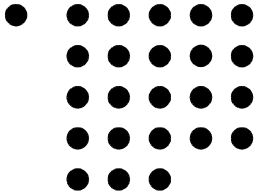


Fachhochschule Köln

University of Applied Sciences Cologne

07 Fakultät für Informations-, Medien- und Elektrotechnik, Studiengang Elektrotechnik/NT

Institut für Nachrichtentechnik

Labor für Datennetze

Diplomarbeit

Diplomand	Holger Moskopp
Matrikelnummer	11006053
Referent	Prof. Dr. Andreas Grebe
Korreferent	Prof. Dr. Heiko Knospe

Thema:

Aufbau einer Sicherheitsarchitektur für das Experimentiernetz unter Berücksichtigung der Anforderungen für SIP-Anwendungen

Hiermit versichere ich, dass ich die Diplomarbeit selbständig angefertigt und keine anderen als die angegebenen und bei Zitaten kenntlich gemachten Quellen und Hilfsmittel benutzt habe.

Holger Moskopp

Danksagung:
An dieser Stelle möchte ich all jenen danken, die durch ihre fachliche und persönliche Unterstützung zum Gelingen dieser Diplomarbeit beigetragen haben. Bei Herrn Prof. Dr. Andreas Grebe für die Vergabe, Betreuung und Diskussion der Diplomarbeit. Bei Herrn Prof. Dr. Heiko Knospe, der so freundlich war, die Stelle des Korreferenten zu übernehmen. Weiterhin danke ich Herrn Björn Zülch vom Fraunhofer IITB und William Metcalf, einem der Snort-Inline Entwickler, für die schier unerschöpfliche Geduld, auf meine E-Mails zu antworten. Desweiteren danke ich Stefan Abu Salah, Achim Marikar, Jörg Grommes und Heinz Dresbach für die engagierte Unterstützung.
Besonderer Dank gebührt meinen Eltern Eyke und Barbara Moskopp und meiner Freundin Karin Ochmann für den Rückhalt und ständige Bestärkung.
Auch danke ich meinen Freunden, Kommilitonen und der Fachschaft NT für all den Kaffee.

1 Inhaltsverzeichnis

1 Vorwort

Das Internet ist unsicher! So, oder so ähnlich, sind viele Aussagen, wenn es darum geht, das eigene Netzwerk zu schützen. Der Schutz eines Netzwerkes geht aber über die bloße Absicherung gegen Angriffsversuche von außen weit hinaus. Vielmehr muss der Sicherheitsbeauftragte eines Netzwerkes sich nach allen Seiten umsehen. Genauso gut können Angriffe von innen kommen, weil einer der Mitarbeiter einfach einmal ausprobieren möchte, wie gut der Sicherheitsbeauftragte gearbeitet hat, oder ein Mitarbeiter möchte vertrauliche Daten stehlen. Auch denkbar ist, dass das eigene Netzwerk Ausgangspunkt eines Angriffs gegen ein ganz anderes Netz werden kann. Der Benutzer eines Netzwerkes ist somit auch ein Risikofaktor, dessen Handlungen eingeschränkt werden müssen. Ihm dürfen nur Dienste zur Verfügung stehen, die er benötigt und die ungefährlich für das Netzwerk sind. Netzwerksicherheit beinhaltet also die Planung, Ausführung, Überwachung und Wartung eines umfassenden Sicherheitssystems für ein Netzwerk.

Doch kann man ein Netzwerk gegen Angriffe jeglicher Art absichern? Eines vorweg, einen allumfassenden Schutz für ein Netzwerk gibt es nicht. Fast täglich werden Sicherheitslücken verschiedener Anwendungen bekannt, die mehr oder weniger kritisch für den Betrieb des eigenen Netzwerkes sind. Zwar gibt es, meist sehr schnell nach bekannt werden einer Sicherheitslücke, Patches oder Updates für die betroffene Software, jedoch bleibt in der Zeit bis der Patch aufgespielt ist, das Netzwerk oder einzelne Komponenten des Netzwerkes ungeschützt.

Ein Sicherheitsbeauftragter eines Netzwerkes kann also nicht eine Sicherheitssoftware installieren, sich dann zurücklehnen und sich sicher fühlen. Das Netzwerk muss ständig überwacht und gewartet werden, denn die Sicherheitskomponenten benötigen ständige Anpassung an neue Gefahrenquellen. Dabei soll die Produktivität des Netzwerkes jedoch unter keinen Umständen eingeschränkt werden. Weiterhin müssen die Sicherheitskomponenten so gestaltet sein, dass sie zuverlässig arbeiten und bei Änderungen im Netzwerk leicht skalierbar sind. Gerade bei neuen Anwendungen, die bisher nicht genutzte Protokolle benötigen, ist dies von elementarer Bedeutung.

Die Internettelefonie ist hierfür ein Paradebeispiel. Der hierbei entstehende Datenverkehr ist für die Sicherheitsmaßnahmen problematisch, da die Daten auf nicht festgelegten Verbindungsendpunkten ausgetauscht werden. Der eigentliche Austausch der Sprachdaten ist nur unter bestimmten Voraussetzungen als solcher zu erkennen.

Alle zuvor genannten Gesichtspunkte sind sicherheitstechnisch zu berücksichtigen, wenn ein Netzwerk effektiv geschützt werden soll. Dies wird in dieser Diplomarbeit beschrieben. Hierbei werden beim Leser grundlegende Kenntnisse zu Betrieb und Nutzung von Datennetzen, sowie die Funktionsweise gängiger Netzwerkprotokolle vorausgesetzt. Protokolle, die nicht vorausgesetzt werden, werden beschrieben oder besitzen einen Verweis auf die entsprechende RFC (Request for Coments).

2 Ziele der Diplomarbeit

Das Experimentiernetz des Labors für Datennetze der Fachhochschule Köln soll durch ein geeignetes Sicherheitskonzept abgesichert werden. Hierbei sind besonders die Anforderungen für VoIP- (Voice over IP) und SIP-Architekturen (Session Initiation Protocol, beschrieben in RFC 3261, früher RFC 2534) zu berücksichtigen.
Attacken auf ein Netzwerk können ohne ausreichenden Schutz verheerende Schäden anrichten. Daher ist das Experimentiernetz mit einer Firewall gegen Angriffe von außen abzusichern. Gleichzeitig sollen SIP-basierte Anwendungen ohne einen relevanten Verlust an Performanz weiterhin nutzbar sein. An die Firewall wird eine DMZ (Demilitarisierte Zone) angegliedert. Weiterhin müssen Sicherheitsmechanismen installiert werden, die andere verwundbare Punkte des Netzwerkes schützen.

Folgende Ziele wurden für die Diplomarbeit definiert:

- Die Schutzmechanismen sollen auf Basis von Linuxanwendungen realisiert werden
- Es ist eine einfache Umsetzung, die auf einer einzelnen Firewall beruht und mit geringem Aufwand erweitert werden kann, gefordert
- Ein- und ausgehender Datenverkehr soll durch eine Firewall streng reglementiert werden
- Unterbindung von unerwünschtem Datenverkehr
- Ausgewählter Datenverkehr soll auf seine Unbedenklichkeit hin untersucht werden
- Vereinfachung der Wartung der Sicherheitskomponeten
- Der Datenaustausch zwischen VoIP-Endgeräten soll durch einen speziellen Netzwerkbereich fließen
- Dem künftigen Systemadministrator soll die Diplomarbeit als Handbuch der verwendeten Komponenten dienen

3 Gefahren für ein Netzwerk

Mit zunehmender Komplexität der Netzwerke wächst auch die Quantität der möglichen Angriffsszenarios. Die Vielzahl der denkbaren Angriffsarten reicht von Attacken auf Protokollbasis bis hin zu komplexen Angriffen wie Viren und Trojanern. Aber auch interne Gefahren können durch falsche Konfiguration von Anwendungen oder durch Anwender, die Kompetenzen überschreiten, im eigenen Netzwerk bestehen.

3.1 Angriffe auf Protokollbasis

DoS-Attacken (Denial of Service) versuchen, einen Server unerreichbar zu machen. Der Angreifer stellt eine Flut an sinnlosen Anfragen an einen Server, um entweder die Leitung zu überlasten oder den Server so stark zu beschäftigen, dass ein Benutzer, der tatsächlich eine Anfrage stellt, nicht mehr bedient werden kann. Hierbei wird zwischen ICMP-Flood und SYN-Flood unterschieden. Ein *ICMP-Flood-Angriff* versucht, die Leitung des Servers zu überlasten, ist aber sehr leicht durch entsprechende Filter zu unterbinden.

Ein *SYN-Flood-Angriff* hingegen versucht, den Server selbst anzugreifen und solange mit Anfragen zu beschäftigen, dass er keine anderen Anfragen mehr annehmen kann. Der Angreifer schickt dazu von seinem Client Rechner ein TCP-Paket (Transmission Control Protocol) mit gesetztem SYN-Flag[1] . Der Server antwortet darauf mit einem Paket, in dem SYN- und ACK-Flag[2] gesetzt sind. Um den normalen Drei-Wege-Handshake abzuschließen würde nun der Client wiederum mit einem SYN/ACK antworten. Diese Antwort wird aber nicht geschickt. Der Server, der auf die Antwort jedoch wartet, geht davon aus, dass sein SYN/ACK Paket verloren gegangen ist und sendet erneut eines, auf das er wiederum keine Antwort erhält. Stattdessen treffen weitere Anfragen von dem Angreifer ein, die der Server genauso behandelt. Alle SYN-Anfragen dieser halboffenen Verbindungen werden in einer speziellen Liste, der Backlog-Queue, gespeichert, bis die Verbindung komplett etabliert ist. Zwar wird ein Eintrag nach einer bestimmten Anzahl von Retrys aus der Liste gelöscht, aber wenn in der zwischenzeit genügend SYN-Floods eintreffen, läuft die Backlog-Queue voll. Wenn dies geschieht, kann der Server auf diesem Port keine weiteren Anfragen mehr annehmen und ist unerreichbar.

1 SYN steht für syncronise und ist die Anfrage den folgenden Verkehr zu synchronisieren.

2 ACK steht für acknowledge und zeigt an, dass die Gegenseite verstanden hat.

Ein Angreifer kann die Anzahl der SYN-Flood Pakete um ein Vielfaches steigern, indem er versucht einen dDoS (distibuted Denial of Service) Angriff zu starten. Dazu muss er die Gewalt über andere Rechner erlangen. Dies kann der Aggressor durch das Einschleusen von speziellen Trojanern und Würmern z.B. in E-Mails erreichen. Diese verteilen sich über befallene Rechner an andere E-Mailadressen, bis zu einem bestimmten Zeitpunkt der gemeinsame Angriff von den Wirtsrechnern gestartet wird.

3.2 Schadsoftware

Schadsoftware wie Viren, Trojaner und Würmer sind Programme, die einzelne Rechner oder Netzwerke schädigen sollen. Dabei können die Absichten, die hinter diesen Angriffsformen stehen sehr unterschiedlich sein. Sie können dazu dienen, einen Wirt auszuspionieren und dem Angreifer für ihn interessante Informationen zu übermitteln. Desweiteren kann das Ziel eines Angriffs sein, den Wirt durch Zerstörung seiner Daten zu schädigen oder einen Angriff eines ganz anderen Zieles vom Wirtsrechner aus vorzubereiten. Solche Schadprogramme nehmen Veränderungen an den Hardware-einstellungen (z.B. Netzwerkeinstellungen), dem Betriebssystem oder Anwendungen vor. Der wichtigste Unterschied zwischen Viren, Trojanern und Würmern ist die Infizierung und Ausführung. Viren sind unselbstständige Programmroutinen, die an Dateien angehängt sind und vom Anwender ausgeführt werden müssen, um aktiv zu werden. Dies geschieht meist durch Dateianhänge in E-Mails, die eine vermeintlich ungefährliche Dateiendung besitzen. Trojaner und Würmer dagegen sind selbstausführend und verfolgen das Ziel, sich auf möglichst vielen Rechnern zu verbreiten. Sie können sich sowohl im aktiven Quellcode[3] einer Webseite als auch in Dateien verstecken.

Aufgrund der Verbreitung und Architektur von Microsoft-Windows Betriebssystemen gibt es sehr viel mehr Viren, die diesem System gefährlich werden können. Zwar gibt es auch Schadsoftware, die UNIX und deren Derivate (z.B.: Linux, BSD, und Mac/OS X) angreifen, jedoch sind diese Betriebssysteme durch die restriktive Rechtevergabe und Architektur weitaus besser geschützt. Diese Umstände ist in Heterogenen Netzen[4] zu berücksichtigen.

Eine Unterart der Trojaner sind sogenannte Exploits (to exploit = ausnutzen). Exploits

3 z.B.: Java oderActiveX

4 Netze, in denen sich verschiedene Betriebssystemarchitekturen befinden.

zielen auf Sicherheitslücken von Anwendungen und verstecken sich in scheinbar harmlosen Dateien, denen jedoch ausführbarer Maschinencode beigefügt wurde. Wird dieser Maschinencode ausgeführt, kann es z.B. dazu führen, dass Pufferbereiche der laufenden Anwendung überschrieben werden. Die Anwendung kann abstürzen, sich anders als gewollt verhalten oder mit anderen Benutzerrechten ausgeführt werden.

3.3 Interne Gefahren

Den Benutzern eines Netzwerkes müssen bestimmte Tätigkeiten untersagt werden, um den einwandfreien Betrieb eines Netzwerkes zu gewährleisten und Computermissbrauch auszuschließen. Ist ein nicht authorisierter Benutzer in der Lage, Sicherheitseinstellungen zu verändern oder auf für ihn nicht bestimmte Daten zuzugreifen, so stellt dies eine erhebliche Gefahr dar. Wartungsarbeiten und das Bearbeiten der Sicherheitsregeln dürfen nur von authorisierten und eingewiesenen Personen durchgeführt werden. Authorisierte Personen sind der Administrator eines Netzwerks oder von ihm beauftragte Personen. Zu Computermissbrauch zählen zum Beispiel das Versenden von Spam-Mail aus dem Netzwerk oder das Starten von Attacken gegen das eigene oder ein fremdes Netzwerk durch Benutzer des eigenen Netzwerkes.

3.4 Gefahren für VoIP-Anwendungen im Netzwerk

Der Betrieb von VoIP-Anwendungen in einem Netzwerk birgt neben den oben beschriebenen Risiken noch indirekte Gefahren wie unerwünschten Gesprächsverkehr und die Gefahr der Abhörbarkeit.

Die Sprachdaten eines VoiP-Gespräches sind im Normalfall unverschlüsselt. Das Abhören im LAN (Local Area Network) ist mit entsprechenden Tools möglich (z.B.: rtpplay oder rtpdump), kann aber durch Verschlüsselungstechniken unmöglich gemacht werden, z.B.: SRTP (Secure Realtime Transport Protocol, beschrieben in RFC 3711) – Die Endgeräte müssen diese Verschlüsselung jedoch unterstützen. Weitere Verschlüsselungsmöglichkeiten sind IPsec oder TLS[5] (Transport Layer Security, beschrieben in RFC 2246). IPsec ist eine Verschlüsselung auf IP Ebene. Dies setzt für beide Geprächspartner allerdings ein

5 TLS ist eine standartisierte Weiterentwicklung von SSL 3.0.

zugehöriges VPN (Virtual Private Network) voraus. TLS ist sehr verbreitet bei VoIP Anwendungen. Ein Abhören des Gespräches außerhalb des Netzwerkes ist möglich. Sollen vertauliche Gepräche über das Internet geführt werden, ist eine Verschlüsselung nötig, denn es ist jedem der Gewalt über einen Knoten besitzt, möglich ein Gepräch abzuhören (Provider, Geheimdienste).

ARP-Spoofing (Adress Resolution Protocol) ermöglicht das Umleiten und Mitschneiden von Gesprächen in lokalen Netzwerken. Hierbei wird ein Rechner in der Mitte als Relay verwendet, auf dem mitgesnifft werden kann. Ein Gespräch kann dann mitgeschnitten werden. Ring-All-SIP-Pakete[6] können zu DoS-Angriffen führen, wenn sie als Packet-Flooding eingesetzt werden.

Durch Änderungen im SIP-Header kann sich ein Angreifer als jemand ausgeben, der er nicht ist. Ähnlich wie bei Phishing-Mail lässt sich auch hier nur schwer eine Manipulation erkennen. Die Authentizität des Anrufers ist somit nicht gewährleistet, denn an sich ist der von außen kommende Anruf vollkommen in Ordnung. Das Mitlesen und Manipulieren von SIP-Daten kann jedoch durch SIPS (SIP over SSL) verhindert werden. Allerdings bieten kaum Anbieter diese Verschlüsselung an, denn sie ist sehr rechenintensiv und geht somit zu Lasten der Performanz. Hier hilft nur dieselbe Verhaltensregel wie bei Phishing-E-Mails : Gesundes Misstrauen.

Bei SPIT (SPAM over Internet Telephony) handelt es sich meist um Werbeanrufe. Diese vollkommen zu unterdrücken ist schwierig, da SPIT erst nach zustandekommen der Verbindung erkannt werden kann. Hier kann man lediglich Anrufer oder Anrufergruppen ausschließen. Allerdings müssten deren Kennungen dazu bekannt sein. Viele Hersteller von IP-Telefonie Geräten schieben genau dieses Tätigkeitsfeld den Providern zu. Es sollen entsprechende *rote* Listen erstellt werden. Diesen Listen sollte man jedoch skeptisch gegenüberstehen, da solche Listen wirkungslos werden, wenn der SIP-Header gefälscht wird.

Eine weitere Gefahr besteht darin, dass ein Angreifer sein Opfer anruft, und in den Nutzdaten Exploits mitschickt.

6 Anrufen aller registrierten Teilnehmer eines SIP-Servers.

4 Voice over IP

Voice over IP ermöglicht die Übertragung von Sprache über ein Netzwerk. Diese Technologie birgt für den Betreiber eines Netzwerkes jedoch die bisher beschriebenen Gefahren und bereitet je nach Architektur des Netzwerkes Schwierigkeiten bei der sicheren Implementierung. Um die Überlegungen der Diplomarbeit besser zu erläutern, soll in diesem Kapitel auf die Funktionsweise von SIP-basierter Kommunikation eingegangen werden.

4.1 Session Inititation Protocol

SIP-Pakete dienen zur Kontaktaufnahme zwischen den Endpunkten und werden üblicherweise als UDP-Verkehr[7] (User Datagram Protocol, beschrieben in RFC 768) auf Port 5060-5062 übertragen. Dies macht eine Kanalisierung in einer Firewall sehr einfach [RUPP-SIP]. Ein SIP-Paket besteht aus einer Start Line, dem Message Header und dem Message Body. Die Start Line enthält unter anderem die *Method*, die es einem SIP-Server ermöglicht, den Zustand einer Verbindung zu erkennen. In der nachfolgenden Tabelle sind die wichtigsten Methoden aufgeführt, die für eine VoIP-Verbindung benötigt werden. Es gibt noch weitere Methoden, die zur Abfrage von Endgerät-Eigenschaften, Verbindungszuständen und der Behandlung von erweiterten Funktionen dienen.

SIP-Method	Beschreibung der Methode
REGISTER	Ein Teilnehmer kann sich mit der Verwendung der REGISTER-Methode bei einem Registrar Server registrieren.
INVITE	Mit der INVITE-Methode wird ein VoIP-Gespräch von einem Teilnehmer initialisiert.
ACK	Die ACK-Methode dient zur Signalisierung, dass die Verbindung zustande gekommen ist, und mit dem Senden der Sprachdaten begonnen werden kann.
BYE	Die BYE-Methode wird verwendet wenn, ein Teilnehmer auflegt und das Gespräch endet.

7 Die Verwendung von TCP als Transportprotokoll ist auch möglich, jedoch wird UDP explizit empfohlen.

SIP-Method	Beschreibung der Methode
CANCEL	Die CANCEL-Methode dient zum Abbrechen des Gespräches, wenn die maximale Wartezeit auf den angerufenen Teilnehmer überschritten wird, oder der anrufende Teilnehmer vor der Reaktion des Angerufenen auflegt.

Tabelle 1: SIP-Methods

Außerdem werden während des SIP-Datenverkehrs unter anderem die Kontaktadressen, Codecs und Transportprotokolle für den nachfolgenden RTP-Datenstrom festgelegt. Dazu gibt es im SIP-Header ein Feld, in dem diese als SDP-Daten (Session Description Protocol, beschrieben in RFC 2327) eingetragen werden. SDP ist kein eigenständiges Übertragungsprotokoll, sondern ein Hilfsmittel, um genaue Parameter für die RTP-Verbindung festzusetzen.

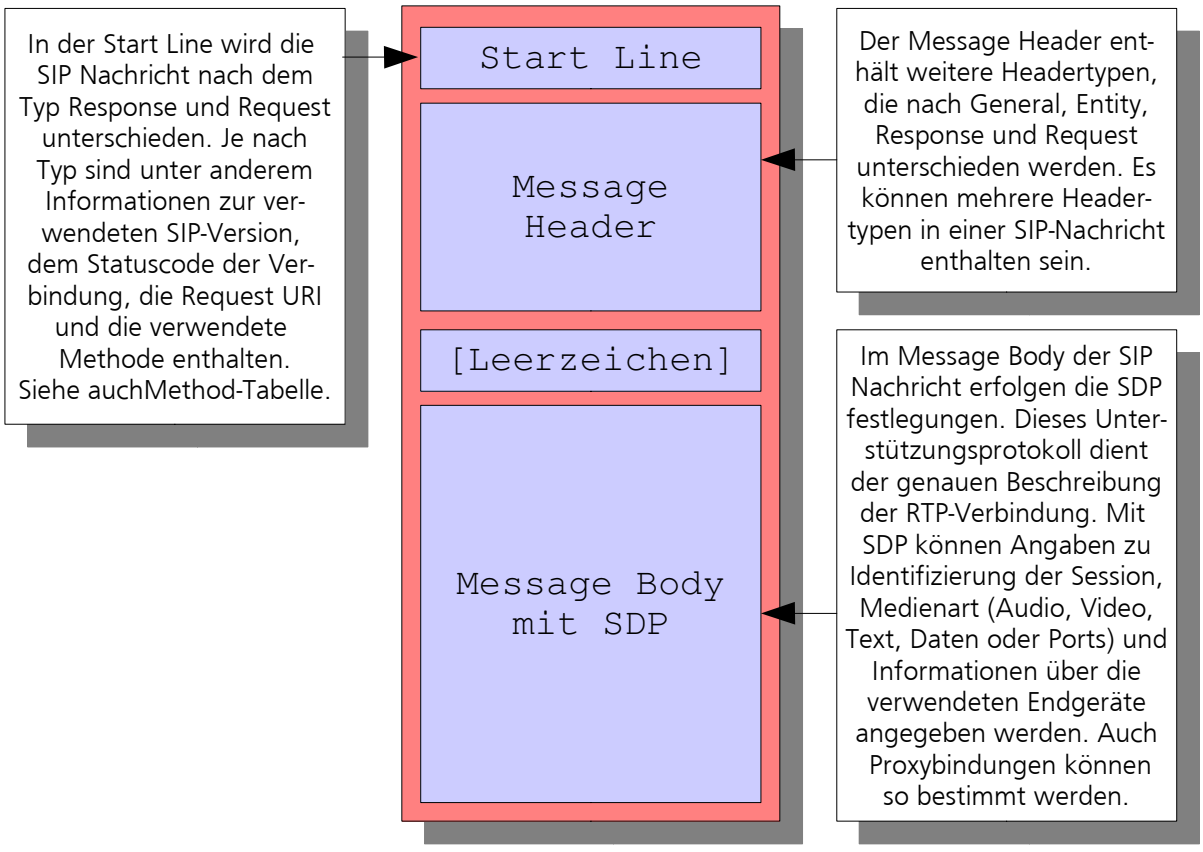


Abbildung 1: Aufbau einer SIP-Nachricht

4.2 Realtime Transportprotokoll

RTP-Verkehr (Realtime Transport Protocol, beschrieben in RFC 3550, früher RFC 1889) besteht aus kodierten Daten in *Stücken* (z.B. 160 Bytes/20 msec, abhängig vom verwendetet Codec) und dient der kontinuierlichen, echtzeitsensitiven Übertragung von Daten. Daher wird beim RTP-Verkehr das verbindungslose Protokoll UDP verwendet, denn ein verlorengegangenes Paket wird zu einem späteren Zeitpunkt nichtmehr benötigt und bedarf keiner erneuten Sendung. Ein großes Problem bei der Kanalisierung von RTP-Datenströmen ist der Bereich der genutzten Ports. In den RFCs wurden keine spezifischen Ports für den RTP-Verkehr festgelegt. Somit liegt der mögliche Bereich im gesamten UDP-Highportrange (Port 1024 bis 65535) [IL-COL]. Zwar beschränken einige Anbieter von VoIP-Endgeräten diesen Bereich, jedoch geschieht dies nicht einheitlich.
Ein weiteres Problem ist der Aufbau des RTP-Paketes selbst. Das Realtime Transport Protocol ist Payload des UDP-Paketes. Von außen gesehen ist ein RTP-Paket somit durch einen Paketsniffer wie Ethereal oder eine Paketfilter-Firewall nicht ohne Weiteres zu erkennen. Ethereal und kommerzielle Firewalls bedienen sich deshalb eines Tricks. Sie kontrollieren die SDP-Daten der vorangehenden SIP-Nachrichten. Hier sind die ausgehandelten Ports, Quell- und Zieladressen enthalten. Diese werden in eine Liste eingetragen. Kommt nun ein UDP-Paket, dass zu einer der gespeicherten Adress/Port-Kombinationen passt, wird dieses Paket als RTP-Paket eingeordnet. Ein einfacher Paketfilter kann dies jedoch nicht.

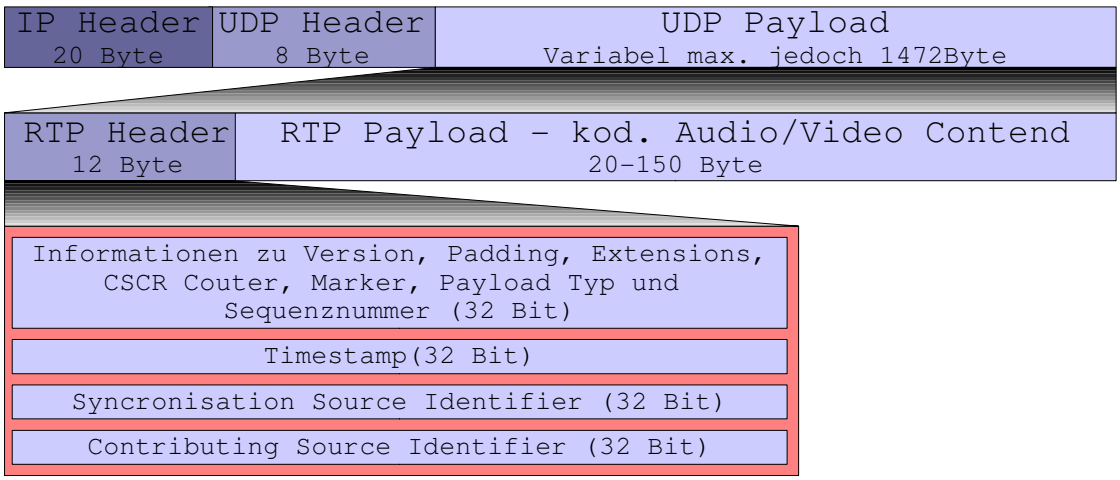


Abbildung 2: Aufbau eines RTP-Paketes

4.3 Wie funktioniert SIP-basierte VoIP Kommunikation ?

Ein Teilnehmer A, der ein VoIP-Gespräch führen möchte, muss sich zunächst bei einem SIP-Server A registrieren, um erreichbar zu sein. Ein VoIP fähiger Computer mit einer VoIP-Software oder ein VoIP-Phone sind vorausgesetzt. Der SIP-Server speichert seine momentane Lokation in Form seiner IP-Adresse. Weiterhin wird diese IP-Adresse, bis zur Abmeldung mit der URI (Uniform Ressource Identifier, beschrieben in RFC 3986) des Teilnehmers verknüpft. Somit ist die URI in Form von sip:TeilnehmerA@SIP-ServerA personenbezogen und unabhängig vom Standort des Teilnehmers. Um die SIP-Technologie zu erläutern, muss nun eine Fallunterscheidung vorgenommen werden.

Der erste Fall beschreibt den Sachverhalt, dass beide Teilnehmer auf dem gleichen SIP-Server registriert sind. Möchte nun ein Teilnehmer B, der ebenfalls am SIP-Server A registriert ist, Teilnehmer A anrufen, wird zunächst ein Verbindungswunsch (INVITE) an den SIP-Server gestellt. Dieser kennt die IP-Adresse von Teilnehmer A, leitet das INVITE an ihn weiter und sendet an Teilnehmer B ein TRYING als Bestätigung zurück. Teilnehmer A sendet daraufhin ein RINGING zurück an den Server, der es seinerseits an Teilnehmer B weiterleitet. Das RINGING signalisiert die Durchstellung des INVITE Paketes. Wird das Gespräch nun von Teilnehmer A angenommen, so sendet dieser ein weiteres Paket OK zurück, welches eine Portaushandlung für den folgenden Sprachdatenverkehr enthält. Dieses Paket wird ebenfalls über den Server vermittelt. Teilnehmer B quittiert dieses Paket wiederum über den Server mit einem ACK Paket. Erst jetzt folgt der eigentliche Sprachdatenverkehr in Form von RTP-Datenverkehr, direkt zwischen den beiden Teilnehmern. Wird das Gespräch nun beispielsweise von Teilnehmer A beendet, sendet er ein BYE an den SIP-Server, der das BYE an den anderen Teilnehmer B weiterleitet. Teilnehmer B antwortet über den SIP-Server mit einem OK. Das Gespräch ist nun beendet und Sockets werden wieder freigegeben.

