Intrusion Detection am Beispiel von Snort

Alexis Hildebrandt
Syscall() AG
afh@syscall.de

Mathias Meyer
mediaworx berlin AG
meyer@mediaworx.com

3. Juli 2001

Zusammenfassung

Intrusion Detection ist noch ein sehr junges Gebiet der Netzwerksicherheit. Dieser Artikel soll Praxis-orientiert in die Grundlagen und Möglichkeiten der Intrusion Detection einführen, um dann anhand des frei verfügbaren IDS Snort eine beispielhafte Konfiguration aufzuzeigen. Hierbei wird auch Wert darauf gelegt, zu zeigen, dass die Einrichtung eines Intrusion Detection Systems nicht zwingend einen großen Aufwand erfordern muss.

Diese Ausarbeitung entstand im Rahmen der Vorlesung "Netzwerksicherheit" im Studiengang Informatik an der Fachhochschule Brandenburg.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Vorwort

Diese Ausarbeitung entstand nicht ohne Grund. Erstens mal trifft sie im Ganzen das Interessengebiet der beiden Autoren und zweitens gibt es kaum deutsche Ausarbeitungen zum Thema Intrusion Detection im allgemeinen und zu Snort im besonderen. Snort wurde zwar schon in mehreren Artikeln in Fachzeitschriften näher behandelt (siehe [14] und [13]), die Autoren sind aber der Ansicht, dass bisher kein Artikel besonders ausführlich auf die Konfiguration eingegangen ist. Allerdings soll das nicht heissen, dass dieser Artikel vollständig ist. Wie viele andere Programme aus dem Open Source Bereich ist auch Snort einer fortschreitenden Entwicklung unterworfen. Neue Features werden wahrscheinlich schon nach Vollendung dieses Artikels implementiert sein. Bei Bedarf sind die Autoren aber gern bereit, ein *Snort monthly* herauszubringen ;). Für neue Features wird zudem in zukünftigen Artikeln mehr als genug Platz sein.

Ein zweiter Anlass war die Vorlesung "Netzwerksicherheit" des Studienganges Informatik an der Fachhochschule Brandenburg unter der Leitung von Prof. Dr. Barbara Wiesner, die das Interesse der Autoren noch weiter anstachelte, um letztenendes diese Ausarbeitung zu bekommen. Letztere erhebt nicht den Anspruch, vollständig zu sein, perfekt zu sein und auch nicht gut geschrieben zu sein. Kritik, Anmerkungen aber auch Lob sind bei den Autoren sehr willkommen und erwünscht.

Kein Artikel dieses Ausmasses kann geschrieben werden, ohne dass dafür Zeit geopfert werden muss. Die Autoren haben sehr viel Zeit in diesen Artikel investiert, nicht um einer guten Note willen, sondern des eigenen Interesses wegen. Aus diesem Grund möchten die Autoren den Menschen danken, die am meisten unter der am Computer verbrachten Zeit leiden mussten, danken. Mathias Meyer dankt hier Jördis Anderson, die trotz langen Abenden vor der Bildröhre immer eine Schulter zum Anlehnen hatte und viel Geduld ob des neuen Interessengebietes und des Euphorismus zeigte. Alexis Hildebrandt möchte Zoltan Toth danken, der ihm trotz der vor dem Rechner verbrachten Zeit immer ein leckeres Essen kochte und sich immer fragte, warum diese Ausarbeitung nicht in Word geschrieben wurde.

Es gibt aber auch Personen, die keinen direkten Einfluss auf diesen Artikel hatten, die den Autoren aber durch andere Dinge hilfreich zur Seite standen (bildlich gesehen). Auch diese sollen hier nicht unerwähnt bleiben.

- Martin Roesch
  Zweifellos eine wichtige Inspiration für diesen Artikel, da in seinem Kopf die Idee zu Snort entstand und auch umgesetzt wurde.

- Olaf Schreck
  Sein fachlicher Rat in Bezug auf Netzwerke und Intrusion Detection hat uns sehr geholfen.

- W. Richard Stevens
  Seine grandiosen Bücher, z.B. *TCP/IP Illustrated* oder *Unix Network Programming*, haben uns sehr viel Wissen vermittelt und werden dies wohl in Zukunft auch noch tun, da sie sehr umfangreich sind und definitiv in jedes Bücherregal gehören.

- Donald E. Knuth
  Gibt es etwas, womit dieser Mann sich noch nicht beschäftigt hat? In seinem Kopf ist TEX entstanden, wofür ihm die Autoren am dankbarsten sind, da diese Ausarbeitung damit entstanden ist und keine Ausarbeitung mehr mit etwas anderem entstehen wird, und er hat die Trilogie *The Art of Computer Programming* geschrieben. Welcher Informatik-Student erinnert sich zudem nicht mit Freuden an Knuth-Morris-Pratt, den legendären Pattern Matching Algorithmus?

- Matthias Kalle Dalheimer
  Gibt es ein Buch bei O'Reilly, was noch nicht von ihm übersetzt wurde? Besonderer Dank gilt ihm aber hauptsächlich für *LATEX- kurz&gut*

Um den Leser nicht weiter mit Ausschweifungen zu langweilen, fahren wir lieber mit dem eigentlich Inhalt dieses Artikels, mit der Intrusion Detection, fort.

1 Grundlagen der Intrusion Detection

1.1 Einführung in den Begriff

Mit der wachsenden Verbreitung des Internets privat und im Geschäftsleben wächst auch der Bedarf an Sicherheit. Vornehmlich Firmen-Netzwerke sind vielen Angriffs-Versuchen sowohl von innen als auch von außen ausgesetzt. Ein PacketSniffer kann hier Abhilfe schaffen, indem er den gesamten Netzwerk-Verkehr protokolliert. Aber es kann nicht die Aufgabe des System- bzw. Netzwerk-Administrators sein, den Tag mit der Analyse der protokollierten Daten zu verbringen und etwaige Angriffe zu erkennen, zumal er dafür auch noch die spezifischen Merkmale der Unmengen möglicher Angriffe zur Hand oder besser im Kopf haben müsste. Hier helfen sogenannte Intrusion Detection Systeme weiter, die den Netzwerkverkehr mittlerweile in Echtzeit analysieren und anhand bestimmter Regeln etwaige Angriffe erkennen und bei Bedarf Aktionen ausführen, um den Angriff abzuwenden, ihn zu protokollieren oder den Administrator zu benachrichtigen.

Hier stellt sich aber zuerst die Frage, was eine Intrusion überhaupt ist. Heberlein, Levitt und Mukherjee von der University of California, Davis haben sie 1991 in [4] wie folgt definiert: Eine Menge von Handlungen, deren Ziel es ist, die Integrität, die Verfügbarkeit oder die Vertraulichkeit eines Betriebsmittels zu kompromittieren. Allgemeiner gefasst kann man eine Intrusion als eine Verletzung der Sicherheitsmassnahmen eines Systemes verstehen. Intrusion Detection ist noch ein sehr junges Gebiet der IT-Sicherheit und ist im allgemeinen der Versuch, Methoden zu entwickeln, mit denen Angriffe auf Rechnersysteme erkennbar sind. Frühe Intrusion Detection Systeme versuchten dies anhand der Analyse der vom Betriebssystem zur Verfügung gestellten Protokolldateien, den sogenannten Auditdaten, die dem Administrator meistens auch zur Verfügung stehen. Dass diese Methode stark verbesserungswürdig ist, liegt auf der Hand. Eine andere Möglichkeit, die auch heute ihren Zweck nicht verfehlt, ist die Überprüfung der Integrität von Dateien mit Hilfe von Tripwire, welches unter [11] zu finden ist und dessen Zweck es ist, sicherzustellen, dass Dateien nicht verändert wurden. Heutzutage kommt es nicht nur auf die Analyse von Protokolldaten eines Hosts an, sondern auf die effektive Analyse von teilweise sehr grossem Netzwerkaufkommen.

Sollte nun ein Angreifer versuchen, in einen Computer oder ein Netzwerk einzudringen, auf dem ein solcher PacketAnalyser installiert ist, so schlägt dieser Alarm, protokolliert den Angriff und kann ihn eventuell abwehren. Letzteres zählt aber eher zur Aufgabe von Intrusion Response Systemen, auf welche hier nur in beschränktem Masse eingegangen werden soll. Hier ist aber noch erwähnenswert, dass manche ID Systeme gewisse Möglichkeiten

bieten, um auf einen Angriff zu reagieren und so auch eine gewisse Intrusion Response Möglichkeit bieten.

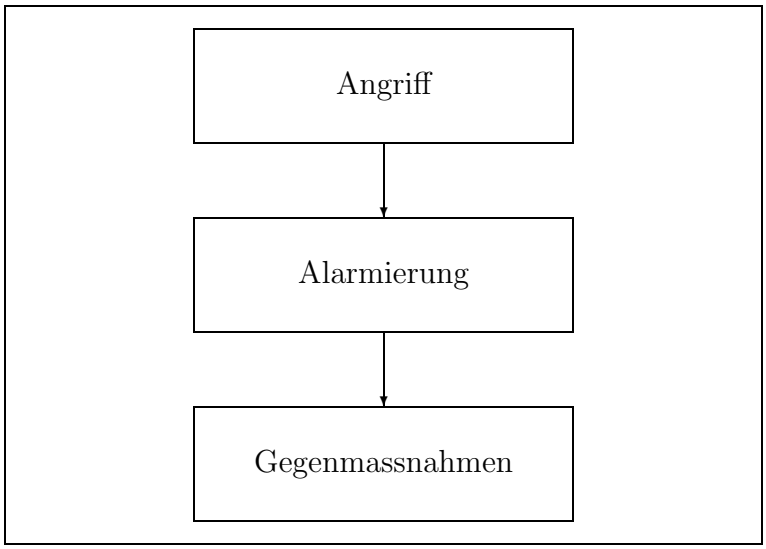

Abbildung 1: Grundprinzip der Intrusion Detection

Generell kann man ein Intrusion Detection System als eine Art Alarmanlage ansehen. Sollte ein Einbruch vom System entdeckt werden, wird der Administrator benachrichtigt (über Pager, SMS, eMail, WinPopup-Message, etc.) und kann dann entscheiden, wie er mit dem Problem umgeht. Im Extremfall wird er sich entscheiden, das System herunterzufahren oder in den Single-User-Mode bringen, um es keinem weiteren Missbrauch auszusetzen und sich ungestört der Behebung des ausgenutzten Fehlers widmen zu können.

Die Konfiguration solcher Systeme ist sehr vielfältig und kann je nach Ausführung auch sehr umfangreich werden. Die grössten Unterschiede liegen wohl darin, ob man ein solches IDS für einen einzelnen Rechner einsetzen möchte oder ein ganzes Netz absichern will. Bei der Konfiguration ist es wichtig, dass die Integrität der Konfigurationsdaten gesichert ist und das im Falle eine Einbruchsmeldung selbige auch garantiert wird, das heisst, dass es sich um einen echten Angriff handelt und nicht etwa um einen Fehlalarm, der als Folge einer Fehlkonfiguration entstand. Nach der Installation eines IDS im Netzwerk wird der Administrator die ersten Wochen nach der Installation mit der Analyse des Netzwerkverkehrs verbringen, um die normale Kommunikation des zu sichernden Netzes von Gefährdungen desselben zu unterscheiden. Tut er dies nicht und verlässt sich der Einfachheit halber auf frei erhältliche Konfigurationen oder auf die Standardkonfiguration, kann es

zu sogenannten False Positives oder auch False Negatives kommen. Die Gefahren einer Fehlkonfiguration werden in Abschnitt 4 genauer beschrieben.

