (Nessus Attack Scripting Language) werden von einer großen Open Source Community laufend neue Plugins zur Verfügung gestellt. Die Plugins sind in logische Gruppen, wie Backdoors, Denial of Service, FTP oder Windows etc. gegliedert. [siehe Abbildung 9-8]

DoS-Attacken oder Shell-Angriffe können beispielsweise direkt durch Nessus getestet werden. Obwohl Nessus eigentlich nur testet und die Schwächen nicht direkt ausnützt, gibt es viele gefährliche Plugins, die den zu überprüfenden Rechner beziehungsweise Dienste zum Absturz bringen können! Abschließend liefert Nessus einen umfangreichen Report [siehe Abbildung 9-7] mit den Sicherheitslücken und eine Risikobewertung der entdeckten Exploits.

[www 05]

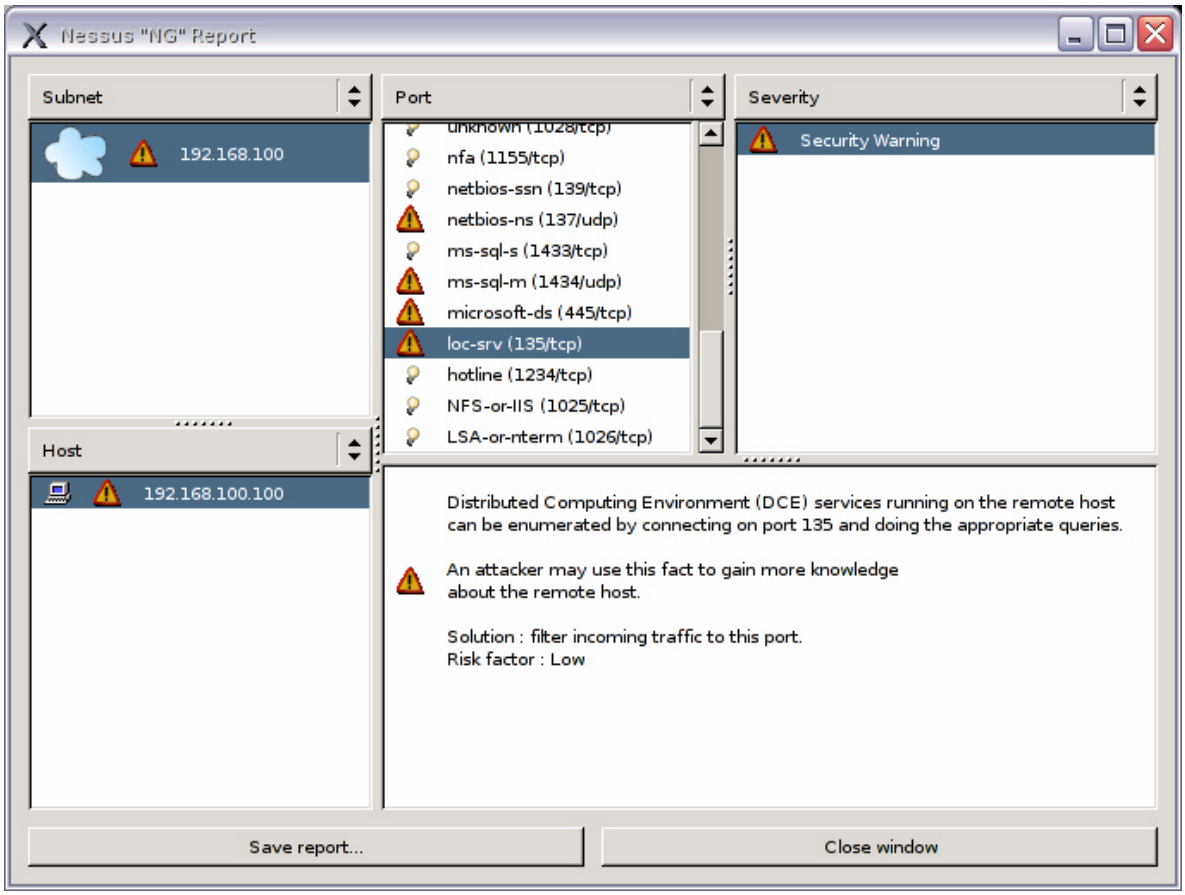


Abbildung 9-7: Nessus Report eines Windows Server System

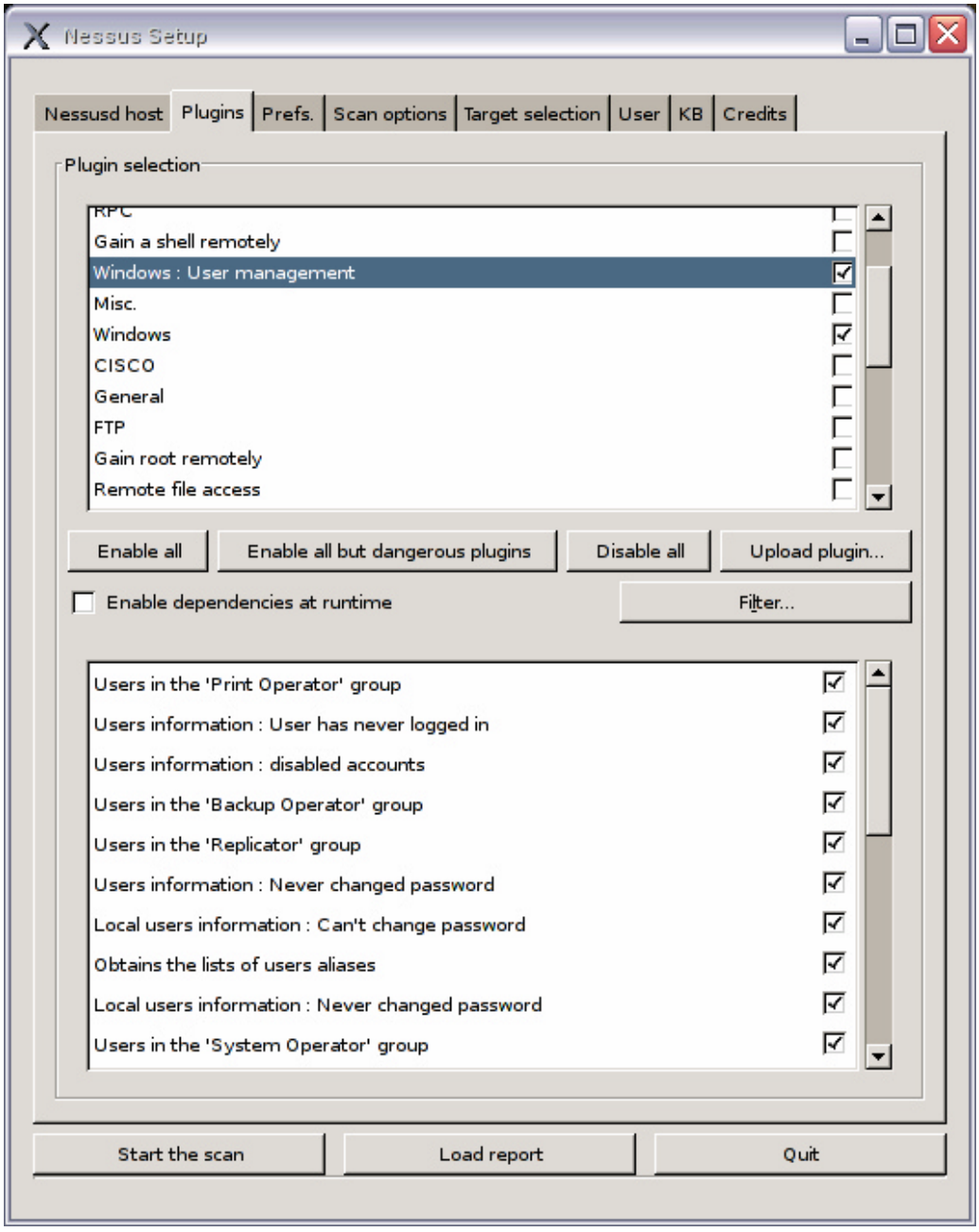


Abbildung 9-8: Nessus Plugin Verwaltung

Abbildung 9-9 zeigt den detaillierte Testaufbau für die Versuche mit Nessus. Zusätzlich zum internen Windows Small Business Server - der die Windows Domäne verwaltet - wird in der demilitarisierten Zone (DMZ) der Fortigate Firewall zusätzlich ein öffentlicher Web- und FTP-Server betrieben. Sowohl der interne Server, als auch der Linux Server in der DMZ stellen begehrte Ziele für den simulierten internen Angreifer dar. Egal welches Objekt getestet (angegriffen) wird, die Datenpakete müssen am ID-System (Mirrorport des Swichtes) vorbei und

sollten somit erkannt werden. Nessus bietet zum Zeitpunkt der Drucklegung dieser Diplomarbeit rund 7700 Plugins[8]. Einfach alle auszuwählen wäre schon aus Gründen der verfügbaren Zeit und der notwendigen Verifizierung unmöglich, und auch nicht sinnvoll. Sehr viele dieser Erweiterungen von Nessus sind auf exakt spezifizierte Systeme (z.B. Cisco Router, Mac OS X) oder Applikationen (z.B. Apache, PostgreSQL) ausgerichtet und daher nicht überall geeignet.