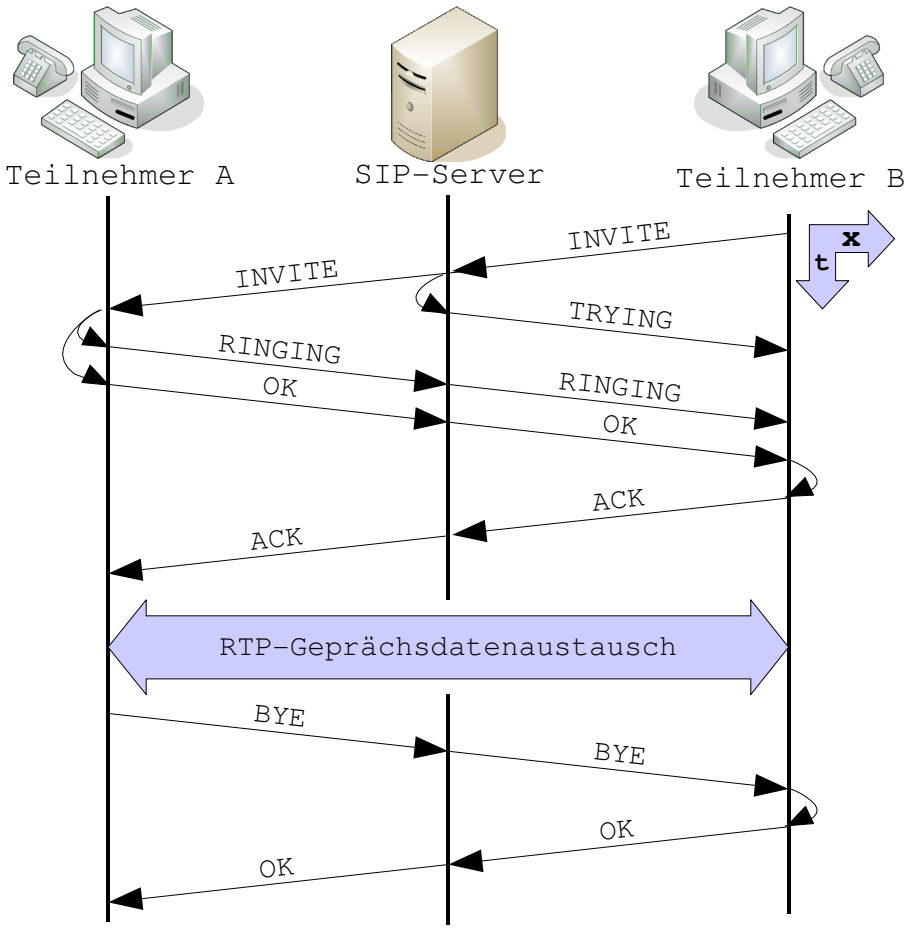


Abbildung 3: Ablauf eines SIP-Nachrichtenaustausches mit einem Server

Im zweiten Fall ist Teilnehmer A auf SIP-Server A registriert und Teilnehmer B auf SIP-Server B registriert. Teilnehmer A möchte nun Teilnehmer B anrufen. Nachdem der SIP-Server A das INVITE empfangen hat, stellt dieser eine DNS-Anfrage an einen DNS-Server, um den Adressteil der URI aufzulösen. Ist die IP-Adresse des gewünschten SIP-Servers B dann bekannt, sendet der SIP-Server A das INVITE weiter und informiert Teilnehmer A mit dem TRYING. Der SIP-Server B kennt die IP des Teilnehmers B, leitet wiederum das INVITE an ihn weiter und quittiert mit einem TRYING an SIP-Server A. Dieses TRYING wird jedoch von SIP-Server A nicht weitergeleitet, sondern dient nur der Signalisierung, dass SIP-Server B das INVITE weitergegeben hat. Der weitere Verlauf ist analog zum ersten Fall, mit der Bindung der SIP Pakete an die SIP-Server.

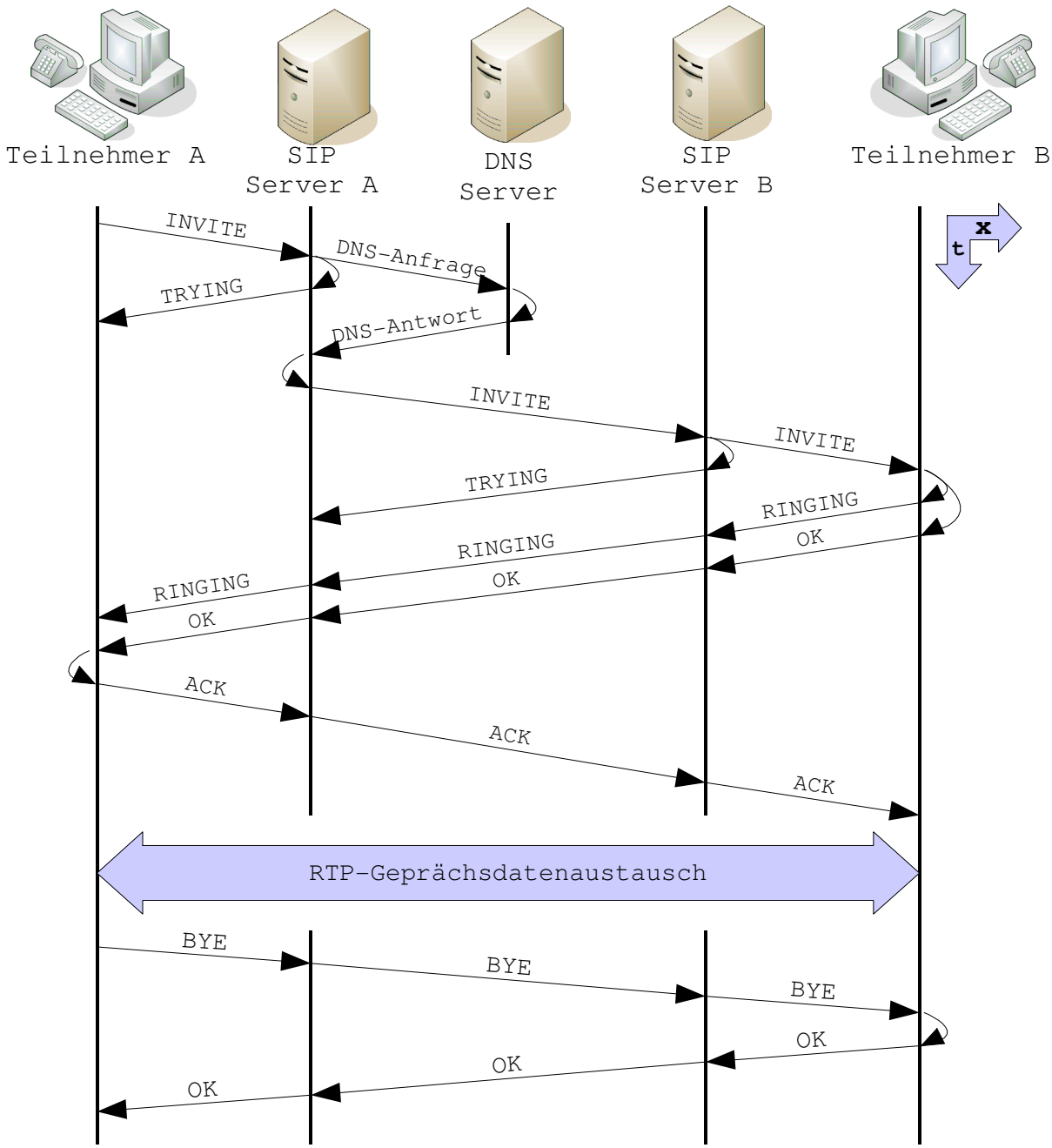


Abbildung 4: Ablauf eines SIP-Nachrichtenaustausches mit zwei Servern

Ein SIP-Server, der Teilnehmer verwalten kann, wird auch Registrar-Server genannt. Um zu vermeiden, dass sich unautorisierte Personen auf einem Server registrieren, muss eine Authentifizierung des Teilnehmers erfolgen.

5 Ansätze zur Netzwerksicherheit

Der Schutz eines Netzwerkes ist mit dem Tunen eines Automotors zu vergleichen. Um einen Automotor zu optimieren, genügt es nicht, nur einen anderen Auspuff zu verwenden, sondern es sind weiterhin Maßnahmen an Luftansaugmenge, Verdichtung und vielem mehr vorzunehmen. Übertragen auf die Sicherheit eines Netzwerkes spielen viele Einzelkomponenten eine Rolle, die nur in ihrer Summe einen maximalen Schutz bieten. Dabei beschränken sich die einzelnen Komponenten einer Sicherheitsarchitektur nicht nur auf Hard- und Software, sondern auch auf Verhaltensregeln in einem Netzwerk.

5.1 Vertrauensbereiche

Um Daten in einem Netzwerk auszutauschen, muss der Zugriff auf sie möglich sein. Betreibt man beispielsweise einen Webserver, auf dem die Internetpräsenz gehostet wird, muss dieser Server auch vom Internet erreichbar sein. Da man einem Fremden weniger vertraut als einem Benutzer des eigenen Netzwerkes, möchte man einem Fremden natürlich keine Einsicht in interne Daten gewähren. Weiterhin traut man einem fremden Netzwerk eher einen potenziellen Angriff zu, als dem eigenen Netzwerk. Dennoch gibt es auch für Benutzer des eigenen Netzwerkes Grenzen, die nicht überschritten werden dürfen. So sollte es ihnen z.B. nicht gestattet sein, sicherheitsrelevante Einstellungen vorzunehmen oder bestimmte Software mit deren Netzwerkprotokollen zu nutzen. Es ist deshalb ratsam, Netzwerke in Vertrauensbereiche einzuteilen und deren Rechte zu reglementieren. Diese Abstufung lässt sich beliebig differenzieren, je nach Anforderung an das eigene Netzwerk. Hier soll eine Abstufung in drei Vertrauensbereiche (nicht, stark eingeschränkt und eingeschränkt vertrauenswürdig) verwendet werden. Das Internet ist nicht vertrauenswürdig und erlangt nur Zugriff auf bestimmte Server und bestimmte Dienste (z.B. Webserver, SIP-Server). Desweiteren soll bestimmter Datenverkehr aus dem Internet nur auf Anforderung des internen Netzes in das eigene Netzwerk gelangen. Benachbarte Subnetze von anderen Abteilungen sind stark eingeschränkt vertrauenswürdig und erhalten zu den Rechten, die der nicht vertrauenswürdige Bereich besitzt, weitere Rechte, wie z.B der Zugriff über SSH auf bestimmte Geräte. Das eigene Netzwerk ist eingeschränkt vertrauenswürdig und erwirbt neben den Rechten der anderen

Vertrauensbereiche wiederum erweiterte Rechte, die zur Erfüllung der Aufgaben der Benutzer benötigt werden. Grundsätzlich sollte kein Bereich als Vertrauenswürdig gelten, da dies maximale Verwundbarkeit eines Netzwerkes bedeutet.

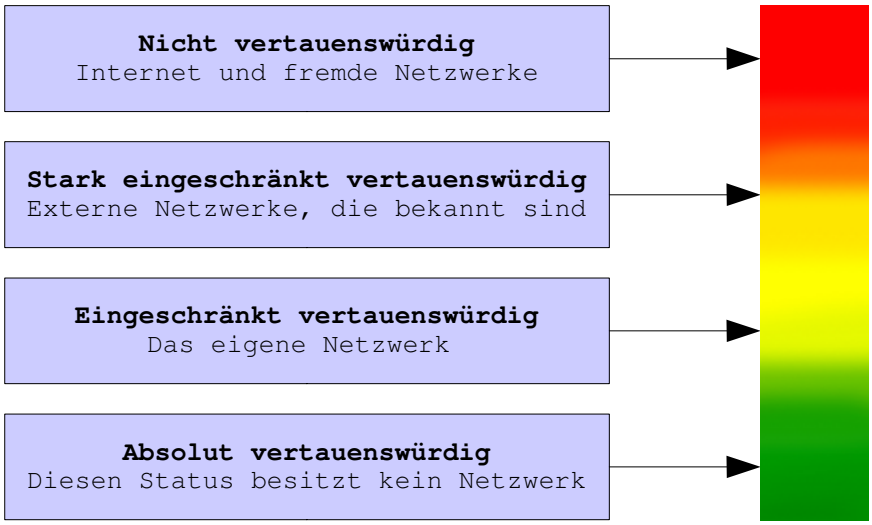


Abbildung 5: Vertrauensbereiche

5.2 Firewalls

Eine Firewall beschränkt den Zugriff zwischen zwei Netzwerken. Zum Schutz eines Netzwerkes kommen hier verschiedene Technologien zum Einsatz. Man unterscheidet Firewalls hinsichtlich Ihrer Betriebsarten und Fähigkeiten.

Bei einer *Paketfilterung* werden Filterregeln in Form einer Liste erstellt, die den erlaubten Datenverkehr enthält. Ein ankommendes Datenpaket wird nun auf das Protokoll, die verwendeten Ports, Ziel- und Quelladresse hin untersucht und mit den Einträgen der Liste verglichen. Trifft eine Regel auf das Paket zu, wird die der Filterregel zugeordnete Verfahrensweise ausgeführt. Das kann eine Erlaubnis für das Paket sein die Firewall zu passieren, aber auch ein Verbot. Um den Fall abzufangen, dass für das Paket keine Filterregel existiert, gibt es eine globale Regel die immer greift, wenn keine es Übereinstimmung zwischen Liste und Paketart gibt. Diese Regel steht am Ende der Liste und verwährt allen anderen Protokolltypen die Passage durch die Firewall. Diese statische Methode der einfachen Paketfilterung prüft jedoch nur die IP-Header.

Eine *Stateful-Inspection* Firewall geht noch einen Schritt weiter und prüft zudem die Flags

und Headeroptionen. Somit kann festgestellt werden, ob das Paket zu einer bestehenden und authorisierten Verbindung gehört. Weiterhin unterstützt eine Stateful-Inspection Firewall NAT (Network Adress Translation, beschrieben in RFC 2663).

ALG-Firewalls (Application Layer Gateway) arbeiten auf Ebene Sieben des OSI-Models, der Anwendungsschicht. Sie überprüfen neben den IP-Headern auch die Nutzdaten eines Paketes [ZIEG-FIRE][TEC].

5.3 Demilitarisierte Zonen und Proxys

Demilitarisierte Zonen sind gesonderte Bereiche eines Netzwerkes, die gelockerte Sicherheitsrichtlinen besitzen, um dem externen Netz Dienste zur Verfügung zu stellen, ohne jedoch dem externen Netz Zugriff auf das interne Netzwerk zu gewähren. Die klassische DMZ besteht aus einem eigenständigen Netzwerk zwischen zwei Firewalls, die sicherstellen, dass nur erwünschter Datenverkehr auf die angebotenen Dienste zugelassen wird. Das zu schützende Netzwerk ist von außen für diese Dienste nicht ansprechbar. Erfolgt nun ein Angriff auf einen der Server, bei dem der Angreifer Gewalt über die Maschine erlangt, ist das zu schützende Netzwerk immer noch durch die zweite Firewall geschützt. In der Praxis werden hier unterschiedliche Firewall-Typen verwendet. Außen wird meist eine Paketfilter-Firewall verwendet, wohingegen die innere Firewall als Application Layer Gateway ausgelegt wird.

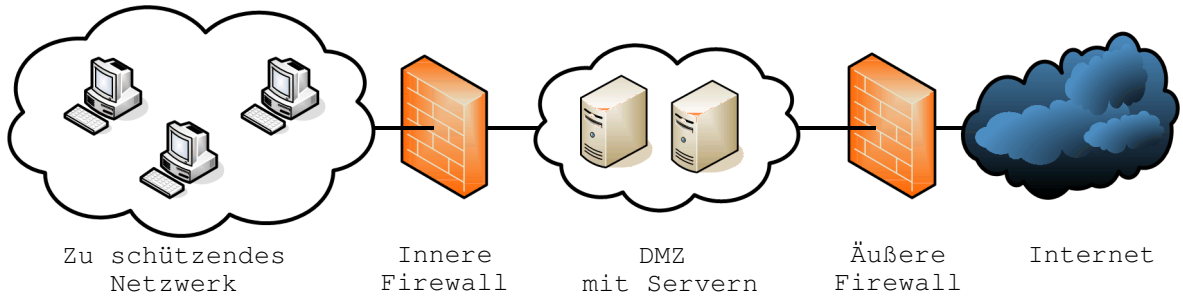


Abbildung 6: Aufbau einer klassischen DMZ

Die innere und äußere Firewall können auf einem Gerät zusammengefasst werden, denn eine Firewall mit mehreren Netzwerkschnittstellen ist in der Lage zu unterscheiden, auf welchem Interface ein Paket eintrifft oder es verlässt. Eine Firewall ist somit auch

Routingfähig. Allerdingst birgt diese Architektur den Nachteil des geringeren Schutzes. Bringt ein Angreifer die Firewall in seine Gewalt, so ist das dahinterliegende Netzwerk ungeschützt. Als Vorteil ist jedoch ein erheblich geringerer Kosten- und Arbeitsaufwand anzuführen.

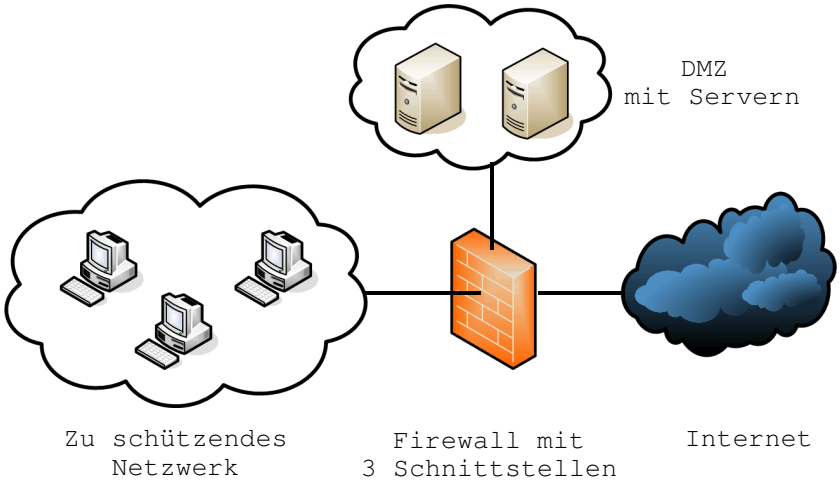


Abbildung 7: DMZ mit nur einer Firewall

Neben Servern, die Dienste für das interne Netzwerk wie auch für Fremdnetze zur Verfügung stellen, werden auch Proxy-Server in demilitarisierten Zonen platziert. Proxy-Server vermitteln oder stellen stellvertretend Anfragen für einen Client an einen Server. Dabei ist er für den Benutzer transparent. Proxy-Server können zusätzliche Funktionen übernehmen. Ein HTTP-Proxy kann beispielsweise bedenkliche Internetseiten sperren, oder häufig angeforderte Inhalte zwischenspeichern und so die Netzlast verringern. Genauso können SIP-Proxys und RTP-Proxys dazu beitragen SIP- und RTP-Verkehr zu lenken und zu kanalisieren.

5.4 Intrusion Detection System

Ein Intrusion Detection System (IDS) dient der Erkennung von Einbrüchen auf einzelnen Computersystemen oder ganzen Netzwerken. Mit Sensoren wird der Datenverkehr untersucht. Wird ein Angriff erkannt, kann die Reaktion vom Erstellen eines Eintages in ein Logfile bis hin zur sofortigen Benachrichtigung des Systemadministrators (z.B.: per E-Mail

oder SMS) gehen. Grundsätzlich gibt es zwei Verfahren bei der Intrusion Detection.
Bei der *Misuse Detection* wird anhand von Angriffsmuster-Signaturen, die in einer Datenbank enthalten sind, ein Angriff erkannt. Ist ein Angriffsmuster nicht bekannt, wird kein Alarm ausgelöst, weshalb die Datenbank ständig aktuell gehalten werden muss. Existiert für eine Angriffsart keine entsprechende Signatur, so bleibt ein Angriff unentdeckt. Das IDS *kennt* bei der zweiten Methode, der *Anormal Detection*, den *normalen* Netzwerkverkehr. Alles was anormal ist, wird als Angriff gewertet. Dies kann beispielsweise eine erhöte Anzahl von Zugriffsversuchen mit falschem Passwort sein, was Rückschlüsse auf einen Bruteforce-Angriff zulässt. Das System erfordert eine lange Einarbeitungszeit, mit einer hohen Anzahl von Fehlalarmen, bis der normale Netzbetrieb bekannt ist.

Desweiteren unterscheidet man zwischen hostbasierten- (HIDS) , netzbasierten Intrusion Detection Systemen (NIDS) und deren Mischung, den Hybridsystemen.

HIDS arbeiten nur auf einzelnen Computersystemen und erkennen:

- insbesondere Angriffe von Innen
- das Ausführen von Adminbefehlen eines Unbekannten
- modifikation von Daten
- Rootkits, Trojaner

NIDS überwachen ein Netzwerk und erkennen:

- DoS und dDoS Attacken
- ungültige Pakete (Ping of death)
- Angriffe auf die Applikationsschicht
- Zugriff auf Daten und Informationsdiebstahl
- Angriffe auf die Firewall und das IDS durch fragmentierte Pakete oder Angriffe auf die Zustandsüberwachung des IDS.
- Spoofing
- Portscans

Der Betrieb eines IDS ist für den Systemadministrator sehr zeitaufwendig, denn alle Alarme müssen ausgewertet werden. Problematisch daran sind Falschmeldungen. Man unterscheidet hier zwischen Falsch-positiv, was einen Fehlalarm darstellt, und Falsch-negativ, welches einen tatsächlich stattgefundenen Angriff nicht anzeigt. Während das erste nur störend ist, kann ein Falsch-negativ fatale folgen haben [SPEN-IDPS].

5.5 Intrusion Prevention System

Intrusion Prevention Systeme (IPS) arbeiten sehr ähnlich wie IDS. Allerdings sind sie in der Lage, neben dem Loggen und Benachrichtigen, selbständig auf Angriffe zu reagieren und Verbindungen zurückzusetzen. Ein Hostbasiertes Intrusion Prevention System (HIPS) arbeitet nur auf einem einzelnen Rechner und dient dazu, nach bestimmten Richtlinien Zugriffe zu beschränken. Netzbasierte Intrusion Prevention Systeme (NIPS) werden üblicherweise auf einem Router installiert und verhindern die Weiterleitung von gefährlichen Paketen [SPEN-IDPS].

5.6 Security Policy

Eine Security Policy (engl. = Sicherheitspolitik) eines Netzwerkes beschreibt sowohl Verhaltensregeln in einem Netzwerk, als auch die Sicherheitsarchitektur selbst und ist für alle Benutzer bindend. Sie legt Arten der Authentifizierung, Aufgabenbereiche und Verantwortlichkeiten fest. Sie spiegelt die Sicherheitsphilosophie wieder. Es handelt sich nicht einfach nur um eine bloße Aufstellung von Regeln, sondern um verbindliche Gesetze. Um qualitative Festlegungen für die IT-Sicherheit zu beschreiben, steht die Definition der Grundwerte am Anfang einer Security-Policy Erstellung. Der Grundschutz und der Schutzbedarf beschreiben, welche Komponenten des Netzwerkes Schutz benötigen und wie hoch der Schutz eingestuft wird. Ebenso wird festgelegt, wo und wann Schutz verfügbar sein muss. Dabei werden Festlegungen zu Verfügbarkeit, Vertaulichkeit, Integrität, Authentizität und Verbindlickeit gemacht. Die *Verfügbarkeit* beschreibt die Gewährleistung Daten, Funktionen oder Betriebsmittel in akzeptabler Zeit zur Verfügung zu stellen. *Vertraulichkeit* bedeutet ausschließlich berechtigten Personen benötigte Informationen zugänglich zu machen. Die *Integrität* eines Sytems beschreibt die physische

und logische Unversehrtheit, sowie den Schutz vor unberechtigten Änderungen an System und Daten. Unter *Authentizität* wird die Echtheit und Zuverlässigkeit des Benutzers oder des Kommunikationspartners verstanden. *Verbindlichkeiten* beschreiben eine fehlerfreie Abwicklung einer Aktion, sodass der Emfang einer Übermittlung von Informationen oder Daten nachweisbar ist.

Im folgenden sind einige Punkte mit kurzen Beispielen aufgeführt, um zu verdeutlichen wie, eine Security Policy zur Steigerung der Sicherheit eines Netzwerkes beitragen kann:

- Deklaration von authorisierten Personen
  Wartungsarbeiten und das Bearbeiten der Sicherheitsregeln dürfen nur von authorisierten und eingewiesenen Personen durchgeführt werden. Authorisierte Personen sind der Administrator des Netzwerkes oder von ihm beauftragte Personen. Eine Security Policy legt diese Personen fest.

- Vermeidung von Computermissbrauch
  Um Computermissbrauch auszuschließen, muss sich jeder Benutzer authentifizieren. Die Security Policy legt Authentifizierungsarten unter Berücksichtigung des Schutzbedarfs fest. Eine Mindestanforderung für Passwörter ist hier denkbar. Zum Beispiel: Eine Länge von acht Zeichen, bestehend aus einer Kombination von Zahlen und Buchstaben. Weiterhin darf das Passwort nicht trivial sein und muss unter Verschluss gehalten werden. Zusätzliche Sicherheit bietet der regelmäßige Wechsel von Passwörtern.
  Auch könnte eine Festlegung für streng vertrauliche Daten getroffen werden, auf die beispielsweise nur durch eine Authentifizierung mit biometrischen Daten zugegriffen werden darf.

- Vereinbarungen über die Veränderung des Sicherheitssystems
  Änderungen an den im Sicherheitssytem enthaltenen Komponenten sind nur mit Zustimmung oder auf Weisung des Administrators durch authorisierte Personen erlaubt. Eine Anpassung kann oder muss eventuell möglich sein, um auf Änderungen oder neue Technologien in Netzwerken zu reagieren.

- Einflussfreiheit der Sicherheitssysteme

  Auf den Komponenten des Security Systems dürfen keine Dienste oder Anwendungen installiert werden, die das System angreifbarer machen. Falls Installationen nötig sind, müssen entsprechende Sicherungsmaßnahmen durchgeführt werden.

- Wartung und Pflege

  Der Administrator oder eine von ihm beauftragte authorisierte Person ist mit der Wartung und Pflege des Security Systems betraut. Hierzu gehört eine regelmäßige Auswertung der Logdateien und das aktualisieren der Regelwerke. Außerdem sind eventuelle Sicherheitspatches der Betriebssysteme zu installieren. Aber auch Verhaltensregeln im Falle eines Angriffs oder Ausfalles werden hier festgelegt.

- Beschreibung der Sicherheitsarchitektur

  Eine Security Policy beinhaltet eine Aufstellung aller Sicherheitskomponeneten sowie einen Netzplan mit deren Position im Netzwerk. Ebenso auch alle anderen Komponenten mit deren Schutzbedarf.

- Zertifizierungen

  Festlegung bestimmter Güteklassen um benötigte Zertifizierungen zu erhalten. Dies kann für IT-Dienstleistungsunternehmen oder E-Commerce-Unternehmen nötig sein, um den Kunden zu belegen, dass verantwortungsbewusst mit deren Daten umgegangen wird.

Dies soll nur ein Überblick sein, zeigt aber sehr genau worauf, eine Securtiy Policy abzielt. Sie kann mitunter das mächtigste Glied eines Sicherheitskonzeptes sein, wenn sie zum Einen konsequent befolgt wird, zum Anderen aber auch so flexibel gestaltet ist, dass sie auch bei unvorhergesehen Zwischenfällen eine klare Verhaltensregel aufzeigt [IL-BSI].

5.7 Sicherheit für VoIP

Um VoIP Komponenten sicher zu machen gibt es verschiedene Ansätze, jedoch noch keine einheitliche Normierung. Neben einigen speziellen Mechanismen von VoIP-Phone Herstellern gibt es verschiedene Verschlüsselungstechniken, die aber zum Einen von der Gegenseite unterstützt werden müssen, und zum Anderen einen hohen administrativen Aufwand mit sich bringen. Will man eine solche Verschlüsselungstechnik einsetzen, kann es möglich sein, dass man einen Großteil der möglichen Teilnehmer, die diese Verschlüsselung nicht unterstützen, ausschließt. Eine globale Erreichbarkeit ist somit nicht mehr gegeben. Verschlüsselungsverfahren sind also nur anwendbar, wenn alle Instanzen (SIP-Provider und Endgeräte) diese unterstützen. Aus diesem Grund wurde im Februar 2005 die VOIPSA (VoIP Security Alliance) gegründet.

> „Die VOIPSA arbeitet an der Entwicklung geeigneter Anforderungsrichtlinien für VoIP. Hierbei werden aktuelle Erkenntnisse über bekannte Bedrohungen sowie Expertenwissen über Sicherheits- und Datenschutzaspekte berücksichtigt."[8]

Bis entsprechende Ansätze weiterverfolgt worden sind, muss ein VoIP fähiges Netzwerk so gestaltet werden, dass es durch die Implementierung der VoIP Komponenten nicht angreifbarer wird. Dies kann durch Verwendung von SIP- und RTP-Proxyservern in einer demilitarisierten Zone geschehen.