1.2 Funktionalität eines IDS

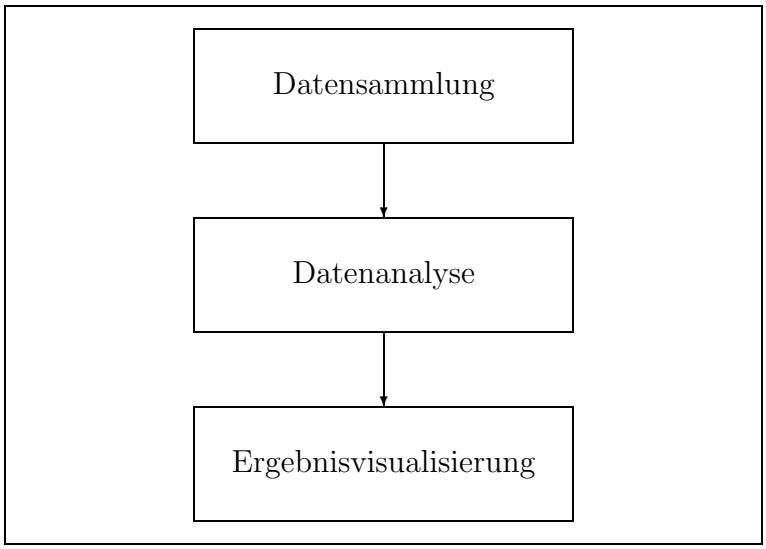

Abbildung 2: Funktionsschritte eines IDS

Im Allgemeinen erledigt ein IDS seine Arbeit in drei Schritten, wie Abbildung 2 veranschaulicht. Im Nachfolgenden werden die Schritte im einzelnen erläutert.

1.2.1 Datensammlung

Im ersten Schritt werden Daten gesammelt. Am Beispiel eines Network Intrusion Detection Systemes handelt es sich hierbei um die Pakete, die das Netzwerk passieren. Somit verfolgen solche Systeme einen präventiven Ansatz, da sie einen Angriff erkennen können während er stattfindet und gegebenenfalls Gegenmassnahmen treffen können. Aber auch Protokolldaten, die sowohl vom Betriebssystem als auch von der auf dem Host laufenden Software abgelegt werden, können als Grundlage für die Entdeckung von Einbrüchen dienen. Sie bieten aber nur Informationen der Applikationsebene und lassen im Idealfall nur auf Angriffe auf bestimmte Programme schliessen. Dieses Vorgehen wird als reaktionäres Intrusion Detection bezeichnet, da es einen Angriff nur im Nachhinein erkennen lässt und der Administrator nur für die Zukunft präventive Massnahmen treffen kann. Ausserdem ist in gewissem

Masse auch die Vergabe von Betriebsmitteln durch das Betriebssystem an die laufenden Prozesse zur Auswertung interessant. Hierzu zählen mitunter die Auslastung der CPU, belegter Speicher, aktive Netzwerkverbindungen, usw.

1.2.2 Datenanalyse

Im zweiten Schritt werden diese Daten vom System analysiert, um dem Administrator in einem dritten Schritt Angriffe benutzergerecht aufzuzeigen. Für die Technik des Erkennungsprozesses existieren zwei Möglichkeiten. Die erste ist die Missbrauchserkennung, die anhand vordefinierter Muster Einbrüche zu erkennen versucht. Hier werden etwa Netzwerkpakete mit Hilfe von vorgegebenen Signaturen überprüft und mit Pattern-Matching Angriffe erkannt. Mit dieser Methode arbeitet auch Snort. Über das Internet kann man auf eine grosse Anzahl von Konfigurationen für eine Unmenge an Angriffssignaturen zurückgreifen. In Abschnitt 2.2 wird darauf näher eingegangen. Daraus lässt sich schliessen, dass die Voraussetzung für die Missbrauchserkennung eine für den stattfindenden Angriff passende Signatur ist und dass das IDS selbstverständlich Zugriff auf diese Signatur hat.

Exkurs: Der letzte Abschnitt führte den Begriff der Signatur ein. Hier bezog sich die der Begriff auf den Fingerprint eines Angriffes. Jeder Angriff erzeugt bestimmte Pakete, die sich entweder durch einen String auf Applikationsebene, wie z.B. `/cgi-bin/phf` *für ein CGI-Probe, durch eine bestimmte Konstellation gesetzter TCP-Flags, wie z.B. der XMas-Tree-Scan, bei dem alle Flags gesetzt sind, oder ähnliche Merkmale auszeichnen. So kommt die Signatur zustande, die jeden Angriff auszeichnet. Ähnliche Bedeutung hat die Signatur, anhand derer ein IDS einen Angriff erkennt. Eigentlich stellt diese Signatur nur das Ebenbild des Fingerprints eines Angriffs dar, allerdings in der Form, in der sie das IDS verarbeiten kann. Ähnlich den Regeln eines Paketfilters, anhand der Pakete passieren lässt oder nicht, benötigt ein IDS eine Reihe von Signaturen, i.e. Regeln, um die entsprechenden Angriffe zu erkennen. Für perfekte Verwirrung sorgt hier noch die digitale Signatur, auf die aber erst später eingegangen wird.*

Eine andere Technik der Analyse ist die Anomalieerkennung. Sie stellt eine Art Heuristik dar und versucht, auch unbekannte Angriffe zu erkennen. Eine Anomalie wäre am Beispiel eines Network Intrusion Detection Systemes eine Abweichung vom normalen Netzwerkverkehr. Natürlich stellt eine Anomalie nicht immer eine Gefahr dar, weswegen eine längere Einlaufzeit für das System wie weiter oben schon erwähnt von elementarer Bedeutung ist, um ihm den normalen Netzwerkverkehr "beizubringen". In einem anderen Umfeld sind hier natürlich auch Dinge wie die Auslastung der CPU von

Bedeutung, da das System hier auf Dauer von einem Durchschnittswert ausgehen können muss, um eine Anomalie zu erkennen. Dieses Vorgehen verfolgt somit statistische Ansätze. Wenn ein zu überwachender Parameter ausserhalb der definierten Akzeptanzschwellen liegt, wird Alarm ausgelöst.

Eine zweite Herangehensweise der Anomalieerkennung verfolgt logische Ansätze. Hierbei wird im Gegensatz zum statistischen Ansatz die zeitliche Abfolge von Ereignissen in Betracht gezogen. Dieser logische Ansatz betrachtet bestimmte Ereignisfolgen als typisch. Beobachtet das System den Anfang einer solchen Ereignisfolge, erwartet es, dass auch der Rest dieser Ereignisfolge abläuft. Passiert dies nicht, schlägt das System Alarm.

1.2.3 Ausgabe der Ergebnisse

Die Darstellung des Ergebnisses der Analyse geschieht dann je nach Erkennungstechnik. Die Ergebnisse der Missbrauchserkennung können in einer einfachen Ja/Nein-Darstellung veranschaulicht werden. Wurde ein Angriff mit Hilfe einer entsprechenden Signatur erkannt, wird dies entsprechend veranschaulicht. Hierzu sei noch angemerkt, dass man nicht immer genau zwischen einer Missbrauchserkennung und einer Anomalieerkennung differenzieren kann. So existieren auch siganturbasierte Intrusion Detection Systeme, die für bestimmte Angriffe Schwellwerte bieten. So muss ein Network Intrusion Detection System zum Beispiel auch einen Portscan erkennen, bei dem die Pakete in längeren Abständen gesendet werden.

Die Ausgabe der Ergebnisse beschränkt sich natürlich nicht auf die Aufbereitung der Daten für die lokale Einsicht, zum Beispiel über ein Web-Interface wie ACID[1] ([8]), sondern muss auch die entsprechende Benachrichtigung des Administrators in geeigneter Form beinhalten.

1.3 Warum Intrusion Detection?

Die Gründe, warum man sein Netzwerk oder auch nur einen Einzelrechner, der wichtige Aufgaben im Firmenumfeld hat, mit einem Intrusion Detection System absichern sollte, liegen heutzutage auf der Hand. Kaum ein Tag vergeht, an dem man nicht von neuen Hackversuchen oder Einbrüchen in Firmennetzwerken hört. So manche Kreditkartennummer hat durch Einbruchsversuche schon den Weg in die Finger böswilliger Cracker gefunden. Die Ursachen für einen möglichen Einbruch können sehr vielfältig sein. Der Angreifer kann zum Beispiel über Fehler in der Implementierung des TCP/IP-Stacks Zugriff auf das System oder dessen Ressourcen erhalten. Ein Profi findet

[1] ACID steht für Analysis Console for Intrusion Databases und ist am CERT Coordination Center entwickelt worden.

anhand einer Untersuchung des TCP/IP-Fingerprints, welchen ein System hinterlässt, heraus, um welches System es sich handelt und kann daraus auf etwaige Fehler in der Implementierung schliessen, um diese dann letzten Endes auszunutzen. Ein Einbruch kann aber auch über Fehler in der auf dem System verwendeten Software erfolgen. Häufige Ursache für solche Fehler sind die sogenannten Buffer Overflows. In letzter Zeit sind immer mehr Fehler in Programmen, seien es Buffer Overflows oder Fehler, durch die ein normaler Benutzer root-Rechte erlangen kann, bekannt geworden. In letzter Zeit hat hier vor allem der DNS-Dämon BIND, welcher der Quasi-Standard unter Unix und seinen Derivaten ist, der FTP-Server Wu-FTP und natürlich der Klassiker, der Microsoft Internet Information Server, von sich reden gemacht. Durch die ersten beiden Programme gelangten die ersten Würmer[2] in die Unix-Welt. Mittlerweile gibt es sogar einen Wurm, der eine Kombination aus Sicherheitlöchern im Microsoft IIS und BIND ausnutzt.

Es mag so scheinen, als ob Einbruchsversuche grundsätzlich nur in Netzwerken geschehen. Tatsächlich ist prinzipiell jeder Portscans u.ä. ausgeliefert, der sich z.B. über sein Modem ins Internet einwählt. Aber eine IP-Adresse, von der man weiss, dass die dahintersteckenden Rechnersysteme häufig wechseln, ist nicht halb so interessant wie ein System, welches dauerhaften Kontakt zum Netzwerk hat oder gar mehrere Systeme in einem Netzwerk, die für einen der gefürchteten Distributed Denial of Service Attacken genutzt werden könnten. So sind zum Beispiel die IP-Adressen, die den DSL-Anschlüssen der deutschen Telekom zugeteilt sind, beliebtes Ziel für Portscans, da die Benutzer, die hinter der IP-Adresse stecken, oftmals tagelang ununterbrochen eine Verbindung zum Internet haben, womit sie sich sehr gut als “Mittäter” eines Distributed Denial of Service eignen.