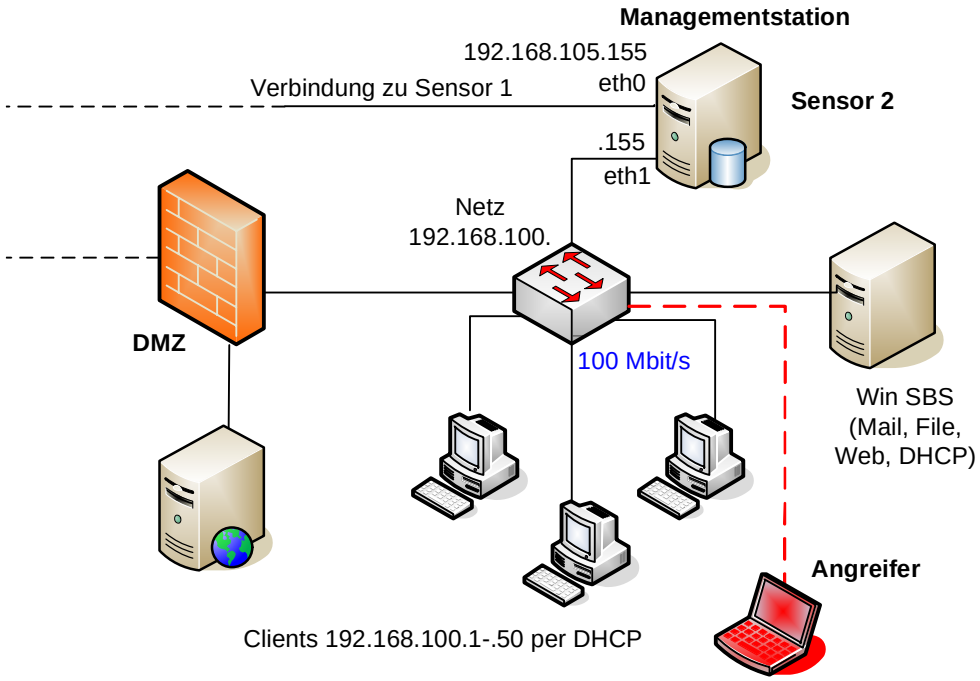


Abbildung 9-9: Detaillierter Testaufbau

Auf beiden Serversystemen (Windows und Linux) wurden einige Angriffsmuster exemplarisch durchgeführt und die Erkennung durch das ID-System überprüft.

Bei der Erkennung wurde zwischen drei Fällen unterschieden [siehe Tabelle 9-3]:

Ja: genau jener Angriff wurde durch eine oder mehrere Signaturen festgestellt.

Nessus direkt: der Angriff wurde entdeckt, und auch das ausführende Tool Nessus wurde aufgrund seiner Eigenheiten durch Signaturen festgestellt.

Indirekt: der Angriff als solcher wurde nicht direkt erkannt, sondern anhand von ungewöhnlichem Traffic oder Portanfragen; keine spezielle Signatur war zutreffend.

[8] http://www.nessus.org/plugins/index.php?view=all

Nein: das Intrusion Detection System nahm keine Kenntnis vom stattgefundenen Angriff

Angriffsgruppe	Plugin (Angriff)	Beschreibung	Erkennung			
			ja	Nessus direkt	indirekt	nein
Backdoors	Bugbear worm	anhand offener Ports werden Passwörter übermittelt			x	
	IIS Possible Compromise	prüft Filezugriff auf Webserver	x			
CGI abuses	ASP Upload Vulnerability	File Uploads durch ASP Lücke			x	
	Check f. dangerous IIS default sites	versucht Files auszulesen	x			
	IIS directory traversal	Attacke mit Unicode in URL	x			
	Outlook Web anonymous access	anonymer Zugriff auf Webmail			x	
Denial of Service	http method overflow	ungültiger http-request erzeugt overflow		x		
	IIS Frontpage	DoS Angriff bzgl. Frontpage-Erweiterungen		x		
	Ping of Death	DoS durch ungültige Fragmentierung	x			
Gain root remotely	http cookie overflow	lange cookie-names und große Werte erzeugen overflow		x		
	http header overflow	extra langer http-Header		x		
	MS RPC Interface Buffer Overrun	Netbios Angriff		x		
General	http server type, version	fragt Typ und Version ab			x	
	MS Exchange Public Folders Information Leak	nutzt bekannte Exchange Schwäche			x	
Misc	Web Server Cross Site Scripting	probiert gefährliches Cross Site Scripting am Webserver		x		
Remote file access	Test HTTP dangerous methods	Prüft ob PUT und DELETE Aktionen möglich sind	x			
SMTP Problems	Exchange Remote Buffer Overflow	bringt Exchange-Dienst zum Absturz			x	
	Mail relaying	prüft ob Server anonyme Mails weiterleitet (Spamgefahr)			x	
Windows	Word/Excel may allow arbitrary code to run	Angreifer kann Befehle über die Programme ausführen	x			
	SMB log in as users	verschiedene Anmeldeversuche	x			
	Win Media Services Remote DoS	DoS-Attacke gegen Media Services	x			
	SMB share access	Prüft Zugriff auf alle Freigaben	x			
	SMB Registry Autologon	erlaubt Angreifer Anmeldung auf Remote-Host	x			

Tabelle 9-3: Nessus Testbericht Windows Server

Angriffsgruppe	Plugin (Angriff)	Beschreibung	Erkennung			
			ja	Nessus direkt	indirekt	nein
Denial of Service	Infinite HTTP request	killt Webdienst durch Vielzahl von Anfragen	x			
	HTTP negative content length DoS	beendet Dienst durch ungültige HTTP requests		x		
	HTTP method overflow	DoS durch HTTP method field	x			
	Too long basic authentication	langes Authentifizierungsfeld		x		
	HTTP unfinished linde deal	URL-Angriff		x		
CGI abuses	Websphere Cross Site Scripting	Script Angriff		x		
	Apache Remote Command Execution	Versuch, Befehle durch Apache Dienst auszuführen		x		
	PHP Nuke opendir	erlaubt Files auszulesen		x		
	Apache Directory Listing	lest wwwroot aus		x		
	PHP.EXE / Apache Vulnerability	Remote user kann PHP Binaries ausführen		x		
Default Unix Accounts	Default password for root	testet Login mit root root				X
	Unpassworded root account	testet Login ohne Passwort				X
RPC	automountd service	prüft ob der Dienst läuft	x			
	nfsd service	ermöglicht Befehlsausführung	x			
Gain a shell remotely	ssh overflow	Bufferoverflow über ssh			x	
	Apache chunked encoding	nützt Bug von Apache	x			
Misc	Apache Connection Blocking DoS	DoS Attacke auf Apache	x			
	Directory Scanner	liest Verzeichnisse vom Webserver	x			
	Web Server Cross Site Scripting	prüft Anfälligkeit auf Cross Site Scripting	x			
	Brute Force Login (Hydra)	probiert Login Kombinationen	x			
General	OS Identification	stellt Betriebssystem fest	x			
	Apache Auth Module SQL Insertion Attack	Security-Lücke bei SQL	x			
FTP	ST FTP Traversal	Versuch aus ftp-root ins Dateisystem zu wechseln	x			
	FTP CWD ~root	Versuch als root anzumelden	X			
	FTP User, Pass overflow	Server beendet Verbindungen bei zu langen Befehlen (DoS)	X			
Gain root remotely	Too long POST command	Codeausführung möglich		X		
	HTTP Cookie overflow	Rootrechte durch Overflow				
	Too long URL	killt Dienst mit langen URLs		X		
	HTTP header overflow	killt Dienst mit langen Header		X		
	Too long authorization	durch langen String können Befehle ausgeführt werden		X		
Remote file access	NFS export				X	
	HTTP dangerous methods	prüft PUT, DELETE Aktionen		X		
Backdoors	Stacheldraht Detect	Trojaner	X			
	BackOrifice	sehr bekannter Trojaner	X			

Tabelle 9-4: Nessus Testbericht Linux Server

Besonders bei web-basierten Angriffen glänzte Snort geradezu. Es konnte den Security-Scanner Nessus oft direkt identifizieren. Nur bei den Login-Versuchen mit Defaultpasswörtern versagte Snort. Sonst wurde jede Überprüfung erkannt, was für die guten und aktuellen Signaturen von Snort spricht.

9.3.6 Filesharing Software

- *e-Mule*

Ebenfalls getestet wurde die Erkennung von Filesharing Software bei Peer-to-Peer Netzen. Vielen Arbeitgebern und Netzwerkbetreuern sind diese Anwendungen ein Dorn im Auge, da Bandbreite verschwendet wird und die Gefahr von Viren und Trojanern erheblich steigt.

< Signature >	< Timestamp >	< Source Address >	< Dest. Address >
url[snort] P2P eDonkey transfer	2005-04-07 14:50:14	62.218.216.137:33125	62.241.53.2:4242
url[snort] P2P eDonkey transfer	2005-04-07 14:50:29	62.241.53.2:4242	62.218.216.137:33125
url[snort] P2P eDonkey transfer	2005-04-07 14:50:29	62.241.53.2:4242	62.218.216.137:33125
url[snort] P2P eDonkey transfer	2005-04-07 14:50:29	62.218.216.137:33126	62.241.53.16:4242
url[snort] P2P eDonkey transfer	2005-04-07 14:50:30	62.218.216.137:33127	62.241.53.17:4242

Abbildung 9-10: Erkennung von Filesharing Software

Anhand des enstprechenden Rule-Sets [siehe Anhang F] erkennt Snort die P2P Software und liefert dem Sicherheitsadministrator wichtige Informationen.

9.4 Ergebnisse auf der WAN-Seite

Die dokumentierten Ergebnisse auf der WAN-Seite (Wide Area Network) sind **nicht** durch eigene Testangriffe produziert. Es handelt sich um Echtdaten und Angriffsversuche von einem gewöhnlichen Internet-Kabel-Breitbandanschluss.

