Sicherheitsarchitekturen
Enterprise Network Security - Defence

Version 3.91

Wintersemester 2018/2019

Fachhochschule Technikum Wien

René Pfeiffer

rene.pfeiffer@technikum-wien.at / pfeiffer@luchs.at

Systemadministrator GNU/Linux® Manages! & Crowes Agency

Inhaltsverzeichnis

1. Einleitung **11**

2. Grundlagen **13**
2.1. Begriffe . . . 13
2.2. Grundkonzeptie von Sicherheitsarchitekturen . . . 14
2.2.1. Bedrohungsanalyse . . . 14
2.2.2. Trennung . . . 14
2.2.3. Inspektion . . . 14
2.2.4. Protokollierung . . . 15
2.3. Die Internetprotokolle . . . 15
2.3.1. Address Resolution Protocol (ARP) . . . 16
2.3.2. Internet Protocol (IPv4 & IPv6) . . . 16
2.3.3. Transmission Control Protocol (TCPv4) . . . 19
2.3.4. User Datagram Protocol (UDPv4) . . . 22
2.3.5. Internet Control Message Protocol (ICMPv4) . . . 23
2.3.6. Internet Control Message Protocol (ICMPv6) . . . 23
2.3.7. Alternative Transportprotokolle . . . 24
2.3.8. Routing und Transport . . . 25
2.3.9. IPv6 . . . 28
2.4. Weitere Layer 2 Protokolle . . . 31
2.5. Namens-/Dienstauflösung . . . 31
2.6. Bedrohungen für vernetzte Systeme . . . 32
2.7. Schutzmechanismen für vernetzte Systeme . . . 34
2.7.1. Verschlüsselung . . . 35
2.7.2. Public Key Infrastructure (PKI) . . . 36
2.7.3. Aufbau einer Certificate Authority . . . 37

3. Arten von Paketfiltern **39**
3.1. Überblick - Wahl der „Waffen” . . . 39
3.2. Statische Paketfilter . . . 39
3.3. Zustandsgesteuerte Paketfilter . . . 40
3.3.1. Nftables Paketfiltersystem . . . 42
3.3.2. BPFilter Paketfiltersystem . . . 42
3.4. Content-basierte Paketfilter . . . 43

4. Aufbau eines Paketfilters **45**
4.1. Sinn und Unsinn von Torwächtern . . . 45
4.2. Unterteilung der Netze durch Filter . . . 45
4.3. Linux® Netfilter im Überblick . . . 46
4.4. Weitere Filterkriterien . . . 48
4.5. Vorgehensweise . . . 50
4.5.1. Default Policies - Grundhaltungen . . . 50
4.5.2. Reihenfolge der Regeln . . . 50
4.5.3. Beispiel eines Paketfilters . . . 51
4.5.4. ICMP . . . 59
4.5.5. Mögliche Schlupflöcher . . . 59
4.5.6. Logging . . . 60
4.5.7. Generelle Filter . . . 61
4.5.8. Filtern von IPv6 . . . 61

5. Testen eines Paketfilters **63**
5.1. Beobachten mit Sniffern . . . 63
5.2. Generieren von Netzwerkverkehr . . . 64
5.2.1. hping . . . 64
5.2.2. isic . . . 64
5.2.3. nmap . . . 65
5.2.4. Seagull . . . 68
5.3. Belastungstests und Auditing . . . 68
5.3.1. Verhalten unter Last . . . 68
5.3.2. Simulation von Angriffen . . . 69
5.3.3. Einsatz von Live-CDs . . . 76

6. Härten von Serversystemen **77**
6.1. Warum Härten? . . . 77
6.2. Checkliste für Betriebssysteme . . . 77
6.2.1. Generelle Vorgehensweise zur Absicherung von Betriebssystemen . . . 78
6.2.2. Basisabsicherung eines Debian Serversystems . . . 78
6.3. Konfigurationsmanagement . . . 81
6.3.1. Konvergenz von Konfigurationszuständen . . . 82
6.3.2. CFEngine . . . 82
6.3.3. Ansible . . . 86
6.4. Checkliste für Applikationen . . . 89
6.4.1. Apache Web Server . . . 89
6.4.2. Webapplikationen . . . 90
6.4.3. MySQL . . . 91
6.4.4. PHP . . . 91
6.4.5. Postfix Mail Transport Agent . . . 94