Letztenendes ist eine Kette immer nur so stark wie ihr schwächstes Glied, was zum Benutzer des Systems führt. Oftmals sind Einbruchsversuche auf falsch gewählte Passwörter zurückzuführen, was einen Einbruch in ein Rechnersystem erheblich erleichtert. Hat ein Einbrecher erstmal einen Zugang zum System, ist es ein leichtes, zu noch höheren oder gar den absoluten, den root-Rechten auf einem Rechner zu gelangen. Läuft auf einem System beispielsweise der finger-Dämon, besteht schonmal die Möglichkeit, an etwaige Benutzerkennungen zu kommen, um mit deren Hilfe mit einem Dictionary bekannte Passwörter zu überprüfen.

Der Angreifer kann natürlich auch über Social Engineering versuchen, an das Passwort des Benutzers zu kommen. Man hört immer wieder von Fällen, in denen dreiste Personen unter einem Vorwand oder mit einer falschen Iden-

[2]Ein Wurm ist ein Programm, dass sich meist über Sicherheitslöcher von Host zu Host ausbreitet und mehr oder weniger Schaden anrichtet.

tität versucht haben, einen Benutzer zur Herausgabe seines Passwortes zu bewegen.

Ein Intrusion Detection System kann einen Login eines normalen Benutzer nicht als Angriff erkennen, auch wenn eine andere Person dahintersteckt. Das IDS kann hier nur noch etwaige ungewöhnliche Aktivitäten erkennen, die in nächster Zukunft von diesem Login ausgehen würden.

1.4 Eingliederung in die lokale Netzwerkstruktur

Ein Intrusion Detection System wird oftmals mit einer Firewall in Verbindung gebracht oder gar mit selbiger verwechselt. So hat ein Intrusion Detecion System nicht die Aufgabe, eine Firewall zu ersetzen, sondern stellt eine sehr sinnvolle Ergänzung zu ihr dar. Eine Firewall legt Regeln fest, welcher Netzverkehr nach aussen bzw. nach innen dringen darf und ein Intrusion Detection System kann überprüfen, ob diese Regeln auch wirklich eingehalten werden.

Im Zusammenhang mit einem Netzwerk stellt sich oftmals die Frage, an welcher Stelle das Intrusion Detection System greifen soll. Wird es vor der Firewall positioniert, kann es den hereinkommenden Verkehr auf etwaige Angriffe überprüfen. Heutzutage stellt aber nicht mehr nur der Verkehr, der von aussen in ein Netzwerk dringt, eine Gefahr für selbiges dar, sondern auch der Verkehr im Innern des Netzwerkes. Der Hauptgegner lauert oftmals nicht mehr nur im Internet oder generell ausserhalb des lokalen Netzes. Oftmals sind es die eigenen Mitarbeiter, Fremde, die sich Zugang zum Netz verschafft haben oder Würmer, Trojaner o.ä., die eine Gefahr für das Netz darstellen. Daraus ergibt sich ein weiterer Faktor, der bei der Positionierung des IDS zu beachten ist. Für die Positionierung des IDS im Innern des Netzwerks spricht noch eine andere Tatsache. Grundlage eines jeden Netzwerkes, vor allem in Firmen, sollte eine vor der Installation von Firewalls und Intrusion Detection Systemen festgelegte Sicherheitspolitik sein. Existiert diese nicht, ist fraglich, was für einen Sinn eine Firewall oder ein IDS machen würde. Davon ausgehend, dass eine existiert, wird mit einer Firewall festgelegt, welcher Verkehr aus einem anderen Netz, sei es das Internet oder jegliche Art externer Verbindungen aus anderen Netzen, die überwacht werden sollen, in das eigene gelangen soll. Die Firewall stellt damit eine Art Wiederspiegelung der Sicherheitspolitik dar. Sollte diese verletzt werden und ein Angreifer seinen Weg in das lokale Netz finden, so hat ein IDS, welches vor der Firewall positioniert ist, keine Chance, dieses Vorgehen zu entlarven. Generell kann es zwar erkennen, dass ein Angriff an der Firewall angekommen ist, aber nicht, ob er wirklich abgeblockt wurde. Somit würde ein Intrusion Detection System, welches den Verkehr, der von der Firewall in das interne Netz gelangt,

analysieren kann, solch einen Angriff erkennen. Somit sollte ein IDS als vor der Firewall positioniert werden und eine hinter der Firewall am Eingang zum lokalen Netz. Abbildung 3 verdeutlicht dies. Mit dieser Konstellation können Angriffe sowohl von innen als auch von aussen erkannte werden.

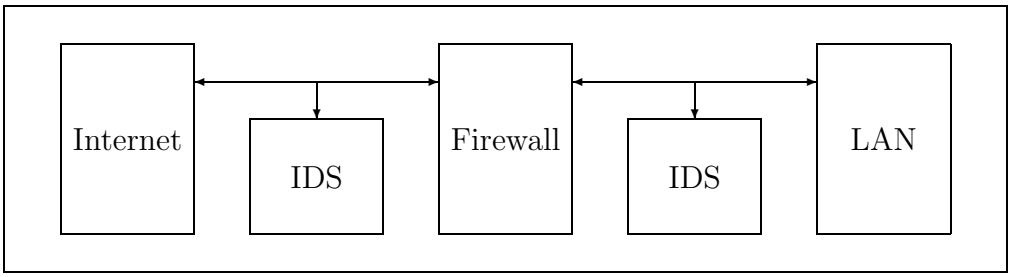

Abbildung 3: Positionierung des IDS

Zum Einsatz von Intrusion Detection Systemen muss allgemein noch etwas angemerkt werden. Wie ein Packet Sniffer muss auch ein IDS die Netzwerkkarte in den sogenannten "promiscuous mode"[3] schalten, um den gesamten Netzverkehr analysieren zu können. Normalerweise aktzeptiert eine Netzwerkkarte nur Pakete, die an sie gerichtet sind. Alle anderen werden einfach ignoriert. Der "promiscuous mode" sorgt dafür, dass die Karte auch alle anderen Pakete akzeptiert. Wäre dies nicht möglich, könnte ein IDS nur Angriffe auf den Host, auf dem es installiert ist, erkennen.

Aus dieser Tatsache ergibt sich noch ein anderes Problem der Positionierung. In einer geswitchten Netzwerkumgebung existieren normalerweise nur Punkt-zu-Punkt-Verbindungen zwischen den einzelnen Hosts. Hängt das IDS an einem normalen Port eines Switches, kann es somit nicht auf den gesamten Verkehr des Netzwerkes zugreifen, sondern hat wiederum nur Zugriff auf Pakete, die an seinen Host adressiert sind. Viele Switches bieten allerdings Monitoring-Ports, auf die der gesamte Verkehr, der den Switch passiert, geleitet wird. Es versteht sich von selbst, dass ein IDS-Host an einen solchen Port angeschlossen werden sollte.

1.5 Praktischer Einsatz und Effizienz von ID Systemen

Da das Gebiet der Intrusion Detection noch sehr jung ist, ist es nicht sehr einfach, ausreichend über den praktischen Einsatz zu berichten. Dies gilt vornehmlich für den Einsatz von Systemen auf Basis der Anomalieerkennung. Systeme wie Snort, die mit Missbrauchserkennung arbeiten, sind hingegen schon häufig und auch erfolgreich im praktischen Einsatz zu finden. Beide

[3] promiscuous (*engl.*): vermischt, aber auch sexuell freizügig ;)

Systeme unterscheiden sich in der Effektivität der Angriffserkennung. Gibt es beim Einsatz einer Missbrauchserkennung eine zum Angriff passende Signatur, so steigt der Aufwand zur Erkennung exponentiell zum Effektivitätsgrad des Angriffs. Ein System mit Anomalieerkennung verhält sich genau entgegengesetzt. Hier wächst der Aufwand zur Erkennung nur exponentiell zum Effektivitätsgrad des Angriffes. Daraus lässt sich schliessen, dass die Anomalieerkennung effektivere Angriffe mit einem geringeren Aufwand als die Missbrauchserkennung registrieren kann.

Ein IDS im allgemeinen und ein Network IDS im besonderen muss natürlich so wenig wie möglich Angriffsfläche bieten. Die Alarmanlage stellt oft das erste Ziel eines Angriffs dar. Ist sie erstmal ausgeschaltet, kann ein Hacker ungestört arbeiten und kann nur noch durch etwaige Aktivitäten ausgemacht werden, die Meldungen in den Protokollen des entsprechenden Systems erzeugen, sofern er sie nicht selbst entfernt hat. Darum heisst es auch bei einem Intrusion Detection System: So viele Dienste wie nötig aber so wenig wie möglich. Je weniger Ports offen sind, je weniger Dienste auf dem IDS-Host laufen und je weniger Benutzerkonten auf dem Host existieren, umso resistenter ist er gegen Angriffsversuche. Eine Absicherung der nötigen Benutzerkonten und vor allem dem des Administrators durch entsprechende Passwörter versteht sich von selbst. Ein Zugriff auf das Benutzerkonto des Administrators ermöglicht es dem Angreifer, ohne grosse Probleme, das IDS zu deaktivieren. Weiterhin muss dem IDS der Zugriff auf seine Konfigurations- und Signaturdateien gewährleistet werden. Vorsicht ist auch geboten beim protokollieren der Angriffe. Es muss gewährleistet werden, dass sich das IDS nicht durch überlaufende Protokolldateien selbst deaktiviert, was durchaus eine Folge eines gezielten Angriffs auf das IDS sein könnte und so einen Denial of Service als Folge hätte.

Für den effektiven Einsatz eines Systemes mit Anomalieerkennung ist es elementar, dass es über Informationen verfügt, wie das zu überwachende System, z.B. ein Netzwerk, im Normalfall funktioniert, d.h. welche Aktivitäten für gewöhnlich stattfinden. Dafür ist eine entsprechende Einarbeitungszeit erforderlich, um das System in die entsprechenden Umstände “einzugewöhnen”, damit es lernt, in welchem Zusammenhang normalerweise Ereignisse stehen, um in Zukunft Abweichungen zu erkennen. Die Vorteile liegen dabei auf der Hand, nämlich dass ein System mit Anomalieerkennung unabhängig davon, welcher konkrete Angriff gerade stattfindet, einen solchen erkennen kann. Ein System auf Basis der Missbrauchserkennung ist auf aktuelle Signaturen angewiesen. Wird die Signaturdatenbank nicht auf einem aktuellen Stand gehalten, nutzt das System auf Dauer wenig. Eine schwellwertgesteuerte Signaturanalyse erfordert zusätzlich zur reinen Signaturdatenbank noch eine Definition der Schwellwerte, die den Akzeptanzbereich definieren.