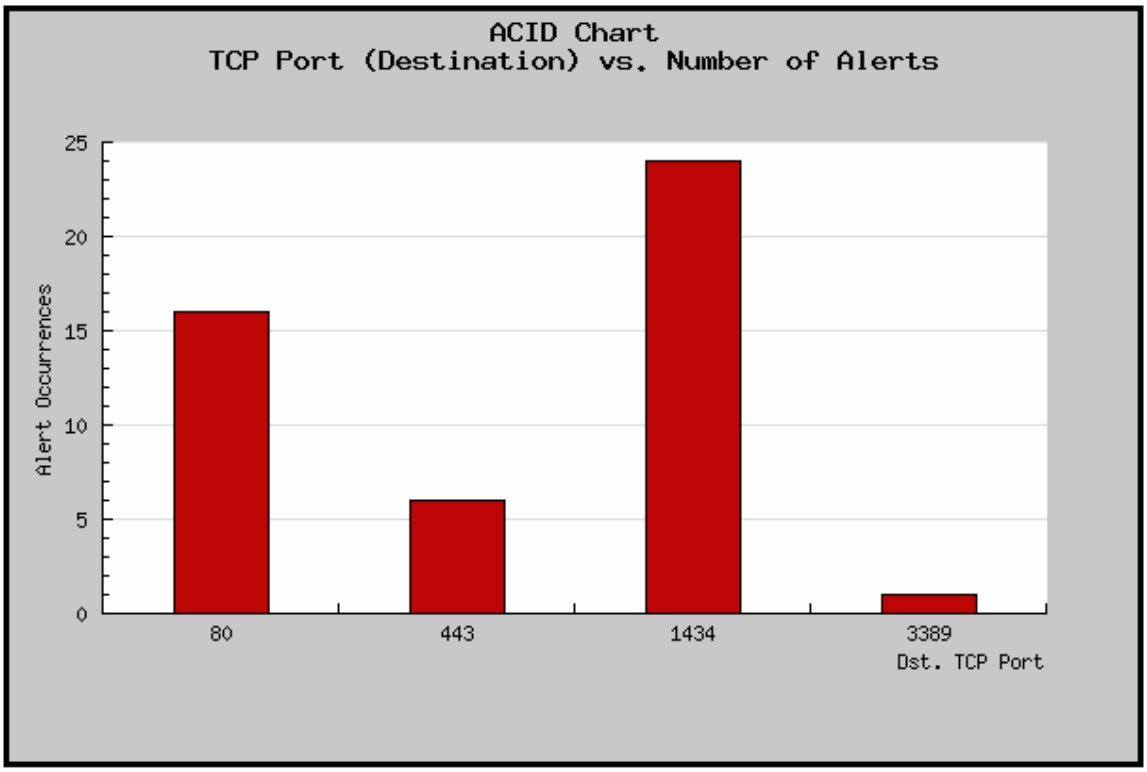


Abbildung 9-11: Typische Angriffsports an der WAN-Seite

Die Informationen, welche durch das Intrusion Detection System vor der Firewall gewonnnen werden, sind für den Administrator - zum Absichern seiner Anwendungen - sehr wertvoll. Die gelungene Grafikfunktion von ACID gibt einen schnellen Überblick, welche Ports für Angreifer interessant sind, und welche Dienste daher besonders sorgfältig abgesichert werden müssen.

Abbildung 9-11 zeigt die Zielports sämtlicher Alarmvorfälle innerhalb eines Tages. Den Spitzenreiter stellt hierbei der Microsoft SQL Server auf Port 1434 neben dem Webserver auf Port 80 dar. Auch bei der SSL Application auf Port 443 gibt es bekannte Schwachstellen, die zu einem Bufferoverflow führen können. Auch auf dem Remote Desktop Port 3389 wurde ein Alarm verzeichnet. Die Abbildung 9-12 zeigt das zugehörige Alarmlisting mit der Klassifizierung und Auftrittshäufigkeit der Alarme.

< Signature >	< Classification >	< Total # >	< Sensor # >	< Src. Addr. >	< Dest. Addr. >	< First >	< Last >
[bugtraq][cve][icat][snort] WEB-CGI redirect access	attempted-recon	1 (2%)	1	1	1	2005-04-04 15:15:13	2005-04-04 15:15:13
[bugtraq][snort] WEB-MISC weblogic/tomcat .jsp view source attempt	web-application-attack	1 (2%)	1	1	1	2005-04-04 15:19:25	2005-04-04 15:19:25
nessus[snort] WEB-FRONTPAGE /_vti_bin/ access	web-application-activity	1 (2%)	1	1	1	2005-04-04 15:20:55	2005-04-04 15:20:55
[bugtraq][cve][icat]url[snort] MISC MS Terminal server request	protocol-command-decode	1 (2%)	1	1	1	2005-04-04 15:24:58	2005-04-04 15:24:58
[arachNIDS][snort] ICMP PING NMAP	attempted-recon	6 (10%)	1	4	2	2005-04-04 16:12:57	2005-04-04 18:47:33
[snort] WEB-CGI calendar access	attempted-recon	10 (17%)	1	1	1	2005-04-05 06:25:54	2005-04-05 06:25:59
[bugtraq][bugtraq][cve][icat]nessusurl[snort] MS-SQL Worm propagation attempt	misc-attack	8 (13%)	1	7	2	2005-04-04 17:41:22	2005-04-05 06:05:34
[bugtraq][bugtraq][cve][icat]nessusurl[snort] MS-SQL Worm propagation attempt OUTBOUND	misc-attack	8 (13%)	1	7	2	2005-04-04 17:41:22	2005-04-05 06:05:34
[bugtraq][cve][icat]nessus[snort] MS-SQL version overflow attempt	misc-activity	8 (13%)	1	7	2	2005-04-04 17:41:22	2005-04-05 06:05:34
[cve][icat][snort] WEB-CGI icat access	web-application-activity	1 (2%)	1	1	1	2005-04-04 20:06:18	2005-04-04 20:06:18
nessus[snort] WEB-MISC robots.txt access	web-application-activity	2 (3%)	1	2	1	2005-04-04 20:39:27	2005-04-05 03:36:32
[bugtraq][cve][icat]url[snort] WEB-MISC PCT Client_Hello overflow attempt	attempted-admin	6 (10%)	1	1	3	2005-04-04 21:14:14	2005-04-05 06:23:02
[snort] ICMP Destination Unreachable Communication Administratively Prohibited	misc-activity	7 (12%)	1	1	2	2005-04-05 04:25:08	2005-04-05 05:56:40

Abbildung 9-12: Alarmlisting auf der öffentlichen Seite (WAN)

9.5 Fazit

Die Erkennungsrate der Signaturen von Snort und die Performance im Zusammenhang mit dem Tool Barnyard sind sehr gut und stehen einer kommerziellen Appliance-Lösung qualitativ nicht nach. Sehr schwierig ist allerdings die Abstimmung auf die herrschende Netzwerkumgebung mit all ihren Protokollen und Eigenheiten, damit der normale Betrieb nicht zu einer „Überschwemmung“ der Datenbank führt. Verwendet man sehr viele Regeln, so wird nahezu jeder Angriff detektiert. Setzt man zu wenige oder die falschen Signaturen ein, werden Intrusions nicht festgestellt. Hier gilt es, durch Erfahrung und Wissen einen guten Kompromiss zu finden, wobei professioneller Support durch den Hersteller eines gekauften IDS sicherlich sehr hilfreich ist.

Interessant ist die Tatsache, dass sich die Anzahl der Alarme auf den beiden Sensoren erheblich unterscheidet. Während vor der Firewall vergleichsmäßig selten verdächtige Pakete analysiert werden, wird der im LAN stehende Sensor nahezu überflutet. Natürlich hängt diese Tatsache mit dem hohen Verkehrsaufkommen (100 MBit/s statt 2 MBit/s) und den vielen Diensten im lokalen Netz zusammen die zu einer hohen false positives Rate (Fehlalarmquote) führen. Die Abstimmung des internen Sensors - bezogen auf die Snort-Regeln - und die Auswahl der Präprozessoren sind echte Herausforderungen, die über Wochen beschäftigen und viel Wissen erfordern. Erst dann werden die angezeigten Daten aussagekräftig und glaubwürdig. Davor kämpft man mit einer Vielzahl von Falschmeldungen aufgrund der hohen Sensibilität des Systems. Hunderte Alarme pro Stunde sind zu Beginn der NIDS-Implementierung keine Seltenheit. Trotzdem dürfen nicht zu viele Regelsätze ausgenommen werden, sonst werden tatsächliche Verletzungen in der IT-Landschaft nicht entdeckt.

Die Aktualität der Snort Versionen und des Regelsatzes ist von ganz entscheidender Bedeutung. Während der Testphase wurden kurze Zeit die Versionen 2.3.2 (letzte verfügbare) und Version 2.1.0 (veraltet) eingesetzt. Die Unterschiede in der Erkennungsrate am selben Netzwerksegment bei den gleichen Testmethoden waren gewaltig.

Nicht alle, aber viele der getesteten Angriffe hatten das Potenzia,l ein System lahm zu legen. Beide Zielobjekte (Server) werden regelmäßig mit Patches aktualisiert und wurden durch keinen der Angriffe in ihrem laufenden Betrieb gefährdet oder gestört. Diese Tatsache zeigt, wie wichtig und nützlich das Updaten von Systemen für die Sicherheit ist.

Dass Open Source Software nicht nur Vorteile besitzt, hat sich während der Installation deutlich gezeigt. Es ist ein sehr großer zeitlicher Aufwand, das ganze System (verteilte Sensoren und Managementstation) zu installieren (Paketabhängigkeiten) und zu konfigurieren (Anpassung der Dateien). Eine Vielzahl von Stunden muss für das Studium von Büchern, Manuals und speziellen Security-Foren aufgebracht werden. Weiters sind für eine Umsetzung auf Basis von Linux gute Vorkenntnisse im Betriebssystem (Prozesse, Dateisystem, Rechte, Editor etc.) und in Netzwerken (Adressierung, Protokolle) unabdingbar, sonst wird

es schnell zu einer „Mission Impossible“. Beim Kauf einer Appliance-Lösung von bekannten Herstellern ist eine schnelle Installation und professioneller Support natürlich inbegriffen, was viel Zeit und Nerven erspart.