7. Härten von Netzwerken **95**
7.1. Härten von Netzwerken? . . . 95
7.2. Zugang zu Layer 1 . . . 95
7.3. Zugang zu Layer 2 . . . 95
7.4. Trennen auf Layer 2 . . . 96
7.5. Trennen auf Layer 3 . . . 97
7.5.1. NAT ist keine Sicherheitsmaßnahme! . . . 97
7.5.2. Netzwerkzugang mittels IPsec . . . 98
7.6. Drahtlose Netzwerke . . . 98
7.6.1. Begriffsverwirrung . . . 98
7.6.2. Trennung . . . 99

8. Webapplikationen **101**
8.1. Das W³ und die Systemadministration . . . 101
8.2. Angriffe und Schwachstellen . . . 101
8.2.1. Vorbereitungen . . . 101
8.2.2. Plattformen . . . 102
8.2.3. Authentisierung . . . 102
8.2.4. Autorisierung . . . 103
8.2.5. Input Validation . . . 104
8.2.6. Datenbanken und Storage . . . 106
8.2.7. Web Services . . . 106
8.2.8. Managementzugänge . . . 107
8.2.9. Webclients . . . 107

9. „Cloud“ Infrastruktur **109**
9.1. Definition der „Cloud“ . . . 109
9.2. Virtualisierung . . . 109

9.3. Sicherheit . . . 111
9.3.1. Managementzugang . . . 111
9.3.2. Virtualisierung . . . 111
9.3.3. Minimalisierung der Systeme . . . 111
9.3.4. Speichern von Daten . . . 111

A. Nützliche Kommandos und Werkzeuge **113**
A.1. Layer 2 . . . 113
A.1.1. mii-diag . . . 113
A.1.2. mii-tool . . . 113
A.1.3. ARP Cache . . . 113
A.2. Layer 3/4 . . . 114
A.2.1. netstat . . . 114
A.2.2. tcptraceroute . . . 114
A.2.3. Telnet . . . 115
A.2.4. mtr . . . 115
A.3. Layer 7 . . . 115

B. Konfigurationen **117**
B.1. Makefile für Root CA . . . 117

C. Beispiele aus dem Netz **119**
C.1. Session einer TCP Verbindung . . . 119
C.2. Beispiel für FTP Connection Tracking . . . 126
C.3. Routing Skript . . . 127
C.4. Einsatz und Aufbau von Bastion Hosts . . . 129
C.4.1. Sichern der Maschine . . . 130
C.4.2. Deaktivieren aller unnötigen Services . . . 130
C.4.3. Installieren oder ändern der benötigten Services . . . 130
C.4.4. Rekonfigurieren der Maschine für Production Environment . . . 131
C.4.5. Testen und Prüfen der Sicherheit (Auditing) . . . 131
C.5. nmap Proben . . . 131
C.5.1. Standard TCP connect() Scan mit Version-ID-Patch . . . 131
C.5.2. TCP connect() Scan - mit Portauswahl . . . 132
C.5.3. ICMP Ping Sweep . . . 132
C.5.4. Ansicht einer Linux 2.2.19 Firewall . . . 133
C.5.5. Linux® 2.2.19 Paketfilter von der LAN Seite gesehen . . . 133

D. Protokolle und Ports **135**
D.1. Protokolle . . . 135
D.1.1. Ports . . . 139

Abbildungsverzeichnis

2.1. Internet Protocol (IPv4) Header . . . 17
2.2. Internet Protocol (IPv6) Header . . . 18
2.3. Transmission Control Protocol (TCP) Header . . . 19
2.4. TCP Verbindungsaufbau und Session . . . 21
2.5. User Datagram Protocol (UDP) Header . . . 22
2.6. Internet Control Message Protocol (ICMP) Header . . . 23
2.7. Routing Tabelle eines Gateways . . . 27

3.1. Paketfluß durch eine Linux® ipchains Firewall . . . 40
3.2. Zustandstabelle einer Netfilter Firewall . . . 41

4.1. Paketfilter trennen Internet, DMZ und LAN . . . 45
4.2. Wirkung von „security level" Angaben . . . 46
4.3. Paketfluß durch eine Linux® Netfilter Firewall . . . 47
4.4. Schematischer Aufbau des Linux® Paketfilters . . . 51
4.5. Aktives FTP . . . 55
4.6. Passives FTP . . . 55
4.7. Lücken in der Firewall durch IPv6 und Tunnel . . . 60