1.6 Gegenmassnahmen und rechtliche Aspekte

Sollte es nun irgendwann einmal zu einem Angriff kommen, ist die Frage, wie man mit den gesammelten Daten umgeht. Nutzt man die Daten, um einen Gegenangriff einzuleiten, sollte man sich die Frage stellen. was man letztenendes damit bewirken wird und ob dies wirklich den Sinn eines Intrusion Detection Systems darstellt. Zieht man nun den Zorn des angreifenden Hackers bzw. Crackers[4] auf sich oder erzielt man mit einer Massnahme gegen den Angreifer tatsächlich eine abschreckende Wirkung? Letzteres dürfte wohl eher selten der Fall sein, denn mit einem Gegenangriff zieht man nur unnötig die Aufmerksamkeit des Angreifers auf das Intrusion Detection System selbst, wenn er selbiges nicht schon angegriffen hat. Im Idealfall zeigt das IDS also überhaupt keine Reaktion nach aussen, sondern geht seiner eigentlichen Aufgabe nach, benachrichtigt den Administrator und sammelt entsprechend Daten über den Angreifer für eine spätere Auswertung zum Einleiten etwaiger rechtlicher Schritte.

Sollte eine Einleitung von Gegenmassnahmen aber dennoch notwendig sein, so sollten diese in einem Masse geschehen, welches dem System oder dem zu überwachenden Netzwerk nicht den Zorn des Angreifers auf sich zieht und diesen zu weiteren Angriffen anstachelt. Hierbei ist eine vorrübergehende oder im Extremfall auch dauerhafte Sperrung der IP-Adresse des Angriffsursprungs denkbar. Letzteres macht aber bei dynamisch vergebenen IP-Adressen wenig Sinn. Ausserem kann nicht garantiert werden, dass der Angriff wirklich von der verwendeten Adresse aus stattfindet, da der Angriff durchaus eine Kombination aus IP-Spoofing, der Verfälschung der eigenen IP-Adresse, und der entsprechenden Angriffstechnik sein kann, was die Identifizierung des Angreifers stark erschwert, da es erstens nahezu unmöglich ist, ein IP-Spoofing zu erkennen, und zweitens die Identifizierung des tatsächlichen Ursprungs des Angriffs dadurch nicht mehr gewährleistet werden kann. Andere Möglichkeiten sind z.B. die Sperrung es entsprechenden UDP/TCP-Ports oder die Terminierung des betroffenen Programms.

Diese Aktionen fallen in den Bereich der Intrusion Response Systeme und sollen hier nicht näher erläutert werden, zumal sie für die Beschreibung von eventuellen Massnahmen schon genügend Aussagekraft haben. Manche Intrusion Detection Systeme bieten auch die Möglichkeit, mit einem RST-Paket die aktive TCP-Verbindung zu beenden, aber auch dies kann wiederum unnötig die Aufmerksamkeit des Angreifers auf das IDS ziehen, da dieser dann merken könnte, dass etwas anderes als der Host, den er angreifen wollte,

[4]Die Erwähnung beider erscheint den Autoren politisch korrekter, denn die Medien haben den Begriff Hacker in letzter Zeit stark in den Schmutz gezogen. Eine hilfreiche Defition und zusätzliche Informationen hat Eric S. Raymond unter [6] zusammengestellt.

die Verbindung beendet hat.

Die gesammelten Auditdaten stellen nach § 416 der Zivilprozessordnung kein rechtsverbindliches Beweismittel dar wie zum Beispiel ein notariell beglaubigtes Schriftstück. Die Daten unterliegen der freien Beweiswürdigung, womit sie den gleichen gerichtlichen Status wie eine Zeugenaussage haben und ihre Beurteilung im Ermessen des Gerichtes liegt. Der Grund für den Status eines nicht rechtverbindlichen Beweismittels liegt in der Tatsache, dass die Auditdaten nachträglich modifiziert und manipuliert werden können. Eine mögliche Umgehung liegt in der digitalen Signierung der Auditdaten durch das IDS, um die Integrität der Daten zu sichern. Doch dafür muss die digitale Signatur[5] vertrauenswürdig sein und der entsprechende private Schlüssel sicher aufbewahrt werden. Dieser Umstand muss im Falle eines Falles auch nachgewiesen werden.

1.7 Das IDS als Angriffspunkt

Um seinen Dienst in einem Netzwerk zuverlässig zu verrichten, muss ein Intrusion Detection System auch gewisse Voraussetzungen erfüllen, um nicht selbst Opfer eines Angriffs zu werden. Hiermit sind vornehmlich Probleme gemeint, die das IDS im Zusammenhang mit den im zu überwachenden Netzwerk installierten Hosts haben kann. Hier gibt es allgemein zwei Techniken, die es ermöglichen, das IDS mit falsch erscheinenden Paketen zu verwirren oder Inkonsistenzen zwischen dem IDS und den Endsystemen im lokalen Netzwerk auszunutzen. Erstere Technik wird “Insertion”[6] genannt und die letztere “Evasion”[7].

1.7.1 Insertion

Insertion basiert darauf, dass ein IDS Pakete akzeptiert, die das Endsystem normalerweise nicht annehmen würde. Das Intrusion Detection System geht also davon aus, dass ein Paket vom entsprechenden Host akzeptiert wird, obwohl der Host das Paket ablehnt. Kein anderer Host im Netzwerk ausser dem IDS kümmert sich also um die Pakete. Der Angreifer fügt quasi Daten in das IDS ein. Generell liegt bei vielen Intrusion Detection Systemen eine grosse Anfälligkeit gegenüber dieser Angriffstechnik. Ein Angreifer könnte “Insertion” benutzen, um signaturbasierte ID Systeme zu umgehen und einen Angriff durchführen, ohne dass dieser vom IDS bemerkt würde, da die Pakete vom

[5] Die erwähnte digitale Signatur ist nicht zu verwechseln mit der Signatur eines Angriffs. Hier ist eine Signatur im kryptografischen Sinne gemeint.

[6] Insertion (*engl.*): Einfügen, Einfügung.

[7] Evasion (*engl.*): Umgehung, Ausweichung.

System als ungefährlich eingestuft wurden und so ungehindert am IDS vorbei zum Zielhost gelangen. Diese Angrifftechnik basiert auf der Tatsache, dass ID Systeme, die signaturbasiert arbeiten, Pakete mit Pattern-Matching überprüfen. Schafft es der Hacker nun aber, dem IDS einen anderen Strom von Daten zukommen zu lassen als dem Zielsystem, da das IDS Pakete akzeptiert, die das Zielsystem normalerweise nicht akzeptieren würde, könnte er beispielsweise eine Reihe von Paketen mit jeweils nur einem Character schicken, welche zusammengesetzt im IDS einen anderen String ergeben als im Zielsystem. Der Angreifer könnte beispielsweise den String "ATTAXCK" senden. Der Angreifer hat aber das Paket mit dem Character "X" so manipuliert, dass der Zielhost es ablehnen wird, die restlichen Pakete aber akzeptiert. Beim Zielhost wird also letztenendes der String "ATTACK" eintreffen, das IDS aber findet nur den String "ATTAXCK", welcher dann nicht als Angriff erkannt wird, da in der Signaturdatenbank nur der String "ATTACK" gefunden wird. Der Angreifer hat es also geschafft, Daten einzufügen und einen Angriff ohne Bemerken durch das IDS durchzuführen. Eine Lösung für dieses Probleme wäre die Anpassung des IDS an die Regeln, nach denen das Zielsystem Pakete akzeptiert oder sie fallenlässt.

1.7.2 Evasion

Die zweite Möglichkeit, "Evasion", basiert auf der Möglichkeit, dass ein IDS Pakete ablehnt, die ein Host im Netzwerk akzeptieren würde. Das Prinzip ist also prinzipiell das gleiche wie bei "Insertion": Das IDS sieht einen anderen Strom von Daten als das Zielsystem. Greift man wieder auf Beispiel mit dem String "ATTACK" zurück, würde das IDS also bei einem Strom aus Paketen mit jeweils nur einem Character bei Manipulation des entsprechenden Paketes mit dem zweiten "A" den String "ATTCK" erkennen, wohingegen beim Zielsystem der String "ATTACK" ankommt. Diese Umgehung des IDS basiert auf der derarten Veränderung des Paketes, welches das entsprechende "A" enthält, so dass das IDS es ablehnt, das Zielsystem es aber akzeptiert.

Im realen Einsatz stellen "Insertion"- und "Evasion"-Attacken keine einfache Lösung dar, um ein IDS zu umgehen. Dafür müsste ein Angreifer die Möglichkeit haben, Pakete in einen Datenstrom einzufügen.

2 Warum Snort?

2.1 Freie Verfügbarkeit

An dieser Stelle sollen nicht die Vorteile von Open Source Software diskutiert werden. Dieser oftmals aussichtslose K(r)ampf hat schon Freundschaf-

ten gespalten und zu sinnlos langen Diskussionen geführt. Vielmehr ist die freie Verfügbarkeit von Snort in Bezug auf Intrusion Detection Systeme ein wichtiger Kostenfaktor. Hier bietet Snort gegenüber kommerziellen Intrusion Detection Systeme einen klaren Vorteil, nämlich die Tatsache, dass nur die Kosten für die Installation, Konfiguration und Pflege des Systems zu beachten sind. Kommerzielle Systeme bringen zusätzlich noch den Kostenfaktor des Kaufes mit sich.

Zudem stellt sich die Frage, ob ein Mitarbeiter der eigenen Firma in der Lage ist, das kommerzielle System mit vorhandenener Dokumentation zuverlässig einzurichten. Mit Snort dürfte dies kein Problem mehr sein, denn dafür wurde schliesslich dieser Artikel geschrieben ;). Sollte es mit Snort allerdings derartige Probleme geben, kann man die Firma Silicon Defense unter [7] kontaktieren und kommerziellen Support für Snort anfordern. In Anbetracht dessen, dass man nicht unbedingt für jedes zukünftige Update von Signaturen auf kommerziellen Support zurückgreifen möchte, wo es doch ein leichtes ist, eigene Signaturen für Snort zu entwickeln, sollte man aber vielleicht doch in Erwägung ziehen, sich näher mit Snort auseinanderzusetzen. Die dafür investierte Zeit und natürlich das investierte Geld dürften sich im Nachhinein auf jeden Fall auszahlen.

Andere Entscheidungsgründe für oder gegen Snort werden wahrscheinlich auch von den Gegebenheiten des entsprechenden Netzwerkes abhängig sein. Ein Administrator, der unter Windows zu Hause ist, wird wohl eher einem IDS mit einer schönen bunten Oberfläche den Vorzug geben, da er es so oder so gewöhnt es, hier und da ein Häkchen anzuklicken und die Sache läuft[8], obwohl es Snort neben anderen Architekturen u.a. auch für Windows gibt. Für einen unter Unix und seinen Derivaten beheimateten Administrator wird die Wahl wohl nicht schwer sein, da er die Kommandozeile sowieso lieben gelernt hat und in Snort den richtigen Partner finden wird. Wie bei so vielen Software-Entscheidungen gilt auch hier der Grundsatz: Puristisch, aber leistungsfähig oder umfangreich und eventuelle Performance-Einbußen. Wobei hier den kommerziellen Systemen nicht ihre Leistungsfähigkeit abgesprochen werden soll, aber wir sind der Ansicht, dass grösserer Umfang nicht unbedingt ein Mehrwert ist, sondern eher zu höheren Ansprüchen führen kann, und Hardware ist letztenendes auch ein Kostenfaktor, der bei der Installation eines Intrusion Detection Systemes entscheidend ist.