Trotzdem, wenn man die Mühe auf sich nimmt und ein Open Source Netzwerk ID-System aufzieht, wird man auch belohnt. Man lernt viel über Netzwerkanalyse dazu und erhält wichtige Daten über das eigene Netz ohne großen Kostenaufwand. Besonders die grafische Konsole ACID ist ein echtes Highlight, welches die Alarmdaten der Datenbank detailliert und übersichtlich aufbereitet.

10 Schwächen von IDS

Das ID-System selbst ist im eigentlichen Sinn nur Beobachter und kann somit nur feststellen. Es benötigt somit weitere externe Netzwerk-Komponenten wie Router oder Firewalls zum Reagieren auf Angriffe. Viele Verbindungen von Endgeräten werden aktiv überwacht, damit ist das System aber auch selbst anfällig gegen Denial-Of-Service Attacken. Unangenehme Begleiterscheinung beim Einsatz von IDS sind häufige Fehlalarme (false positives), gerade am Beginn einer Implementierung, aber auch im laufenden Betrieb. Es kann also passieren, dass der Administrator zu vergeblichen Verfolgungsjagden nach nicht existierenden Hackern verleitet wird. Ist das ID-System zu empfindlich eingestellt, werden vielleicht tatsächliche Angriffe im Lauf der Zeit vom Sicherheitsbeauftragten nicht mehr ernsthaft wahrgenommen.

In geswitchten Umgebungen muss ein Mirrorport eingerichtet werden, dabei kann es aber zu Problemen bei der Verarbeitung kommen, da sehr große Datenmengen in kürzester Zeit analysiert werden müssen. Häufig kommt es zur Überlastung durch unzureichende Dimensionierung des Systems. Manche Angriffsmethoden oder Protokollimplementierungen wie IP-Fragmentierung (Pakete werden für die Übertragung zerteilt und machen die Erkennung des Inhalts schwierig) oder Distributed Denial-of-Service (DoS-Attacke von mehreren Systemen gleichzeitig) bereiten speziell bei Network ID-Systemen Probleme. Hostbasierte ID-Systeme zeigen dafür bei sich ständig ändernden Datenbeständen Schwächen.

Weitere Probleme beim Einsatz von ID-Systemen sind:

- *hohe Investitionskosten*
- *hoher Wartungsaufwand zur Erhaltung der Atkualität*
- *verschlüsselter Datenverkehr*
- *Datenschutz [siehe Kapitel 6]*

Fehlendes Know-How des Sicherheitsadministrators kann sicher auch zu einem Problem führen. Da das Gebiet in der Informations-Technologie noch relativ jung ist, gibt es auch keine herstellerübergreifenden Standards für Kommunikations-protokolle zwischen den ID-Sensoren und dem Managementsystem. Ansätze um Standards zu schaffen gibt es aber bereits:

- *Common Intrusion Detection Framework (CIDF) Projekt*
- *Intrusion Detection Working Group (IDWG)*

[vgl. BSI 02a]

11 Kosten und Nutzen, die Management-Entscheidung

Um nicht an der Realität vorbei zu gehen, muss festgestellt werden, dass Intrusion Detection teuer ist. Entweder man schafft sich schnelle Computer (pro Stück etwa ab 1200 €) an und setzt auf Open Source Software, oder man greift zu einer kommerziellen Lösung, wo die kleinsten Systeme ab 3000 € aufwärts zu haben sind. Zusätzlich muss eventuell das Netzwerk verändert werden und ein Netzanalytiker beschäftigt werden (Personalkosten). Um diese hohen Investitionen zu rechtfertigen, muss für das Management ein bedeutender Nutzen erkennbar sein. Intrusion Detection kann die jährlich zu erwartenden Verluste, die durch Angriffe gegen die eigene Organisation entstehen, verringern, indem die Firewall und andere Verteidigungsmittel durch die Analyse des IDS optimiert werden können. [vgl. Northcutt 04]

Das Management muss sehen, dass sich die Investitionen in das einbruchserkennende System lohnen. Einfache Zahlenbeispiele und Fragestellungen können in „Chef-Etagen“ bereits viel bewirken:

- *Was kosten zwei oder mehrere Tage Stillstand der EDV im Unternehmen?*
- *Wie viele Mitarbeiter müssen untätig (und trotzdem bezahlt) zusehen?*
- *Welche Spezialisten müssen bei der Wiederinbetriebnahme des Rechenzentrums (z.B. EMC Storage Area Network, Oracle DB) vor Ort sein?*
- *Was kostet ein modernes ID-System für die Anforderungen des Unternehmens?*

Einige dieser Punkte sollte man sich gut vor Augen halten, wenn es um die Anschaffung eines Intrusion Detection Systems geht. Bei einem Neuerwerb bzw. Upgrade sollte man sich nicht mit einem veralteten System zufrieden geben. Nach dem Motto „wenn schon, denn schon“, sollte man zu einem leistungsfähigen „state of the art“ - System greifen, sonst hinkt man den Angreifern bald wieder hinterher!

Wenn man nicht bereit ist in diese notwendige Technologie zu investieren, muss man mit dem bestehendem Risiko leben, und Verantwortung im Falle eines Desasters übernehmen.

Die Investitionskosten sind ohne Frage sehr hoch, aber trotz allem begrenzt und kalkulierbar. Es wird keinen maximalen Schutz zu minimalen Kosten geben, aber sicher eine vernünftige Relation zwischen den Kosten der Sicherheitsinvestitionen und dem zu erwartenden Nutzen.

12 Trends und Entwicklungen

Das Gebiet der Intrusion Detection entwickelte sich aus erweiterten Auditmaßnahmen und stellt heute eine eigenständige Produktkategorie dar. Viele Unternehmen überlegen in Zukunft den Erwerb eines IDS für ihre IT-Sicherheit.

Es ist zu erwarten, dass Intrusion Detection Systeme in wenigen Jahren Standard in Unternehmensnetzwerken jeder Größe sein werden. Wie diese Diplomarbeit zeigt, bietet eine Firewall allein keinesfalls einen ausreichenden Schutz in dieser fortgeschrittenen globalen IT-Landschaft. ID-Systeme sind eine sinnvolle und notwendige Ergänzung zu anderen Sicherheitsmaßnahmen im Netz. Es bedeutet allerdings auch zunehmende Anforderungen an das Sicherheitspersonal. Eine klare Trennung der Methoden (misuse detection und anomaly detection) sowie der Arten (NIDS, HIDS, IRS, IPS [siehe Kapitel 4]) wird in Zukunft nicht mehr möglich sein. Bei vielen großen Herstellern (z.B. ISS, McAfee, Cisco) sieht man den Trend zur Verschmelzung der Techniken zu einem ausgereiften, verteilten System, welches die Vorteile der verschiedenen Ansätze nützt.

Im Open Source Bereich gewinnt Snort weiterhin an Bedeutung und erhält auch im kommerziellen Umfeld (z.B. Sourcefire, PacketAlarm, SynSpector) weiter Einzug. Während die Anschaffungskosten bei Open Source punkto Software gänzlich entfallen, muss mit hohem Personalaufwand und kostenpflichtigem professionellen Support gerechnet werden.

In der ersten Einführungsphase von Firewall-Produkten waren große Finanzinstitute, Versicherungen und Behörden nahezu die einzigen Kunden. Nach und nach wurde der Preis für solche Systeme gesenkt und sie daher auch für Klein- und Mittelbetriebe, beziehungsweise den SOHO-Bereich interessant. Dasselbe spiegelt sich nun bei den ID-Systemen wieder. Die Preispolitik ändert sich und die Produkte werden in den breiten Massenmarkt gedrückt. Wichtige Aspekte hierfür sind natürlich Benutzerfreundlichkeit, Plug-and-Play und Unterstützung für Windows Systeme, welche die breite Masse bevorzugt. Viele Hersteller erkennen das Potential, das der Security-Sektor bietet und werben mit Sicherheitslösungen „aus einer Hand“. ID-Systeme werden mit weiteren

Sicherheits-Werkzeugen von einem Hersteller angeboten. Ob sich dieser Ansatz einer Universallösung in Zukunft durchsetzen wird, bleibt abzuwarten.

Langsam, aber doch, vollzieht sich ein bei den Herstellern erkennbarer Trend von den ursprünglich passiven Intrusion Detection Systemen zu der neuen Generation von aktiven Intrusion Prevention Systemen, die Gefahren in Echtzeit erkennen und verhindern. Über diese Systeme muss der gesamte Datenverkehr geleitet werden, womit hohe Performance nötig ist, damit keine Latenzzeiten für den normalen Arbeitsbetrieb entstehen. Häufig werden Intrusion Prevention System-Lösungen in Infrastrukturkompenenten (Router, Switches, Firewalls), die an neuralgischen Punkten des Netzwerks stehen, bereits implementiert.

13 Resümee

Die Zahl der Angriffe und Sicherheitsverletzungen in Netzwerken ist in den letzten Jahren stark gestiegen. Zustände von Systemen können sich aufgrund neuer Entwicklungen, höherer Bandbreiten oder neuer Sicherheitslücken täglich oder sogar stündlich ändern. Vorbei sind die Tage, in denen technische Lösungen, wie Firewall und Virenschutz, für die Unternehmenssicherheit ausreichend waren. Ergänzend zur bestehenden Sicherheitsinfrastruktur (meist Firewall und Virenschutz) sollte ein Intrusion Detection System integriert werden. Das ID-System bringt natürlich nicht nur Vorteile mit sich. Es ist ein komplexes System das ständig angepasst und aktualisiert werden muss. Der Aufwand für die Administratoren erhöht sich stark und sollte nicht unterschätzt werden.