8 Jonathan Zar, Sr. Director Business Development bei SonicWALL und Secretary der VOIPSA. In einem Interview des Onlinemagazins www.tomsnetworking.de [IL-TOM]

6 Sicherheitskonzept für das Experimentiernetz

Da das Experimentiernetz bereits besteht und teilweise schon über Sicherheitsmechanismen verfügt, ist hier keine komplette Neukonzepierung des Netzwerkes vorgesehen, sondern eine flexible Anpassung der neuen Sicherheitskomponenten an das bestehende Netzwerk. Die vorangegangenen Kapitel umreißen die Gesichtspunkte, die für die Diplomarbeit berücksichtigt werden müssen.

6.1 Experimentiernetz

Durch Einsatz neuer Technologien im Experimentiernetz des DN-Labors der FH Köln befindet sich das Netzwerk im stetigem Umbau. Der folgende Netzplan kann daher nur eine Momentaufnahme darstellen.

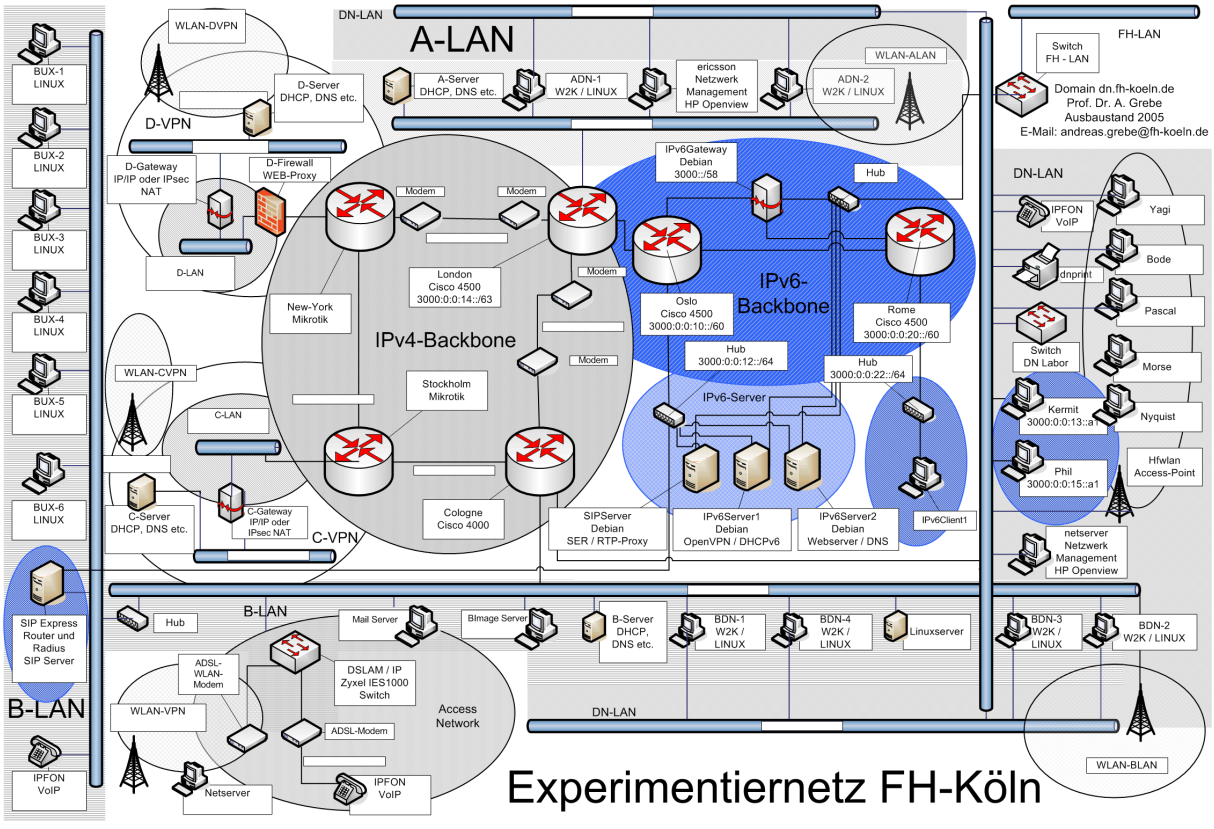


Abbildung 8: Netzplan des Experimentiernetzes

Dieses Forschungs- und Lehrnetzwerk enthält mehrere Subnetze, die von einem Verbund

aus Routern und Modems als Backbone, verbunden werden. Im Rahmen von mehreren Paktikumsversuchen können Studenten hier verschiede Messungen vornehmen. Weiterhin befindet sich ein IPv6 Netzwerk im DN-Netzwerk.

Durch Praktika, Studienarbeiten und Diplomarbeiten fluktuiert der Zugriff von verschiedenen Benutzern (Studenten) ständig , wodurch der Schutzbedarf nicht nur für das DN-Netz vorhanden ist, sondern auch für die angeschlossenen Netzwerke. So sollen beispielsweise Angriffe vom DN-Netzwerk nach außen unterbunden werden. Aus diesem Umstand ergeben sich die Position der Sicherheitskomponenten im Netzwerk und die Vertrauensbereiche.

6.2 Festlegung des Ortes und der Vertauensbereiche

Der Teil des Netzwerks, der durch die Diplomarbeit abgesichert werden soll, wird im folgenden als DN-Netz und das vorgelagerte Netzwerk als NT-Netz bezeichnet. Beide Netze befinden sich im FH-Netz, welches wiederum mit dem Internet verbunden ist. Zwischen DN-Netz und NT-Netz ist nun ein Sicherheitsmechanismus zu implementieren, der Flexibilität, Sicherheit und Kommunikationsmöglichkeiten bietet.

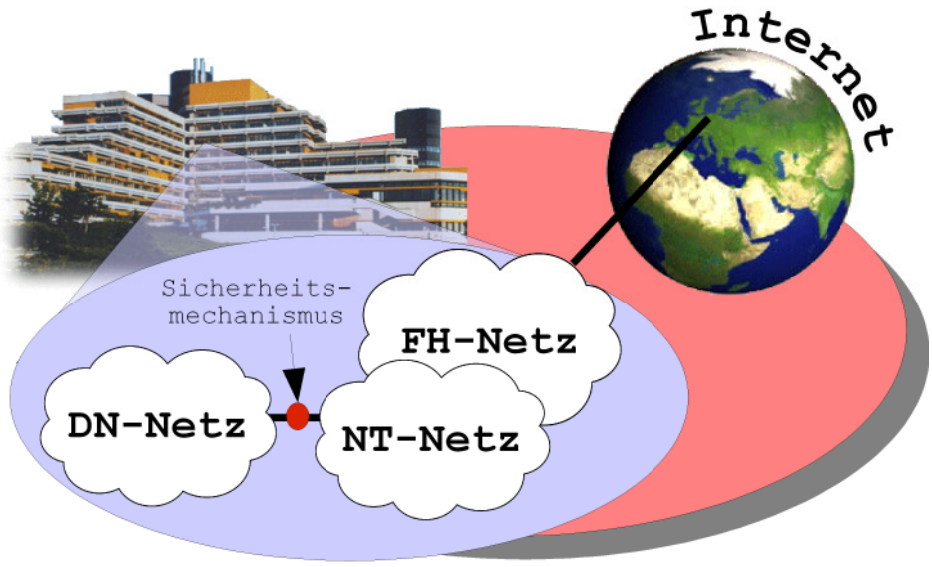


Abbildung 9: Anordnung der relevanten Netzwerke

Aus der Anordnung der Netzwerke und deren Benutzern ergeben sich die Vertauensbereiche. Das DN-Netz gilt als eingeschränkt vertrauenswürdig. Aufgrund der fachbereichseigenen Benutzer (Studenten) bleibt, obwohl man Lernwilligkeit und Identifikation mit dem eigenen Fachbereich voraussetzen kann, ein Restrisiko. Das vorgelagerte NT-Netz ist als stark eingeschränkt vertrauenswürdig anzusehen, da sich in diesem Netz auch andere Labore mit Studenten und Mitarbeitern des Fachbereichs befinden. Ein Angriff von diesem Netz ist vermutlich nicht zu erwarten, dennoch bewegen sich für das DN-Netz unbekannte Personen in ihm. Das FH-Netz und das daran angebundene Internet sind nicht vertrauenswürdig. In diesen Netzen bestehen potenzielle Gefahren durch gänzlich unbekannte Benutzer.

Netzwerk	Vertrauensbereiche
DN-Netz	Eingeschränkt vertrauenswürdig
NT-Netz	Stark eingeschränkt vertrauenswürdig
FH-Netz/Internet	Nicht vertrauenswürdig

Tabelle 2: Übersicht der Vertrauensbereiche

Zwischen das DN-Netz und das NT-Netz wird eine Firewall mit einer DMZ, in der sich ein SIP/RTP-Proxy befindet, implementiert.

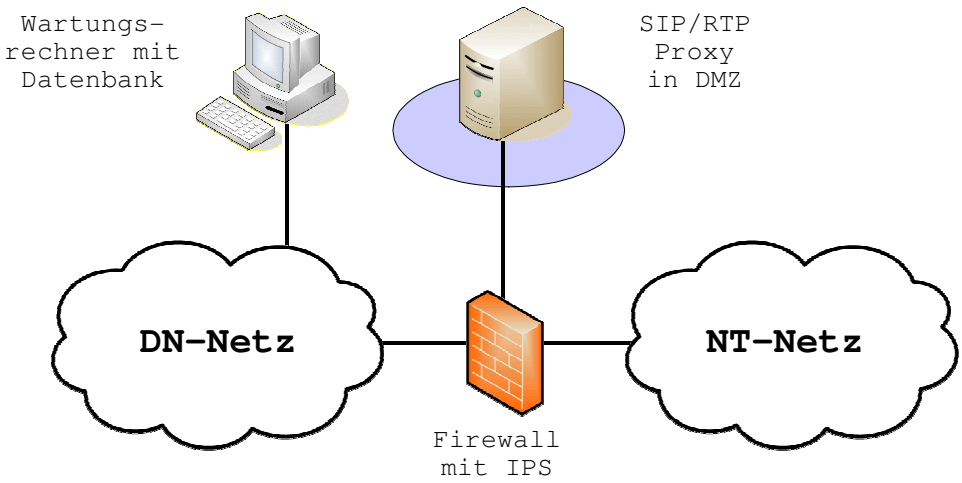


Abbildung 10: Aufbau der Sicherheitskomponenten

Die Firewall wird auf einem Linux-Rechner realisiert und beherrscht Stateful-Inspection. Weiterhin enthält die Firewall ein Misuse Detection IPS, an das auch ein Virenscanner angeschlossen ist. Mit dem SIP/RTP-Proxy Server wird der gesamte VoIP-Verkehr an die demilitarisierte Zone gebunden. Im DN-Netz wird ein Wartungsrechner installiert, von dem aus sich sowohl die Firewall als auch der Server in der DMZ verschlüsselt ansprechen lässt. Außerdem enthält dieser Rechner eine Datenbank, in der sämtliche abgewendeten Einbrüche, die das IPS bemerkt hat, eingetragen werden. Diese Datenbank kann mithilfe eines graphischen Frontends ausgewertet werden.

6.3 Auswahl der Software

Bevor auf die Hardware-Komponenten des Sicherheitssystems eingegangen wird, soll zunächst die verwendete Software vorgestellt werden. Die beschrieben Eigenschaften der Software erleichtern das Verständnis für das Gesamtkonzept und erklären somit auch die Auswahl der Software. Sämtliche verwendete Software unterliegt der GNU GPL (General Public Licence). Das GNU-Projekt wurde mit dem Ziel gegründet, vollständig freie Betriebssysteme und Anwendungen zu entwickeln. Die GNU General Public License ist eine von der Free Software Foundation herausgegebene Lizenz für *freie Software*. Freie Software bedeutet, dass sie beliebig genutzt, studiert, verwendet und kopiert werden darf.

6.3.1 Debian

Debian GNU/Linux ist ein Open Source Linux Betriebssystem, dass aufgrund seiner Stabilität und den umfangreichen vorkompilierten Softwarepaketen, die eine einfache Installation ermöglichen, ausgewählt wurde. Das Paket-Managementsystem APT erleichtert es, aktuellere Versionen oder neue Software, Utilitys oder Tools zu installieren oder zu deinstallieren. Außerdem werden sogenannte dependencies (engl. = Abhängigkeiten) berücksichtigt. Soll eine Software installiert werden, die bestimmte Bibliotheken benötigt, werden diese von APT mitinstalliert. Durch die große Nutzergemeinde sind fast täglich Sicherheitsupdates sowie neue Anwendungen

verfügbar, die stabil und sicher laufen. Für diese Diplomarbeit wurde auf allen drei Rechnern ein Debian Sarge 3.1 mit einem Kernel 2.6 verwendet. [IL-DEB] [ID-DEB].

6.3.2 iptabels

Das Programm iptables dient zur Steuerung der bestehenden Paketfilter (netfilter) im Kernel 2.4 und 2.6. Es wird über die Kommandozeile ausgeführt und kann damit sehr einfach in ein Shellscript eingebettet werden. Dabei lassen sich komplexe Filterregelketten erstellen, und per Modul Zusatzfunktionen hinzuladen, um eine Firewall zu erstellen. Zwar gibt es fertig kompilierte Firewalls für Linux (teils mit graphischem Frontend), jedoch unterstützen diese nicht die für diese Diplomarbeit benötigten Funktionen. Für Iptabels gibt es auch Administrierungswerkzeuge wie den FWBuilder, der mit einer graphischen, komfortablen Benutzeroberfläche aufwarten kann. Doch leider unterstützt auch dieser nicht das benötigte IP-Queueing, das das IPS dringend benötigt. Daher wurde die Firewall in einem einfachen Shellscript ausgeführt [IL-NET].

6.3.3 Snort-Inline

Snort-Inline ist ein netzbasiertes Intrusion Prevention System, das auf dem IDS Snort[9] von Martin Roesch beruht. Snort-Inline wird momentan von William Metcalf und Victor Julien betreut und weiterentwickelt. Wo Snort lediglich vermerken kann, dass ein Angriff stattgefunden hat, kann Snort-Inline reagieren. Snort-Inline kann Pakete von iptables aus einer Queue übernehmen und nach einem Regelsatz prüfen. Nachdem die Pakete von Snort-Inline entgegengenommen wurden, stehen mehrere Präprozessoren zur Verfügung, die auch in der Lage sind mehrere ankommende Pakete in Echtzeit zusammenzusetzen und deren zusammenhängenden Inhalt zu überprüfen. So kann zum Beispiel ein Exploit erkannt werden, auch wenn er über mehrere Pakete verteilt ist. Snort-Inline kann dann selbstständig die Verbindung zurücksetzen oder die Pakete einfach droppen. Weiterhin besteht die Möglichkeit, die Pakete nach Viren zu durchsuchen [IL-SNO][ID-SNO][SYN-SNO].

9 Weitere Informationen zu Snort unter www.snort.org

6.3.4 ClamAV

Der Open-Source Virenscanner ClamAV besitzt neben einer sehr umfangreichen Virensignaturdatenbank (über 40000 Viren, Würmer und Trojaner) auch den Dämon Freshclam, der nach einer frei einstellbaren Zeit, diese Datenbank über das Internet aktualisiert. ClamAV ist in der Lage, von der Kommandozeile aus Dateien zu scannen, aber auch mit einem entsprechenden Plugin E-Mails nach Würmern zu untersuchen. Auf der Firewall wird er jedoch nur benötigt, um einem Präprozessor von Snort-Inline die Signaturdatenbank zu Verfügung zu stellen, damit Snort-Inline die Pakete auf Viren hin untersuchen kann [IL-CLA].

6.3.5 MYSQL

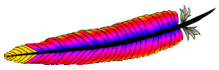
Der Vorteil einer MySQL-Datenbank liegt in ihrem relationalen Aufbau, der aus Tabellen und Spalten besteht, die sich aufeinander beziehen. Solche Beziehungen werden als Schlüsselwert in einer Spalte abgelegt. Hier wird das menschliche Denken immitiert, viele Einzelobjekte zu Gruppen zusammenzufassen. Durch diese Herangehensweise ist eine relationale Datenbank wie MySQL intuitiver und performanter als ein hierarchisches Datenbankmodel, das alle Objekte in einem großen Speicherbereich ablegt. Die Pakete, die von Snort-Inline als bedrohlich einstuft, werden nicht nur verworfen, sondern auch geloggt. Dies kann in einem einfachen Logfile geschehen, aber auch als Eintrag in einer MySQL-Datenbank. MySQL wird von der Firma MySQL AB weiterentwickelt und steht sowohl unter der GPL als auch unter einer kommerziellen Lizenz zur Verfügung. Hier wird die Open Source Variante als Debian-Softwarepaket verwendet [IL-MYS].

6.3.6 BASE

BASE steht für *Basic Analysis and Security Engine* und basiert auf dem Code des *Analysis Console for Intrusion Databases* (ACID) Projekts. Betreut wird das Projekt über das Sourceforge.net von Kevin Johnson. BASE ist ein Web-Frontend, zur Analyse der Datenbankeinträge eines Snort oder Snort-Inline. Dies

gestaltet sich sehr Komfortabel, denn protokollierte Angriffe können nach verschieden Kriterien sortiert und auch graphisch dargestellt werden. So sind beispielsweise Sortierungen nach Angriffsziel oder Quelle, Art des Angriffs oder Angriffe über einem bestimmten Zeitraum möglich [IL-BASE] [ID-BASE].

6.3.7 Apache mit GD und PHP Unterstützung



Apache ist ein Web-Server, der dem Web-Frontend BASE eine Schittstelle bietet, um abgefragte Einträge der Datenbank in einem Browser dazustellen. Der Webserver muss BASE die Bildmanipulationsbibliothek GD und PHP zur Verfügung stellen. Weiterhin ist der Apache Web-Server aufgrund seiner Modularität in der Lage, mit verschieden Skriptsprachen (z.B.: PHP, Perl) dynamische Inhalte darzustellen und diese aus Datenbanken wie beispielsweise MySQL abzurufen. Unter Debian gibt es den Apache Webserver als vorkompiliertes Softwarepaket, der die benötigte Unterstützung von GD und PHP bietet.

6.3.8 SIP-Express Router

Der SIP-Express Router (SER) ermöglicht den Verbindungsauf- und abbau eines VoIP-Gepräches und wurde von der Gruppe iptel.org, die dem Fraunhofer Institut FOKUS[10] angehört, entwickelt. Dabei wurde die Software möglichst performant ausgelegt, sodass auch auf einem einfachem Personal Computer mehrere 1000 Gepräche gleichzeitg verarbeitet werden können. Der SER ist nicht nur auf VoIP beschränkt, sondern unterstützt zudem andere Medienformate wie Messaging (Jabber) und Videostreaming. Da der SER modular aufgebaut ist, bietet er die Möglichkeit viele Erweiterungen zu implementieren. So kann die Software als SIP-Registrar, SIP-Proxy oder SIP-Redirect Server eingesetzt werden. Der SER ist in der Lage, eine eigene Benutzerverwaltung in Form einer Datenbank zu führen, sowie Schnittstellen für MySQL, oder Radius Authentifizierung anzubieten [IL-SER][ID-SER].

10 Das Fraunhofer Institut für Offene Kommunikationssysteme FOKUS ist eine Einrichtung der Fraunhofer Gesellschaft zur Förderung der angewandten Forschung e.V.

6.3.9 RTP-Proxy

Ein RTP-Proxy kann nicht nur RTP-Datenströme an sich binden. Er kann ebenfalls helfen Probleme bei NAT (Network Adress Translation), der Übersetzung von IP-Adressen zu umgehen. Weiterhin ist mit Ihm eine Kanaliserung des RTP-Datenstroms möglich, was zur Kontrolle der Gesprächdaten genutzt werden kann. Sei es um die Geprächsdauer zu Abrechnungszwecken zu ermitteln, oder um ein Gespräch abzuhören. Der Portaone *RTPproxy* kann er über einen Socket mit dem SER kommunizieren. Trifft am SER ein INVITE ein, so wird vom RTP-Proxy zunächst überprüft, ob bereits ein Port vergeben wurde; ist dies nicht der Fall, so weist der RTP-Proxy den ersten freien UDP-Port aus dem Portrange diesem Gespräch zu. Der SIP-Express Router übernimmt diese Vorgabe in den SDP-Bereich der SIP-Nachricht. Somit werden die folgenden RTP-Datenpakete über den RTP-Proxy fließen [IL-RTP][ID-RTP].

6.4 Vorstellung der Hardwarekomponenten

Die einzelnen Softwarekomponenten werden, wie aus der folgenden Abbildung ersichtlich, auf den einzelnen Hardware Komponenten des Sicherheitssystems verteilt.

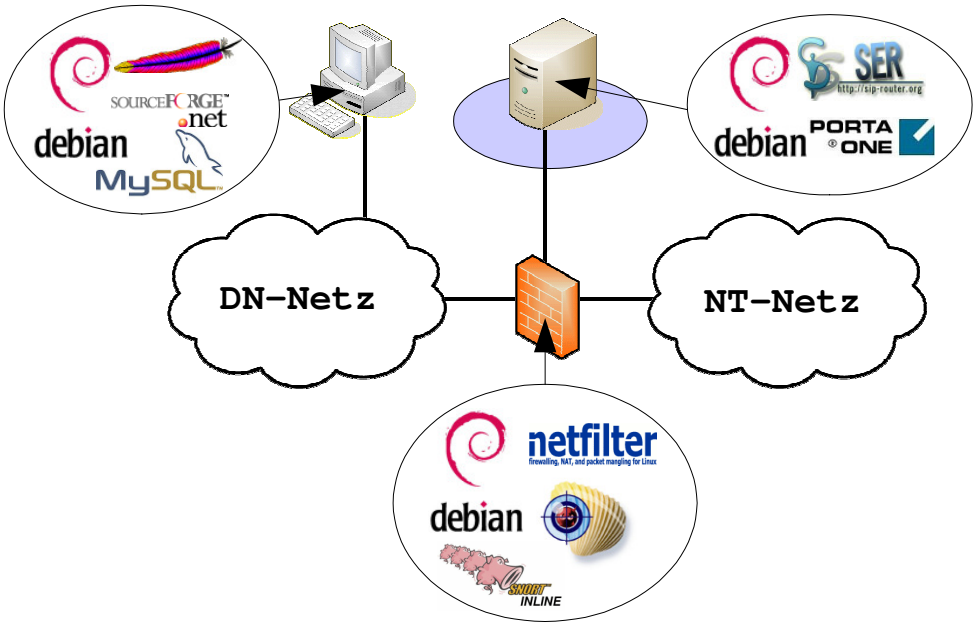


Abbildung 11: Verteilung der Software

6.4.1 Firewall

Das Ziel einer Firewall ist den Datenverkehr zu reglementieren und unerwünschten Datenverkehr herauszufiltern. Es sollen beispielsweise die angeschlossenen Netze nicht mit NetBIOS-Nachrichten[11] der Windows-Systeme überflutet und unerwünschte oder unsichere Protokolle vermieden werden. Benötigte Protokolle sollen jedoch verwendet werden können. Dazu werden Richtlinien festgelegt, welcher Verkehr die Firewall passieren darf. Alles was nicht erlaubt ist, ist verboten. Daher müssen nur die benötigten Protokolle berücksichtigt werden. Wichtig ist bei der Stateful Inspection die Richtung und der Zustand der Verbindung. So können Regeln aufgestellt werden, in denen ein Paket von außen, die Firewall nur passieren darf, wenn es zuvor von innen angefordert wurde. Der Verbindungszustand kann jedoch nur für TCP-Verkehr genutzt werden, da der TCP-Header diese Information enthält.

Das ICMP Protokoll nimmt mit dem Ping-Befehl eine Sonderstellung ein. Da sich aus der Antwort auf einen Ping Informationen über den *angepingten* entfernten Rechner und seine *Existenz* herleiten lassen, soll dies vom DN-Netz aus möglich sein. Um der Aussenwelt jedoch sowenig Angriffsfläche wie möglich zu bieten[12], soll Ping zum einen nur aus dem NT-Netz möglich sein und dem Nachfrager nur mitgeteilt werden, dass die Maschine existiert. Die Firewall lässt also lediglich den ICMP Typ 8 (echo request) zu. Mit dieser Maßnahme können DoS-Broadcast-Angriffe nicht abgewehrt werden, sondern lediglich erschwert werden.

Die umseitige Tabelle enthält die Protokolle, die benötigt werden und die Firewall kontrolliert passieren dürfen. Die Richtungspfeile zeigen an, von welcher Seite ein Verbindung initiiert werden darf.

11 Windows ist ein sehr mitteilsames Betriebssystem, das in regelmäßigen Abständen seinen NetBIOS-Namen versendet. Auf diese Weise umgeht Microsoft das Routing. Nach einer Zeit ist jeder Windowsrechner allen anderen Windowsrechnern in einem Netzwerk bekannt.

12 Es gilt: Was man nicht sehen kann, kann man nur schwer angreifen.

Dienst	Zielports	Protokoll	Richtung
FTP	20 ; 21	UDP	DN-Netz → Alle Netze, nicht DMZ
Telnet	23	TCP	DN-Netz → Alle Netze
HTTP	80 bzw. 8080	TCP	DN-Netz → Alle Netze, nicht DMZ Firewall → Alle externen Netze DMZ → Alle externen Netze
SSH	22	TCP	DN-Netz ↔ NT-Netz DN-Netz → Alle Netze
DNS	53	TCP	DN-Netz → Alle Netze, Nicht DMZ
SNMP	161 ; 162	UDP	DN-Netz ↔ NT-Netz
HTTPS	443	TCP	DN-Netz → Alle Netze, nicht DMZ
SIP	5060-5062	UDP	DN-Netz ↔ DMZ ↔ Alle Netze
RTP	dynamisch	UDP	DN-Netz ↔ DMZ ↔ Alle Netze
MySQL Transfer	3306	TCP	Firewall → Wartungsrechner
RIP	520	UDP	DN-Netz ↔ NT-Netz
Andere Dienste	Übrige Ports	TCP/UDP	DN-Netz ! Alle Netze

Tabelle 3: Erlaubte Protokolle und deren Intiierungsrichtung

Diese Kanalisierung wird durch iptabels ermöglicht. Eine Beschreibung zur Erstellung von iptabes-Regeln findet sich im folgenden Kapitel. Das komplette Firewallscript ist im Anhang zu finden. Das zuladbare ip_queue Modul sorgt dafür, dass Pakete in eine FIFO-Schlange eingegliedert werden und dort weiterverarbeitet werden können.

Mit dem Start der Firewall wird Snort-Inline gestartet. Snort-Inline besteht grundsätzlich aus vier Segmenten. Der *Paketsniffer* nimmt die Pakete von iptables aus der Queue entgegen und identifiziert sie nach Typ und Protokoll. Die *Präprozessoren* nehmen die Rohpakete entgegen und prüfen sie gegen bestimmte Plugins. Der stream4-Präprozessor setzt beispielsweise zusammengehörige Einzelpakete zusammen. Durch dieses *reassembling* ist Snort-Inline in der Lage, auch zusammengehörige fragmentierte Pakete einem Signatur Mustervergleich zu unterziehen. So kann nach einem einzelnen String eines Exploits gesucht oder ein Signaturvergleich mit der ClamAV Virendatenbank

durchgeführt werden. Die *Detection Engine* überprüft die Pakete gegen eine Reihe von Regeln und entscheidet anhand dieser Regeln, ob ein Paket eine Gefahr darstellt oder nicht. Die Regeln liegen in Listenform für das jeweilige Bedrohungsszenario vor (z.B. exploit.rules, ftp.rules, dos.rules) und müssen über das Konfigurationsscript nach bedarf eingebunden werden. Eine Regel[13] setzt sich aus dem Regel_Header und den Regel-Optionen zusammen.

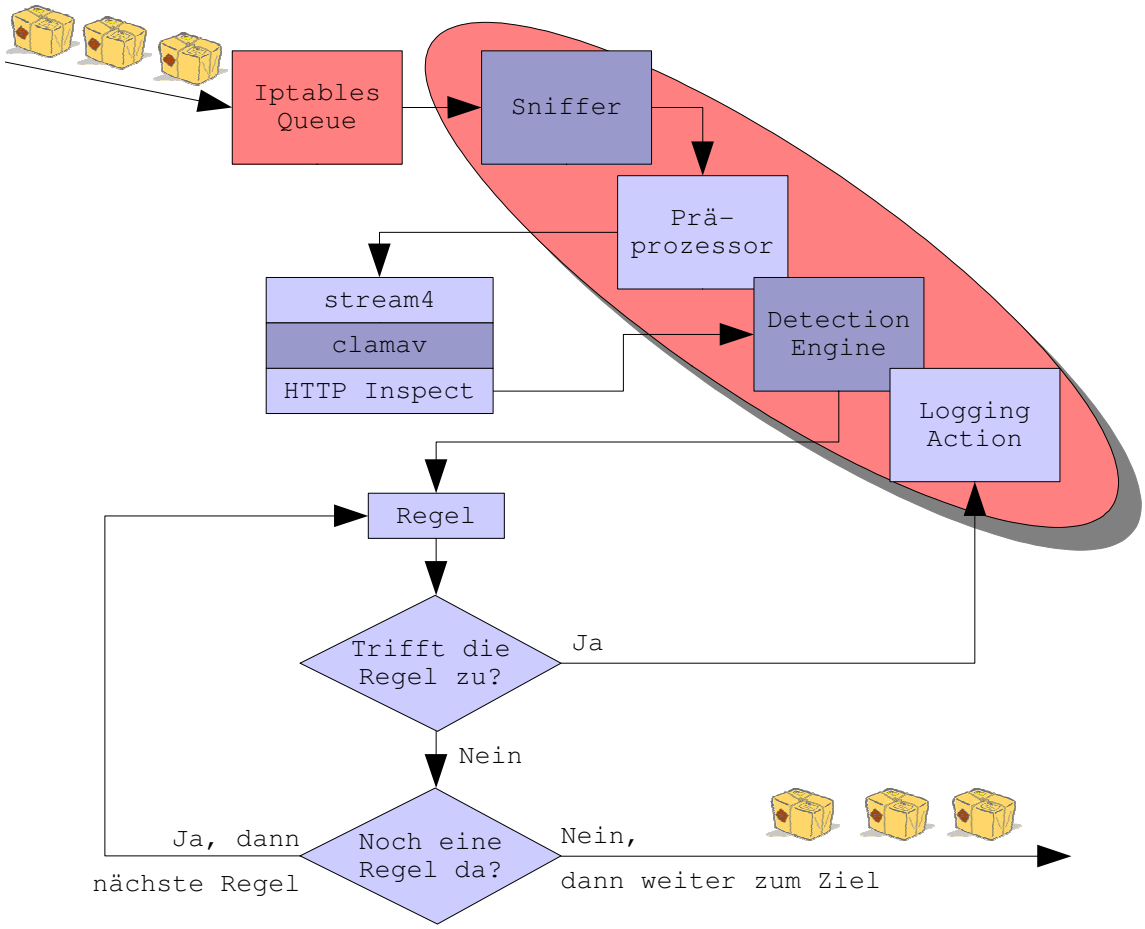


Abbildung 12: Arbeitsweise von Snort-Inline

13 An dieser Stelle soll auf das Buch [SYN-SNO] hingewiesen werden, das ausgezeichnete Beschreibungen zum Schreiben von Regeln und Präprozessoren enthält. Die Regeln werden von der Snort-Fangemeinde ständig aktuell gehalten, sodass die Selbsterstellung einer Regel jedoch meist nicht nötig ist.