Was den Vergleich mit seinen kommerziellen Gegnern angeht, braucht sich Snort nicht hinter diesen zu verstecken. Snort wurde für einen guten Datendurchsatz und hohe Leistung optimiert und wird deswegen auch als

[8] Der Leser verzeihe den Autoren den Sarkasmus, aber wir haben schon genug Erfahrung mit solchen Leuten gemacht. Hoch lebe die MCSE-Gehirnwäsche ;)

Lightweight Intrusion Detection System bezeichnet (siehe [5]). Zwar bietet es keine so schönen Oberflächen für die Visualisierung, aber hier wurde vom CERT mit ACID und von der Firma Silicon Defense mit SnortSnarf Abhilfe geschaffen.

2.2 Update der Signaturen

Das Update der Signaturen für neue Angriffe ist im allgemeinen nicht nur eine Kostenfrage, sondern auch eine Frage der Geschwindigkeit. Snort kann mit einer mittlerweile grossen Community aufwarten, die dafür sorgt, dass für neue Angriffe innerhalb von Stunden nach Bekanntwerden die entsprechenden Signaturen verfügbar sind. Zudem kann ein Administrator, der sowohl mit dem Entwerfen von Rules für Snort vertraut ist und sich auch mit den entsprechenden Angriffssignaturen auskennt, schnell und ohne grösseren Aufwand eigene Signaturen schreiben. So ist also bei der Installation eines IDS wichtig, festzustellen, wie schnell der Hersteller mit entsprechenden neuen Signaturen aufwarten kann. Es ist möglicherweise inakzeptabel, dass die Aktualisierungen zwei bis drei oder mehr Tage benötigen, wenn sich doch mit Snort eine derart unkomplizierte und schnelle Möglichkeit der Aktualisierung bietet. Ausserdem sollte man sich bei solch einer Entscheidung auch über Kosten, die die Updates vom Hersteller verursachen, im Klaren sein und diese mit dem Arbeitsaufwand, den Snort hier verursacht, aufwiegen.

3 Snort und seine Möglichkeiten

3.1 Analysemöglichkeiten

Snort arbeitet bei der Analyse der gesammelten Daten nach dem Prinzip der Missbrauchserkennung, welches in Abschnitt 1.2.2 erklärt wurde. Die zahlreichen Optionen von Snort erlauben Zugriff auf alle wichtigen Bestandteile der zu untersuchenden Pakete. Snort kann anhand der Quell- und Ziel-IP-Adressen, Quell- und Zielports, TCP-Flags, des Datenteils und noch diverser anderer Merkmale Pakete analysieren. Welche das im Ganzen sind, zeigt Abschnitt 6. Den Datenteil eines Paketes kann Snort anhand von Binärmustern in Form von Hexadecimal-Code analysieren, oder mit Pattern-Matching auf Strings untersuchen.

Sollte man doch nach einer Möglichkeit der Anomalieerkennung suchen, bietet Snort über seine Plugin-Schnittstelle das Plugin Spade[9], welches von

[9]Statistical Packet Anomaly Detection Engine

Silicon Defense ([7]) entwickelt wird. Es bietet eine statistische Anomalieerkennung, ist aber noch in seinem frühen Entwicklungsstadium. Nichtsdestotrotz bietet Spade schon eine beachtliche Stabilität und wird im praktischen Umfeld schon erfolgreich eingesetzt. Allerdings wird empfohlen, für den Einsatz von Spade einen gesonderten Snort-Prozess laufen zu lassen, der nur für den Zweck der Anomalieerkennung konfiguriert ist.

Im allgemeinen bietet die Plugin-Schnittstelle von Snort die Möglichkeit, Pakete zu analysieren und gegebenenfalls zu verändern, noch bevor die eigentliche Analyse von Snort die Pakete untersucht.

Snort bietet noch diverse andere Präprozessoren an, zum Beispiel für die Erkennung von Portscans. Mehr dazu folgt in Abschnitt 6.2.1.

3.2 Alarmierungs- und Logmöglichkeiten

Snort bietet diverse Möglichkeiten, auf einen erkannten Einbruch aufmerksam zu machen. Sie lassen sich entweder einzeln oder kombiniert nutzen, aber mehr dazu in Abschnitt 6.1.1 und Abschnitt 6.2.2. Die erste Möglichkeit, quasi der Klassiker, ist die Benachrichtigung über *syslog*. Soll Snort Alarmmeldungen in seinem eigenen Logfile protokollieren, so kann es dies auf zwei Arten tun, entweder schnell oder vollständig. Die erste Methode empfiehlt sich bei hohem Netzaufkommen, da hier nicht alle Paket Header gesichert werden. Daraus ergibt sich schon, was die zweite Methode tut, nämlich alle Fehler samt kompletten Paket Headern gesichert. Die letztere Methode ist allerdings dadurch auch erheblich langsamer, da hier ein nicht zu verachtender Aufwand für die Formatierung der entsprechenden Daten betrieben werden muss.

So man als Administrator ein Fan von WinPopup-Messages ist, kann man auf die Alarmierung per SMB-Message zurückgreifen und sich so über einen Angriff oder Einbruch benachrichtigen lassen. Dies bietet sich vor allem dann an, wenn der Administrator unter Windows zu Hause ist, aber auch mit Samba lassen sich diese Nachrichten sinnvoll auswerten.

Ein noch experimentelles Feature ist die Benachrichtigung über einen Unix-Socket. Hierfür erstellt Snort einen Socket, aus dem andere Programme dann entsprechend die Daten auslesen und auswerten können.

Neben den Alarmierungsmöglichkeiten, bietet Snort noch diverse Möglichkeiten, in denen es Paket-Informationen sichern kann. Dies kann im *tcpdump*-Format geschehen. Hier besteht der Vorteil in den vielen Programmen, die dieses Format verstehen und auswerten können. Eine andere Variante ist das Formatieren mit Hilfe von XML. Am CERT wurde die sogenannte SNML[10]

[10]Simple Network Markup Language

entwickelt, mit deren Hilfe Snort die Daten formatiert und in eine Datei oder eine Datenbank sichern kann.

Weiterhin kann Snort in eine Datenbank wahlweise loggen oder seine Alarmmeldungen absetzen. Mehr dazu folgt in Abschnitt 3.3.

3.3 Visualisierung der Ergebnisse

Nach einem erfolgten Angriff ist es elementar, die gesammelten Daten zu sichten, um sie für etwaige Massnahmen auszuwerten. Mit diversen Zusatzprogrammen ist es möglich, die Daten grafisch aufzuarbeiten. Hier stehen ACID und SnortSnarf an erster Stelle. ACID greift auf Daten zurück, die Snort in eine Datenbank loggt. Bis dato bietet Snort Unterstützung für Oracle, MySQL, PostgreSQL oder ODBC[11]. ACID basiert auf PHP[12] und bereitet die gesammelten Daten grafisch auf. SnortSnarf wurde von der Firma Silicon Defense ([7]) entwickelt und besteht aus einer Reihe von Perl-Scripten. Prinzipiell bereitet es die Daten auch auf, aber trotzdem ist der Output teilweise noch sehr kryptisch. Hier hat ACID die Nase vorn.

Neben der grafischen Aufbereitung steht dem Administrator natürlich nichts im Weg, wenn er selbst Einblick in die Logdateien nehmen will. Snort sammelt seine Daten in einem Verzeichnis und diversen Unterverzeichnissen, die nach der IP-Adresse benannt werden, von der der Angriff ausging. Daten über Portscans werden in einer gesonderten Datei gesichert.

3.4 Gegenmassnahmen

Snort bietet in beschränktem Maße Möglichkeiten der Intrusion Response. Über die *libnet* lässt sich die sogenannte Flexible Response nutzen. Somit kann man über die Rules von Snort Reaktionen auf Angriffe auslösen. Die Möglichkeiten erlauben eine explizite Beendigung der TCP-Verbindung über ein RST-Paket oder über ICMP. Bei letzterem besteht die Wahl zwischen “Net unreachable”, ein “Host unreachable”, ein “Port unreachable” oder allen dreien auf einmal.

Prinzipiell sei aber auch noch einmal in Frage gestellt, ob eine Response vonnöten ist. Mehr dazu findet sich in Abschnitt 1.6.

[11] ODBC steht für Open DataBase Connectivity und stellt ein standardisiertes Interface dar, welches eine Abstraktion von der dahinterstehenden Datenbank ermöglicht.

[12] PHP bedeutet *PHP: Hypertext Preprocessor* und ist eine freie und mittlerweile sehr verbreitete Scriptsprache.

4 Gefahren einer Fehlkonfiguration

Wie schon in Abschnitt 1.5 erwähnt, erfordert es einen gewissen Aufwand, das System in das lokale Netzwerk einzuarbeiten. Setzt man kein Network Intrusion Detection System, sondern ein Host Based Intrusion Detection System, so erfordert dies nichtsdestotrotz eine gewisse Einarbeitung in die Umstände des zu überwachenden Hosts. Daten, die den Standardbetrieb des Systems ausmachen, wollen gesammelt und Akzeptanzschwellen definiert werden. Je nach Art der Analyse sind unterschiedliche Methoden der Einarbeitung vonnöten. Für Missbrauchserkennung muss der Administrator entweder den Netzwerkverkehr, der normalerweise das Netzwerk passiert, kennen oder selbigen vor der Konfiguration des Intrusion Detection Systems analysieren, um eine möglichst effiziente Zusammenstellung von Signaturen zu finden. Anomalieerkennung erfordert eine quasi Eingewöhnung in die lokalen Umstände, um zu erkennen, welcher Netzverkehr sich in normalen Schranken bewegt und welcher vom normalen Verkehr abweicht. Hier ist eine relativ lange Eingewöhnungsphase nötig, da auch Verkehr, der vielleicht nur einmal im Monat standfindet, etwa durch ein Netzwerkbackup o.ä, berücksichtigt werden muss. Bei einem Intrusion Detection System, welches mit Missbrauchserkennung arbeitet, ist ein Fehlalarm nicht unbedingt dramatisch, da der Administrator schnell noch eine oder mehrere weitere Signaturen hinzufügen kann. Löst ein System, welches mit Anomalieerkennung arbeitet, einen Fehlalarm aus, kann es aufwändig sein, dem IDS den neuen Umstand, den es als normal erkennen soll, beizutragen.

Die schon erwähnten Fehlalarme werden in der Praxis mit dem Anglismus False Positives bezeichnet. Ein Angriff, der dem Intrusion Detection System verborgen geblieben ist, heisst demnach False Positive. Auf beide wird nachfolgend kurz eingegangen, da sie im Zusammenhang mit Intrusion Detection auf keinen Fall unter den Tisch gefallen lassen werden dürfen.

4.1 False Positives

Wie schon erwähnt handelt es sich bei den False Positives um Fehlalarme. Diese können die Folge einer nicht ausreichenden Konfiguration in Form der entsprechenden Signaturen sein oder dadurch entstanden sein, dass sich der Administrator auf ein Paket im Internet verfügbarer, auf bestimmte Netzwerkumstände angepasste Signaturdatenbanken verlassen hat.