Für Unternehmen mit hohem Sicherheitsbewusstsein und vielen kritischen Daten ist dieser Aufwand sicherlich vertretbar. Die beste technische Lösung kann allerdings nur wenig Sicherheit bieten, wenn es schon an der Sicherheitsorganisation im Unternehmen grundsätzlich scheitert. Viele Unternehmen widmen sich IT-Herausforderungen nur technisch. Es bedarf aber einer gründlichen organisatorischen Einbindung des ID-Systems. Intrusion Detection ist keinesfalls eine Technik, die man nach der Integration sich selbst überlassen kann. Wichtig ist es klare Sicherheitsrichtlinien zu erstellen, die von IT-Anwendern und Administratoren auch eingehalten und kontrolliert (Firewall und ID-System) werden.

Firewall, ID-System und Virenschutz können auch nie hundertprozentige Sicherheit bieten, sie können jedoch die Gefahr von erfolgreichen Angriffen gegen die eigene IT-Umgebung deutlich verringern. Sicherheit ist ein Prozess, der einer ständigen Veränderung unterliegt und viel Zeit, Mühe, Erfahrung und Wissen benötigt. Absolute Sicherheit ist trotz allem ein Mythos!

Anhang A: Literatur- und Quellenverzeichnis

[Anonymous 03]
Anonymous: *Hacker´s Guide*
Markt + Technik Verlag 2003, München

[Beale 03]
Jay Beale, Anne Carasik, Scott Dentler, Adam Doxtater, Wally Eaton, Jeremy Faircloth, James Foster, Vitaly Osipov: *Snort 2.0 Intrusion Detection*
Mitp-Verlag 2003, Bonn

[Brutschy 03]
Arne Brutschy*: Intrusion Detection Systems; Problemseminar Mobilität und Sicherheit im Internet*
Universität Leipzig, Juli 2003

[BSI 02a]
Einführung von Intrusion Detection Systemen – Grundlagen
Bundesamt für Sicherheit in der Informationstechnik, Oktober 2002

[BSI 02b]
Einführung von Intrusion Detection Systemen – Leitfaden für die Einführung
Bundesamt für Sicherheit in der Informationstechnik, Oktober 2002

[BSI 02c]
Einführung von Intrusion Detection Systemen – Rechtliche Aspekte
Bundesamt für Sicherheit in der Informationstechnik, Oktober 2002

[CERT 02]
Howard Lipson F., Ph.D.: *Tracking and Tracing Cyber Attacks*
CERT Coordination Center, November 2002

[CSI 04]
2004 CSI/FBI Computer Crime and Security Survey
Computer Security Institute
http://www.gocsi.com/

[Faulhaber 02]
Arndt Faulhaber, Emal Alekozai: *Portscanner und Network Intrusion Detection Systeme*

[Helden 98]
Dr. Josef Helden, Dr. Stefan Karsch: *Grundlagen, Forderungen und Marktübersicht für Intrusion Detection Systeme*
Debis IT Security Services, Oktober 1998

[Hildebrandt 01]
Alexis Hildebrandt, Matthias Meyer: *Intrusion Detection am Beispiel Snort*
Juli 2001

[Leitold 03]
DI H. Leitold, DI T. Rössler, Dr. I. Schaumüller-Bichl: *Österreichisches IT-Sicherheitshandbuch,* Version 2.1 Mai 2003

[Müller 03]
Burkhard Müller: *Netzwerke*
Markt + Technik Verlag 2003, München

[Northcutt 04]
Stephen Northcutt, Judy Novak: *Network Intrusion Detection*
Hüthig Verlag 2004, Bonn

[Rafeeq 03]
Rafeeq Ur Rehman: *Intrusion Detection Systems with Snort*
Pearson Education 2003, New Jersey

[Sobirey 99]
Michael Sobirey: *Datenschutzorientiertes Intrusion Detection*
F. Vieweg & Sohn 1999, Braunschweig/Wiesbaden

[Spenneberg 03]
Spenneberg Ralf: *Intrusion Detection für Linux Server*
Markt + Technik Verlag 2003, München

[Tett 04]
Matt Tett: *Detektive fürs LAN: Intrusion Detection Systeme im Test*
ZDNet Australia, Oktober 2004

[www 01]
RFC 793 – Transmission Control Protocol
Transmission Control Protocol:
http://www.ietf.org/rfc/rfc0760.txt?number=793

[www 02]
IANA Wellknown-Ports
http://www.iana.org/assignments/port-numbers

[www 03]
RFC 760 – Internet Protocol
Internet Protocol: http://www.ietf.org/rfc/rfc0760.txt?number=760

[www 04]
Scantechniken
http://www.tcp-ip-info.de/security/portscans.htm

[www 05]
Frank Koormann, Jan Wagner: *Nessus-Benutzerhandbuch*
http://ftp.intevation.de/boss/doc/users-manual-de-20050103.pdf

[www 06]
Internet Security Systems
http://www.iss.net

[www 07]
Cisco Systems - Deutschland
http://www.cisco.com/global/DE/index.shtml

[www 08]
Enterasys Networks
http://www.enterasys.com/products/ids/

[www 09]
Snort-Projekt
http://www.snort.org/

[www 10]
Tripwire-Projekt
http://www.tripwire.org/

[www 11]
SANS (SysAdmin, Audit, Network, Security) Top 20
http://www.sans.org/top20/#threats

[www 12]
Computer – Technik – Security: *Spoofing-FAQs*
http://www.computec.ch/dokumente/unsortiert/spoofing_faq.html

[www 13]
McAfee IntruShield Network IPS
http://www.mcafeesecurity.com/de/products/mcafee/network_ips/intrushield_appliances.htm

[www 14]
Sourcefire Network Security
http://www.sourcefire.com

[www 15]
Juniper Networks – Intrusion Detection & Prevention
http://www.juniper.net/products/intrusion/

Alle Quellenangaben entsprechen dem Stand vom 11. Mai 2005.

Anhang B: Abbildungsverzeichnis

Anhang C: Tabellenverzeichnis

Anhang D: Danksagungen

Ich bedanke mich bei meinem Betreuer Herrn Dipl.-HTL-Ing. **Schaupp Andreas** MSc der mir dieses interessante Diplomarbeitsthema ermöglicht hat. Er war jederzeit für mich erreichbar, gab mir wertvolle Hinweise aus der Praxis, und bot mir eine kompetente Betreuung bei meinem Themengebiet.

Ein besonderer Dank geht an meine Eltern, **Angelika** und **Gerhard Reikerstorfer**, die mich während meiner Ausbildungsjahre in jeglicher Hinsicht unterstützten und somit einen wesentlich Beitrag für meinen erfolgreichen Abschluss leisteten.

Anhang E: Glossar

ACID
= Analysis Console for Intrusion Databases; stellt eine grafische Benutzeroberfläche für Snort zur Verfügung.

ADSL
= Asymmetric Digital Subscriber Line; digitaler Internet-Breitband-Anschluss.

Anomaly Detection
Erkennung von Angriffen durch atypisches Verhalten.

ARP
= Address Resolution Protocol; liefert zu einer bekannten IP-Adresse die MAC-Adresse.

ASCII
= ein Akronym für American Standard Code for Information Interchange und basiert auf dem lateinischem Alphabet.

BSI
= Bundesamt für Sicherheit in der Informationstechnik (Deutschland)

CERT
= Computer Emergency Response Team; eine Notfallmannschaft die rasch reagiert, falls ein Angriff stattfindet oder stattgefunden hat.

CSI
= Computer Security Institute

CPU
= Central Processing Unit

DIDS
= Distributed Intrusion Detection System; eine zentrale Management-Station sammelt die Daten mehrerer Sensoren.

DMZ
= demilitarisierte Zone; entkoppelt öffentlich zur Verfügung gestellte Dienste (z.B. http oder mail) vomlokalen Netzwerk.

DoS
= Denial of Service; bezeichnet Angriffsmethoden, welche die Verfügbarkeit eines Systems verhindern.

Ethereal
= Packet-Sniffer, der auf verschiedenen Plattformen erhältlich ist.

False negatives
Angriffe die vom ID-System nicht erkannt wurden.

False positives
Das ID-System meldet fälschlich einen Angriff.

Firewall
Stellt ein schützendes Glied zwischen Netzwerken dar, indem nur bestimmte Datenverbindungen erlaubt werden.

Gnome
= grafische Oberfläche auf Linux-Systemen.

GPL
= GNU General Public License; Lizenz freier Software.

HIDS
= Host ID-Systeme; analysiert Daten auf einem Rechner, um Angriffe festzustellen.

How-To´s
Ist eine kurze Anleitung für das Lösen einer ganz bestimmten Aufgabe oder eines Problems.

https
= hyper text transfer protocol secure; verschlüsselte Variante zu http.

ICMP
= Internet Control Message Protocol

IDS
Siehe ID-System.

ID-System
= Intrusion Detection System; ein Überwachungssystem in einem lokalen Netzwerk oder auf einem Rechner. In der Literatur auch oft mit IDS abgekürzt.

IDP
= Intrusion Detection and Prevention System; eine aktive Struktur, die potentiellen Bedrohungen vorbeugt.

IP
= Internet Protocol

IRS
= Intrusion Response System; reagiert aktiv auf Angriffe.

IT
= Information Technology

KDE
= grafische Oberfläche auf Linux-Systemen.

Kompromittierung
= ein Oberbegriff für die Verletzung einer Vertraulichkeit.

LAN
= Local Area Network; bezeichnet ein Netzwerk das sich räumlich auf ein Gebäude oder Firmengelände beschränkt.

MAC
= Medium Access Control; stellt ein Adressierungsschema auf Schicht 2 des ISO/OSI-Modells dar.

Misuse Detection
Erkennen von Angriffen durch Signaturen.

MTU
= Maximum Transmission Unit; wie viele Bytes in einem Block auf dem Medium übertragen werden können (z.B. Ethernet 1500 Byte).

NIDS
= Netzwerk ID-System; stellt Einbrüche durch Überwachung der Netzwerk-aktivitäten fest.

OSI
= Open Systems Interconnection; dient als Grundlage für die Kommunikation zwischen Rechnernetzwerken. Ist ein theoretisches Kommunikationsmodell der International Standard Organisation (ISO).