Der Regel-Header enthält Action, Quelle, Richtung und Ziel des Paketes. Die Regel-Option enthält beispielsweise die Strings, auf die das Paket hin untersucht werden soll. Letztendlich sorgen die *Logging- und Actionkomponenten* dafür, dass im Alarmfall ein Eintrag in das Logfile oder die Datenbank geschrieben und das Paket entsprechend der Anweisung behandelt wird. Als Anweisung stehen neben den Anweisungen des ursprünglichen Snort noch die erweiterten Anweisungen drop, sdrop und reject zur Verfügung. Drop verwirft das Paket und protokolliert es, wohingegen sdrop es nur verwirft. Bei reject erzeugt Snort-Inline für ein TCP Paket ein TCP-RST Paket und für ein UDP-Paket ein ICMP-Unreachable. Nachdem das Paket protokolliert wurde, wird Iptabels angewiesen, das Paket zu verwerfen. Wird das Paket als gefahrlos eingestuft passiert es erst jetzt die Firewall zum Ziel.

6.4.2 SIP/RTP Proxy

Die Kanalisierung des SIP-Verkehrs liesse sich sehr leicht bewerkstelligen, da dieser Verkehr lediglich die UDP-Ports 5060-5062 benötigt. Jedoch handelt es sich bei SIP um ein verbindungsloses Protokoll, das von der Sateful Inspection nicht berücksichtigt werden kann. Somit müssten die UDP-Ports 5060-5062 auch in Richtung des eigenen SIP-Registrars geöffnet werden, was den SIP-Registrar auf diesem Portrange praktisch ungeschützt liesse. Daher bietet es sich an, den SIP-Verkehr über einen Proxy in der DMZ abzuwickeln.

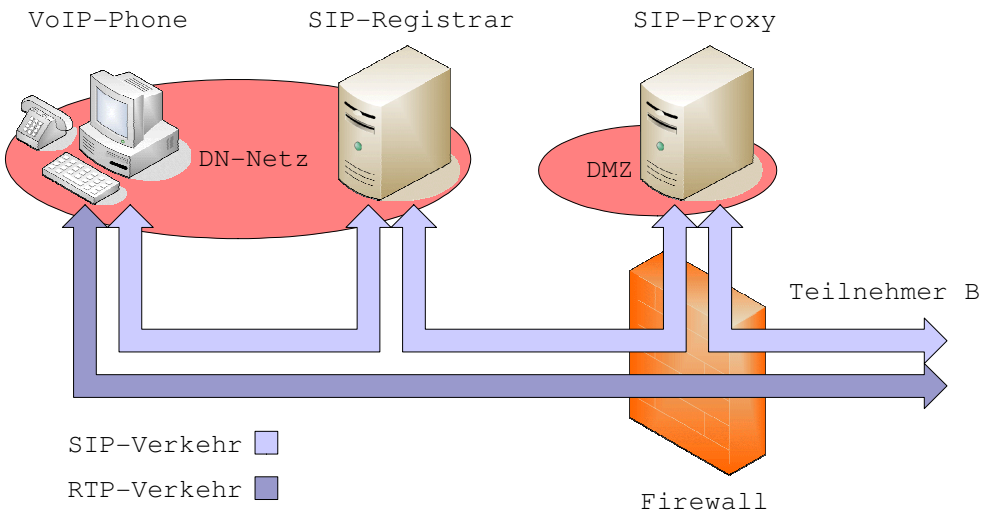


Abbildung 13: Kanalisierung des SIP-Verkehrs mittels SIP-Proxy

Der nach der Signalisierungsphase folgende RTP-Verkehr fließt direkt zwischen den Endgeräten. Um diesen RTP-Verkehr der SIP-Verbindung zuordnen zu können, muss die Firewall in der Lage sein, den SDP-Anteil der SIP-Nachrichten auszuwerten. Der RTP-Verkehr kann dann der SIP-Signalisierung zugeordnet und die benötigten Ports dynamisch geöffnet werden. Aus diesem Grund müssen keine Firewallports (>1024) von vornherein freigeschaltet werden. Wie bereits auf Seite 16 erwähnt, sind dazu nur kommerzielle Firewalls (z.B.: Cisco PIX[14]) fähig – Iptabels kann dies nicht.

Daher wird der RTP-Proxy von Portaone auf dem SIP-Proxy installiert, um den RTP-Verkehr an den Proxy zu binden und den gesammten RTP-Verkehr durch die DMZ fließen zu lassen. Ein weiterer Vorteil ist, dass der RTP-Proxy keine Pakete annimmt und weiterleitet, die nicht zu einer mit dem SIP-Proxy vereinbarten bestehenden Gesprächsverbindung gehören.

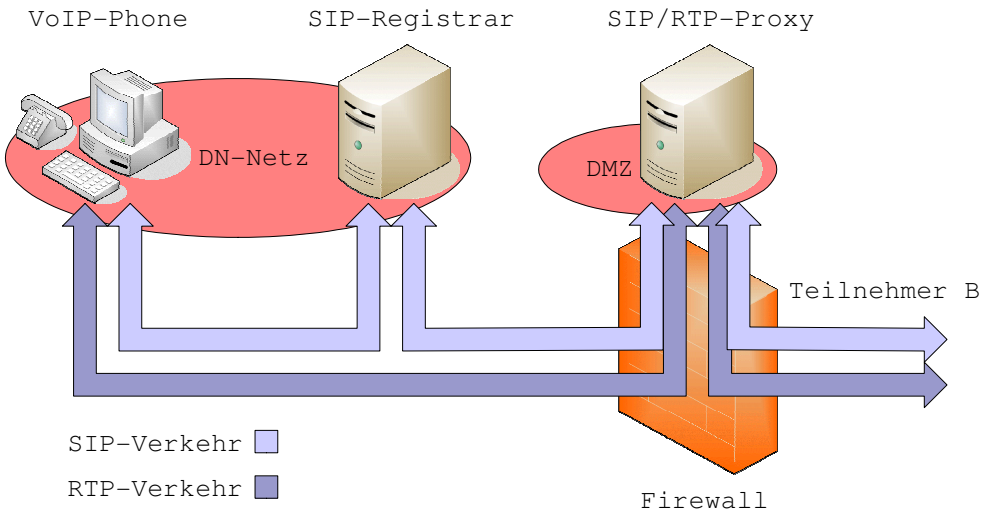


Abbildung 14: Kanalisierung der SIP/RTP-Daten mittels SIP/RTP-Proxy

Da es sich um einen von außen erreichbaren Server handelt, muss dieser Server gehärtet werden. Unter dem *Härten* eines Systems, versteht man das Abschalten aller nicht benötigten Dienste und das Installieren der aktuellen Sicherheitspatches.

14 In der Dokumentation von Cisco ist dieses Feature beschrieben [IL-CISC]

Um SYN-Flood-Attacken auf den Netzwerkpuffer (Backlog-Queue) des Servers entgegenzuwirken werden, Syncookies eingesetzt. Hierbei wird, wenn sich die Backlog-Queue füllt, statt des ACK ein Cookie gesendet. Dieser besteht aus einer Sequenznummer, Absender und Empfängeradresse, einem geheimen Schlüssel und einem kurzen Timeout. Wird der Cookie beantwortet, kann der Server von einem gültigen SYN ausgehen [ZIEG-FIRE].

Weiterhin ist festzuhalten, dass es sich bei dem bestehenden SIP-Registrar Server im DN-Netz um einen Server handelt, der eine Radius-Authentifizierung verlangt. Diese Implementierung wurde in vorangegangenen Diplomarbeiten behandelt und weiterentwickelt [IL-DIPL].

6.4.3 Wartungsrechner

Der Wartungsrechner enthält im Gegensatz zu Firewall und Proxy-Server ein Debian mit Desktop Umgebung, um dem Administrator hier ein komfortables Arbeitsumfeld mit Editoren und dem Browser für BASE zur Verfügung zu stellen. Er ist im DN-Netz plaziert und dient dazu, Änderungen an der Firewall oder dem Proxyserver via SSH vorzunehmen. Auf den Wartungsrechner, und somit auch auf die Firewall und den Proxy-Server, dürfen nur der Sicherheitsbeauftragte oder von ihm beauftragte Personen Zugriff haben, um Computermissbrauch auszuschließen. Außerdem erhält der Wartungsrechner von Snort-Inline die Einträge der geblockten Pakete für die MySQL-Datenbank.

7 Installation und Konfiguration

Diese Installationsbeschreibung soll die im Rahmen der Diplomarbeit durchgeführten Arbeiten dokumentieren und dem künftigen Sicherheitsbeauftragten des DN-Netzes als Handbuch dienen.

Aus sicherheitstechnischen Gründen ist es allgemein üblich, verwendete IP-Adressen im eigenen Netzwerk der Öffentlichkeit nicht preiszugeben. Daher soll die Installation an einem beispielhaften Aufbau mit privaten IP-Adressen eines Klasse C-Netzes erläutert werden. Der Aufbau stellt die Sicherheitskomponenten und die implementierten Fähigkeiten der benötigten Rechner dar. Diese Benennung dient dem besseren Verständnis. Eine Portierung auf beliebige Netzbereiche ist leicht möglich.

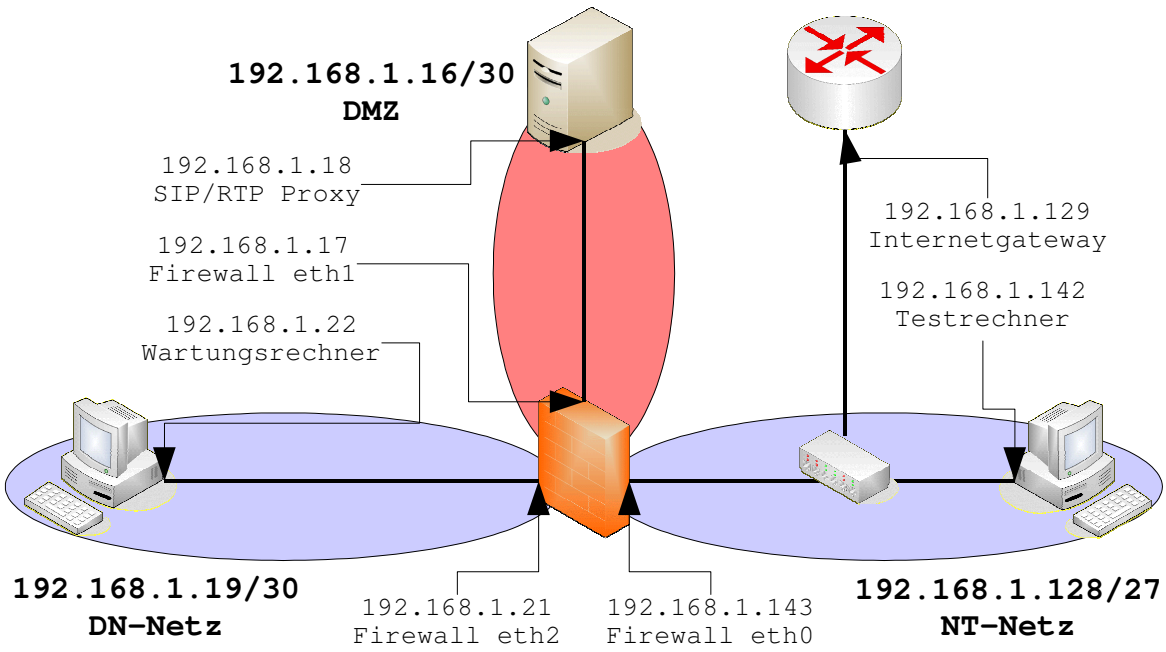


Abbildung 15: Aufbau des Beispielnetzes

Die reine Installation des Debian-Betriebssystems wird nicht beschrieben. Hier beschränkt sich die Beschreibung auf sicherheitsrelevante Einstellungen am Betriebssystem. Startscripte sowie die kompletten Konfigurationsfiles befinden sich im Anhang.

Für die Installation von Snort-Inline und BASE wurden einige Tutorials aus dem Internet verwendet. Diese bezogen sich jedoch auf andere Rahmenbedingungen und konnten daher größtenteils nur auszugsweise Verwendung finden [IL-SAV] [IL-GUR] [IL-LUG].

7.1 Debian Einstellungen

Um die Systeme zu härten, wurden alle nicht benötigten Dienste abgeschaltet. Insbesondere der `inetd` ist bei Debian ein Dämon, der Dienste *on demand* startet und zur Verfügung stellt. Dazu wird in der Datei `/etc/init.d/inetd` in der ersten Zeile `exit 0;` eingefügt, damit das Skript beim Start nicht ausgeführt wird. Dieser Dienst steht somit nicht mehr zur Verfügung.

Alle anderen nicht benötigten Dienste können mit dem `apt-get` Befehl entfent werden.

```
apt-get --purge remove telnet finger nfs-common pppoe ➪
    pppoeconf ppp hotplug
```

Anschließend wird der GRUB-Bootmanager besser gegen unbefugten Zugriff geschützt. Dazu sind in der Datei `/boot/grub/menu.lst` folgende Werte zu ändern:

```
timeout 3
passwort GEHEIM
```

Der Timeoutwert sorgt dafür, dass das System schneller gestartet wird. Das Passwort muss Security-Policy-konform sein.

Der User-Login muss ebenfalls abgesichert werden. In `/etc/login.defs` können diese Einstellungen vorgenommen werden.

```
FAIL_DELAY          10
FAILLOG_ENAB        yes
```

```
LOG_UNFAIL_ENAB     yes
SYSLOG_SU_ENAB      yes
SYSLOG_SG_ENAB      yes
```

Wenn das Login Passwort falsch eingegeben wurde, wird mit `FAIL_DELAY` die Zeit eingestellt, nach der eine erneute Anmeldung möglich ist. Dies erschwert Brute-Force Angriffe. Fehlgeschlagene Anmeldungen werden mit `FAILLOG_ENAB` in ein Logfile geschrieben, wobei mit `LOG_UNFAIL_ENAB` auch unbekannte Loginnamen vermerkt werden. `SYSLOG_SU_ENAB` und `SYSLOG_SG_ENAB` zeichnen die Benutzung des Kommandos `su` bzw. `sg` auf [RON-DEB].

Zunächst wurde auf allen drei Rechnern Debian installiert, um festzustellen, dass die Netzwerkverbindungen funktionieren. Außerdem ist das Bearbeiten der verschiedenen Scripte auf einem Editor (z.B.: Kate, Bluefish) in Desktopumgebung vom Wartungsrecher aus komfortabler als eine Bearbeitung auf der Konsole. Die SER-Konfigurationssdateien sowie das Firewallscript können nach der Bearbeitung mittels des Befehls `scp` über SSH SSL-verschlüsselt an die Zielrechner übertragen werden.

7.2 Installation der Firewall

7.2.1 iptables Firewallscript

Am Anfang eines Firewall-Scriptes werden die benötigten Module geladen, wobei je nach Bedarf noch Variablen für Ethernetschnittstellen, IP-Adressen oder Subnetze gesetzt werden können. Üblicherweise folgt nun das Aktivieren des Forwardings, um Routerfunktionalität zu erhalten, und das Setzen der Syncookies.

```
# Forwarding aktivieren
echo "1" > /proc/sys/net/ipv4/ip_forward
# Syncookies aktivieren – gegen flood Attacken
echo "1" > /proc/sys/net/ipv4/tcp_syncookies
```

Dies ist bei Debian nicht nötig, da diese Werte in `/etc/network/options` gesetzt werden können, wurde aber aus Gründen der Portierbarkeit in das Firewallscript aufgenommen.
Im Anschluß beginnt das Regelwerk mit dem Löschen alter, eventuell vorhandener Regeln und dem Setzen der Global Policy. Die Global Policy steht am Ende aller Regeln und ist die Defaultregel. Wenn die Regelkettenlisten durchlaufen wurden und keine Regel zutrifft, soll diese Regel greifen. So wird sichergestellt, dass alles verboten ist, was nicht ausdrücklich erlaubt ist.
Anschließend werden die Einstellungen für die Statefull Inspection und Snort-Inline festgelegt, woraufhin das eigentliche Regelwerk der erlaubten Verbindungen folgt. Es gibt zwei unterschiedliche Ansätze, eine Firewallarchitektur mit mehreren Ethernetschnittstellen unter iptables aufzubauen. Die Schnittstellen- und die Protokollarchitektur. Bei der *Schnittstellenarchtitektur* werden die Regelketten um die Schnittstellen angeordnet und die Regeln für jede Schnittstelle einzeln nach ein- und ausgehendem Verkehr sortiert. Dazu ist die Deklaration eigener Ketten nötig, um die Firewall für die Administartion übersichtlich zu halten. Auch verlängert diese Architektur das Firewallscript. Bei der *Protokollarchitektur* findet diese Zuordnung nicht statt. Es handelt sich um eine einzige Liste, die nach Protokolltypen geordnet wird. Da dies Änderungen der Firewall bezüglich eines Protokolls erleichtert, wurde diese Architektur hier verwendet.
Eine Regel baut sich aus Aufruf, (Tabelle), Kette, Protokoll, Quelle/Ziel, Match und Target auf, die teilweise mehrfach zur Spezifizierung verwendet werden können. Für jedes dieser Elemente gibt es eine Reihe von Prä- und Suffixen. Die einzelnen Elemente einer Regel können sehr umfangreich variiert werden, sodass an dieser Stelle lediglich eine kurze Beschreibung des Regelaufbaus erfolgen soll. Das folgende Beispiel erlaubt HTTP-Pakete vom DN-Netz durch die Firewall zu leiten, wenn sie zu einer neuen Verbindung gehören.

```
iptables -A FORWARD -p tcp -s 192.168.1.19/30 --syn ↪
    --dport 80 -m state --state NEW  ACCEPT
```

Wäre dies die einzige Regel, würde das Antwortpaket jedoch blockiert werden. Um der Antwort die Passage zu ermöglichen, muss die Stateful Inspection Einstellung am Anfang

gesetzt sein, oder eine explizite Antwortregel erstellt werden.

Ankommenden Paketen kann nicht nur eine Passage erlaubt werden, sie können auch an bestimmte Ziele umgeleitet werden. Dies geschieht hier für ankommende SIP-Pakete. Wichtig sind hierbei die ersten Pakete (INVITE), die an den SIP-Server gerichtet sind. Sollte beipielsweise ein SIP-Paket vom FH-Netz and den SIP-Server des DN-Netzes gerichtet sein, müssten die entsprechenden Firewallports in Richtung des DN-Netzes geöffnet sein, um den Server zu erreichen. Daher werden diese ankommenden SIP-Pakete mittels NAT, PREROUTING und DNAT in die DMZ umgeleitet und dort vom SIP-Proxy weiterverarbeitet.

```
iptables -t nat -I PREROUTING -p udp -i $EXTERN_ETH ↪
    --dport 5060:5062 -j DNAT --to 192.168.1.18
```

Die umseitige Abbildung zeigt anhand der oben genannten HTTP-Regel die wichtigsten Elemente und den Aufbau einer iptables-Regel. Sie zeigt die Kombinationsmöglichkeiten auf und soll als Kurzanleitung zum Erstellen neuer iptables-Regeln fungieren. Weitergehende Informationen finden sich in den Quellen [PUR-IPT] [IL-NET].

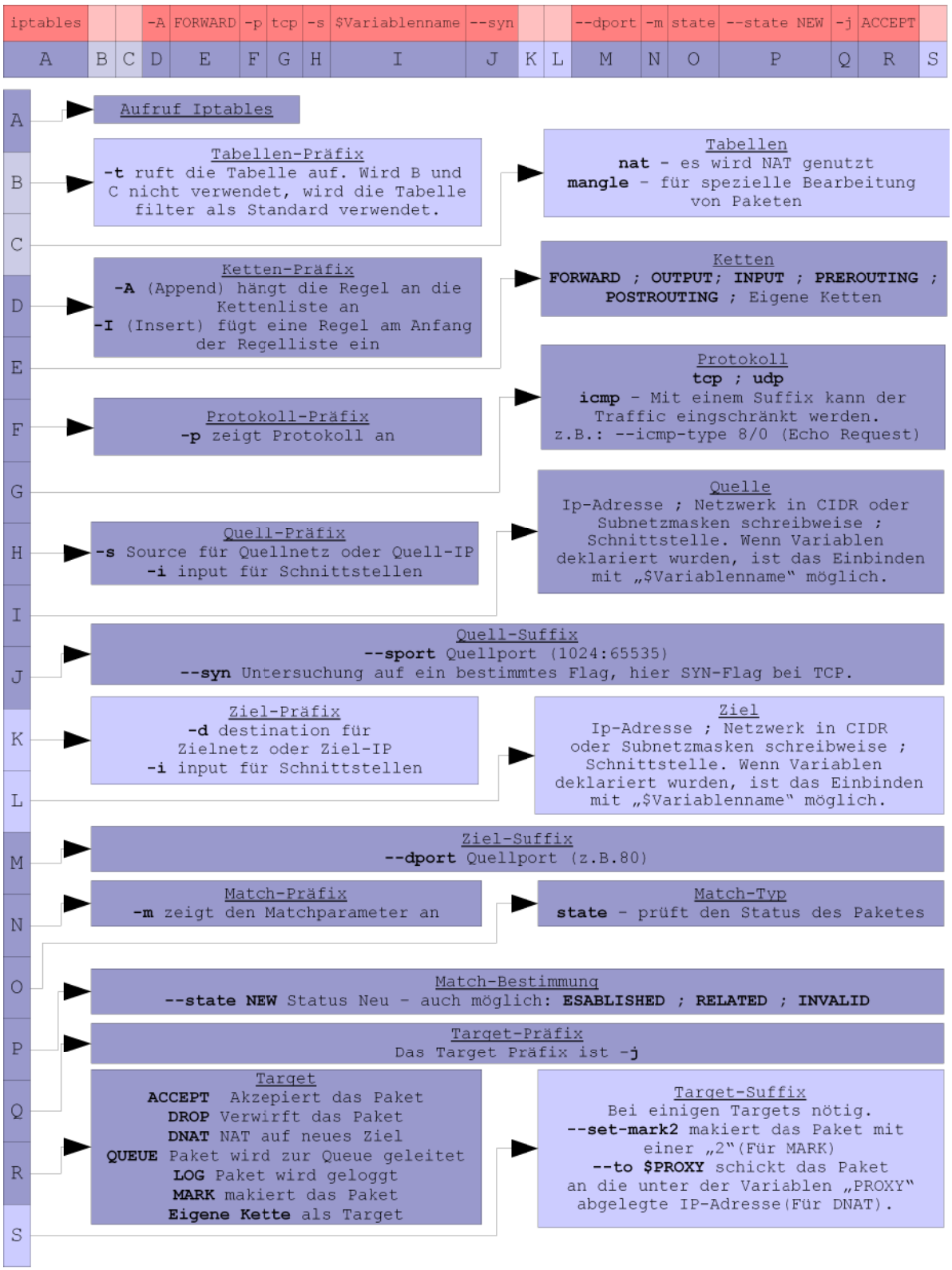


Abbildung 16: Aufbau einer iptables Regel

Um die Pakete an Snort-Inline weiterzugeben, werden sie in der Queue abgelegt. Damit Snort-Inline den Unterschied zwischen einem neuen und einem etablierten Paket erkennen kann, werden die Pakete von iptables vorher entsprechend markiert. Dies geschieht durch die Tabelle `mangle` und das Target `MARK`. Erst danach werden die Pakete tatsächlich in die Queue geschoben.

```
iptables -t mangle -A FORWARD -p tcp -s $DN_NETZ --syn
--dport 8080 -m state --state NEW -j MARK --set-mark 1

iptables -t mangle -A FORWARD -p tcp --sport 8080-m
state --state  ESTABLISHED  -j MARK --set-mark 2

iptables -I FORWARD -m mark --mark 1 -j QUEUE
iptables -I FORWARD -m mark --mark 2 -j QUEUE
```

In diesem Beispiel wird ein neues TCP-SYN Paket, das an den Port 8080 (HTTP-Proxy der FH-Köln) gerichtet ist, mit einer *1* und das Antwortpaket von Port 8080 mit einer *2* markiert. Eine Überprüfung der Quell-IP oder des Quellnetzes ist nicht nötig, da das zurückkommende Paket nur bearbeitet wird, wenn es zu einer angeforderten und etablierten Verbindung gehört. Es werden hier also vom DN-Netz ausgehenden SYN-Pakete inspiziert. Sollen auch alle anderen von innen kommenden Pakete inspiziert werden, muss der Regelsatz erweitert werden. Diese Funktion kann nicht in einer Regel zusammengefasst werden, da hier der Status des Paketes geprüft wird.

```
iptables -t mangle -A FORWARD -p tcp -s $DN-NETZ --syn
-m state --state ESTABLISHED --dport 8080 -j MARK
--set-mark 2
```

Beim Anlegen von Regeln ist das sogenannte Rule-shadowing zu beachten. Da iptables die Regeln innerhalb einer Kette der Reihe nach abarbeitet und es ebenfalls eine Bearbeitungsreihenfolge für die Tabellen gibt, kann eine Regel von einer anderen

überdeckt werden, sodass sie niemals zur Ausführung kommt. Um dies zu vermeiden, muss die Abarbeitungsweise der Tabellen und Ketten beachtet werden. Das folgende Bild zeigt die möglichen Wege eines Paketes durch iptables.

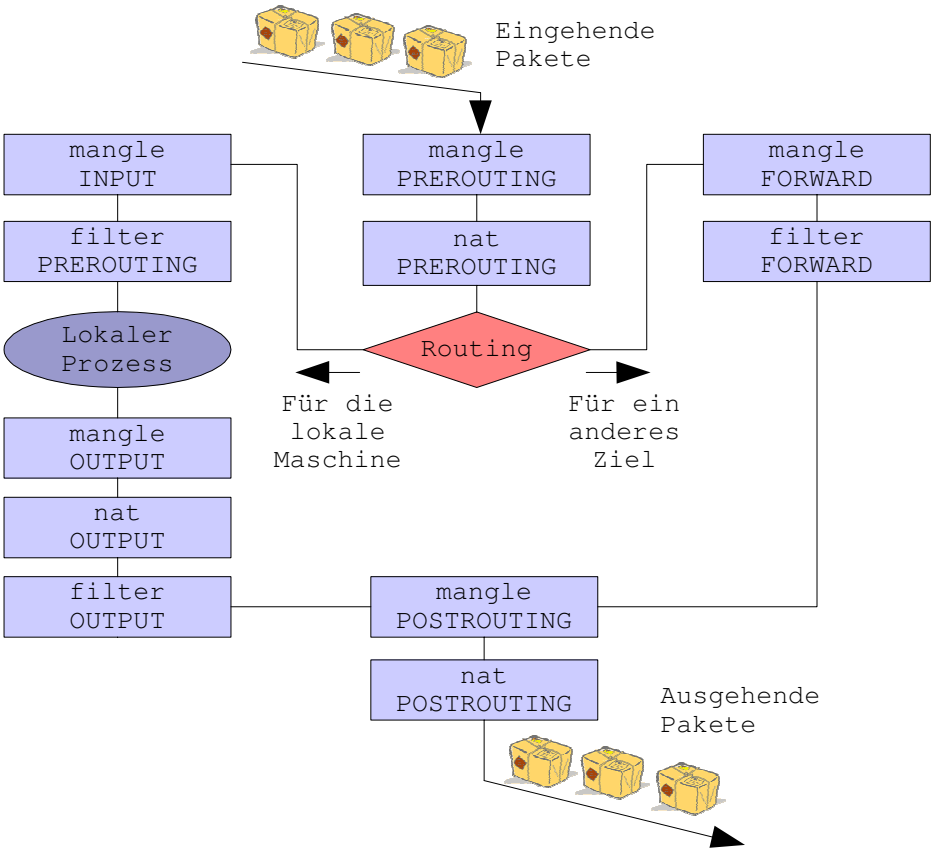


Abbildung 17: Abarbeitungsreihenfolge von iptables

Der Weg eines Paketes durch die Regelketten soll im Folgenden an zwei Beispielen ausschnittsweise aufgezeigt werden. Im ersten Beispiel ist ein SIP-Paket für einen lokalen Prozess bestimmt.

```
iptables -A INPUT -p udp -i $eth0 -d $LOKALE_IP
--dport 5060 -j ACCEPT
```

Zunächst wird das Paket von der Tabelle `mangle` aufgenommen und mit den `PREROUTING` Regeln verglichen. Wird keine passende Regelkette gefunden, wird das

Paket an die Tabelle `nat` und die Kette `PREROUTING` weitergeleitet. Nachdem auch hier keine passende Regel gefunden wurde, findet eine Routingentscheidung statt. Hier gelangt das Paket weiter zur Tabelle `mangle` in die `INPUT` Kette. Gibt es auch hier keine passende Regel, so wird das Paket an die Tabelle `filter` in die Regelkette `INPUT` weitergeleitet, wiederum vorausgesetzt dass keine passende Regel existiert. Letztendlich wird es an den lokalen Prozess weitergeleitet.