Generell sollten False Positives mit dem Hinzufügen einer entsprechenden Signatur oder dem Anpassen der Anomaliedaten behoben werden, da sonst die Gefahr besteht, dass man sich an die falschen Alarmmeldungen gewöhnt. Das hat vielleicht zur Folge, dass man zwischen vielen Falschmel-

dungen einmal wichtige und tatsächliche Angriffe überliest. Zudem sollte man auch bedenken, dass man als Administrator eines Tages mal nicht im Hause ist und eine andere Person mit der Einsicht der Daten beauftragt. Dieser weiss vielleicht nicht, dass die entsprechende Fehlmeldung nicht beachtenswert ist und wird eventuell versuchen, Gegenmassnahmen einzuleiten oder informiert einfach seinen Vorgesetzten oder eine andere dritten Person. Man stelle sich die Standpauke vor, die einem Administrator dann blühen dürfte.

Zu guter Letzt darf man auch den Fakt nicht vergessen, dass ein Angreifer sich die Generierung eines False Positive zu Nutze machen könnte, indem er die Protokolldaten zum Überlaufen bringt. Dieser Gedanke mag einem zwar abwägig erscheinen, denn woher soll der Angreifer wissen, dass das System einen Fehlalarm auslöst, aber man sollte diesen Fakt trotzdem nicht ausser Acht lassen, denn wer weiss, über welche Wege Angreifer an ihre Informationen gelangen. Das Überlaufen der Protokolldaten zu verhindern ist zudem eine Frage, die schon beim Aufbau des Intrusion Detection Systems geklärt werden sollte, da ein Angreifer das Überlaufen auch durch eine Unmenge tatsächlicher Angriffe auslösen könnte.

4.2 False Negatives

False Negatives bedeuten, dass dem Administrator ein tatsächlicher Angriff durch die Lappen gegangen ist, weil er entweder vergessen hat, eine entsprechende Regel zu entwerfen oder weil es sich um einen Angriff handelt, der zum Zeitpunkt der Einrichtung des Intrusion Detection Systems noch nicht bekannt war. Der Grund für False Positives ist also im allgemeinen analytischer Natur. Entweder hat der Administrator einen Angriff schlichtweg übersehen oder er hat sich nicht auf dem laufenden gehalten über neue Angrifftechniken. Hier wird deutlich, dass auch ein IDS ständige Wartung benötigt genauso wie eine Firewall. Das Informieren über neue Einbrüche, neue Angriffstechniken und neue Angriffsprogramme ist genauso elementar wie das entsprechende Anpassen der Signaturdatenbank. Ein schlecht gepflegtes Intrusion Detection System bietet nach gewisser Zeit ohne Wartung ebenso wenig Garantie für die Entdeckung neuer Angriffe wie eine schlecht gewartete Firewall Schutz vor eben diesen Angriffen bietet.

5 Installation

Die Installation von Snort auf Unix-ähnlichen Systemen gestaltet sich relativ einfach, zuerst werden die Quellen der neuesten Version von Snort[13], sowie die Quellen der Libpcap[14] benötigt, letztere wird von Snort für das Abfangen der Pakete auf unterster Ebene verwendet. Nun sollten die beiden Tarballs in ein persönliches Arbeitsverzeichnis kopiert werden, dort packt ein

```
$ tar zxf snort-1.7.tar.gz RETURN
```

die Quellen von Snort in ein Unterverzeichnis aus. Nachdem in letzteres gewechselt wurde zeigt ein

```
$ ./configure --help RETURN
```

die diversen Konfigurationsmöglichkeiten zum Übersetzen. Interessant dürften hier die unterschiedlichen Unterstützungen für die verschiedenen Datenbanksysteme und die OpenSSL-Bibliothek, sowie die *'Flexible Responses'* und die Alarmierung via SMB[15] sein. Einige dieser Optionen verlangen nach weiteren `include`-Files, die in den Sourcecode-Distributionen der jeweiligen Programme enthalten sind, beispielsweise werden für die MySQL-Unterstützung die MySQL-Headerfiles und ihre Clientbibliothek benötigt[16]. Falls die Flexible Responses verwendet werden sollen - diese erweitern Snort um eine Intrusion Response Komponente (siehe dazu Abschnitt 3.4) - wird zusätzlich die Libnet benötigt[17]. Wenn die Software für die jeweiligen Extras vorhanden, kompiliert und installiert ist, kann mit dem Übersetzungsvorgang für Snort begonnen werden:

```
$ ./configure --prefix=/usr/local --enable-smbalerts && make RETURN
```

Nach erfolgreichem Kompilieren installiert ein

```
$ make install RETURN
```

als Superuser Snort und seine Komponenten nach `/usr/local/`. An diesem Punkt angelangt, kommen wir nun zum nächsten wohl wichtigsten Schritt, der Konfiguration.

[13] `http://www.snort.org`
[14] eine Packet Capture Library `http://www.tcpdump.org`
[15] WinPop-Up Message
[16] `http://www.mysql.com`
[17] `http://www.packetfactory.net`

6 Die Konfiguration von Snort

Da Snort in erster Linie ein regelbasiertes IDS ist, benötigt es eine Art Wissensbasis, eine 'Datenbank' in der die sogenannten Signaturen definiert sind. Als Signaturen werden in diesem Kontext die Charakteristika eines Pakets bezeichnet, die dieses Paket und seine Eigenschaften (z.B. Flags, Source- und Destination-Port und/oder -IP-Adresse) oder 'bösartig'[18] definieren. Ohne eine solche Wissensbasis ist Snort quasi blind, es kann zwar die Pakete von der Netzwerkkarte dekodieren, weiss aber nicht welche dieser Pakete unerwünscht sind und welche nicht.

Wie sieht nun eine solche Wissensbasis aus? Die Wissensbasis ist ein einfaches Textfile, das sich aus Regeln[19] (im folgenden auch *rules* genannt) zusammensetzt, anhand derer Snort weiss, wie es mit den einzelnen Paketen zu verfahren hat. Wie wir sehen, dreht sich bei der Konfiguration von Snort alles um solche Regeln. Betrachten wir nun den Aufbau und die Funktionsweise solcher *rules*.

6.1 Aufbau einer Rule

Eine Regel setzt sich aus zwei Teilen zusammen, dem sogenannten *rule header* und den *rule options*. Da Snort nicht mit Zeilenumbrüchen innerhalb der *rules* umgehen kann ist es enorm wichtig folgendes zu beachten:

Regeln müssen immer in einer Zeile stehen!

Snort beendet sich andernfalls mit einer entsprechenden Fehlermeldung und verweigert die Arbeit.

6.1.1 Der Rule Header

Der Rule Header definiert, was geschehen soll, wenn eine bestimmte Signatur in einem Paket gefunden wird und trifft eine grobe Vorauswahl über Protokoll, IP-Adresse und Portnummer:

$$\underbrace{\texttt{alert}}_{action}\ \underbrace{\texttt{TCP}}_{protocol}\ \underbrace{\texttt{any}\quad\texttt{any}}_{SourceIP-Address\ Port}\ \underbrace{\texttt{->}}_{direction}\ \underbrace{\texttt{!192.168.0.0/24}\quad\texttt{:1024}}_{DestinationIP-Address\ Port}$$

action hier stehen folgende Optionen zur Verfügung: Alarmieren `alert`, Protokollieren `log` und Ignorieren `pass`. Es können auch eigene *actions* definiert werden (siehe 6.2.2).

[18] eine passendere Übersetzung des Englischen *'malicious'* liess sich nicht finden

[19] Daher leitet sich auch die Bezeichnung *regelbasiertes IDS* ab.

protocol definiert, welches Protokoll beobachtet werden soll; zur Zeit werden die 3 gängigsten Protokolle `TCP` `UDP` `ICMP` unterstützt.

Portnumber & IP-Address Zuerst wird die Source IP-Adresse in CIDR[20]-Notation angegeben, das `!` dient hier als Negationsoperator und definiert so alle IP-Adressen, die *nicht* 192.168.0.0/24 entsprechen. Als Portnummer kann jede Zahl zwischen 1 und 65535 gewählt werden, Bereiche werden mit dem `:`-Operator gekennzeichnet (hier alle Ports $\geq$ 1024). `any` definiert alle gültigen Werte für Port und/oder IP-Adresse. Nach dem *direction*-Operator wird noch die Destination IP-Adresse und Portnummer angegeben.

direction Der *direction*-Operator legt die Richtung des Traffics fest; unidirektional `->` oder bidirektional `<>`.

Sicherlich lässt sich über den Sinn und Unsinn des oben aufgeführten *rule headers* streiten, doch er soll hier nur als Beispiel und Veranschaulichung der diversen Operatoren dienen. Mit dem rule header bekommt Snort schon einen guten Einblick in die Wünsche des Netzwerkadministrators, doch die eigentlichen Gefahren entgehen Snort, hierfür ist ein Blick in die jeweils spezifischen Details der einzelnen Pakete notwendig (Data Segment, Flags).

6.1.2 Die Rule Options

Die *rule options* lassen feinere Abstufungen bei der Überwachung des Netzwerks und bei der Angriffserkennung zu, mit ihnen wird definiert was genau in einem Paket als verdächtig gilt, abgesehen von seinem Ursprungs- und Zielort. Die rule options schliessen sich dem rule header an und werden durch `(` und `)` zusammengefasst. Die einzelnen Optionen bestehen immer aus Schlüsselwort und Wertepaaren, die voneinander mit `;` getrennt sind. Die verschiedenen Optionen hängen stark von dem zuvor im rule header spezifizierten Protokoll ab, hier ein Beispiel:

$$\texttt{(}\ \underbrace{\texttt{msg}}_{keyword}\ \underbrace{\texttt{:}}_{separator}\ \underbrace{\texttt{LogMessage}}_{value}\ \underbrace{\texttt{;}}_{separator}\ \ldots\ \underbrace{\texttt{flags:SA+;}}_{option\ n}\ \texttt{)}$$

keyword Das Schlüsselwort beschreibt, nach welchem Kriterium gesucht werden soll

`:` Trennt Schlüsselwort und Wert.

[20] Classless Internet Domain Routing

value definiert den Wert, den ein mit *keyword* angegebenes Kriterium aufweisen muss.

`;` Trennt die einzelnen Schlüsselwort-Wertepaare voneinander.

Die *rule options* funktionieren alle nach demselben oben gezeigten Prinzip. Alle Optionen detailliert zu erklären würde den Rahmen diese Dokuments sprengen, deshalb soll hier lediglich eine kurze Übersicht die grundlegende Bedeutung jeder Einzelnen erklären, für eine ausführliche Beschreibung sei auf [12] und [22] verwiesen.