Packet-Sniffer
Siehe Sniffer.

Promiscuous Mode
Netzwerkkarte nimmt alle Pakete, unabhängig von der Adressierung, auf.

rpc
= remote procedure call.

RPM
= RedHat Packet Manager.

SCP
= secure copy; verschlüsselte Übertragung von Dateien.

Sniffer
= eine Software die den Datenverkehr im Netzwerk aufnehmen und darstellen kann.

Snort
= ein Open Source Intrusion Detection System.

SOHO
= Small-Office Home-Office; bezeichnet Kleinbetrieb oder Heimarbeitsplatz.

SPADE
= Statistical Packet Anomaly Detection Engine; eine Erweiterung zu Snort um auch Anomalieerkennung zu ermöglichen.

SPAN
Abk. für Switch Port Analyser. An diesem Port können allen Datenströme die durch den Switch gelangen, kopiert werden.

Spoofing
Spoofing bedeutet die Vorgabe einer falschen Identität (z.B. IP-Adresse) und das Eindringen in ein fremdes Netz mit den Privilegien der vorgegebenen IP-Adresse.

SSH
= secure shell; verschlüsselte Fernwartung auf Terminal-Ebene.

SSL
= Secure Socket Layer; Übertragungsprotokoll für sichere Kommunikation.

Stack
= Stapelspeicher; eine dynamische Speicherstruktur, wo ein Programm während der Laufzeit Daten ablegen kann.

TCP
= Transmission Control Protocol; ein verbindungsorientiertes Protokoll.

telnet
Application, die Remotezugriff auf entfernte Rechner per Commandline ermöglicht.

UDP
= User Datagram Protocol; ein verbindungsloses Protokoll.

URL
= Uniform Resource Locator; identifiziert eine Quelle (Dokument) an einem bestimmten Ort (Adresse) im Internet eindeutig.

USB
= Universal Serial Bus; ist ein Bussystem, das externe Geräte wie Drucker, Scanner oder USB-Stick mit dem Computer verbindet.

WLAN
= Wireless Local Area Network; bezeichnet ein drahtloses lokales Netzwerk.