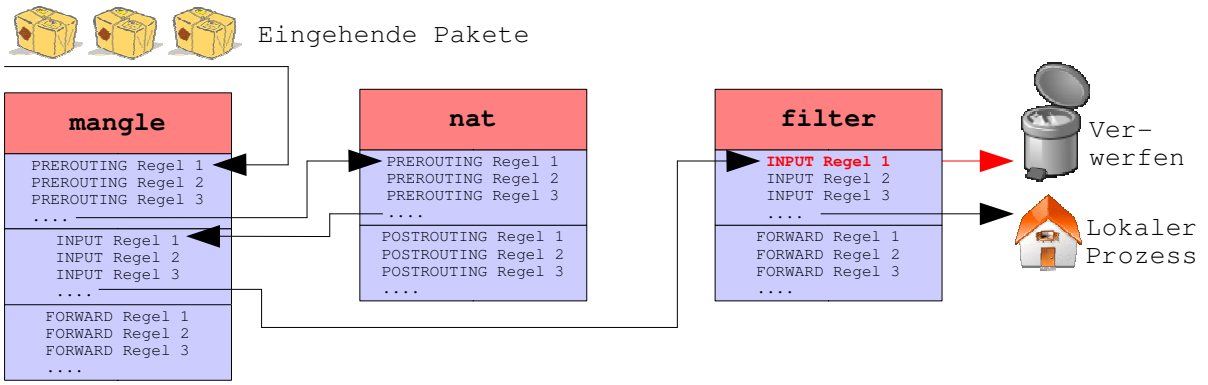


Abbildung 18: Beispiele einer Regelanwendung

Nun wird die folgende Regel hinzugefügt.:

```
iptables -I INPUT -p udp -i $eth0 -d $LOKALE_IP -j DROP
```

Diese Regel verbietet explizit, dass der lokale Rechner über die Ethernetschnittstelle eth0 per UDP angesprochen wird. Das Paket durchläuf also ebenso die einzelnen Tabellen mit ihren Listen wie im ersten Beispiel. Allerdings nur bis zur Tabelle `filter`. Hier greift die erste `INPUT` Regel und verwirft das Paket.

Die *SIP-INPUT-Regel* aus dem ersten Beispiel ist somit vom Rule-shadowing betroffen und wird niemals zur Ausführung kommen.

7.2.2 ClamAV

ClamAV kann über das Debian Paketmanagement installiert werden.

```
apt-get install clamav libclamav-dev
```

Das Paket `clamav` beinhaltet den Scanner, die Virendatenbank sowie den Updater `freshclam`. In `libclamav-dev` ist die von Snort-Inline benötigte `clamav.h` Datei enthalten. Dem Freshclam Dämon muss eine Möglichkeit gegeben werden, sich aus dem Internet neue Datensignaturen zu laden. Dies ist nur über HTTP möglich. Bei der Installation wird nach einem eventuellen HTTP-Proxy Server gefragt, der in die Konfigurationsdatei `/etc/clamav/freshclam.conf` eingetragen wird. In dieser Datei können Proxy-Anpassungen sowie der Updatezyklus eingestellt werden.

7.2.3 Snort-Inline Installation

Nicht nur weil Snort-Inline als Debian-Paket nicht zur Verfügung steht, muss es aus den Sourcen kompiliert werden. Snort-Inline bringt eine Fülle von Funktionalitäten mit, die aber nicht von jedem Anwender benötigt werden. Um Snort-Inline schlank zu halten, werden jeweils nur die benötigten Komponenten im Konfigurationsskript angewählt.

7.2.3.1 Abhängigkeiten

Snort-Inline besitzt sowohl in der Standartausführung als auch in der hier gewünschten Ausführung einige Abhängigkeiten. Es werden verschiedene Bibliotheken zum Kompilieren benötigt.

`libipq`

Diese Bibliothek ist im Debian Developer-Paket `iptables-dev` enthalten. Sie sorgt dafür, dass Snort-Inline mit iptables kommunizieren kann, und wird über das APT-System Installiert.

```
apt-get install iptables-dev
```

libnet-1.0.2a

Die Datei libnet ermöglicht es Snort-Inline Pakete zurück ins Netzwerk zu senden, wenn keine Regel für sie vorliegt. Zur Installation wird die Datei `libnet-1.0.2a.tar.gz` von [ID-LIBNET] nach `/usr/src/` kopiert und entpackt.

```
tar xvfz libnet-1.0.2a.tar.gz
```

Diese Datei findet sich mehrfach im Internet und ist oft fehlerhaft. Sie führt bei der Ausführung von `./configure` zu einer Fehlermeldung. Der Fehler liegt in der Datei `/usr/src/Libnet-1.0.2a/include/libnet.h`. Hier müssen mit einem Editor die Zeilen 87-89 in eine einzige Zeile geschrieben werden und das Kommentarzeichen vor dem `define` entfernt werden.

Nun kann die Bibliothek installiert werden.

```
./configure
make
make install [15]
```

Die Libnet Bibliothek muss in der Version 1.0.x vorliegen. Snort-Inline kann nicht mit der neueren 1.1.x Version zusammmen arbeiten.

pcre-6.1

Snort-Inline benönigt die `pcre` Bibliothek (Perl-Compatible Regular Expression). Die Datei `pcre-6.1.tar.gz` kann von [ID-PCRE] in das Verzeichnis `/usr/src/` kopiert werden. Nach dem Entpacken der Datei mit dem tar-Befehl wird sie per CMMI installiert.

libpcap-0.9.3

Die libpcap Datei ist eine Paket-Capture-Library, die Pakete analysieren kann. Zur Installation wird die aktuelle Version von [ID-LIBP] in das Verzeichnis `/usr/src/` kopiert

15 Der Block wird im folgenden mit CMMI abgekürzt.

werden und nach dem entpacken mit CMMI installiert.

mysqlclient

Um Snort-Inline mit MySQL Unterstützung zu installieren, werden die `mysqlclientlibrarys` benötigt. Diese sind im Debian Developer-Paketen enthalten .

```
apt-get install libmysqlclient10-dev
```

7.2.3.2 Installation und Konfiguration von Snort-Inline

Die hier verwendete Version ist snort_inline-2.2.0a. Sobald alle Abhängigkeiten vorhanden sind, kann die Datei `snort_inline-2.2.0a.tar.gz` von [ID-SNO-IN] in das Source-Verzeichnis `/usr/src/` kopiert werden. Nach dem Entpacken wird das Konfigurationsskript `./configure` mit den benötigten Optionen gestartet.

```
./configure --prefix=/opt/snort-inline/ ↵
--with-libipq-includes=/usr/include/libipq ↵
--enable-flexresp ↵
--enable-inline ↵
--enable-clamav ↵
--with-mysql
make
make install
```

Mit der ersten Option wird der Pfad angegeben, in dem Snort-Inline installiert wird. Das Verzeichnis `opt` ist bei Debian standardmäßig angelegt und ist für die Installation optionaler Software vorgesehen. Der zweite Punkt gibt den Pfad für die lilipq an und der dritte aktiviert die flexresp-Funktion, die für das Weiterleiten der Pakete verantwortlich ist. Die folgenden drei Optionen binden die gewünschten Funktionalitäten ein.

Zunächst wird ein Unterverzeichnis erstellt, in das die benötigten Konfigurationsdateien und Regelsets kopiert werden.

```
mkdir -p /opt/snort-inline/etc/snort-inline/
cp -R /usr/src/snort_inline-2.2.0a/etc/* ➥
/opt/snort-inline/etc/snort-inline/
cp -R /usr/src/snort_inline-2.2.0a/rules/* ➥
/opt/snort-inline/etc/snort-inline/
```

Da Snort-Inline aus dem Code von Snort entstanden ist, müssen noch einige Anpassungen im Pfad `/opt/snort-inline/etc/snort-inline/` vorgenommen werden. Im Konfigurationsscript `snort_inline.conf` sind folgende Änderungen mittels eines Editors vorzunehmen.

- Um den Regelpfad anzupassen, muss der Regelsatz in der Variablen `RULES_PATH` angegeben werden.

```
var RULE_PATH rules
```

- Der Präprozessor `stream4` erhält Informationen über den Status der Pakete.

```
preprozessor stream4: disable_evasion_alerts,➥
iptablesnewmark,iptablesestmark,forceipstate
```

- Die Pfade der zweier Konfigurationsdateien werden angepasst. Dazu wird die Variable `$RULE_PATH` entfernt.

```
include classification.config
include reference.config
```

- Dem Präprozessor `clamav` wird mitgeteilt, welcher Verkehr zu untersuchen ist und wie er verfahren soll, wenn eine Virensignatur erkannt wird.

```
preprozessor clamav: ports all !22 !433,action-drop
```

Da die Regeln für Snort erstellt wurden und Snort nicht über die Drop-Funktion verfügt, müssen die Regelsets angepasst werden. Hierzu haben die Snort-Inline Entwickter ein Script beigefügt mit dem ALERTs gegen DROPs ausgetauscht werden.

```
cd /opt/snort-inline/etc/snort-inline/
cp convert.sh rules/
cd rules
chmod +rx convert.sh
./convert.sh
```

An dieser Stelle kann Snort-Inline, sofern entsprechende iptables bestehen, getestet werden.

```
cd /opt/snort-inline/bin
./snort_inline -Qvc /opt/snort-inline/etc/snort-inline/
snort_inline.conf 2&>/tmp/test
```

Snort-Inline schreibt seine Startbildschirmausgabe in die Datei `/tmp/test`. Sie ist sehr ausführlich, weshalb sie zum Testen in die Datei `test` umgelenkt werden sollte. In der Datei `test` kann dann bequemer kontrolliert werden, ob Snort-Inline einwandfrei startet. Sollte die Installation noch Fehlerhaft sein, läßt sich das in dieser Datei sehr leicht prüfen.

Eine Fehlermöglichkeit ist die Bibliothek `libpcre.so.0` und ihr Pfad. Snort-Inline sucht diese Bibliothek im Pfad `/usr/local/lib`. Ist sie Datei dort nicht vorhanden, muss ein Softlink auf `/usr/lib/libpcre.so.3` gesetzt werden.

```
ln -S /usr/lib/libpcre.so.3 /usr/local/lib/libpcre.so.0
```

Im Anschluß muss Snort-Inline mitgeteilt werden, wo sich die MySQL-Datenbank befindet.

Hierzu wird die `snort_inline.config` um folgende Einträge ergänzt.

```
### MySQL Datenbank-Ort
output database: log, mysql, user=snort password=russel
dbname=snort host=192.168.1.22
```

Obwohl Snort-Inline nun *weiß,* wo sich die MySQL-Datenbank befindet, kann es vorkommen, dass gefundene Virensignaturen nur im Logfile vermerkt werden. Dies liegt am `clamav` Präprozessor. Wenn dies der Fall ist, muss in der Datei `spp_clamav.c` nach der Zeile

```
CallAlertFuncs(p, outstring, NULL, &event);
```

die Zeile

```
CallLogFuncs(p, outstring, NULL, &event);
```

eingefügt werden. Diese nachträgliche Änderung wird erst nach einem erneuten `make` und `make install` wirksam.

Ist nur das Aufzeichnen in die MySQL-Datenbank erwünscht, müssen die folgenden Zeilen in der `snort-inlne.conf` kommentiert werden:

```
#output alert_full: snort_inline-full
#output alert_fast: snort_inline-fast
#output log_tcpdump: tcpdump.log
```

Ansonsten legt Snort-Inline in `/var/log/snort/` entsprechende Logdateien an, die schnell sehr groß werden können. Sind diese Logfiles dennoch erwünscht, müssen sie gepflegt werden. Dies kann beispielsweise durch den Dämon `syslog.d` erfolgen.

Bevor Snort-Inline zum ersten Mal mit MySQL-Unterstützung gestartet werden kann, muss die MySQL-Datenbank auf dem Wartungsrechner in Betrieb sein. Ist dies nicht der Fall, wird Snort-Inline nicht gestartet, da keine Einträge in die Datenbank geschrieben werden können.

Die Firewall-Installation ist abgeschlossen. Um sie sofort nach dem Neustart verfügbar zu machen, muss das Startscript in den Runlevel 2 eingebunden werden.

```
ln -S /etc/snortfw_start.sh /etc/rc2.d/S92snortfw-start
```

7.3 Installation des Wartungsrechners

7.3.1 MySQL

Die MySQL-Datenbank ist als Debian-Paket verfügbar.

```
apt-get install mysql-server
apt-get install libsqlplus-dev
```

Das Paket `mysql-server` enthält den Client, die Datenbank und den Server. Einige Developer-Bibliotheken sind im `libsqlplus-dev` Paket enthalten.

Die Installation ist sofort benutzbar, besitzt jedoch noch kein Passwort. Nach Vergabe eines Passwortes wird eine Datenbank mit dem Namen snort erstellt. Auf diese Datenbank erhält der Benutzer snort@192.168.1.21 Zugriffsrechte[16]. Zunächst wird die MySQL-Konsole gestartet. Eine MySQL-Eingabe kann über mehrere Zeilen durch *Return* getrennt werden und ist erst mit dem Semikolon abgeschlossen. MySQL beantwortet jede Eingabe mit der Ausgabe eines OK und der Zeit die für die Bearbeitung benötigt wurde. Im folgenden ist diese Ausgabe nur nach der ersten Anweisung aufgeführt.

16 MySQL Grundlagen, Zugriffsrechte und Steuerbefehle werden in [WID-MYS] erklärt.

```
mysql
mysql> set password root@localhost=password('geheimwort');
>Query OK, 0 rows affected (0,05 sec)
mysql> create database snort;
>Query OK, 0 rows ...
mysql> grant INSERT,SELECT on root.* to
snort@192.168.1.21;
mysql> set password for snort@192.168.1.21=
password('russel');
mysql> grant CREATE,INSERT,SELECT;DELETE,UPDATE on
snort.* to snort@192.168.1.21;
mysql> grant CREATE,INSERT,SELECT;DELETE,UPDATE on
snort.* to snort;
mysql>exit
>bye
```

Die Datenbank-Tabellen, die Snort-Inline benötigt, kann das Script `create_mysql` erstellen. Es befindet sich in `/usr/src/snort_inline2.2.0a` auf der Firewall und kann per SSH auf den Wartungsrechner kopiert werden. Anschließend wird es ausgeführt.

```
mysql -u root -p < /home/create_mysql
```

Um sicherzugehen, dass die Snort-Datenbank durch das Skript erstellt wurde, kann sie in der MySQL-Konsole kontrolliert werden.

```
mysql> use snort;
mysql> show tables;
```

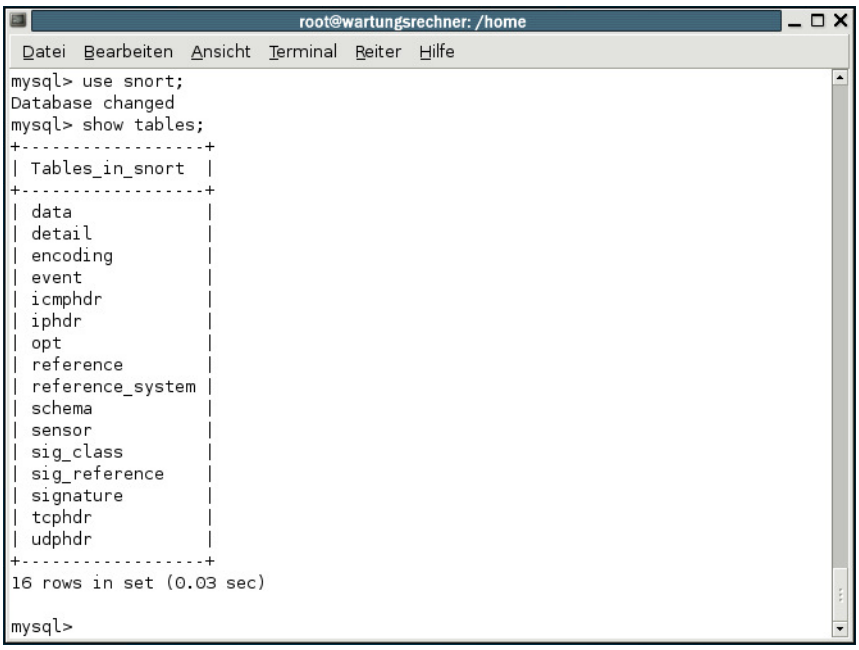


Abbildung 19: Snort Datenbankaufbau

7.3.2 Apache

Der Apache-Server mit den benötigten Unterstützungen steht als Debian-Softwarepaket zur Verfügung.

```
apt-get install apache
apt-get install libapache-mod-php4
```

Nach der Installation wird der Apache-Webserver unter Debian automatisch gestartet. Daher sollte nach der Installation des Paketes `libapache-mod-php4` der Apache gestoppt und neu gestartet werden.

```
apache stop
apache start
```

Anschließend werden die benötigten Unterstützungen für PHP4 installiert.

```
apt-get install php4-mysql php4-gd php4-pear libphp-adodb
```

Apt trägt die entsprechenden Extensions in die Datei `php.ini` ein. Sind diese nicht vorhanden, müssen sie per Hand hinzugefügt werden.

```
extension=mysql.so
extension=gd.so
```

Nach dieser Veränderung wird der Apache-Server wieder neu gestartet, damit alle Änderungen wirksam werden [IL-APA].

7.3.3 BASE

Um das graphische Frontend BASE zu installieren, müssen zunächst einige Einstellungen an den PHP-Bibliotheken vorgenommen werden, damit BASE vom Browser einwandfrei dargestellt werden kann. Hierzu wird das PHP-Instalationsprogramm PEAR verwendet, das im vorangegangen Kapitel mitinstalliert wurde.

```
pear config-set http-proxy www.eigenerProxyserver.de:Port
pear install Image_Color
pear install Number_Roman
pear install http://pear.php.net/get/
Numbers_Words-0.13.1.tgz
pear install http://pear.php.net/get/
Image_Graph-0.3.0dev4.tgz
```

BASE ist nicht als Debian-Paket erhältlich und muss daher aus den Sourcen kompiliert werden. Dazu wird die Datei `base-1.1.4.tar.gz` in das Verzeichnis `/usr/src` kopiert [ID-BASE]. Nach dem Auspacken müssen die BASE-Dateien in ein Verzeichnis gelegt werden, auf das der Apache-Webserver zugreifen kann.

```
cp -r /usr/src/base-1.1.4 /var/www/base
```

Anschließend werden in der BASE-Konfigurationsdatei `base.conf.php` mittels eines Editors einige Anpassungen vorgenommen. Zunächst läßt sich die verwendete Sprache auf Deutsch einstellen. Der URL-Pfad ist ein relativer Pfad innerhalb des Apache-Verzeichnisses. Die Abstraktionsbibliothek `ADOdb` ermöglicht es BASE unabhängig von der Datenbankform auf diese zuzugreifen. Der Datenbanktyp MySQL wird BASE dennoch mitgeteilt. Zuletzt erhält BASE die Zugangsdaten zur Snort-Inline Datenbank (Name, Ort, User und Passwort).

```
$BASE_language = "german";
...
$BASE_urlpath= "/base";
...
$DBlib_path = "usr/share/adodb";
...
$DBtype = "mysql";
...
$alert_dbname = "snort";
$alert_host = "localhost";
$alert_port = "";
$alert_user = "snort";
$alert_password = "russel";
```

BASE ist nun installiert und kann in einem beliebigen Browser in der Adresszeile mit

```
http://localhost/base
```

aufgerufen werden.

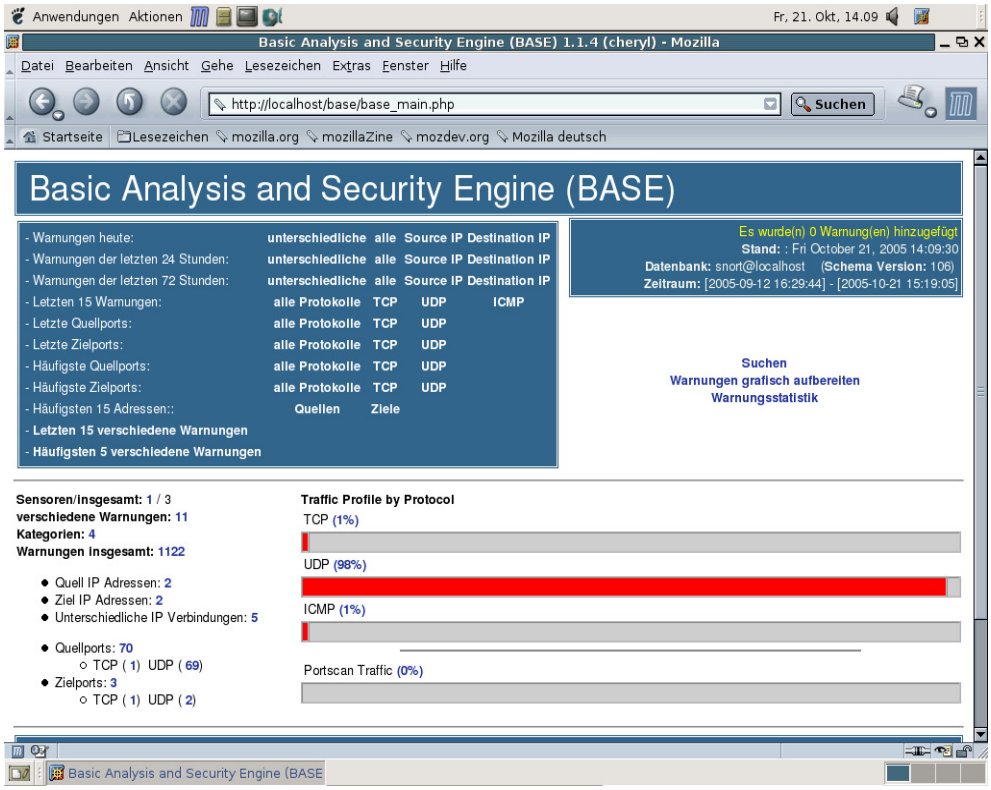


Abbildung 20: BASE Startbildschirm

Nach dem ersten Aufruf wird BASE eine Meldung anzeigen, dass der Datenbankaufbau nicht konform ist. Allerdings bietet BASE an, dies zu beheben. Dieser Dialog muss mit „Ja" beantwortet werden, denn erst nach dieser Anpassung ist BASE in der Lage, die Datenbank zu verwenden.

Sobald Snort-Inline gestartet wird und in die MySQL Datenbank geloggt hat, lassen sich die Einträge nach beliebigen Kriterien durchsuchen oder nach gewünschten Strukturen geordnet anzeigen. Die Gestaltung ist sehr intuitiv und selbsterklärend.

7.4 Installation des SIP/RTP Proxys

7.4.1 RTP Proxy

Der Portaone RTPproxy liegt nicht als Debian-Softwarepaket vor. Nach dem Kopieren in das Verzeichnis `/usr/src` kann die Datei `rtpproxy.tar.gz` entpackt werden. Anschließend ist sie per CMMI zu installieren. Es müssen keine Konfigurationen vorgenommen werden.

Achtung: Die Version `rtpproxy.tar`, die bei berlios.de herunterladbar ist, lässt sich nicht verwenden. Diese Datei ist beschädigt oder veraltet.

7.4.2 SIP Express Router

Der SIP-Express Router wird von [ID-SER] in das Verzeichnis `/usr/src` kopiert. Hier wurde der SER in der Version 0.8.14 verwendet. Nach dem Entpacken der Datei `ser-0.8.14_src.tar.gz` wird der SER mit CMMI in `/usr/local/etc/ser` installiert.

Die SER-0.8.14 ist die derzeit stabile Version und auch als Debian-Paket unter [ID-SER-D] erhältlich. Dieses Paket muss dann mit `dpkg -i <Paketname>` installiert werden, da es nicht über die APT-Ressourcen zur Verfügung steht.

Um die Proxyfunktionen und die RTP-Proxyanbindung zu erhalten, sind Veränderungen in der SIP Express Router Konfigurationsdatei `ser.cfg` nötig. Da das Standartskript auf einen Registrar ausgelegt ist, bietet es sich an, das Skript komplett auszutauschen, zumal sehr viele Konfigurationszeilen einfach wegfallen oder ersetzt werden. Das vollständige Konfigurationsskript befindet sich in Textform im Anhang und in Datenform auf der CD. Die wichtigsten Änderungen, die für die Proxyfunktion nötig sind, werden im folgenden beschrieben.

Zunächst werden die benötigten Module geladen.

```
...
loadmodule "/usr/local/lib/ser/modules/sl.so"
loadmodule "/usr/local/lib/ser/modules/tm.so"
loadmodule "/usr/local/lib/ser/modules/rr.so"
loadmodule "/usr/local/lib/ser/modules/maxfwd.so"
loadmodule "/usr/local/lib/ser/modules/nathelper.so"
loadmodule "/usr/local/lib/ser/modules/textops.so"
...
```

Achtung: Das Modul `registrar.so` darf nicht geladen werden, da es sonst möglich ist, dass sich ein Nutzer ohne irgendeine Art der Authentifizierung auf diesem Proxy registrieren kann.

Dem Modul `nathelper` wird der Pfad des RTP-Proxy-Sockets mitgeteilt, da dieses Modul für die Kommunikation zwischen RTP-Proxy ind SER zuständig ist.

```
...
modparam("nathelper", "rtpproxy_sock",↩
"/var/run/rtpproxy.sock")
...
```

Mit dem Alias wird dem SER seine Identität zugewiesen.

```
...
alias="192.168.1.18"
...
```

Der folgende Konfigurationsdateiausschnitt sorgt dafür, dass SIP-Pakete, die von außerhalb kommen, an den internen SER-Registrar weitergeleitet werden. Hierbei achtet der SER nicht auf die Zieladresse des UDP-Headers, denn im UDP-Header wurde durch die Umleitung des PREROUTINGs der Firewall die Adresse des SIP-Proxyservers als Zieladresse

eingetragen. Das Paket war ursprünglich an den Registrar im DN-Netz gerichtet, weshalb die Zieladresse, die im SDP-Anteil vermerkt ist, vom SER weiterverarbeitet wird.
Ist die Zieladresse des Registrars im SDP-Header enthalten, wird bei einem INVITE der folgende RTP-Verkehr an den Proxy-Server gebunden und muss den „Umweg" über die DMZ gehen. Bei einem BYE-Paket wird die RTP-Bindung wieder freigegeben.

```
if (dst_ip==192.168.1.22) {
     if (method == "INVITE") {
               if (force_rtp_proxy("FAII")){
                    t_on_reply("1");
          } else {
               sl_send_reply("403", "User nicht erreichbar
oder unbekannt!");
               break;
          };
     };
     if (method == "BYE" || method == "CANCEL")
          unforce_rtp_proxy();
forward("192.168.1.22");
break;
};
```

Für aus dem DN-Netz kommende SIP-Pakete gilt die gleiche RTP-Bindung an die DMZ, jedoch findet hier keine Überprüfung des Zieles statt. Das im SDP-Header eingetragene Ziel wird in den UDP-Header übernommen.

```
if (method == "INVITE") {
          if (force_rtp_proxy("FAII")){
                    t_on_reply("1");
               } else {
               sl_send_reply("403", "User nicht erreichbar
oder unbekannt!");
```

```
                break;
            };
        };
```

7.4.3 SIP/RTP Proxy Startskript

Nach dem Start des Rechners soll der SIP/RTP-Proxy sofort zur Verfügung stehen. Dazu wird auf das Startscript `sip-start.sh` im Runlevel 2 ein Softlink gesetzt.

```
ln -S /etc/init.d/sip-start.sh /etc/rc2.d/S92sip-start
```

Das Skript stoppt zunächst den SER und den RTP-Proxy. Die vom SER benötigte SIP Domain wird mit `export` global verfügbar gemacht. Im Anschluß wird der RTP-Proxy gestartet. Dieser muss vor dem SIP-Proxy gestartet werden, da der RTP-Proxy in der Konfigurationsdatei des SER angegeben wurde. Erst danach wird der SER gestartet.

```
#!/bin/bash
#Startskript fuer rtpproxy und SER - sip-start.sh
echo "DER PROXY WIRD JETZT GESTARTET"
echo "------------------------------"
echo " Stoppen des SER zum start der ben. Komponenten"
killall ser
killall rtpproxy
export SIP_DOMAIN="PROXY.dn.fh-koeln.de"
echo "Starten des rtpproxy"
/usr/local/bin/rtpproxy -l 192.168.1.18/192.168.1.18
sleep 2
echo "Neustart des SER"
/usr/local/sbin/serctl start
echo "SIP/RTP Proxy ist nun Bereit"
```

8 Auswertung

Die Durchführung verschiedener Tests, die Testergebnisse und deren Auswertung soll ebenfalls an Versuchsaufbauten erläutert werden.