Tabelle 1: Die verschiedenen *rule options* von Snort

Keyword	Value	Beschreibung
`msg`	`"Log Message"`	schreibt den in Anführungszeichen angegebenen String in die Logs
`logto`	FileName	definiert eine Datei, in die Pakete, die diese *rule* matchen, geloggt werden sollen.
`ttl`	0 - 255	testet den Wert des Time To Live Fields im IP-Header
`tos`	Number	testet den Wert des Type Of Service Fields
`id`	Number	überprüft das ID field eines IP-Pakets
`ipopts`	es ist nur eine der Optionen pro Regel erlaubt!	
	`rr`	Record Route
	`eol`	End Of List
	`nop`	No Operation
	`ts`	Time Stamp
	`sec`	IP Security Option
	`lsrr`	Loose Source Code Routing
	`ssrr`	Strict Source Code Routing
	`satid`	Stream Identifier
`fragbits`	`RB`	Reserved Bit
	`DF`	Don't Fragment
	`MF`	More Fragments

Fortsetzung auf nächster Seite

Fortsetzung von vorheriger Seite

Keyword	Value	Beschreibung
`flags`	`F`	FIN
	`S`	SYN
	`R`	RST
	`P`	PSH
	`A`	ACK
	`1`	Reserved Bit 1
	`2`	Reserved Bit 2

Um `fragbits` & `flags` besser matchen zu können, stehen folgende Operatoren bereit:

- `+` matched alle angegebenen Flags plus beliebige Andere.
- `*` matched mindestens eines der definierten Flags.
- `!` ist der schon bekannte Negationsoperator und matched, wenn die spezifizierten Flags *nicht* gesetzt sind.

`dsize`	Number	Testet die Grösse des Datensegments in einem Paket, damit lassen sich Buffer Overflows leichter identifizieren. Bereiche werden mit `<` und `>` definiert.
`content`	`"|0f430a|str"`	Mit dieser Option wird ein Pattern Matching nach dem Boyer-Moore-Algorithmus im Datensegment durchgeführt. Der Match-String muss in `"` stehen, es ist neben normalen strings auch möglich Bytecode anzugeben, dieser muss durch ein `|` am Anfang und Ende gekennzeichnet werden.

Fortsetzung auf nächster Seite

Fortsetzung von vorheriger Seite

Keyword	Value	Beschreibung
`nocase`	–	schaltet die Unterscheidung zwischen Gross- & Kleinschreibung beim Pattern Matching ab.
`depth`	Number	Gibt die Tiefe an, mit der die einzelnen Pakete beim Pattern Matching durchsucht werden sollen.
`offset`	Number	Definiert den Startpunkt des Pattern Matching.
`seq`	Number	Testet die Sequence Nummer des Pakets.
`ack`	Number	Testet die Acknowledge Nummer.
`itype`	Type	Testet das ICMP Type Field, wobei Type ein String ist.
`icode`	Number	Testet den numerischen Wert des ICMP Type Fields.
`icmp_id`	Number	Testet das ID Field in ICMP ECHO Paketen. Diese Option wurde speziell für *Stacheldraht*-DDoS Attacken entwickelt.
`icmp_seq`	Number	Testet die Sequence Nummer in ICMP ECHO Paketen.
`session`	Extrahiert USER Data aus TCP-Sessions.	
	`all`	Schreibt alle Zeichen
	`printable`	Schreibt nur druckbare Zeichen
`rpc`	Number	Dekodiert Applikation, Prozedur und Version von Remote Procedure Calls.
`resp`	Flexible Responses erlauben Snort, die Verbindung zwischen den jeweiligen Systemen auf unterschiedliche Art zu beenden.	
	`rst_snd`	Sendet `TCP_RST` zum sendenden Host.
	`rst_rcv`	Sendet `TCP_RST` zum empfangenden Host.
	`rst_all`	Sendet `TCP_RST` zu allen involvierten Hosts.

Fortsetzung auf nächster Seite

Fortsetzung von vorheriger Seite		
Keyword	Value	Beschreibung
	`icmp_net`	Sendet `ICMP_NET_UNREACH` zum sendenden Host.
	`icmp_host`	Sendet `ICMP_HOST_UNREACH` zum sendenden Host.
	`icmp_port`	Sendet `ICMP_PORT_UNREACH` zum sendenden Host.
	`icmp_all`	Sendet alle der obigen `ICMP_*_UNREACH` zum sendenden Host.

Es ist einleuchtend, dass die Reihenfolge, in der die Regeln abgearbeitet werden, die Effizienz des IDS ungemein beeinflussen, sowohl positiv als auch negativ. Die Regeln werden *nicht* in der Reihenfolge verarbeitet, in der sie in der Wissensbasis aufgeführt sind, stattdessen weren generell Content Matching Regeln zuletzt ausgeführt, da sie am rechenintensivsten sind. Durch geschicktes Schreiben der Regeln, in etwa das Überprüfen der Flags oder Paketgrösse, lassen sich die Regeln, mit denen für ein Paket wirklich ein Content Matching durchgeführt wird, minimieren. Alle Regeln in einer Wissensbasis werden disjunktiv und die einzelnen Elemente einer rule konjunktiv verknüpft und müssen sich als wahr erweisen, so dass diese Regeln greift, bzw. die angegebene action ausgeführt wird.

6.2 Weiterführende Konfiguration

Regeln sind allerdings nicht die einzigen Hilfsmittel, auf die Snort und der Netzwerkadministrator zurückgreifen können, um einem Eindringling auf die Spur zu kommen.

6.2.1 Preprocessors

Neben diesen umfangreichen Möglichkeiten zur Angriffserkennung stehen Snort noch weitere Konfigurationsoptionen zuf Verfügung, die eine Anpassung an die eigenen Bedürfnisse erleichtert. Eines der wohl besten Features ist die Plugin-Schnittstelle (siehe Abschnitt 3.1), die Snort bietet. Dies ist User Code, der von Snort zwar nach der Paketdekodierung, jedoch vor dem *rule matching* ausgeführt wird. Es sind schon einige sogenannte Präprozessoren geschrieben worden, unter anderem Spade, welches in Abschnitt **??** etwas näher beschrieben wurde, und die Liste der erhältlichen Plugins verlängert sich ständig. Einige der im Moment zur Verfügung stehenden *preprocessors* sollen im folgenden erläutert werden:

`minfrag <Threshold>` untersucht fragmentierte Pakete auf einen Schwellwert; kein Computer sollte Pakete $\leq$ 512 Bytes erzeugen, das heisst, wenn wir kleinere Pakete in unserem Netz vorfinden, könnte dies ein Indiz für einen Angriffsversuch sein, der mittels Miniaturpaketen versteckt werden soll.

`defrag` ist eine Weiterentwicklung von `minfrag` und erlaubt Snort eine vollständige Defragmentierung von IP.

`stream: timeout <Timeout>, ports <Ports>, maxbytes <Number>` erlaubt Snort die Reassemblierung von TCP-Streams, die jeweiligen Parameter legen den Timeout in Sekunden, die Ports, auf denen die Stream-Reassembly vorgenommen werden soll, und die maximale Grösse der von Snort rekonstruierten Pakete fest.

`http_decode <Port_numbers>` dekodiert den URL-encodierten Datenstrom in einen reinen ASCII-Datenstrom auf den angegebenen Ports. Dies ist für die Erkennung von CGI-Attacken sehr hilfreich.

`portscan: <Network> <Number_of_Ports> <Detection_Period> <Logdir/Filename>` überwacht das definierte Netzwerk auf Portscans, wobei mit `<Number_of_Ports>` angegeben wird, wieviele Ports in welcher Zeit 'gescannt' werden müssen, bevor dieses Plugin seine Daten in das angegebene Logfile schreibt.

`portscan-ignorehosts: <Host_List>` Die Liste der hier angegebenen Host-IP-Adressen lässt Portscans von diesen Hosts zu, es ist eventuell erwünscht die Rechner im eigenen Netz von Zeit zu Zeit einem Portscan zu unterziehen, da sich so eventuell ein Port entdecken lässt, der von einem bisher unbemerkten Angreifer für seine Zwecke geöffnet wurde.

`spade` erweitert die Fähigkeiten von Snort um eine statistische Analyse, mit der Snort nicht mehr allein von den Signaturen abhängig ist. Da sich der Code noch im Betastadium befindet ist von einer Benutzung in sicherheitskritischen Netzwerken abzuraten. Eine ausführliche Dokumentation ist in der Sourcecode-Distribution von Snort enthalten.

6.2.2 Output-Modules

Damit die erkannten Angriffe auch nach persönlichen Vorstellungen und Vorlieben mitgeteilt werden, stellt Snort sogenannte *output modules* bereit. So können quasi eigene *actions* definiert werden.

```
ruletype myAction
```

```
{
type alert
output alert_smb: workstation.list
output log_tcpdump: myAction.log
}
```

`ruletype <MyAction>` bindet den Namen *MyAction* an die neudefinierte *action*.

`type <Action>` kategorisiert die neue *action* grob nach den uns schon bekannten *actions* (siehe 6.1.1).

`output <Module>: <Options>` Für jede eigene *action* lassen sich beliebig viele *output modules* angeben. Sie beschreiben wie die jeweiligen Daten für die *action* wo abgelegt werden sollen[21]. Einen kleinen Überblick der verschiedenen Module und ihrer Optionen verschafft die folgende Tabelle.

Tabelle 2: Die Output-Modules von Snort

Module	Optionen	Beschreibung
`alert_fast`	Filename	Schreibt einen kurzen 'Einzeiler' in Filename.
`alert_full`	Filename	Schreibt den vollständigen Paketheader in Filename. Da das Module einige Umformatierungen der Daten vornimmt, ist es eher langsam.
`alert_smb`	Filename	Schickt eine kurze Nachricht an alle in Filename (pro Zeile ein NetBIOS Name) angegebenen Rechner. Es ist zu beachten, dass dieses Module den `smbclient` benutzt, um die Nachricht zu versenden.
`alert_unixsock`	–	Kreiert einen Unix domain socket, den andere Programme abhören können, um die Ausgaben von Snort in Echtzeit zu analysieren.

Fortsetzung auf nächster Seite

[21] Filenameangaben sind hier meist relativ zum Snort-Logverzeichnis (siehe **??**) angegeben

Fortsetzung von vorheriger Seite

Module	Optionen	Beschreibung
`alert_syslog`	Facility Priority Options	Dieses Modul erlaubt das Logging über *syslog*. Die Facility gibt an, welche Art des Programs in den system log schreibt, die Priority wie kritisch die Meldung für den Betrieb des Systems ist. Genauere Erläuterungen finden sich in der manpage zu *syslog*.
`log_tcpdump`	Filename	Schreibt Snort's Meldungen in die Datei Filename im *tcpdump*-Format, dies ist vorallem für die nachträgliche Analyse der Daten sehr hilfreich.
`xml`	`log | alert`, Parameter List	Ermöglicht Snort seine Daten in der Simple Network Markup Language (XML formatiert) zu schreiben. Die Optionen der Parameter-Liste werden in Tabelle 3 erklärt. `log` oder `alert` assoziieren das *output modul* mit der jeweiligen *action*.
`database`	`log | alert`, Database, Parameter List	Erlaubt Snort Einträge in die Datenbank *Database*, so dass diese dann der Datenpräsentation dienlich ist (siehe 3.3). Die Unterstützung für das jeweilige Datenbankformat muss in Snort mit einkompiliert werden (siehe Abschnitt 14). Eine detaillierte Erklärung der Parameter-Liste findet sich in Tabelle 4. `log` oder `alert` s.o.

Die Parameter-Listen des Database- und XML-Moduls bestehen aus `SPACE`-separierten (␣) *keyword=value*-Paaren, mit denen die beiden Module konfiguriert werden. Die folgenden Tabellen geben Aufschluss über die bestehenden Möglichkeiten.