Anhang F: Snort Konfigurationsdatei

```
#--------------------------------------------------
#   http://www.snort.org     Snort 2.3.2 Ruleset
#     Contact: snort-sigs@lists.sourceforge.net
#--------------------------------------------------
# $Id: snort.conf,v 1.144.2.8 2005/03/08 16:28:12 jhewlett Exp $
#
##################################################
# This file contains a sample snort configuration.
# You can take the following steps to create your own custom configuration:
#
#  1) Set the network variables for your network
#  2) Configure preprocessors
#  3) Configure output plugins
#  4) Customize your rule set
#
##################################################
# Step #1: Set the network variables:
#
# You must change the following variables to reflect your local network. The
# variable is currently setup for an RFC 1918 address space.
#
# You can specify it explicitly as:
#
var HOME_NET 192.168.100.0/24
#
# or use global variable $<interfacename>_ADDRESS which will be always
# initialized to IP address and netmask of the network interface which you run
# snort at.  Under Windows, this must be specified as
# $(<interfacename>_ADDRESS), such as:
# $(\Device\Packet_{12345678-90AB-CDEF-1234567890AB}_ADDRESS)
#
# var HOME_NET $eth0_ADDRESS
#
# You can specify lists of IP addresses for HOME_NET
# by separating the IPs with commas like this:
#
# var HOME_NET [10.1.1.0/24,192.168.1.0/24]
#
# MAKE SURE YOU DON'T PLACE ANY SPACES IN YOUR LIST!
#
# or you can specify the variable to be any IP address
# like this:

#var HOME_NET any

# Set up the external network addresses as well.  A good start may be "any"
var EXTERNAL_NET any

# Configure your server lists.  This allows snort to only look for attacks to
# systems that have a service up.  Why look for HTTP attacks if you are not
# running a web server?  This allows quick filtering based on IP addresses
# These configurations MUST follow the same configuration scheme as defined
# above for $HOME_NET.

# List of DNS servers on your network
var DNS_SERVERS 192.168.100.100

# List of SMTP servers on your network
var SMTP_SERVERS 192.168.100.100

# List of web servers on your network
var HTTP_SERVERS 192.168.100.100

# List of sql servers on your network
var SQL_SERVERS 192.168.100.100

# List of telnet servers on your network
# var TELNET_SERVERS $HOME_NET
```

```
# List of snmp servers on your network
# var SNMP_SERVERS $HOME_NET

# Configure your service ports.  This allows snort to look for attacks destined
# to a specific application only on the ports that application runs on.  For
# example, if you run a web server on port 8081, set your HTTP_PORTS variable
# like this:
#
# var HTTP_PORTS 8081
#
# Port lists must either be continuous [eg 80:8080], or a single port [eg 80].
# We will adding support for a real list of ports in the future.

# Ports you run web servers on
#
# Please note:  [80,8080] does not work.
# If you wish to define multiple HTTP ports,
#
## var HTTP_PORTS 80
## include somefile.rules
## var HTTP_PORTS 8080
## include somefile.rules
var HTTP_PORTS 80

# Ports you want to look for SHELLCODE on.
var SHELLCODE_PORTS !80

# Ports you do oracle attacks on
#var ORACLE_PORTS 1521

# other variables
#
# AIM servers.  AOL has a habit of adding new AIM servers, so instead of
# modifying the signatures when they do, we add them to this list of servers.
var                                                                    AIM_SERVERS
[64.12.24.0/23,64.12.28.0/23,64.12.161.0/24,64.12.163.0/24,64.12.200.0/24,205.188.3.0/24,20
5.188.5.0/24,205.188.7.0/24,205.188.9.0/24,205.188.153.0/24,205.188.179.0/24,205.188.248.0/
24]

# Path to your rules files (this can be a relative path)
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\rules
var RULE_PATH ../rules

# Configure the snort decoder
# ============================
#
# Snort's decoder will alert on lots of things such as header
# truncation or options of unusual length or infrequently used tcp options
#
#
# Stop generic decode events:
#
# config disable_decode_alerts
#
# Stop Alerts on experimental TCP options
#
# config disable_tcpopt_experimental_alerts
#
# Stop Alerts on obsolete TCP options
#
# config disable_tcpopt_obsolete_alerts
#
# Stop Alerts on T/TCP alerts
#
# In snort 2.0.1 and above, this only alerts when a TCP option is detected
# that shows T/TCP being actively used on the network.  If this is normal
# behavior for your network, disable the next option.
#
# config disable_tcpopt_ttcp_alerts
#
# Stop Alerts on all other TCPOption type events:
#
# config disable_tcpopt_alerts
#
```

```
# Stop Alerts on invalid ip options
#
# config disable_ipopt_alerts

# Configure the detection engine
# ==============================
#
# Use a different pattern matcher in case you have a machine with very limited
# resources:
#
# config detection: search-method lowmem

# Configure Inline Resets
# =======================
#
# If running an iptables firewall with snort in InlineMode() we can now
# perform resets via a physical device. We grab the indev from iptables
# and use this for the interface on which to send resets. This config
# option takes an argument for the src mac address you want to use in the
# reset packet.  This way the bridge can remain stealthy. If the src mac
# option is not set we use the mac address of the indev device. If we
# don't set this option we will default to sending resets via raw socket,
# which needs an ipaddress to be assigned to the int.
#
# config layer2resets: 00:06:76:DD:5F:E3

##################################################
# Step #2: Configure preprocessors
#
# General configuration for preprocessors is of
# the form
# preprocessor <name_of_processor>: <configuration_options>

# Configure Flow tracking module
# ------------------------------
#
# The Flow tracking module is meant to start unifying the state keeping
# mechanisms of snort into a single place. Right now, only a portscan detector
# is implemented but in the long term,  many of the stateful subsystems of
# snort will be migrated over to becoming flow plugins. This must be enabled
# for flow-portscan to work correctly.
#
# See README.flow for additional information
#
preprocessor flow: stats_interval 0 hash 2

# frag2: IP defragmentation support
# ------------------------------
# This preprocessor performs IP defragmentation.  This plugin will also detect
# people launching fragmentation attacks (usually DoS) against hosts.  No
# arguments loads the default configuration of the preprocessor, which is a 60
# second timeout and a 4MB fragment buffer.

# The following (comma delimited) options are available for frag2
#    timeout [seconds] - sets the number of [seconds] that an unfinished
#                        fragment will be kept around waiting for completion,
#                        if this time expires the fragment will be flushed
#    memcap [bytes] - limit frag2 memory usage to [number] bytes
#                     (default:  4194304)
#
#    min_ttl [number] - minimum ttl to accept
#
#    ttl_limit [number] - difference of ttl to accept without alerting
#                         will cause false positves with router flap
#
# Frag2 uses Generator ID 113 and uses the following SIDS
# for that GID:
#  SID     Event description
# -----   ------------------
#   1       Oversized fragment (reassembled frag > 64k bytes)
#   2       Teardrop-type attack

preprocessor frag2

# stream4: stateful inspection/stream reassembly for Snort
```

```
#--------------------------------------------------------------------
# Use in concert with the -z [all|est] command line switch to defeat stick/snot
# against TCP rules.  Also performs full TCP stream reassembly, stateful
# inspection of TCP streams, etc.  Can statefully detect various portscan
# types, fingerprinting, ECN, etc.

# stateful inspection directive
# no arguments loads the defaults (timeout 30, memcap 8388608)
# options (options are comma delimited):
#   detect_scans - stream4 will detect stealth portscans and generate alerts
#                  when it sees them when this option is set
#   detect_state_problems - detect TCP state problems, this tends to be very
#                           noisy because there are a lot of crappy ip stack
#                           implementations out there
#
#   disable_evasion_alerts - turn off the possibly noisy mitigation of
#                            overlapping sequences.
#
#
#   min_ttl [number]       - set a minium ttl that snort will accept to
#                            stream reassembly
#
#   ttl_limit [number]     - differential of the initial ttl on a session versus
#                             the normal that someone may be playing games.
#                             Routing flap may cause lots of false positives.
#
#   keepstats [machine|binary] - keep session statistics, add "machine" to
#                         get them in a flat format for machine reading, add
#                         "binary" to get them in a unified binary output
#                         format
#   noinspect - turn off stateful inspection only
#   timeout [number] - set the session timeout counter to [number] seconds,
#                       default is 30 seconds
#   memcap [number] - limit stream4 memory usage to [number] bytes
#   log_flushed_streams - if an event is detected on a stream this option will
#                         cause all packets that are stored in the stream4
#                         packet buffers to be flushed to disk.  This only
#                         works when logging in pcap mode!
#
# Stream4 uses Generator ID 111 and uses the following SIDS
# for that GID:
#  SID     Event description
# -----   ------------------
#   1       Stealth activity
#   2       Evasive RST packet
#   3       Evasive TCP packet retransmission
#   4       TCP Window violation
#   5       Data on SYN packet
#   6       Stealth scan: full XMAS
#   7       Stealth scan: SYN-ACK-PSH-URG
#   8       Stealth scan: FIN scan
#   9       Stealth scan: NULL scan
#   10      Stealth scan: NMAP XMAS scan
#   11      Stealth scan: Vecna scan
#   12      Stealth scan: NMAP fingerprint scan stateful detect
#   13      Stealth scan: SYN-FIN scan
#   14      TCP forward overlap

preprocessor stream4: disable_evasion_alerts

# tcp stream reassembly directive
# no arguments loads the default configuration
#   Only reassemble the client,
#   Only reassemble the default list of ports (See below),
#   Give alerts for "bad" streams
#
# Available options (comma delimited):
#   clientonly - reassemble traffic for the client side of a connection only
#   serveronly - reassemble traffic for the server side of a connection only
#   both - reassemble both sides of a session
#   noalerts - turn off alerts from the stream reassembly stage of stream4
#   ports [list] - use the space separated list of ports in [list], "all"
#                  will turn on reassembly for all ports, "default" will turn
#                  on reassembly for ports 21, 23, 25, 53, 80, 143, 110, 111
#                  and 513
```

```
preprocessor stream4_reassemble

# http_inspect: normalize and detect HTTP traffic and protocol anomalies
#
# lots of options available here. See doc/README.http_inspect.
# unicode.map should be wherever your snort.conf lives, or given
# a full path to where snort can find it.
preprocessor http_inspect: global \
    iis_unicode_map unicode.map 1252

preprocessor http_inspect_server: server default \
    profile all ports { 80 8080 8180 } oversize_dir_length 500

#
#  Example unique server configuration
#
#preprocessor http_inspect_server: server 1.1.1.1 \
#    ports { 80 3128 8080 } \
#    flow_depth 0 \
#    ascii no \
#    double_decode yes \
#    non_rfc_char { 0x00 } \
#    chunk_length 500000 \
#    non_strict \
#    oversize_dir_length 300 \
#    no_alerts


# rpc_decode: normalize RPC traffic
# ---------------------------------
# RPC may be sent in alternate encodings besides the usual 4-byte encoding
# that is used by default. This plugin takes the port numbers that RPC
# services are running on as arguments - it is assumed that the given ports
# are actually running this type of service. If not, change the ports or turn
# it off.
# The RPC decode preprocessor uses generator ID 106
#
# arguments: space separated list
# alert_fragments - alert on any rpc fragmented TCP data
# no_alert_multiple_requests - don't alert when >1 rpc query is in a packet
# no_alert_large_fragments - don't alert when the fragmented
#                            sizes exceed the current packet size
# no_alert_incomplete - don't alert when a single segment
#                       exceeds the current packet size

preprocessor rpc_decode: 111 32771

# bo: Back Orifice detector
# -------------------------
# Detects Back Orifice traffic on the network.  Takes no arguments in 2.0.
#
# The Back Orifice detector uses Generator ID 105 and uses the
# following SIDS for that GID:
#  SID     Event description
# -----   ------------------
#   1       Back Orifice traffic detected

preprocessor bo

# telnet_decode: Telnet negotiation string normalizer
# ---------------------------------------------------
# This preprocessor "normalizes" telnet negotiation strings from telnet and ftp
# traffic.  It works in much the same way as the http_decode preprocessor,
# searching for traffic that breaks up the normal data stream of a protocol and
# replacing it with a normalized representation of that traffic so that the
# "content" pattern matching keyword can work without requiring modifications.
# This preprocessor requires no arguments.
# Portscan uses Generator ID 109 and does not generate any SID currently.

preprocessor telnet_decode

# Flow-Portscan: detect a variety of portscans
# --------------------------------------------
# Note:  The Flow preprocessor (above) must first be enabled for Flow-Portscan to
```

```
# work.
#
# This module detects portscans based off of flow creation in the flow
# preprocessors.  The goal is to catch one->many hosts and one->many
# ports scans.
#
# Flow-Portscan has numerous options available, please read
# README.flow-portscan for help configuring this option.

# Flow-Portscan uses Generator ID 121 and uses the following SIDS for that GID:
#  SID     Event description
# -----   ------------------
#   1       flow-portscan: Fixed Scale Scanner Limit Exceeded
#   2       flow-portscan: Sliding Scale Scanner Limit Exceeded
#   3       flow-portscan: Fixed Scale Talker Limit Exceeded
#   4      flow-portscan: Sliding Scale Talker Limit Exceeded

# preprocessor flow-portscan: \
#      talker-sliding-scale-factor 0.50 \
#      talker-fixed-threshold 30 \
#      talker-sliding-threshold 30 \
#      talker-sliding-window 20 \
#      talker-fixed-window 30 \
#      scoreboard-rows-talker 30000 \
#      server-watchnet [10.2.0.0/30] \
#      server-ignore-limit 200 \
#      server-rows 65535 \
#      server-learning-time 14400 \
#      server-scanner-limit 4 \
#      scanner-sliding-window 20 \
#      scanner-sliding-scale-factor 0.50 \
#      scanner-fixed-threshold 15 \
#      scanner-sliding-threshold 40 \
#      scanner-fixed-window 15 \
#      scoreboard-rows-scanner 30000 \
#      src-ignore-net [192.168.1.1/32,192.168.0.0/24] \
#      dst-ignore-net [10.0.0.0/30] \
#      alert-mode once \
#      output-mode msg \
#      tcp-penalties on

# sfPortscan
# ----------
# Author: Dan Roelker
# Portscan detection module.  Detects various types of portscans and
# portsweeps.  For more information on detection philosophy, alert types,
# and detailed portscan information, please refer to the README.sfportscan.
#
# -configuration options-
#     proto { tcp udp icmp ip_proto all }
#       The arguments to the proto option are the types of protocol scans that
#       the user wants to detect.  Arguments should be separated by spaces and
#       not commas.
#     scan_type { portscan portsweep decoy_portscan distributed_portscan all }
#       The arguments to the scan_type option are the scan types that the
#       user wants to detect.  Arguments should be separated by spaces and not
#       commas.
#     sense_level { low|medium|high }
#       There is only one argument to this option and it is the level of
#       sensitivity in which to detect portscans.  The 'low' sensitivity
#       detects scans by the common method of looking for response errors, such
#       as TCP RSTs or ICMP unreachables.  This level requires the least
#       tuning.  The 'medium' sensitivity level detects portscans and
#       filtered portscans (portscans that receive no response).  This
#       sensitivity level usually requires tuning out scan events from NATed
#       IPs, DNS cache servers, etc.  The 'high' sensitivity level has
#       lower thresholds for portscan detection and a longer time window than
#       the 'medium' sensitivity level.  Requires more tuning and may be noisy
#       on very active networks.  However, this sensitivity levels catches the
#       most scans.
#     memcap { positive integer }
#       The maximum number of bytes to allocate for portscan detection.  The
#       higher this number the more nodes that can be tracked.
#     logfile { filename }
#       This option specifies the file to log portscan and detailed portscan
```

```
#       values to.  If there is not a leading /, then snort logs to the
#       configured log directory.  Refer to README.sfportscan for details on
#       the logged values in the logfile.
#     watch_ip { Snort IP List }
#     ignore_scanners { Snort IP List }
#     ignore_scanned { Snort IP List }
#       These options take a snort IP list as the argument.  The 'watch_ip'
#       option specifies the IP(s) to watch for portscan.  The
#       'ignore_scanners' option specifies the IP(s) to ignore as scanners.
#       Note that these hosts are still watched as scanned hosts.  The
#       'ignore_scanners' option is used to tune alerts from very active
#       hosts such as NAT, nessus hosts, etc.  The 'ignore_scanned' option
#       specifies the IP(s) to ignore as scanned hosts.  Note that these hosts
#       are still watched as scanner hosts.  The 'ignore_scanned' option is
#       used to tune alerts from very active hosts such as syslog servers, etc.
#
preprocessor sfportscan: proto  { all } \
                         memcap { 10000000 } \
                         sense_level { low }

# arpspoof
#----------------------------------------
# Experimental ARP detection code from Jeff Nathan, detects ARP attacks,
# unicast ARP requests, and specific ARP mapping monitoring.  To make use of
# this preprocessor you must specify the IP and hardware address of hosts on
# the same layer 2 segment as you.  Specify one host IP MAC combo per line.
# Also takes a "-unicast" option to turn on unicast ARP request detection.
# Arpspoof uses Generator ID 112 and uses the following SIDS for that GID:

#  SID     Event description
# -----   ------------------
#   1       Unicast ARP request
#   2       Etherframe ARP mismatch (src)
#   3       Etherframe ARP mismatch (dst)
#   4       ARP cache overwrite attack

preprocessor arpspoof
#preprocessor arpspoof_detect_host: 192.168.40.1 f0:0f:00:f0:0f:00

# Performance Statistics
# ----------------------
# Documentation for this is provided in the Snort Manual.  You should read it.
# It is included in the release distribution as doc/snort_manual.pdf
#
# preprocessor perfmonitor: time 300 file /var/snort/snort.stats pktcnt 10000

##################################################################
# Step #3: Configure output plugins
#
# Uncomment and configure the output plugins you decide to use.  General
# configuration for output plugins is of the form:
#
# output <name_of_plugin>: <configuration_options>
#
# alert_syslog: log alerts to syslog
# ----------------------------------
# Use one or more syslog facilities as arguments.  Win32 can also optionally
# specify a particular hostname/port.  Under Win32, the default hostname is
# '127.0.0.1', and the default port is 514.
#
# [Unix flavours should use this format...]
# output alert_syslog: LOG_AUTH LOG_ALERT
#
# [Win32 can use any of these formats...]
# output alert_syslog: LOG_AUTH LOG_ALERT
# output alert_syslog: host=hostname, LOG_AUTH LOG_ALERT
# output alert_syslog: host=hostname:port, LOG_AUTH LOG_ALERT

# log_tcpdump: log packets in binary tcpdump format
# -------------------------------------------------
# The only argument is the output file name.
#
# output log_tcpdump: tcpdump.log

# database: log to a variety of databases
```

```
# ------------------------------------
# See the README.database file for more information about configuring
# and using this plugin.
#
# output database: log, mysql, user=root password=test dbname=db host=localhost
# output database: alert, postgresql, user=snort dbname=snort
# output database: log, odbc, user=snort dbname=snort
# output database: log, mssql, dbname=snort user=snort password=test
# output database: log, oracle, dbname=snort user=snort password=test

# unified: Snort unified binary format alerting and logging
# -------------------------------------------------------------
# The unified output plugin provides two new formats for logging and generating
# alerts from Snort, the "unified" format.  The unified format is a straight
# binary format for logging data out of Snort that is designed to be fast and
# efficient.  Used with barnyard (the new alert/log processor), most of the
# overhead for logging and alerting to various slow storage mechanisms such as
# databases or the network can now be avoided.
#
# Check out the spo_unified.h file for the data formats.
#
# Two arguments are supported.
#    filename - base filename to write to (current time_t is appended)
#    limit    - maximum size of spool file in MB (default: 128)
#
output alert_unified: /var/log/snort/snort.alert
output log_unified: /var/log/snort/snort.log

# You can optionally define new rule types and associate one or more output
# plugins specifically to that type.
#
# This example will create a type that will log to just tcpdump.
# ruletype suspicious
# {
#   type log
#   output log_tcpdump: suspicious.log
# }
#
# EXAMPLE RULE FOR SUSPICIOUS RULETYPE:
# suspicious tcp $HOME_NET any -> $HOME_NET 6667 (msg:"Internal IRC Server";)
#
# This example will create a rule type that will log to syslog and a mysql
# database:
# ruletype redalert
# {
#   type alert
#   output alert_syslog: LOG_AUTH LOG_ALERT
#   output database: log, mysql, user=snort dbname=snort host=localhost
# }
#
# EXAMPLE RULE FOR REDALERT RULETYPE:
# redalert tcp $HOME_NET any -> $EXTERNAL_NET 31337 \
#   (msg:"Someone is being LEET"; flags:A+;)

#
# Include classification & priority settings
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\etc\classification.config
#

include classification.config

#
# Include reference systems
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\etc\reference.config
#

include reference.config

##################################################################
# Step #4: Configure snort with config statements
#
# See the snort manual for a full set of configuration references
```

```
config flowbits_size: 256

##################################################################
# Step #5: Customize your rule set
#
# Up to date snort rules are available at http://www.snort.org
#
# The snort web site has documentation about how to write your own custom snort
# rules.
#
# The rules included with this distribution generate alerts based on on
# suspicious activity. Depending on your network environment, your security
# policies, and what you consider to be suspicious, some of these rules may
# either generate false positives ore may be detecting activity you consider to
# be acceptable; therefore, you are encouraged to comment out rules that are
# not applicable in your environment.
#
# The following individuals contributed many of rules in this distribution.
#
# Credits:
#   Ron Gula <rgula@securitywizards.com> of Network Security Wizards
#   Max Vision <vision@whitehats.com>
#   Martin Markgraf <martin@mail.du.gtn.com>
#   Fyodor Yarochkin <fygrave@tigerteam.net>
#   Nick Rogness <nick@rapidnet.com>
#   Jim Forster <jforster@rapidnet.com>
#   Scott McIntyre <scott@whoi.edu>
#   Tom Vandepoel <Tom.Vandepoel@ubizen.com>
#   Brian Caswell <bmc@snort.org>
#   Zeno <admin@cgisecurity.com>
#   Ryan Russell <ryan@securityfocus.com>



#========================================
# Include all relevant rulesets here
#
# The following rulesets are disabled by default:
#
#   web-attacks, backdoor, shellcode, policy, porn, info, icmp-info, virus,
#   chat, multimedia, and p2p
#
# These rules are either site policy specific or require tuning in order to not
# generate false positive alerts in most enviornments.
#
# Please read the specific include file for more information and
# README.alert_order for how rule ordering affects how alerts are triggered.
#========================================

include $RULE_PATH/local.rules
include $RULE_PATH/bad-traffic.rules
include $RULE_PATH/exploit.rules
include $RULE_PATH/scan.rules
include $RULE_PATH/finger.rules
include $RULE_PATH/ftp.rules
include $RULE_PATH/telnet.rules
include $RULE_PATH/rpc.rules
include $RULE_PATH/rservices.rules
include $RULE_PATH/dos.rules
include $RULE_PATH/ddos.rules
include $RULE_PATH/dns.rules
#include $RULE_PATH/tftp.rules

include $RULE_PATH/web-cgi.rules
include $RULE_PATH/web-coldfusion.rules
#include $RULE_PATH/web-iis.rules
include $RULE_PATH/web-frontpage.rules
include $RULE_PATH/web-misc.rules
include $RULE_PATH/web-client.rules
include $RULE_PATH/web-php.rules

include $RULE_PATH/sql.rules
#include $RULE_PATH/x11.rules
include $RULE_PATH/icmp.rules
include $RULE_PATH/netbios.rules
```

```
include $RULE_PATH/misc.rules
include $RULE_PATH/attack-responses.rules
#include $RULE_PATH/oracle.rules
include $RULE_PATH/mysql.rules
include $RULE_PATH/snmp.rules

include $RULE_PATH/smtp.rules
include $RULE_PATH/imap.rules
#include $RULE_PATH/pop2.rules
include $RULE_PATH/pop3.rules

include $RULE_PATH/nntp.rules
include $RULE_PATH/other-ids.rules
include $RULE_PATH/web-attacks.rules
include $RULE_PATH/backdoor.rules
include $RULE_PATH/shellcode.rules
# include $RULE_PATH/policy.rules
# include $RULE_PATH/porn.rules
# include $RULE_PATH/info.rules
include $RULE_PATH/icmp-info.rules
include $RULE_PATH/virus.rules
# include $RULE_PATH/chat.rules
# include $RULE_PATH/multimedia.rules
include $RULE_PATH/p2p.rules
include $RULE_PATH/experimental.rules

# Include any thresholding or suppression commands. See threshold.conf in the
# <snort src>/etc directory for details. Commands don't necessarily need to be
# contained in this conf, but a separate conf makes it easier to maintain them.
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\etc\threshold.conf
# Uncomment if needed.
# include threshold.conf
```