8.1 Nessus

Nessus ist ein unter der GPL veröffentlichter[17] Netzwerkscanner, der in der Lage ist, einen Großteil der bekannten verwundbaren Punkte eines Netzwerkes aufzuzeigen. Es werden verschiedene vorbereitende Maßnahmen auf typische Angriffsmuster ausprobiert, und deren Ergebniss angezeigt [IL-NESS].

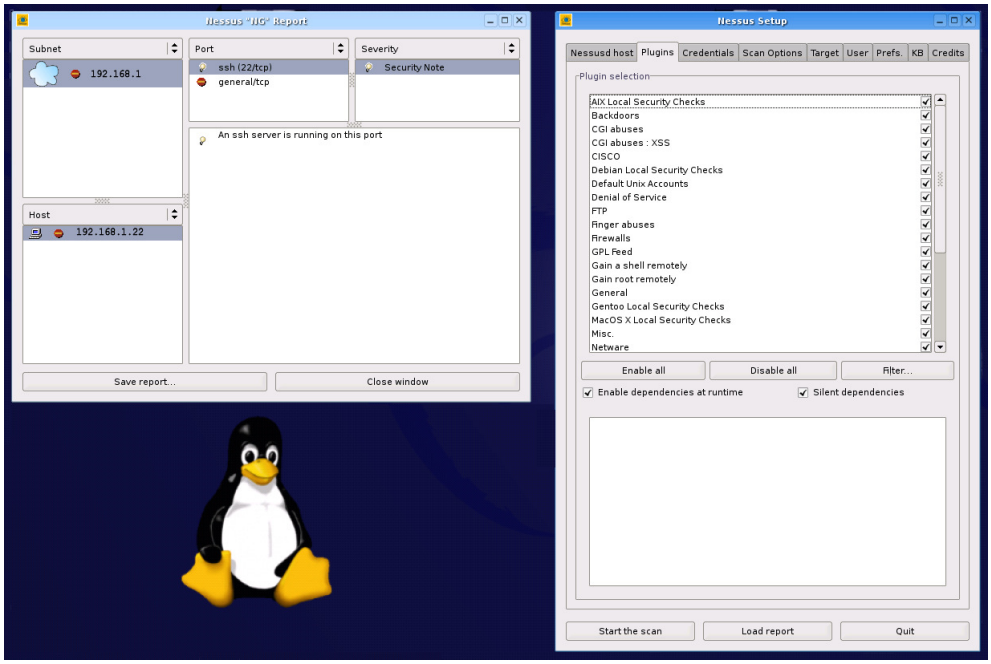


Abbildung 21: Nessus mit Scan Report

Mit Nessus wurde ein Netzwerkscan auf die Adresse 192.168.1.22 im DN-Netz vom Testrechner mit der Adresse 192.168.1.142 über die Firewall durchgeführt. Der Scan Report ergab, dass der Wartungsrechner über SSH ansprechbar ist. Dies ist jedoch nur aus dem NT-Netz möglich. Weiterhin wurde ein NIDS im Netzwerk erkannt, welches Falsch-positiv Alarme auslösen könnte. Beides war jedoch bekannt. Der ausführliche Nessus Scan Report befindet sich im Anhang.

17 Ab Version 3.0 soll Nessus nur noch unter proprietärer Lizenz weiterentwickelt werden.

8.2 Virentransfer

Der Virentransfer wurde an einer Insellösung getestet, um auszuschließen, dass sich Viren verbreiten können. Es wurde versucht, während einer FTP-Sitzung Viren vom Rechner *Virenschleuder* auf den Wartungsrechner zu übertragen.

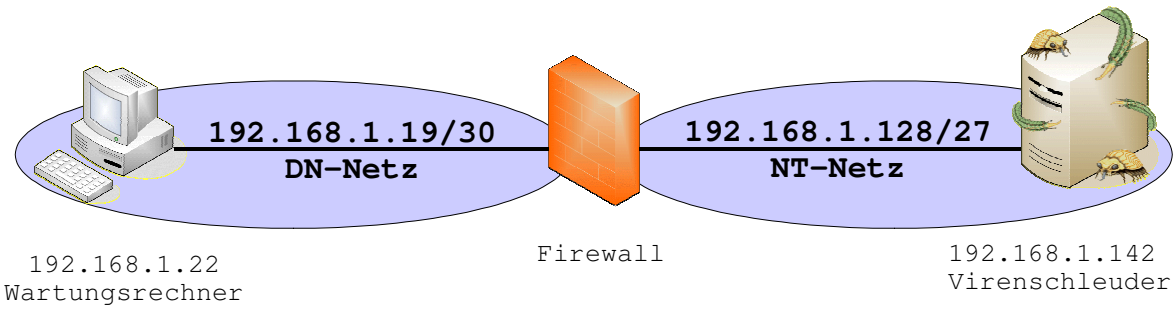


Abbildung 22: Testaufbau Insellösung

8.3 Auswertung mit BASE

8.3.1 Allgemeine Hinweise zur Auswertung

Sobald BASE aufgerufen wird, informiert Base darüber, ob neue Warnmeldungen vorliegen. Die Einträge in der Datenbank können komfortabel gesichtet werden, wobei die Sortierkriterien vielfältig einstellbar sind. Zu jeder Warnmeldung wird die SID-Nummer angegeben, die die Meldung ausgelöst hat. Darüberhinaus gibt es einen direkten Link, der zum entsprechenden snort.org Signaturdatenbank Eintrag führt. Hier ist aufgeführt, warum die Warnmeldung entstanden ist. Es finden sich detailierte Informationen zu Auswirkungen, betroffenen Systemen, bekannten Falsch-positiv und Falsch-negativ Meldungsszenarios und Verweise auf Nessus Refernzen. Weiterhin werden Vorschläge unterbreitet, wie das Problem behoben werden kann oder ob überhaupt etwas unternommen werden muss.

Ein Großteil der Warnmeldungen kann über frei skalierbare Parameter wie Zeiträume, Netzbereiche oder Protokollarten graphisch dargestellt werden, was eine Kontolle der Meldungsarten übersichtlicher gestaltet. So kann beispielweise die Meldungshäufigkeit eines Vorfalls oder die Quelle des Verursachers besser beurteilt werden.

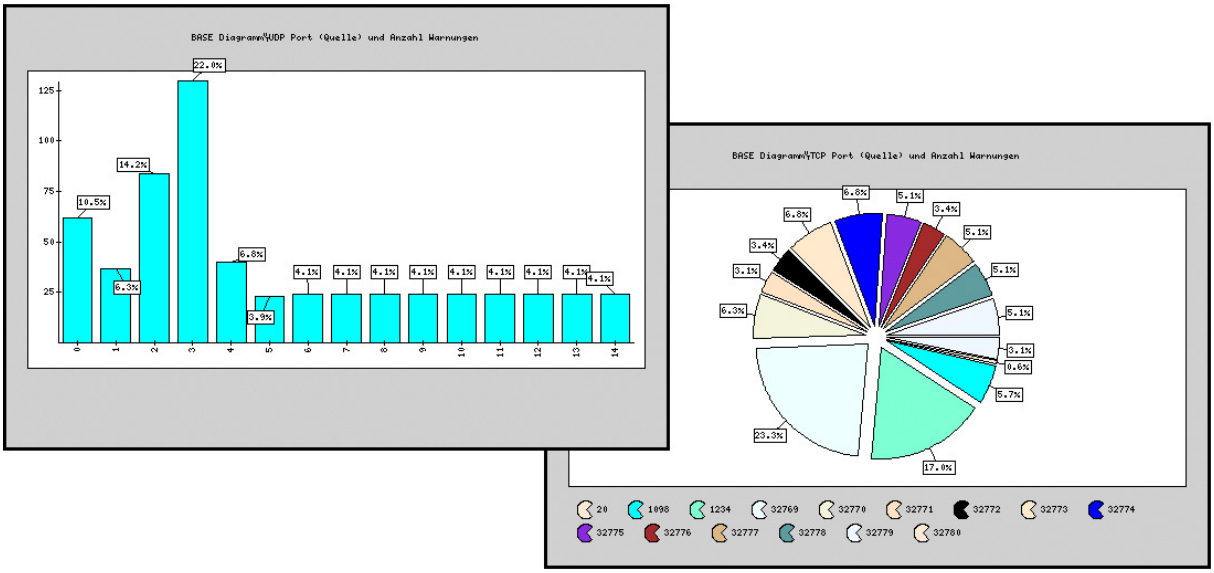


Abbildung 23: Beispiele einer graphischen Auswertung.

Aufgrund des beträchtlichen Umfanges der Auswertungsmöglichkeiten mit BASE, sollen hier nur einige exemplarische Warnmeldungen aufgeführt werden.

8.3.2 Auswertung des Nessus Scans

Der Nessus-Scan verursachte weit über 400 Meldungen, die sich zu 90% auf die UDP-Ports 161-162 beziehen. Da das SNMP-Protokoll nur vom NT-Netz zugelassen ist, und der Nessus Scan aus diesem Netz initiiert wurde, wird die Gefahr hier als gering eingestuft. Ein SNMP-Zugriff von außerhalb ist nicht möglich.

Weiterhin konnten durch Snort-Inline einige DoS und DDoS Angriffe abgewehrt werden. So wurde beispielsweise der DoS ath (SID 274), der bei Modems einen Verbindungsabbruch auslösen kann, und eine DDoS TFN Sondierrung (SID 221), die einen DDoS einleiten soll, abgefangen. Die Signaturdatenbank gibt Hinweise darauf wie verfahren werden sollte, wenn die Quelladressen dieser Angriffe im eigenen Netzwerk liegen. Da dies nicht der Fall ist, wurde also ein Angriff von außen erfolgreich abgewehrt.

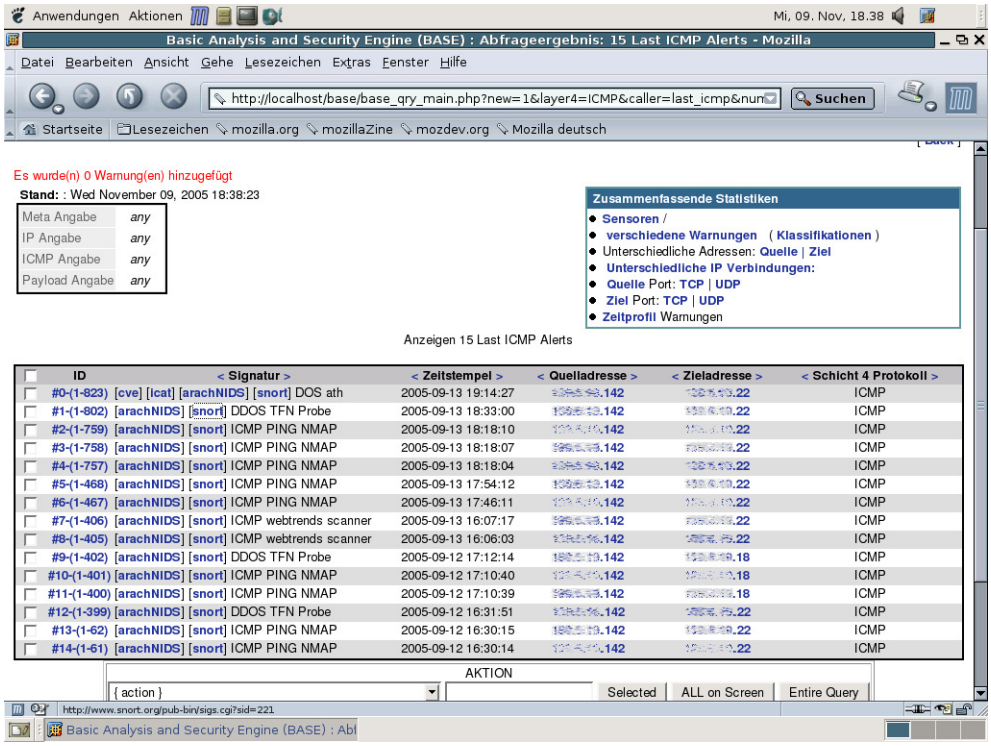


Abbildung 24: Übersicht über die ICMP Warnmeldungen

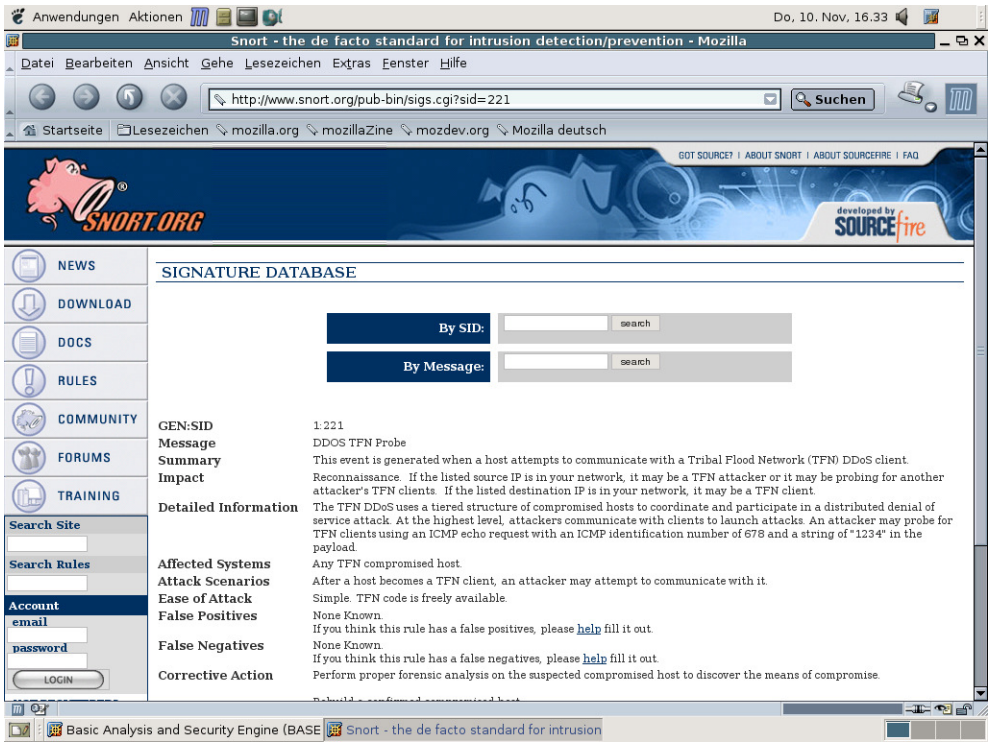
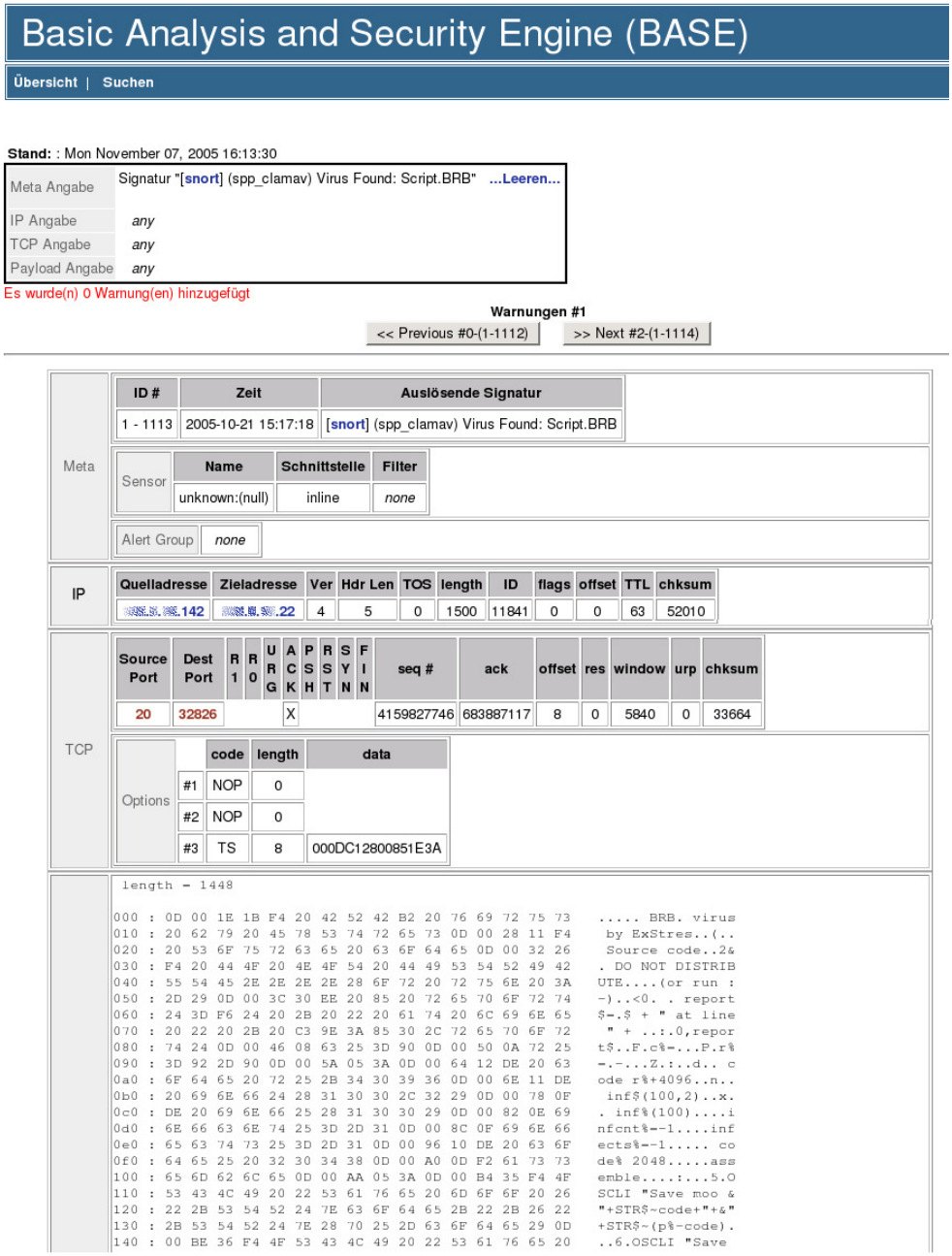


Abbildung 25: Snort Signaturdatenbank (SID 221)

8.3.3 Auswertung des Virentransfers

Beim Versuch, Viren bei einer FTP-Sitzung auf den Wartungsrechner zu übertragen, wurde die Verbindung von Snort-Inline abgebrochen. Ein passender Eintrag fand sich mittels BASE in der Datenbank.



Basic Analysis and Security Engine (BASE)

Übersicht | Suchen

Stand: : Mon November 07, 2005 16:13:30

Meta Angabe	Signatur "[snort] (spp_clamav) Virus Found: Script.BRB" ...Leeren...
IP Angabe	*any*
TCP Angabe	*any*
Payload Angabe	*any*

Es wurde(n) 0 Warnung(en) hinzugefügt

Warnungen #1

<< Previous #0-(1-1112) >> Next #2-(1-1114)

Meta

ID #	Zeit	Auslösende Signatur
1 - 1113	2005-10-21 15:17:18	[snort] (spp_clamav) Virus Found: Script.BRB

Sensor	Name	Schnittstelle	Filter
	unknown:(null)	inline	*none*

Alert Group	*none*

IP

Quelladresse	Zieladresse	Ver	Hdr Len	TOS	length	ID	flags	offset	TTL	chksum
[redacted].142	[redacted].22	4	5	0	1500	11841	0	0	63	52010

TCP

Source Port	Dest Port	R 1	R 0	URG	ACK	PSH	RST	SYN	FIN	seq #	ack	offset	res	window	urp	chksum
20	32826				X					4159827746	683887117	8	0	5840	0	33664

Options		code	length	data
	#1	NOP	0	
	#2	NOP	0	
	#3	TS	8	000DC12800851E3A

```
length = 1448

000 : 0D 00 1E 1B F4 20 42 52 42 B2 20 76 69 72 75 73   ..... BRB. virus
010 : 20 62 79 20 45 78 53 74 72 65 73 0D 00 28 11 F4    by ExStres..(..
020 : 20 53 6F 75 72 63 65 20 63 6F 64 65 0D 00 32 26    Source code..2&
030 : F4 20 44 4F 20 4E 4F 54 20 44 49 53 54 52 49 42   . DO NOT DISTRIB
040 : 55 54 45 2E 2E 2E 28 6F 72 20 72 75 6E 20 3A   UTE....(or run :
050 : 2D 29 0D 00 3C 30 EE 20 85 20 72 65 70 6F 72 74   -)..<0. . report
060 : 24 3D F6 24 20 2B 20 22 20 61 74 20 6C 69 6E 65   $=.$ + " at line
070 : 20 22 20 2B 20 C3 9E 3A 85 30 2C 72 65 70 6F 72    " + ..:.0,repor
080 : 74 24 0D 00 46 08 63 25 3D 90 0D 00 50 0A 72 25   t$..F.c%=...P.r%
090 : 3D 92 2D 90 0D 00 5A 05 3A 0D 00 64 12 DE 20 63   =.-...Z.:..d.. c
0a0 : 6F 64 65 20 72 25 2B 34 30 39 36 0D 00 6E 11 DE   ode r%+4096..n..
0b0 : 20 69 6E 66 24 28 31 30 30 2C 32 29 0D 00 78 0F    inf$(100,2)..x.
0c0 : DE 20 69 6E 66 25 28 31 30 30 29 0D 00 82 0E 69   . inf%(100)....i
0d0 : 6E 66 63 6E 74 25 3D 2D 31 0D 00 8C 0F 69 6E 66   nfcnt%=-1....inf
0e0 : 65 63 74 73 25 3D 2D 31 0D 00 96 10 DE 20 63 6F   ects%=-1..... co
0f0 : 64 65 25 20 32 30 34 38 0D 00 A0 0D F2 61 73 73   de% 2048.....ass
100 : 65 6D 62 6C 65 0D 00 AA 05 3A 0D 00 B4 35 F4 4F   emble....:...5.O
110 : 53 43 4C 49 20 22 53 61 76 65 20 6D 6F 6F 20 26   SCLI "Save moo &
120 : 22 2B 53 54 52 24 7E 63 6F 64 65 2B 22 2B 26 22   "+STR$~code+"+&"
130 : 2B 53 54 52 24 7E 28 70 25 2D 63 6F 64 65 29 0D   +STR$~(p%-code).
140 : 00 BE 36 F4 4F 53 43 4C 49 20 22 53 61 76 65 20   ..6.OSCLI "Save
```

Abbildung 26: Erkannter Virus "Script.BRB"

Der Virus „Script.BRB“ aus dem Beispielbild wurde nicht auf den Wartungsrechner übertragen. Leider gibt es hierzu keine Direktlinks, um festzustellen, wie der Virus arbeitet oder welche Ressourcen von ihm betroffen sind. Mit dem Namen kann jedoch in der ClamAV Online-Virendatenbank [IL-CLAMDB] nach dem Virus gesucht werden.

Achtung: Snort-Inline ist mit dem Präprozessor `clamav` nicht in der Lage, Viren in gepackten Dateien zu erkennen.

8.4 VoIP-Testgespräche

Um verschiedene Szenarios für die Proxyfunktionen auszutesten, wurde der Versuchsaufbau erweitert. Dazu wurde auf dem Wartungsrechner ein zusätzlicher SER-Registrar und Softphone installiert. Diese Umgebung repräsentiert das DN-Netz mit dem Teilnehmer *holleinnen,* der am SER-Registrar registriert ist.

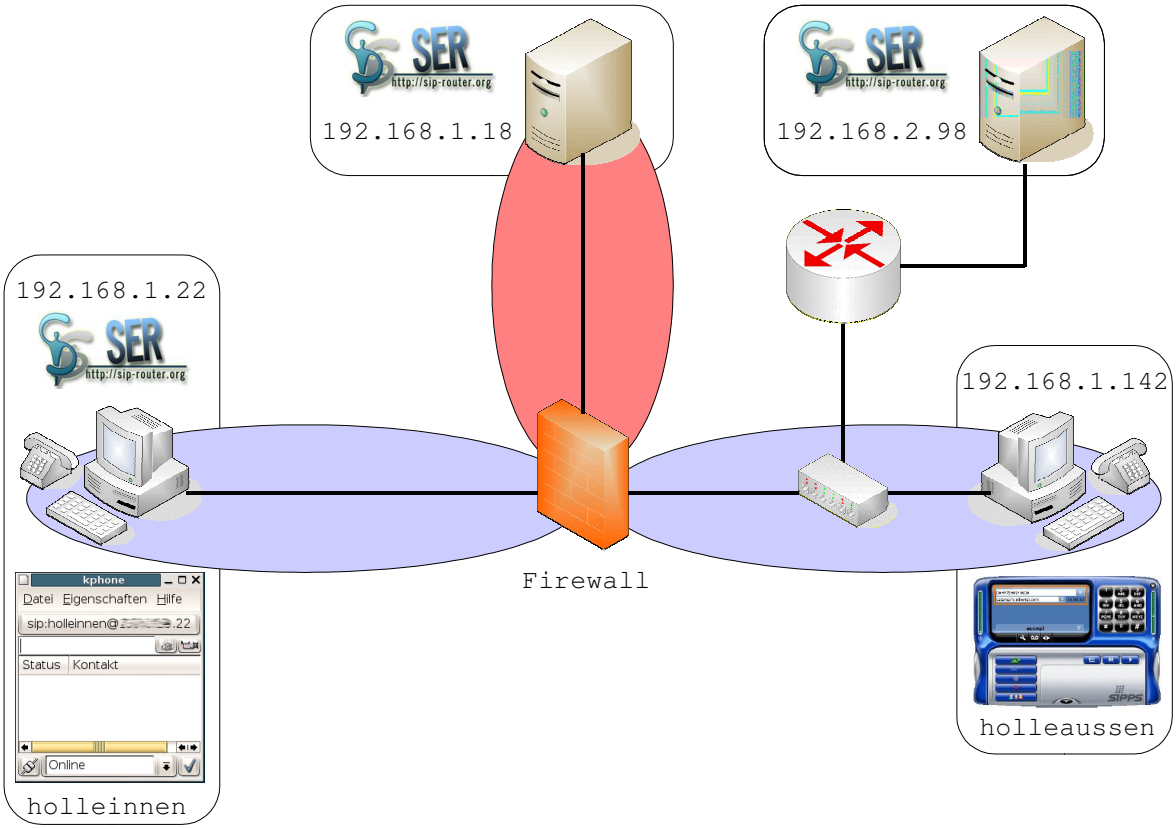


Abbildung 27: VoIP Testaufbau zum Test der Proxyfunktionen

Weiterhin wurde unter der IP-Adresse 192.168.2.98 ein externer SER-Registrar-Server eingerichtet. Auf dem Testrechner wurde unter dem Betriebssystem Windows 2000 das SIPPS[18] Softphone installiert und der Teilnehmer *holleaussen* angelegt. Dieser kann sich sowohl am Registrar-Server im DN-Netz als auch am externen Registrar-Server anmelden. Weiterhin wurden einige Einstellungen am internen SER-Registrar vorgenommen. Die `ser.cfg` Konfigurationsdatei befindet sich im Anhang.

Auf eine Authentifizierung wurde bei beiden Registrar-Servern verzichtet, da in der Testumgebung geprüft werden soll, in welchen Bahnen der SIP- und RTP-Verkehr kanalisiert wird. Bei einer Authentifizierung wird das verschlüsselte Passwort im Header des SIP-Paketes übertragen, weshalb es für die Testumgebung keinen Unterschied macht, ob eine Authentifizierung am Server erfolgt oder nicht.

Um die Proxyfunktionen zu verdeutlichen, wurde mit `tethereal` der Netzverkehr auf dem SIP/RTP-Proxy gesnifft.

Als Beispiel soll der Teilnehmer *holleaussen* am externen SIP-Registrar angemeldet sein und erhält die URI sip:holleaussen@192.168.2.98. Möchte er nun den Teilnehmer *holleinnen* anrufen, so sendet er ein INVITE an den Registrar 192.168.1.22. Diese Paket wird jedoch durch die Firewall umgeleitet und erreicht den SIP-Proxy auf 192.168.1.18.

```
  0.000000 192.168.1.142 -> 192.168.1.18  SIP/SDP Request: INVITE
sip:holleinnen@192.168.1.22, with session description
```

Das Endgerät wendet sich mit dem INVITE direkt an 192.168.1.22, da das Ziel bekannt ist. Wäre die URI beispielsweise sip:holleinnen@dn-netz.de, müsste sich das Endgerät zuerst an seinen eigenen Registrar wenden.

Der SIP-Proxy beantwortet das INVITE mit einem TRYING und leitet das INVITE an 192.168.1.22 weiter.