Tabelle 3: XML Parameter-Liste

Keyword	Value	Beschreibung
`file`	Filename	Wenn dies der einzige Parameter ist, so wird in dieses File geschrieben, andernfalls wenn im `protocol http[s]` angegeben ist, dient das File als Script, welches auf dem entferntern Rechner ausgeführt wird.
`protocol`	Legt das Protokoll für die Übertragung der Daten fest.	
	`http`	Sendet einen `POST`-Request an den Server. Der `file`-Parameter ist Voraussetzung.
	`https`	Sendet den `POST`-Request via SSL. Der OpenSSL-Support muss allerdings miteinkompiliert worden sein. Voraussetzung sind hier die Parameter: `file, cert, key`
	`tcp`	Es wird eine einfache TCP-Verbindung hergestellt, `port` wird benötigt.
	`iap`	Leider ist die Implementierung des Intrusion Alert Protocols noch nicht abgeschlossen, doch ist eine Unterstützung von Seiten Snort bereits vorgesehen.
`host`	`IP-Address`	Bestimmt die IP oder den Hostnamen des zu kontaktierenden Servers.
`port`	1-65535	Definiert zu Portnummer für die Verbindung.
`cert`	Filename	Gibt die Datei, in der das Zertifikat des zu kontaktierenden Servers steht, an (PEM formatiert).

Fortsetzung auf nächster Seite

Fortsetzung von vorheriger Seite

Keyword	Value	Beschreibung
`key`	Filename	Gibt die Datei, in der der Private Key des Client-Rechners steht, an (PEM formatiert).
`ca`	Filename	Gibt die Datei, in der das Certification Authority Zertifikat zur Authentifizierung des Servers steht, an (PEM formatiert).
`server`	Filename	Gibt die Datei, in der eine Liste der von Snort anerkannten Server steht. Dies dient der Authentifizierung der Server.
`sanitize`	Network / Netmask	Verschleiert die IP-Adressen, um die Privatsphäre zu bewahren.
`encoding`	Definiert die zu speichernde Datenrepräsentation	
	`hex`	Schreibt die Daten als Hex-Strings, gut durchsuchbar, schlecht lesbar, doppelte Dateigrösse.
	`base64`	Schreibt die Daten als *base64*-String, ohne Nachbearbeitung nicht durchsuch- oder lesbar, aber lediglich 1,3-fache Dateigrösse.
	`ascii`	Leider gehen bei dieser Repräsentation einige Informationen verloren, da der ASCII-Zeichensatz nicht alle Daten darstellen kann, dennoch ist bei guter Les- und Durchsuchbarkeit eine geringfügig grössere Dateigrösse zu *base64* zu erwarten.

Fortsetzung auf nächster Seite

Fortsetzung von vorheriger Seite		
Keyword	Value	Beschreibung
`detail`	Gibt die Menge der zu loggenden Informationen an.	
	`full`	Schreibt alle Headerinformationen (TCP, IP) und das Datensegment in den angegebenen Pfad.
	`fast`	Es werden lediglich die wichitgsten Informationen, wie z.B Protokoll, IP-Adressen und Ports, Flags, Signatur und Zeitstempel protokolliert.

Tabelle 4: Database Parameter-Liste

Keyword	Value	Beschreibung
`host`	String	Der Host, auf dem der Datenbankserver läuft.
`port`	Number	Der Port, unter dem der Datenbankserver auf dem Host erreichbar ist.
`dbname`	String	Name der Datenbank, in die geschrieben werden soll.
`user`	String	Name des Datenbankbenutzers, mit dem Snort seine Daten in die Datenbank eintragen soll.
`password`	String	Passwort des oben erwähnten Datenbankbenutzers.
`sensor_name`	String	Ein Name für dieses *output module*. Falls dieser Parameter ausgelassen wird, generiert Snort selbst einen.
`encoding`	siehe Tabelle 3 XML Parameter Liste.	
`detail`	siehe Tabelle 3 XML Parameter Liste.	

6.2.3 Variablen

Snort erlaubt die Benutzung von Variablen zur bequemeren Wartung der *rule*-Files. `var MYVAR myval` deklariert die Variable `MYVAR` und initialisiert sie mit dem Wert `myval`. Dereferenziert werden Variablen nach alter Shellmanier mit dem `$`-Operator, Zusätzlich steht hier auch noch der `!`-Negationsoperator zur Verfügung. Die Deklaration von Meta-Variablen ist ebenfalls möglich kontrollieren, so initialisiert `$(MYVAR:-defaultval)` die Variable `MYVAR` mit dem

Wert `defaultval`, falls `MYVAR` bis dahin noch nicht gesetzt ist. `$(MYVAR:?message)` gibt den Wert von `MYVAR` zurück oder beendet Snort mit der Fehlermeldung *message*, je nachdem, ob `MYVAR` gesetzt ist oder nicht.

6.2.4 Die Kommandozeilenoptionen

Die diversen Kommandozeilenoptionen von Snort erleichtern das 'Debugging' der Rules in der Anfangsphase, später können die diversen Optionen ein wenig zur Sicherheit des System beitragen, so dass Snort ungestört seine Arbeit verrichten kann.

Zu Begin empfiehlt es sich, Snort mit `-i` das Interface mitzuteilen, auf welchem es die Pakete aus dem Netz beziehen soll. `-v` lässt Snort etwas gesprächiger werden und mit `-c` wird Snort die Konfigurationsdatei angegeben, in der es die *rules* vorfindet. Zum Interface sei noch angemerkt, dass Snort auf einem Host laufen sollte, dessen Interfaces keine IP-Adresse zugewiesen wurden, um zwar den Empfang der Pakete zu gewährleisten, aber den Host keinen direkt an ihn adressierten Paketen auszusetzen.

Um immer auf dem neuesten Stand der Dinge zu sein, zumindest was die Signaturen betrifft, ist es ratsam, sich täglich die `vision.conf` von White Hats[22] zu besorgen. Diese lässt sich mit wenigen Handgriffen an das eigene Netzwerk anpassen. Dazu bedarf es 3 Konfigurationsdateien, der `vision.conf`, einer Datei für die lokalen Anpassungen `local.conf` und einer Hauptkonfigurationsdatei `snort.conf`. In der `local.conf` werden die Regeln mit einem `pass` versehen und eingetragen, die *nicht* im lokalen Netzwerk eingehalten werden sollen. In der `snort.conf` wird die `local.conf` *vor* der `vision.conf` mittels dem `include`-statement geladen, und die Reihenfolge der Regelabarbeitung mit der Kommandozeilenoption `-o` auf pass, alert, log gesetzt. Das `include`-statement funktioniert ähnlich dem aus C bekannten `include`, es fügt den Inhalt der `include`-Datei in die aktuelle ein. So lassen sich eigene Bibliotheken von Signaturen erstellen, die dann entsprechend von der Hauptkonfigurationsdatei eingebunden werden.

Um Snort nun im tagtäglichen Betrieb laufen zu lassen, ist es ratsam, sich ein kleines Skript[23] zu schreiben, welches Snort mit den entsprechenden Optionen aufruft.

```
$ snort -D -o -i eth0 -l /var/log/snort -c /etc/snort/snort.conf
RETURN
```

startet Snort als Daemon im Hintergrund, der auf dem Interface eth0 'snifft', seine Logdateien unter `/var/log/snort` ablegt und die Konfiguration aus

[22] `http://www.whitehats.com`
[23] ähnlich einem rc-script

`/etc/snort/snort.conf` liest. Die Optionen `-u 505 -g 505 -t directory` lassen den Snort-Prozess mit der Benutzer- und Gruppenkennung 505 in einer *change root*-Umgebung in *directory* laufen. Was die Zugriffsrechte des Prozesses angeht, sollte dies eine ausreichende Sicherheit sicherstellen.

Literatur

[1] Snort: *Homepage*, `http://www.snort.org`

[2] Thomas H. Ptacek, Timothy N. Newsham: *Insertion, Evasion and Denial of Service: Eluding Network Intrusion Detection*, `http://www.snort.org/IDSpaper.pdf`, 1998

[3] Felix Mack: *Workshop "Firewalls unter Linux"*, `http://www.pl-berichte.de/work/firewall/index.html`, 2001

[4] Biswanath Mukherjee, L. Todd Heberlein, Karl N. Levitt: *Network Intrusion Detection*, `http://seclab.cs.ucdavis.edu/papers/mhl94.pdf`, 1994

[5] Martin Roesch: *Snort - Lightweight Intrusion Detection for Networks*, `http://www.snort.org/lisapaper.ps`, 1999

[6] Eric S. Raymond: *How To Become A Hacker*, `http://www.tuxedo.org/~esr/faqs/hacker-howto.html`, 2001

[7] Silicon Defense: *Homepage*, `http://www.silicondefense.com`

[8] AIR-CERT: *Analysis Console for Intrusion Databases*, `http://www.cert.org/kb/acid/`, 2001

[9] Bundesamt für Sicherheit in der Informationstechnik: *Grundlagen, Forderungen und Marktübersicht für Intrusion Detection Systeme und Intrusion Response Systeme*, `http://www.bsi.de/literat/studien/ids/ids-irs.htm`, 1998

[10] tcpdump, libpcap: *Homepage*, `http://www.tcpdump.org`

[11] Tripwire Open Source Project: *Homepage*, `http://www.tripwire.org`

[12] Martin Roesch: *Writing Snort Rules*, `http://www.snort.org/writing_snort_rules.htm`

[13] Martin Freiss, Jürgen Schmidt: *Einbrecheralarm - Intrusion Detection mit Snort*, c't 8/2001 S. 212-219

[14] Ralf Hildebrandt: *Kain und Abel - Snort und nmap - zwei Seiten derselben Münze*, Linux Magazin 12/2000 S. 102-109

[15] Sicherheit im Internet: *Einbruchserkennung-Intrusion Detection*, `http://www.sicherheit-im-internet.de`

[16] Jeremy Frank: *Artificial Intelligence and Intrusion Detection: Current and Future Directions*, `http://seclab.cs.ucdavis.edu/papers/ncsc.94.ps`, 1994

[17] Steven Cheung, Karl N. Levitt, Calvin Ko: *Intrusion Detection for Network Infrastructures*, `http://seclab.cs.ucdavis.edu/papers/clk95.ps`, 1995

[18] John McHugh, Alan Christie, Julia Allen: *Defending Yourself: The Role of Intrusion Detection Systems*, `http://www.stsc.hill.af.mil/crosstalk/2001/jan/mchugh.asp`

[19] Elizabeth D. Zwicky, Simon Cooper, D. Brent Chapman: *Einrichten von Internet Firewalls, 2. Auflage*, O'Reilly Verlag 2001

[20] Rebecca Gurley Bace: *Intrusion Detection*, Macmillan Technical Publishing 2000

[21] Stephen Northcutt, Judy Novak: *Network Intrusion Detection: An Analysts's Handbook, Second Edition*, New Riders 2000

[22] W. Richard Stevens: *TCP/IP Illustrated Vol. 1: The Protocols*, Addison Wesley 1994