Anhang G: Barnyard Konfigurationsdatei

```
#--------------------------------------------------------
#   http://www.snort.org    Barnyard 0.1.0 configuration file
#          Contact: snort-barnyard@lists.sourceforge.net
#--------------------------------------------------------
# $Id: barnyard.conf,v 1.9 2004/05/01 16:43:29 andrewbaker Exp $
##########################################################
# Currently you want to do two things in here: turn on
# available data processors and turn on output plugins.
# The data processors (dp's) and output plugin's (op's)
# automatically associate with each other by type and
# are automatically selected at run time depending on
# the type of file you try to load.
##########################################################

# Step 1: configuration declarations
# To keep from having a commandline that uses every letter in the alphabet
# most configuration options are set here

# enable daemon mode
config daemon

# use localtime instead of UTC (*not* recommended because of timewarps)
#config localtime

# set the hostname (currently only used for the acid db output plugin)
config hostname: snorthost

# set the interface name (currently only used for the acid db output plugin)
config interface: eth0

# set the filter (currently only used for the acid db output plugin)
config filter: not port 22

# Step 2: setup the output plugins

# alert_fast
#-----------------------------
# Converts data from the dp_alert plugin into an approximation of Snort's
# "fast alert" mode.  Argument: <filename>

#output alert_fast

# log_dump
#-----------------------------
# Converts data from the dp_log plugin into an approximation of Snort's
# "ASCII packet dump" mode.  Argument: <filename>

#output log_dump

# alert_syslog
#-----------------------------
# Converts data from the alert stream into an approximation of Snort's
# syslog alert output plugin.  Same arguments as the output plugin in snort.

#output alert_syslog

# log_pcap
#-----------------------------
# Converts data from the dp_log plugin into standard pcap format
# Argument: <filename>

#output log_pcap

# acid_db
#-----------------------------
# Available as both a log and alert output plugin.  Used to output data into
# the db schema used by ACID
# Arguments:
```

```
#      $db_flavor           - what flavor of database (ie, mysql)
#      sensor_id $sensor_id - integer sensor id to insert data as
#      database $database   - name of the database
#      server $server       - server the database is located on
#      user $user           - username to connect to the database as
#      password $password   - password for database authentication
# output alert_acid_db: mysql, sensor_id 1, database snort, server localhost, user root
output alert_acid_db: mysql, sensor, database snort, server 192.168.100.155, user root,
password secret, detail full
```