18 SIPPS wird von der Firma Ahead in einer 30 Tage Trial Version angeboten.

```
  0.003834  192.168.1.18 -> 192.168.1.142 SIP Status: 100 trying --
your call is important to us
  0.003950  192.168.1.18 -> 192.168.1.22  SIP/SDP Request: INVITE
sip:holleinnen@192.168.1.22, with session description
```

Der SER-Registrar im DN-Netz sendet dann das RINGING und das OK zurück, das er vom Softphone erhalten hat. Das SIPPS Softphon antwortet wiedrum mit einem ACK.

```
  0.045666  192.168.1.22 -> 192.168.1.18  SIP Status: 180 Ringing
  0.046021  192.168.1.18 -> 192.168.1.142 SIP Status: 180 Ringing
 11.011611  192.168.1.22 -> 192.168.1.18  SIP/SDP Status: 200 OK, with
session description
 11.014362  192.168.1.18 -> 192.168.1.142 SIP/SDP Status: 200 OK, with
session description
 11.022835 192.168.1.142 -> 192.168.1.18  SIP Request: ACK
sip:holleinnen@192.168.1.22:5062;transport=udp
 11.023258  192.168.1.18 -> 192.168.1.22  SIP Request: ACK
sip:holleinnen@192.168.1.22:5062;transport=udp
```

Es folgt der RTP Datenaustausch, der durch den RTP-Proxy an die DMZ gebunden wird. Der folgende Ausschnitt zeigt anhand der Sequenznummern, dass ein eingehendes RTP-Paket weitergeleitet wird.

```
 11.200715 192.168.1.142 -> 192.168.1.18  RTP Payload type=ITU-T G.711
PCMU, SSRC=720603238, Seq=13564, Time=659145
 11.200822  192.168.1.18 -> 192.168.1.22  RTP Payload type=ITU-T G.711
PCMU, SSRC=720603238, Seq=13564, Time=659145
 11.205132  192.168.1.22 -> 192.168.1.18  RTP Payload type=ITU-T G.711
PCMU, SSRC=167772160, Seq=8, Time=2038110643
 11.205240  192.168.1.18 -> 192.168.1.142 RTP Payload type=ITU-T G.711
PCMU, SSRC=167772160, Seq=8, Time=2038110643
```

Das Gespräch endet mit BYE und OK.

```
 13.873404  192.168.1.22 -> 192.168.1.18  SIP Request: BYE
sip:dnprak@192.168.1.142
```

```
 13.874263  192.168.1.18 -> 192.168.1.142 SIP Request: BYE
sip:dnprak@192.168.1.142
 14.612736 192.168.1.142 -> 192.168.1.18  SIP Status: 200 OK
 14.613133  192.168.1.18 -> 192.168.1.22  SIP Status: 200 OK
```

Die folgende Tabelle zeigt die ausgetesteten möglichen Scenarios.

holleaussen@ 192.168.2.98	☎⇨	holleinnen@ 192.168.1.22	✓
holleinnen@ 192.168.1.22	☎⇨	holleaussen@ 192.168.2.98	✓
holleaussen@ 192.168.1.22	☎⇨	holleinnen@ 192.168.1.22	✓
holleinnen@ 192.168.1.22	☎⇨	holleaussen@ 192.168.1.22	✓

Die Auswertung der tethereal-Mitschnitte zeigt, dass die gewünschte Funktionalität erreicht wurde. INVITE SIP-Nachrichten wurden ebenso vom SIP-Proxy weitergeleitet wie REGISTER SIP-Nachrichten. Somit kann ein Teilnehmer, der sich nicht an einem Rechner im DN-Netz befindet, jedoch registrierungsberechtigt ist, den SIP-Registrar im DN-Netz für VoIP-Anwendungen nutzen.

9 Wartung - How to keep your system alive

Um die Sicherheitskomponenten betriebsbereit und aktuell zu halten, ist die installierte Software auf dem neuesten Stand zu halten. Damit ist nicht unbedingt gemeint, immer die neueste Softwareversion zu installieren, sondern vielmehr regelmäßig Sicherheitspatches und Updates einzuspielen. Eine Ausnahme bildet ClamAV (siehe 9.2).

Um die einzelnen Sicherheitskomponenten im Fehlerfall, der beispielsweise durch einen Hardwaredefekt ausgelöst wurde, schnell wieder in Betrieb nehmen zu können, muss ein Datensicherungskonzept bestehen. Hierbei bietet es sich an, sofort nach der Installation ein Image des lauffähigen Systems zu erstellen. Im DN-Experimentiernetz existiert ein Image-Server, der dafür genutzt werden kann. Auch das Image ist in regelmäßigen Abständen auf dem aktuellen Stand zu halten, um es im Fehlerfall mit möglichst geringen Anpassungen wieder in Betrieb zu nehmen.

Da die Funktion des Sicherheitsbeauftragten im DN-Netz durchaus von einem Team ausgeübt werden kann, ist es wichtig, sämtliche Veränderungen ausführlich und zentral zu dokumentieren. Sollte die Person, die eine authorisierte und Security-Policy-konforme Veränderung vorgenommen hat, nicht verfügbar sein, so ist es dem Rest des Teams zumindest mit Hilfe der Aufzeichnungen möglich, die Veränderungen nachzuvollziehen und den Betrieb des Netzwerkes aufrecht zuerhalten. Aber auch für den Fall des Wechsels eines Sicherheitbeauftragten ist eine Dokumentation der ausgeführten Veränderungen unverzichtbar.

9.1 Debian

Wenn eine Sicherheitslücke des Debianbetriebssystems oder einer Anwendung bekannt wird, reagieren die Debian-Entwickler in der Regel sehr schnell und stellen entsprechende Debianpakete zur Verfügung. Die Sichheitsupdates können bei Debian jedoch nur nach einer Aktualisierung des APT-Paket-Systems durchgeführt werden. Dazu muss in der Quellendatei `/etc/apt/sources.list` die Sicherheitsupdatequelle hinzugefügt werden.

```
deb http://www.debian.org/debian-security ↩
stable/updates main contrib non-free
```

Wird nun Software über das apt-Paketsystem mit `update` ausgeführt, prüft Debian alle Quellen auf aktualisierungen. Sicherheitspatches können dann mit dem Kommando `upgrade` installiert werden.

```
apt-get update
apt-get upgrade
```

Das erste Kommando aktualisiert zunächst die Paketliste, wohingegen das zweite die Aktualisierung durchführt.

9.2 ClamAV

Der Dämon freshclam sorgt dafür, dass die Virendatenbank von ClamAV täglich mehrmals geprüft und aktuell gehalten wird. ClamAV selbst ist jedoch zu aktualisieren, wenn ein neuer Release erscheint. Dies kann ebenfalls durch die Debian-Software-Pakete geschehen. Allerdings sind die neuesten ClamAV-Debianpakete nicht in den Debian Standard-Quellen verfügbar, denn die ClamAV Softwarepakete werden vom Debian Volatile Project gepflegt. In der Datei `/etc/apt/sources.list` ist daher die folgende Zeile hinzuzufügen, um über diese Quelle verfügen zu können.

```
deb http://ftp2.de.debian.org/debian-volatile ↩
sarge/volatile main
```

Mit den folgenden Kommandos werden die Paketlisten aktualisiert und alle verfügbaren ClamAV-Pakete gesucht.

```
apt-get update
apt-cache search clamav
```

9.3 Snort-Inline

Um auf neue Bedrohungen zu reagieren, sollten die Snort-Regel-Datenbanken im Internet regelmäßig auf neue Regeln geprüft werdendie man unter snort.org [ID-SNO] findet. Um die Regel herunterzuladen, benötigt man eine Registrierung bei snort.org.

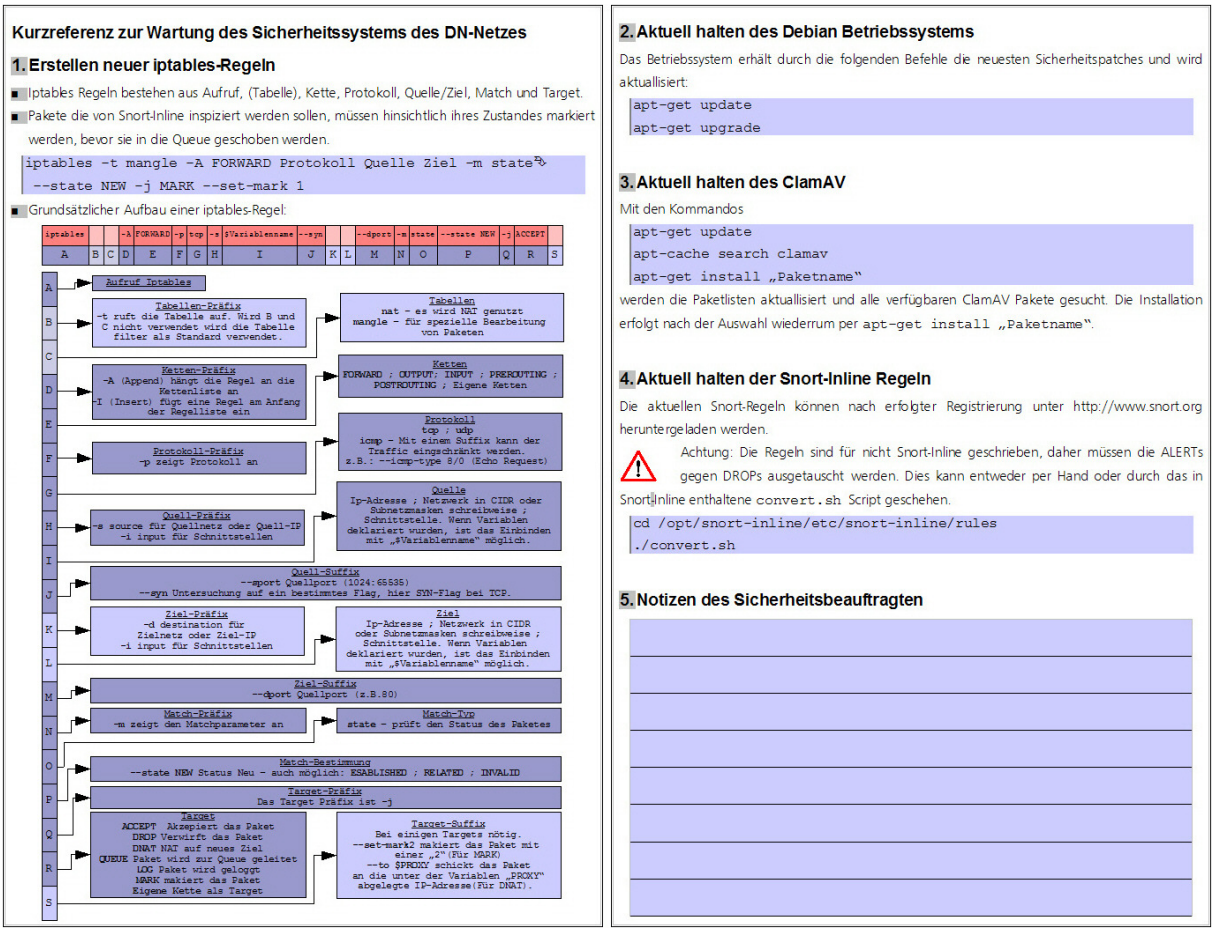
Achtung: Die Regeln sind nicht für Snort-Inline geschrieben, daher müssen die ALERTs gegen DROPs ausgetauscht werden. Dies kann händisch oder durch das in Snort-Inline enthaltene `convert.sh` Skript geschehen, siehe Seite 56.
Selbstverständlich brauchen nur die Regelsets aktualisiert werden, die in die Konfigurationsdatei `snort_inline.conf` eingebunden wurden.

Eine Möglichkeit, aktuelle Regeln automatisiert herunterzuladen, bietet das Tool Oinkmaster [ID-OINK]. Allerdings ist diese Software ebenfalls für Snort geschrieben worden, und bietet daher für Snort-Inline kaum Vorteile. An dieser Stelle soll lediglich auf Oinkmaster verwiesen werden. Es bleibt dem Sicherheitsbeauftragten überlassen, dieses Feature zu nutzen.

Weiterhin empfiehlt es sich, für den Sicherheitsbeauftragten die Snort-Inline Mailingliste im Auge zu behalten. William Metcalf und Victor Julien informieren hier über neue Funktionalitäten der neuen Snort-Inline Releases. Der Systembeauftragte kann dann entscheiden, ob er Snort-Inline updaten will um die neuen Funktionalitäten zu nutzen. Das Archiv wird täglich aktualisiert. Wenn die Informationen sofort per E-Mail gewünscht sind oder Fragen gestellt werden sollen, ist eine Registrierung nötig.[IL-SNO-MAIL]

9.4 Kurzreferenz

Die Kurzreferenz in der Umschlagklappe soll dem Sicherheitsbeauftragten des DN-Netzes die wichtigsten Punkte auf einen Blick zeigen, um das System betriebsbereit und aktuell zu halten. Sie faßt das Kapitel zum Erstellen einer iptables Regel und die Wartungs-Kapitel zusammen. Sie ist auch auf der beigefügten CD als PDF zu finden.



Kurzreferenz zur Wartung des Sicherheitssystems des DN-Netzes

1. Erstellen neuer iptables-Regeln

- Iptables Regeln bestehen aus Aufruf, (Tabelle), Kette, Protokoll, Quelle/Ziel, Match und Target.
- Pakete die von Snort-Inline inspiziert werden sollen, müssen hinsichtlich ihres Zustandes markiert werden, bevor sie in die Queue geschoben werden.

```
iptables -t mangle -A FORWARD Protokoll Quelle Ziel -m state
 --state NEW -j MARK --set-mark 1
```

- Grundsätzlicher Aufbau einer iptables-Regel:

iptables A -A B C D FORWARD E -p F tcp G -s H $Variablenname I --syn J K L --dport M -m N state O --state NEW P -j Q ACCEPT R S

A: Aufruf Iptables

B: Tabellen-Präfix
-t ruft die Tabelle auf. Wird B und C nicht verwendet wird die Tabelle filter als Standard verwendet.

Tabellen
nat - es wird NAT genutzt
mangle - für spezielle Bearbeitung von Paketen

C: Ketten-Präfix
-A (Append) hängt die Regel an die Kettenliste an
-I (Insert) fügt eine Regel am Anfang der Regelliste ein

Ketten
FORWARD ; OUTPUT; INPUT ; PREROUTING ; POSTROUTING ; Eigene Ketten

D: Protokoll-Präfix
-p zeigt Protokoll an

Protokoll
tcp ; udp
icmp - Mit einem Suffix kann der Traffic eingschränkt werden.
z.B.: --icmp-type 8/0 (Echo Request)

E: Quell-Präfix
-s source für Quellnetz oder Quell-IP
-i input für Schnittstellen

Quelle
Ip-Adresse ; Netzwerk in CIDR oder Subnetzmasken schreibweise ; Schnittstelle. Wenn Variablen deklariert wurden, ist das Einbinden mit „$Variablenname" möglich.

F: Quell-Suffix
--sport Quellport (1024:65535)
--syn Untersuchung auf ein bestimmtes Flag, hier SYN-Flag bei TCP.

G: Ziel-Präfix
-d destination für Zielnetz oder Ziel-IP
-i input für Schnittstellen

Ziel
Ip-Adresse ; Netzwerk in CIDR oder Subnetzmasken schreibweise ; Schnittstelle. Wenn Variablen deklariert wurden, ist das Einbinden mit „$Variablenname" möglich.

H: Ziel-Suffix
--dport Quellport (z.B.80)

I: Match-Präfix
-m zeigt den Matchparameter an

Match-Typ
state - prüft den Status des Paketes

J: Match-Bestimmung
--state NEW Status Neu - auch möglich: ESABLISHED ; RELATED ; INVALID

K: Target-Präfix
Das Target Präfix ist -j

L: Target
ACCEPT Akzepiert das Paket
DROP Verwirft das Paket
DNAT NAT auf neues Ziel
QUEUE Paket wird zur Queue geleitet
LOG Paket wird geloggt
MARK makiert das Paket
Eigene Kette als Target

Target-Suffix
Bei einigen Targets nötig.
--set-mark2 makiert das Paket mit einer „2" (Für MARK)
--to $PROXY schickt das Paket an die unter der Variablen „PROXY" abgelegte IP-Adresse(Für DNAT).

2. Aktuell halten des Debian Betriebssystems

Das Betriebssystem erhält durch die folgenden Befehle die neuesten Sicherheitspatches und wird aktuallisiert:

```
apt-get update
apt-get upgrade
```

3. Aktuell halten des ClamAV

Mit den Kommandos

```
apt-get update
apt-cache search clamav
apt-get install „Paketname"
```

werden die Paketlisten aktuallisiert und alle verfügbaren ClamAV Pakete gesucht. Die Installation erfolgt nach der Auswahl wiederrum per `apt-get install „Paketname"`.

4. Aktuell halten der Snort-Inline Regeln

Die aktuellen Snort-Regeln können nach erfolgter Registrierung unter http://www.snort.org heruntergeladen werden.

Achtung: Die Regeln sind für nicht Snort-Inline geschrieben, daher müssen die ALERTs gegen DROPs ausgetauscht werden. Dies kann entweder per Hand oder durch das in Snort_inline enthaltene `convert.sh` Script geschehen.

```
cd /opt/snort-inline/etc/snort-inline/rules
./convert.sh
```

5. Notizen des Sicherheitsbeauftragten

Abbildung 28: Kurzreferenz

10 Schlußbetrachtung

Die im Rahmen dieser Diplomarbeit gesteckten Ziele konnten durch die ausgeführten Arbeiten erreicht werden. Das Sicherheitskonzept wurde so bemessen, dass es mit möglichst geringem Aufwand und in absehbarer Zeit erweitert werden kann. Dies ist auf die Art des Aufbaus, sowie auf die Verwendung von modularer Open Source Software zurückzuführen. Die Vereinfachung der Wartung wurde durch die Zentralisierung des Wartungsrechners ermöglicht. Zwar waren anfangs bedienerfreundlichere Lösungen angedacht, jedoch erfüllten diese nicht die Erwartungen oder verfügten nicht über dringend benötigte Funktionen. Durch die Firewall und das IPS Snort-Inline war es möglich, den Datenverkehr zu reglementieren, unerwünschten Datenverkehr vollkommen auszuschließen und den Datenverkehr auf seine Gefährlichkeit hin zu untersuchen. Auch ist es gelungen, VoIP-Gepräche in das gesicherte Netzwerk und aus ihm herraus zu ermöglichen, ohne das DN-Netz der Gefahr eines Angriffes auszusetzen. Weiterhin wurde diese Diplomarbeit so verfasst, dass es dem künftigen Systemadministrator oder Sicherheitsbeauftragten möglich sein sollte, das System zu warten und aktuell zu halten.

An dieser Stelle sollen einige Punkte erwähnt werden, die nicht in der Diplomarbeit behandelt wurden, aber dennoch für die Sicherheit und den Betrieb eines Netzwerkes bedeutsam sind. Auf die Berücksichtigung dieser Punkte wurde bewusst verzichtet, da sie entweder durch die gesteckten Ziele schon definiert, oder Aufgrund dessen, dass es sich beim DN-Netz um ein Netzwerk für Lehre und Forschung handelt, nicht zweckmäßig waren.

- Datenschutz und Verwendung

  Beim Loggen des Datenverkehrs von Datenverbindungen in einem Netzwerk mit einem IPS werden unter Umständen Verbindungsdaten und personenbezogene Daten gesammelt. Aus diesem Grund sind Aspekte des Datenschutzes und deren Verwendbarkeit vor Gericht zu beachten. Personenbezogene Daten müssen anonymisiert werden[19]. Hierzu kann es reichen, Verbindungsdaten und Personendaten an unterschiedlichen Orten zu speichern. Die Verarbeitung von personenbezogenen

19 §3 Abs.1 Bundesdatenschutzgesetz (BDSG) - Datenvermeidung und Datensparsamkeit [IL-BDSG]

Daten ist nur zulässig, wenn eine schriftliche Genehmigung der Person vorliegt[20]. Da Logs leicht manipulierbar sind oder einfach komplett neu erstellt werden können, sind sie vor Gericht nur verwertbar, wenn eine Manipulation ausgeschlossen werden kann[21]. Durch das Schreiben auf ein Write-only-Medium oder durch eine digitale Signatur sind sie verwertbar. Ebenso ist es untersagt, Nutzungsdaten eines VoIP-Gespräches zu erheben, die nicht für den Betrieb des Dienstes nötig sind[22].

Da im Experimentiernetz keine personenbezogenen Daten vorgehalten werden, sind Datenschutzaspekte sowie die Verwertbarkeit vor Gericht für das Sicherheitssystem ohne Belang. Ein Student, der im Experimentiernetz arbeitet, ist dem System weder mit Namen noch mit Matrikelnummer bekannt. VoIP-Geprächs-Verbindungen werden vom Sicherheitssystem ebenfalls nicht geloggt, und der Dienst steht lediglich den im Experimentiernetz forschenden Mirarbeitern und Diplomanden zur Verfügung. Die vollständigen Gesetzestexte sind unter [IL-BDSG], [IL-STPO], [IL-ZPO] und [IL-TDDSG] nachzulesen.

- Redundanz

  Unter Redundanz versteht man im Allgemeinen das mehrfache Vorhandensein einer funktional gleichen Ressource. Dies verhindert den Verlust einer Fähigkeit, falls eine Ressource ausfallen sollte. So verfügen beispielsweise große Unternehmensnetzwerke (z.b. in der E-Commerce Branche oder im Finanzwesen) über mehrere Internetzugänge die, mit je einem Sicherheitsobjekt[23] abgesichert sind. Fällt eines dieser Sicherheitsobjekte aus, ist das Netz durch dynamisches Routing in der Lage, über die anderen Schnittstellen mit dem Internet verbunden zu bleiben. Dies ist je nach gewünschtem Redundanzumfang mit einem erheblichen finanzellen und materiellen Aufwand verbunden.

  Für das DN-Experimentiernetz wurde eine doppelte Netzanbindung von vorneherein aus wirtschaftlichen und organisatorischen Gründen ausgeschlossen. Dennoch ist die Implementierung einer weiteren Verbindung zum NT-Netz durchaus möglich.

20 §4a Abs.1 Bundesdatenschutzgesetz - Einwilligung [IL-BDSG]

21 Dies ist in der Strafprozessordnung (StPO) [IL-STPO] und der Zivilprozessordnung (ZPO) [IL-ZPO] festgelegt.

22 §6 Abs.1 Teledienstdatenschutzgesetz (TDDSG) - Nutzungsdaten [IL-TDDSG]

23 Je nach Sensibilität kann dieses Sicherheitsobjekt auch aus mehreren Komponenten bestehen.

- Ausblick

  Im Rahmen der Diplomarbeit fiel auf, dass es keine Präprozessoren oder Regeln in Snort-Inline für SIP oder RTP Anwendungen gibt. Durch die rasende Verbreitung, die VoIP-Anwendungen zur Zeit erfahren, bleibt abzuwarten, ob die Snort bzw. Snort-Inline Entwickler entsprechende Implementierungen vornehmen. Vielleicht ist dieser Punkt aber auch für eine künftige Diplomarbeit geeignet. Hierbei müsste unter anderem geprüft werden, ob ein solcher Präprozessor oder das Prüfen Anhand von Regeln den Ansprüchen des QoS (Quality of Service) gerecht werden kann.

  Gerade in den letzten Tagen, während der Entstehung der Diplomarbeit wurde der Snort-Inline-Wedding-Release[24] veröffentlicht. Die wichtigsten Neuerungen sind Verbesserungen am `clamav` und `stream4` Präprozessor, ein neuer Präprozessor (`bait and switch`), der Attacken auf eine IP-Adresse umleiten kann, und die Abschaffung der `libnet` Bibliothek.

24 Victor Julien machte Wiliam Metcalf damit ein Hochzeitsgeschenk, da er am Erscheinugstag heiratete.

11 Legende und Abkürzungsverzeichnis

11.1 Legende

Verweise auf Literatur- und Internetquellen sind in eckige Klammern gesetzt.

Kürzelschlüssel	Quelle
[Autorkürzel-Titelkürzel]	Literaturquellen
[IL-textquellenkürzel]	Internettextquellen
[ID-Downloadquellenkürzel]	Software-Downloadquellen

Wichtige Punkte sind mit einem Achtungsymbol versehen.

Konsoleneingaben oder Quelltextpassagen wurden unterlegt.

```
apt-get moo
```

Ist eine Codezeile zu lang, wird sie duch einen abgeknickten Pfeil in der nächsten Zeile fortgeführt.

```
Franz jagt im komplett verwahrlosten Taxi quer durch ➥
Bayern
```

11.2 Abkürzungsverzeichnis

Abkürzung	Beschreibung	RFC
ALG	Aplication Layer Gateway	
ARP	Address Resolution Protocol	RFC 826
BASE	Basic Analysis and Security Engine	
BDSG	Bundesdatenschutzgesetz	
DDoS	distributed Denial of Service	
DMZ	Demilitarisierte Zone	
DNS	Domain Name System	RFC 882
DoS	Denial of Service	

Abkürzung	Beschreibung	RFC
FIFO	First in Firts out	
FTP	File Transfer Protocol	RFC 959
GD	Bildmanipulationsbibliothek	
GPL	General Public License	
HIDS	Hostbased Intrusion Detection System	
HIPS	Hostbased Intrusion Prevention System	
HTTP	Hypertext Transfer Protocol	RFC 1945 und RFC 2616
HTTPS	Hypertext transfer protocol secure	RFC2828
ICMP	Internet Control Message Protocol	RFC 792
IDS	Intrusion Detection System	
IP	Internet Protocol	
IPS	Intrusion Prevention System	
LAN	Local Area Network	
MySQL	My Structured Query Language	
NAT	Network Adress Translation	
NIDS	Network Intrusion Detection System	
NIPS	Network Intrusion Prevention System	
PDF	Portable Document Format	
PHP	Hypertext Preprocessor	
QoS	Quality of Service	
RFC	Request for Coments	
RIP	Routing Information Protocol	RFC 1085
RTP	Realtime Tranport Protocol	RFC 3550, früher RFC 1889
SDP	Session Description Protocol	RFC 2327
SER	SIP Express Router	
SIP	Session Initiation Protocol	RFC 3261, früher RFC 2534
SNMP	Simple Network Management Protocol	RFC 1157, früher RFC 1067
SPAM	Unverlangte Massen-E-Mail	
SPIT	SPAM over Internet Telefony	
SRTP	Secure Realtime Transport Protocol	RFC 3711
SSH	Secure shell	
SSL	Secure Sockets Layer	RFC 2246

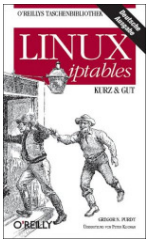
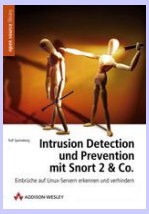
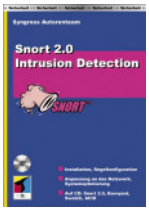
Abkürzung	Beschreibung	RFC
STPO	Strafprozessordnung	
TCP	Transmission Control Protocol	RFC 793
TDDSG	Teledienstdatenschutzgesetz	
TLS	Transport Layer Security	RFC 2246
UDP	User Datagram Protocol	RFC 768
URI	Uniform Ressource Identifier	RFC 3986
URL	Uniform Resource Locator	RFC 1738
VoIP	Voice over IP	
VOIPSA	VoIP Security Alliance	
VPN	Virtual Private Network	
ZPO	Zivilprozessordnung	

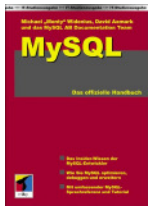
12 Abbildungsverzeichnis

Anmerkung: Die in Kapitel 7.3 verwendeten Abbildungen zeigen die jeweiligen Logos zur verwendeten Software.

13 Quellenverzeichnis

13.1 Literaturquellen

Bezeichnung	Bild	Titel
[PUR-IPT]		**Linux iptables - kurz & gut** Autor: Gregor N. Purdy ISBN: 3-8972-1506-3 Verlag: O'Reilly
[RON-DEB]		**Debian GNU/Linux 3.1-Anwenderhandbuch** Autor: Frank Ronneburg ISBN: 3-8273-2303-7 Verlag: Addison-Wesley
[RUPP-SIP]		**SIP, Multimediale Dienste im Internet** Autor: Rupp/Siegmund/Lautenschläger ISBN: 3-89864-167-8 Verlag: dpunkt
[SPEN-IDPS]		**Intrusion Detection und Prevention mit Snort 2 & Co** Autor: Ralf Spenneberg ISBN: 3-8273-2134-4 Verlag: Addison-Wesley
[SYN-SNO]		**Snort 2.0 Intrusion Detection** Autor: Syngress-Autorenteam ISBN: 3-8266-1304-X Verlag: Mitp-Verlag
[TEC]		**Tecchannel Compact - Netzwerk-Know-How** Ausgabe Jan./Feb./März 2005

Bezeichnung	Bild	Titel
[WID-MYS]		MySQL Autor: Michael Widenius und David Axmark ISBN: 3-8266-1351-1 Verlag: Mitp-Verlag
[ZIEG-FIRE]		**Linux-Firewalls** Autor: Robert L. Ziegler ISBN: 3-8272-6703-X Verlag: Markt & Technik

13.2 Internet-Informationsquellen

Bezeichnung	Link
[IL-APA]	http://docs.php.net/en/install.unix.debian.html
[IL-BASE]	http://secureideas.sourceforge.net/
[IL-BDSG]	http://bundesrecht.juris.de/bundesrecht/bdsg_1990/
[IL-BSI]	http://www.bsi.de/gshb/index.htm
[IL-CISC]	http://www.cisco.com/univercd/cc/td/doc/product/iaabu/pix/pix_sw/v_63/config/fixup.htm#wp1104084
[IL-CLAMDB]	http://clamav-du.securesites.net/cgi-bin/clamgrok
[IL-COL]	http://www.cs.columbia.edu/~hgs/rtp/faq.html#ports
[IL-DEB]	http://www.debian.de/
[IL-DIPL]	http://dnserver.nt.fh-koeln.de/grebe/Diplom/Diplom.htm
[IL-GUR]	http://www.internetsecurityguru.com/documents/Snort_and_BASE_on_CentOS_RHEL_or_Fedora.pdf
[IL-LUG]	http://zentralschweiz.lugs.ch/treff-20050421/SnortBase_Installation.pdf#search='snort%20igor'
[IL-MYS]	http://www.mysql.de/
[IL-NESS]	http://www.nessus.org/
[IL-NET]	http://www.netfilter.org/index.html
[IL-RTP]	http://www.voip-info.org/wiki/view/Portaone+rtpproxy

Bezeichnung	Link
[IL-SAV]	http://gd4.tuwien.ac.at/opsys/linux/lg/117/savage.html
[IL-SER]	http://sip-router.org
[IL-SNO-MAIL]	http://sourceforge.net/mailarchive/forum.php?forum=snort-inline-users
[IL-SNO]	http://snort-inline.sourceforge.net/
[IL-STPO]	http://bundesrecht.juris.de/bundesrecht/stpo/
[IL-TDDSG]	http://bundesrecht.juris.de/bundesrecht/tddsg/
[IL-TOM]	http://www.tomsnetworking.de/content/reports/j2005a/background_voip_security/page5.html
[IL-ZPO]	http://bundesrecht.juris.de/bundesrecht/zpo/

13.3 Internet Software-Download Quellen

Bezeichnung	Link
[ID-BASE]	http://sourceforge.net/project/showfiles.php?group_id=103348
[ID-DEB]	http://www.debian.de/CD/netinst/
[ID-LIBNET]	http://www.packetfactory.net/libnet/dist/deprecated/libnet-1.0.2a.tar.gz
[ID-LIBP]	http://www.tcpdump.org/
[ID-OINK]	http://oinkmaster.sourceforge.net/download.shtml
[ID-PCRE]	ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/
[ID-RTP]	http://www.portaone.com/resources/downloads/
[ID-SER]	ftp://ftp.berlios.de/pub/ser/
[ID-SER-D]	ftp://ftp.berlios.de/pub/ser/0.8.14/packages/debian/stable/
[ID-SNO-IN]	http://snort-inline.sourceforge.net/download.html
[ID-SNO]	http://www.snort.org/

14 Anhang

14.1 Firewallscript snortfw-start.sh

```
#!/bin/sh -vx
#
# Firewall Script fuer Firewall mit DMZ und Snort-Inline
# Holger Moskopp
#

############################################################

# 1. Deklarieren der IP-Adressen, Subnetze und der Ethernetkarten.

# Aussenseite
EXTERN_IP="192.168.1.143"
EXTERN_ETH="eth0"

# DMZ
DMZ_IP="192.168.1.17"
DMZ_NET="192.168.1.16/30"
DMZ_ETH="eth1"

#Innenseite
INTERN_IP="192.168.1.21"
INTERN_NET="192.168.1.20/30"
INTERN_ETH="eth2"

# Besondere Rechner
fwclient="192.168.1.22"             # Wartungsrechner
prox="192.168.1.18"                 # SIP/RTP Rroxy

# Besondere Netze
NT_NET="192.168.1.128/27"

# localhost, der Firewallrechner himself ;-)
LOOP="lo"
LOOP_IP="127.0.0.1"

# 2. Laden der benoetigten Module und Konfigurationen
/sbin/modprobe ip_tables
/sbin/modprobe ip_conntrack
/sbin/modprobe ip_conntrack_ftp
/sbin/modprobe iptable_filter
/sbin/modprobe iptable_mangle
/sbin/modprobe ipt_LOG
/sbin/modprobe ipt_state
/sbin/modprobe ip_queue # fuer Snort - Inline
```

```
# Forwarding aktivieren
echo "1" > /proc/sys/net/ipv4/ip_forward
# Syncookies aktivieren - gegen flood Attacken
echo "1" > /proc/sys/net/ipv4/tcp_syncookies

# iptabels deklarieren
IPTABLES="/sbin/iptables"

# 3. Aufraeumen - loeschen der alten Regeln.
$IPTABLES -F # loescht alle Ketten
$IPTABLES -X # loescht alle benutzerdefinierten Ketten

# 4. Globale Policy setzen - alles droppen

$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP


# 5. Statefull Inspection anordnen und Vorbereitungen fuer Snort - Inline
# 5.1  Pakete bestehender und verwandter Verbindungen akzeptieren
$IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

# 5.2 Pakete die zu keiner der oberen Verbindungen passen droppen
$IPTABLES -A INPUT -m state --state INVALID -j DROP
$IPTABLES -A OUTPUT -m state --state INVALID -j DROP
$IPTABLES -A FORWARD -m state --state INVALID -j DROP

# 5.3 Markierte Pakete fuer Snort-Inline loggen
$IPTABLES -I OUTPUT -m mark --mark 1 -j QUEUE
$IPTABLES -I OUTPUT -m mark --mark 2 -j QUEUE
$IPTABLES -I INPUT -m mark --mark 1 -j QUEUE
$IPTABLES -I INPUT -m mark --mark 2 -j QUEUE
$IPTABLES -I FORWARD -m mark --mark 1 -j QUEUE
$IPTABLES -I FORWARD -m mark --mark 2 -j QUEUE


# 6. Einstellungen fuer die Localhost - Der King darf auch nicht alles.
$IPTABLES -A INPUT -p ALL -i $LOOP -s $LOOP_IP -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LOOP -s $INTERN_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -o $LOOP -d $LOOP_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -o $LOOP -d $INTERN_IP -j ACCEPT
#Verkehr fuer die MYSQL Daten von Snort-Inline $EXTERN_ETH nach innen zulassen.
$IPTABLES -A OUTPUT -p tcp -o $INTERN_ETH -d $fwclient --dport 3306 -m state --state
NEW,ESTABLISHED,RELATED -j ACCEPT
```

```
# 7. Regeln nach Protokollen geordnet

# 7.1 DNS Anfagen erlauben.

# Von innen nach aussen
$IPTABLES -A FORWARD -p udp -s $INTERN_NET --sport 1024:65535 -o $EXTERN_ETH --dport 53 -m
state --state NEW -j ACCEPT
# Von Firewall nach aussen
$IPTABLES -A OUTPUT -p udp -o $EXTERN_ETH --sport 1024:65535 --dport 53 -m state --state
NEW,ESTABLISHED,RELATED -j ACCEPT
# Von DMZ nach aussen
$IPTABLES -A FORWARD -p udp -s $DMZ_NET --sport 1024:65535 -o $EXTERN_ETH --dport 53 -m state
--state NEW,ESTABLISHED,RELATED -j ACCEPT



# 7.2 SSH

# SSH von Wartungsrechner nach Firewall
# Mit Snort alles anschauen
$IPTABLES -t mangle -A INPUT -s $fwclient -p tcp --dport 22 -m state --state NEW -j MARK --
set-mark 1
$IPTABLES -t mangle -A INPUT -s $fwclient -p tcp --dport 22 -m state --state ESTABLISHED -j
MARK --set-mark 2
$IPTABLES -t mangle -A OUTPUT -o $INTERN_ETH -p tcp --sport 22 -d $fwclient -m state --state
ESTABLISHED -j MARK --set-mark 2
# SSH von Wartungsrechner nach DMZ
# Mit Snort alles anschauen
$IPTABLES -t mangle -A FORWARD -s $fwclient -p tcp -d $DMZ_NET --dport 22 -m state --state
NEW -j MARK --set-mark 1
$IPTABLES -t mangle -A FORWARD -s $fwclient -p tcp -d $DMZ_NET --dport 22 -m state --state
ESTABLISHED -j MARK --set-mark 2
$IPTABLES -t mangle -A FORWARD -s $DMZ_NET -p tcp --sport 22 -d $fwclient -m state --state
ESTABLISHED -j MARK --set-mark 2
# SSH nach von innen nach aussen.
# Mit Snort nur Initiator EINMAL anschauen
$IPTABLES -t mangle -A FORWARD -s $INTERN_NET -p tcp -o $EXTERN_ETH --dport 22 -m state --
state NEW -j MARK --set-mark 1
$IPTABLES -t mangle -A FORWARD -s $EXTERN_ETH -p tcp --sport 22 -d $INTERN_NET -m state --
state ESTABLISHED -j MARK --set-mark 2
$IPTABLES -A FORWARD -p tcp -s $INTERN_NET --sport 1024:65535 -o $EXTERN_ETH --dport 22 -m
state --state NEW -j ACCEPT

# SSH von aussen auf besondere Rechner
#                     Von NT-Netz auf Wartungsrechner im Internen Netz
#                     Mit Snort alles anschauen
$IPTABLES -t mangle -A FORWARD -s $NT_NET -p tcp -d $fwclient --dport 22 -m state --state NEW
-j MARK --set-mark 1
```

```
$IPTABLES -t mangle -A FORWARD -s $NT_NET -p tcp -d $fwclient --dport 22 -m state --state
ESTABLISHED -j MARK --set-mark 2
$IPTABLES -t mangle -A FORWARD -s $fwclient -p tcp --sport 22 -d $NT_NET -m state --state
ESTABLISHED -j MARK --set-mark 2
$IPTABLES -A FORWARD -p tcp -s $NT_NET--sport 1024:65535 -d $fwclient --dport 22 -m state --
state NEW -j ACCEPT

# Von NT-Netz auf einen Rechner in der DMZ
#                    Vermeiden, dass Firewall SSH-Verbindung zu DMZ aufnehmen kann
$IPTABLES -A OUTPUT -p tcp -o $DMZ_ETH -m state --state NEW -j DROP
#                    Mit Snort alles anschauen
$IPTABLES -t mangle -A FORWARD -s $NT_NET-p tcp -d $prox --dport 22 -m state --state NEW -j
MARK --set-mark 1
$IPTABLES -t mangle -A FORWARD -s $NT_NET-p tcp -d $prox --dport 22 -m state --state
ESTABLISHED -j MARK --set-mark 2
$IPTABLES -t mangle -A FORWARD -s $prox -p tcp --sport 22 -d $NT_NET-m state --state
ESTABLISHED -j MARK --set-mark 2
$IPTABLES -A FORWARD -p tcp -s $NT_NET--sport 1024:65535 -d $prox --dport 22 -m state --state
NEW -j ACCEPT


# 7.3 SIP UND RTP via DMZ

# RTP Verkehr
$IPTABLES -I FORWARD -p udp -i $DMZ_ETH --sport 1024:65535 -o $EXTERN_ETH --dport 1024:65535
-m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -I FORWARD -p udp -i $EXTERN_ETH --sport 1024:65535 -o $DMZ_ETH --dport 1024:65535
-m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -I FORWARD -p udp -i $INTERN_ETH --sport 1024:65535 -o $DMZ_ETH --dport 1024:65535
-m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -I FORWARD -p udp -i $DMZ_ETH --sport 1024:65535 -o $INTERN_ETH --dport 1024:65535
-m state --state NEW,ESTABLISHED,RELATED -j ACCEPT

# SIP Verkehr
$IPTABLES -t nat -I PREROUTING -p udp -i $EXTERN_ETH --dport 5060:5062 -j DNAT --to $prox
$IPTABLES -t nat -I PREROUTING -p udp -i $INTERN_ETH --dport 5060:5062 -j DNAT --to $prox


# 7.4 FTP

# FTP von innen nach aussen
# Mit Snort nur Initiator EINMAL anschauen
$IPTABLES -t mangle -A FORWARD -i $INTERN_ETH -o $EXTERN_ETH -p tcp --dport 20:21 -m state --
state NEW -j MARK --set-mark 1
$IPTABLES -t mangle -A FORWARD -i $EXTERN_ETH -o $INTERN_ETH -p tcp --sport 20:21 -m state --
state ESTABLISHED -j MARK --set-mark 2

# 7.5 Telnet
```

```
#     Telnet von innen nach aussen.
#             Mit Snort alles anschauen
$IPTABLES -t mangle -A FORWARD -p tcp -s $INTERN_NET --syn -m state --state NEW --dport 23 -j
MARK --set-mark 1
$IPTABLES -t mangle -A FORWARD -p tcp -s $INTERN_NET --syn -m state --state ESTABLISHED --
dport 23 -j MARK --set-mark 2
$IPTABLES -t mangle -A FORWARD -p tcp -m state --state ESTABLISHED --sport 23 -j MARK --set-
mark 2

# 7.6 Rund um HTTP (HTTP Proxy, HTTPS) - Snort-inline

# HTTP via Proxy
# Mit Snort nur Initiator EINMAL anschauen
$IPTABLES -t mangle -A FORWARD -p tcp -s $INTERN_NET --syn -m state --state NEW --dport 8080
-j MARK --set-mark 1
$IPTABLES -t mangle -A FORWARD -p tcp -m state --state ESTABLISHED --sport 8080 -j MARK --
set-mark 2
# HTTPS
# Mit Snort nur Initiator EINMAL anschauen
$IPTABLES -t mangle -A FORWARD -p tcp -s $INTERN_NET --syn -m state --state NEW --dport 443
-j MARK --set-mark 1
$IPTABLES -t mangle -A FORWARD -p tcp -m state --state ESTABLISHED --sport 443 -j MARK --set-
mark 2
# HTTP von der Firewall nach aussen - zum updaten von Debian und Co.
# Mit Snort nur Initiator EINMAL anschauen
$IPTABLES -t mangle -A OUTPUT -o $EXTERN_ETH -p tcp --dport 8080 -m state --state NEW -j MARK
--set-mark 1
$IPTABLES -t mangle -A INPUT -i $EXTERN_ETH -p tcp --sport 8080 -m state --state ESTABLISHED
-j MARK --set-mark 2
# HTTP von der DMZ nach aussen - zum updaten von Debian
# Mit Snort nur Initiator EINMAL anschauen
$IPTABLES -t mangle -A FORWARD -i $DMZ_ETH -o $EXTERN_ETH -p tcp --dport 8080 -m state --
state NEW -j MARK --set-mark 1
$IPTABLES -t mangle -A FORWARD -i $EXTERN_ETH -o $DMZ_ETH -p tcp --sport 8080 -m state --
state ESTABLISHED -j MARK --set-mark 2


# 7.7 SNMP

# von aussen auf bestimmte Rechner
#             Mit Snort alles anschauen
$IPTABLES -t mangle -A FORWARD -p udp -s $NT_NET-d $fwclient --dport 161:162  -m state --
state NEW -j MARK --set-mark 1
$IPTABLES -t mangle -A FORWARD -p udp -s $NT_NET-d $fwclient --dport 161:162  -m state --
state ESTABLISHED -j MARK --set-mark 2
$IPTABLES -t mangle -A FORWARD -p udp -s $fwclient --sport 161:162 -d $NT_NET -m state --
state ESTABLISHED -j MARK --set-mark 2
```

```
# von innen nach NT-Netz
#       Mit Snort alles anschauen
$IPTABLES -t mangle -A FORWARD -p udp -s $INTERN_NET -d $NT_NET--dport 161:162  -m state --
state NEW -j MARK --set-mark 1
$IPTABLES -t mangle -A FORWARD -p udp -s $INTERN_NET -d $NT_NET--dport 161:162  -m state --
state ESTABLISHED -j MARK --set-mark 2
$IPTABLES -t mangle -A FORWARD -p udp -s $NT_NET--sport 161:162 -d $INTERN_NET  -m state --
state ESTABLISHED -j MARK --set-mark 2

# 7.8 ICMP - Ping und Konsorten

# Von NT-Netz nach Innen
#             Echo Request
#             Mit Snort alles anschauen
$IPTABLES -t mangle -A FORWARD -s $NT_NET-p icmp --icmp-type 8/0 -d $INTERN_NET -m state --
state NEW -j MARK --set-mark 1
$IPTABLES -t mangle -A FORWARD -s $NT_NET-p icmp --icmp-type 8/0 -d $INTERN_NET -m state --
state ESTABLISHED -j MARK --set-mark 2
$IPTABLES -t mangle -A FORWARD -s $INTERN_NET -p icmp --icmp-type 8/0 -d $NT_NET-m state --
state ESTABLISHED -j MARK --set-mark 2
$IPTABLES -A FORWARD  -p icmp  --icmp-type 8/0 -s $NT_NET-d $INTERN_NET -m state --state NEW
-j ACCEPT
# Von NT-Netz nach DMZ
#             Echo Request
#             Mit Snort alles anschauen
$IPTABLES -t mangle -A FORWARD -s $NT_NET-p icmp --icmp-type 8/0 -d $DMZ_NET -m state --state
NEW -j MARK --set-mark 1
$IPTABLES -t mangle -A FORWARD -s $NT_NET-p icmp --icmp-type 8/0 -d $DMZ_NET -m state --state
ESTABLISHED -j MARK --set-mark 2
$IPTABLES -t mangle -A FORWARD -s $DMZ_NET -p icmp --icmp-type 8/0 -d $NT_NET-m state --state
ESTABLISHED -j MARK --set-mark 2


# von Innen nach Aussen
#     Jede Art von ICMP nach Aussen
#             Mit Snort nur Initiator EINMAL anschauen
$IPTABLES -t mangle -A FORWARD -i $INTERN_ETH -p icmp -o $EXTERN_ETH -m state --state NEW -j
MARK --set-mark 1
$IPTABLES -t mangle -A FORWARD -i $EXTERN_ETH -p icmp -o $INTERN_ETH -m state --state
ESTABLISHED -j MARK --set-mark 2
#     Jede Art von ICMP nach DMZ
#             Mit Snort nur Initiator EINMAL anschauen
$IPTABLES -t mangle -A FORWARD -i $INTERN_ETH -p icmp -d $DMZ_NET -m state --state NEW -j
MARK --set-mark 1
$IPTABLES -t mangle -A FORWARD -s $DMZ_NET -p icmp -o $INTERN_ETH -m state --state
ESTABLISHED -j MARK --set-mark 2
```

```
# 7.9 RIP Routingprotokoll

# von innen nach aussen
$IPTABLES -A FORWARD -i $INTERN_ETH -p udp -o $EXTERN_ETH --dport 520 -m state --state NEW -j
ACCEPT
# von aussen nach innen
$IPTABLES -A FORWARD -i $EXTERN_ETH -p udp -o  $INTERN_ETH --dport 520 -m state --state -j
ACCEPT

sleep 2

# snort-inline starten
/opt/snort-inline/bin/./snort_inline -Qc /opt/snort-inline/etc/snort-inline/snort_inline.conf
```

14.2 Snort-Inline Konfigurationsscript snort_inline.conf

```
# Honeynet snort_inline configuration file
# Version 0.5
# Last modified 01 January, 2004
#
# Standard Snort configuration file modified for inline
# use.  Most preprocessors currently do not work in inline
# mode, as such they are not included.
#

### Network variables
var HOME_NET 192.168.1.20/30
var HONEYNET any
var EXTERNAL_NET any
var SMTP_SERVERS any
var TELNET_SERVERS any
var HTTP_SERVERS any
var SQL_SERVERS any

# 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 8080

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

```
# Ports you do oracle attacks on
var ORACLE_PORTS 1521

### As of snort_inline 2.2.0 we drop
### packets with bad checksums. We can
### log bad checksum packets by adding "log"
### as an option to checksum_mode
config checksum_mode: all

# Path to your rules files (this can be a relative path)
var RULE_PATH rules

### Preprocessors
# usage guidelines:  if the plugin normalizes the packet so that the
# detection engine can better interpret the data, the plugin can be
# used with the snort_inline safely.  If the plugin itself makes
# the alert decisions, then we have to modify it to drop packets.

# Done by IPTables.  Iptables assembles fragments when we use connection
#                    tracking; therefore, we don't have to use frag2
# preprocessor frag2

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

preprocessor stream4: disable_evasion_alerts,iptablesnewmark,iptablesestmark,forceipstate
preprocessor stream4_reassemble: both

# ClamAV virusscanning preprocessor
#

preprocessor clamav: ports all, action-drop

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

# rpc_decode: normalize RPC traffic
# --------------------------------

preprocessor rpc_decode: 111 32771

# bo: Back Orifice detector
# ------------------------

preprocessor bo
```

```
# telnet_decode: Telnet negotiation string normalizer
# ---------------------------------------------------

preprocessor telnet_decode

# Flow-Portscan: detect a variety of portscans
# --------------------------------------


### Logging alerts of outbound attacks
#output alert_full: snort_inline-full
#output alert_fast: snort_inline-fast

### If you want to log the contents of the dropped packets, remove comment
#output log_tcpdump: tcpdump.log

### MYSQL Datenbankort
output database: log, mysql, user=snort password=russel dbname=snort host=192.168.1.22

# Include classification & priority settings
include classification.config
include reference.config

### The Drop Rules
# Enabled
#include $RULE_PATH/test.rules
include $RULE_PATH/exploit.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/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/pop3.rules
include $RULE_PATH/pop2.rules
include $RULE_PATH/web-attacks.rules
include $RULE_PATH/virus.rules
include $RULE_PATH/nntp.rules

### Disabled

# include $RULE_PATH/other-ids.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/chat.rules
# include $RULE_PATH/multimedia.rules
# include $RULE_PATH/p2p.rules
```

14.3 Startskript Proxy sip-start.sh

```
#!/bin/bash
#Startskript fuer rtpproxy und SER - sip-start.sh
echo "DER PROXY WIRD JETZT GESTARTET"
echo "------------------------------"
echo " Stoppen des SER zum start der ben. Komponenten"
killall ser
killall rtpproxy
export SIP_DOMAIN="PROXY.dn.fh-koeln.de"
echo "Starten des rtpproxy"
/usr/local/bin/rtpproxy -l 192.168.1.18/192.168.1.18
sleep 2
echo "Neustart des SER"
/usr/local/sbin/serctl start
echo "SIP/RTP Proxy ist nun Bereit"
```

14.4 SER-Konfigurationsscript ser.cfg

```
# SER_PROXY Script
debug=3 # debug level (cmd line: -ddddddddddd)
fork=yes
log_stderror=yes # (cmd line: -E)

check_via=no # (cmd. line: -v)
dns=no # (cmd. line: -r)
rev_dns=no # (cmd. line: -R)
port=5060
children=4
fifo="/tmp/ser_fifo"

# ------------------ module loading ----------------------------------

loadmodule "/usr/local/lib/ser/modules/sl.so"
loadmodule "/usr/local/lib/ser/modules/tm.so"
loadmodule "/usr/local/lib/ser/modules/rr.so"
loadmodule "/usr/local/lib/ser/modules/maxfwd.so"
loadmodule "/usr/local/lib/ser/modules/nathelper.so"
loadmodule "/usr/local/lib/ser/modules/textops.so"


#Der RTP-Proxy
modparam("nathelper", "rtpproxy_sock", "/var/run/rtpproxy.sock")

alias="192.168.1.18"

# -- Tests der Headerfelder --
route[1] {

# -- Initialer Test auf zu viele Weiterleitungen --
if (!mf_process_maxfwd_header("10")) {
sl_send_reply("483","Zu viele Weiterleitungen");
break;
};

# -- Test auf eine zu grosse Nachricht --
if ( msg:len > max_len ) {
sl_send_reply("513", "Nachricht zu gross");
break;
};
}

# -- alle Proxy-Hops mitloggen --
route[2] {

record_route();
```

```
loose_route();

}

# -- alle Benutzer mit privaten IPs korrigieren --
#route[3] {

#if (search("(c|C)ontact:.*10\.*")) {
#fix_nated_contact();
#if (method=="INVITE") {
#fix_nated_sdp("2");
#};
#};
#}

# Verbindungsrouting
route{
# -- SIP-Nachrichten Tests --
route(1);
# -- Hops-Speichern --
route(2);
# -- NAT --
#route(3);

# -- Weiterleitung an internen SER, wenn Nachricht von Aussen oder DMZ kommt --
if (dst_ip==192.168.1.22) {

if (method == "INVITE") {
                        if (force_rtp_proxy("FAII")){
                                t_on_reply("1");
                } else {
                        sl_send_reply("403", "User nicht erreichbar oder unbekannt!");
                        break;
                };
        };
        if (method == "BYE" || method == "CANCEL")
                unforce_rtp_proxy();

log(3, "Weiterleitung an internen SER\n");
forward("192.168.1.22");
break;
};

# der RTP-Proxy
        if (method == "INVITE") {
                                if (force_rtp_proxy("FAII")){
                                        t_on_reply("1");
                } else {
```

```
                        sl_send_reply("403", "User nicht erreichbar oder unbekannt!");
                        break;
                };

        };

        if (method == "BYE" || method == "CANCEL")
                unforce_rtp_proxy();

        # Do strict routing if pre-loaded route headers present
        if (loose_route()) {
        t_relay();
        break;
        };

# -- Fehlerhafte Verbindungen ignorieren --
if (!t_relay()) {
sl_reply_error();
};
}
        onreply_route[1] {
        if (!(status=~"183" || status=~"200"))
                break;
        force_rtp_proxy("FA");
}
```

14.5 Konfigurationsdatei für SER-Registrar im Testaufbau

```
#
# $Id: ser.cfg,v 1.21.4.1 2003/11/10 15:35:15 andrei Exp $
#
# simple quick-start config script
#

# ----------- global configuration parameters ------------------------

debug=3         # debug level (cmd line: -dddddddddd)
fork=yes
log_stderror=yes  # (cmd line: -E)

/* Uncomment these lines to enter debugging mode
debug=7
#fork=no
log_stderror=yes
*/

check_via=yes  # (cmd. line: -v)
dns=no           # (cmd. line: -r)
```

```
rev_dns=no      # (cmd. line: -R)
port=5060
#children=4
fifo="/tmp/ser_fifo"

# ------------------ module loading ----------------------------------

# Uncomment this if you want to use SQL database
#loadmodule "/usr/local/lib/ser/modules/mysql.so"

loadmodule "/usr/local/lib/ser/modules/sl.so"
loadmodule "/usr/local/lib/ser/modules/tm.so"
loadmodule "/usr/local/lib/ser/modules/rr.so"
loadmodule "/usr/local/lib/ser/modules/maxfwd.so"
loadmodule "/usr/local/lib/ser/modules/usrloc.so"
loadmodule "/usr/local/lib/ser/modules/registrar.so"

loadmodule "/usr/local/lib/ser/modules/nathelper.so"

# Uncomment this if you want digest authentication
# mysql.so must be loaded !
#loadmodule "/usr/local/lib/ser/modules/auth.so"
#loadmodule "/usr/local/lib/ser/modules/auth_db.so"

# ----------------- setting module-specific parameters ---------------

# -- usrloc params --
#Der RTP-Proxy
modparam("nathelper", "rtpproxy_sock", "/var/run/rtpproxy.sock")
modparam("usrloc", "db_mode",   0)

# -- rr params --
# add value to ;lr param to make some broken UAs happy
modparam("rr", "enable_full_lr", 1)

# -------------------------  request routing logic -------------------

# main routing logic

route{
        # initial sanity checks -- messages with
        # max_forwards==0, or excessively long requests
        if (!mf_process_maxfwd_header("10")) {
                sl_send_reply("483","Too Many Hops");
                break;
        };
        if ( msg:len > max_len ) {
                sl_send_reply("513", "Message too big");
```

```
                break;
        };

        # we record-route all messages -- to make sure that
        # subsequent messages will go through our proxy; that's
        # particularly good if upstream and downstream entities
        # use different transport protocol
        record_route();
        # loose-route processing
        if (loose_route()) {
                t_relay();
                break;
        };

        # if the request is for other domain use UsrLoc
        # (in case, it does not work, use the following command
        # with proper names and addresses in it)
        if (uri==myself) {

                if (method=="REGISTER") {
                        save("location");
                        break;
                };

                # native SIP destinations are handled using our USRLOC DB
                if (!lookup("location")) {
                        sl_send_reply("404", "Not Found");
                        break;
                };

# der RTP-Proxy

        if (method == "INVITE") {
                if (!dst_ip==192.168.1.22 && uri==myself) {
        log(3, "Weiterleitung an DMZ SER\n");
                        if (force_rtp_proxy("FAII")) {
                                t_on_reply("1");
                        forward("192.168.1.18");
                } else {
                        sl_send_reply("403", "User nicht erreichbar oder unbekannt!");
                        break;
                };
                }
        };

        if (!dst_ip==192.168.1.22) {
        log(3, "Weiterleitung an DMZ SER\n");
```

```
                        if (force_rtp_proxy("FAII")) {
                                t_on_reply("1");
                        forward("192.168.1.18");
                } else {
                        sl_send_reply("403", "User nicht erreichbar oder unbekannt!");
                        break;
                };
                }
        };
        if (method == "BYE" || method == "CANCEL")
                unforce_rtp_proxy();

        # Do strict routing if pre-loaded route headers present
        if (loose_route()) {
        t_relay();
        break;
        };

        #if (method == "INVITE")
        #record_route();

# -- Fehlerhafte Verbindungen ignorieren --
if (!t_relay()) {
sl_reply_error();
};
}

onreply_route[1] {
        if (!(status=~"183" || status=~"200"))
                break;
        force_rtp_proxy("FA");
}
```

14.6 Nessus Scan Report

Nessus Scan Report

This report gives details on hosts that were tested and issuses that were found. Please follow the recommended steps and procedures to eradicate these threats.

Scan Details

Host s which were alive and responding during test	1
Number of security holes found	0
Number of security warnings found	0

Host List

Host(s)	Possible Issuse
192.168.1.22	Security note(s) found

Analysis of Host

Adress of Host	Port/Service	Issue regarding Port
192.168.1.22	general/tcp	Security note(s) found
192.168.1.22	ssh (22/tcp)	Security note(s) found

Security Issues and Fixes: 192.168.1.22

Type	Port	Issue and fix
Informational	general/tcp	TCP split NIDS evasion function is enabled. Some tests might run slowly and you may get some false negative results. Nessus ID : 10889
Informational	general/tcp	TCP fake RST NIDS evasion is enabled. Some tests might run slowly and you may get some false negative results. Nessus ID : 10889
Informational	ssh (22/tcp)	Remote SSH version: SSH-2.0-OpenSSH_3.8.1p1 Debian-8Sarge.sarge.4 Nessus ID : 10267
Informational	ssh (22/tcp)	The rmote SSH deamon supports the following versions of the SSH protocol: .1.99 .2.0 Nessus ID 10881