5.1. Login mit dem Nessus Client . . . 70
5.2. Auswählen der zu verwendenen Plugins für den Nessus Scan . . . 71
5.3. Voreinstellungen für die verwendeten Nessus Plugin-Module . . . 72
5.4. Optionen für den bevorstehenden Nessus Scan . . . 73
5.5. Ziele für den bevorstehenden Nessus Scan . . . 74
5.6. Scan in Progress . . . 74
5.7. Darstellung der Ergebnisse . . . 75

6.1. Beispielspartitionierung eines Webservers . . . 80
6.2. CFEngine Failsafe Modus . . . 83
6.3. Ansible Multi-Node Workflow . . . 86

7.1. 802.1X NAC Übersicht . . . 96

8.1. CAPTCHA generiert mit Gimp . . . 103
8.2. Prinzipielle Funktionsweise von Web Services . . . 107

9.1. Hypervisor im Native Mode . . . 110
9.2. Hypervisor im Hosted Mode . . . 110

Tabellenverzeichnis

2.1. Die Schichten im OSI Modell 15
2.2. Typen von IPv6 Erweiterungskopfdaten 19
2.3. Codes zur ICMP Typ 3 Meldung `Destination Unreachable` 23
2.4. MTU Werte für verschiedene Netzwerktechniken 28

4.1. Filtereigenschaften des SSH Protokolls 53
4.2. Filtereigenschaften des File Transfer Protokolls 55
4.3. Filtereigenschaften des DNS Protokolls 57

6.1. Typisches Partitionslayout eines Servers 79
6.2. Modernes Partitionslayout eines Servers 79

1. Einleitung

Dieses Skriptum hat eine lange Geschichte hinter sich. Ursprünglich war es das Begleitmaterial für die Vorlesung zum Thema *Firewalls/Paketfilter*. Der Fokus lag damals auf der Netzwerkebene (TCP/IP Layer 2, 3 und 4). Die Applikationsschicht wurde durch Proxysysteme beschützt. Das alles war noch vor der (Wieder)Erfindung der *Cloud*, den Smartphones, vielen Social Media Plattformen und dem Internet der Dinge.

Im Laufe der Jahre kamen Themen wie die Härtung von Applikationen und Systemen hinzu. Schließlich landete man bei der Architektur der Informationstechnologie. Verglichen mit den Anfängen ist heute viel mehr vernetzt. Die drahtlosen Netzwerke sind nahezu überall anzutreffen, in welcher Form auch immer.

Überraschenderweise hat sich an den essentiellen Vorgehensweisen sehr wenig geändert. Die Technologe hat große Sprünge gemacht. Man kann aber implementierte Informationstechnologie nach wie vor mit den Methoden der vergangenen Generationen in Kombination mit neuen Werkzeugen absichern. Es ist oberste Priorität sich nicht von technologischen Fähigkeiten beeindrucken zu lassen. Die IT Sicherheitsexpertin reduziert stetig und versucht die Kommunikation, die Akteure (seien es Menschen oder Maschinen) und die transportierten sowie verabeiteten Daten zu erfassen. Es geht letztlich immer um Zugriffe und Interaktion, egal ob diese blockiert, geregelt oder erlaubt werden soll. Hat man diese Prinzipien begriffen, so kann man beliebige Architekturen absichern.

2. Grundlagen

Die folgenden Unterkapitel sollen einen Überblick über die verwendeten Begriffe und Techniken geben, die in späteren Abschnitten zum Einsatz kommen. Bei Bedarf sind zusätzliche Informationen über die angesprochenen Themen in den zitierten Referenzen im Anhang dieses Dokuments zu finden.

Der Begriff *Sicherheit*, ob mit oder ohne Zusammenhang mit der Informationstechnologie, birgt leider Potential für Mißverständnisse in sich. Sicherheit ist ein relativer Zustand, der sich jederzeit ändern kann. Man muß also immer mit einer zeitlichen Beschränkung rechnen. Bei komplexen Systemen ist es unmöglich Risiken völlig auszuschließen. Es kann also trotz einer sicheren Situation ein Ereignis eintreten, welches Schaden verursacht.

Darüber hinaus kennt man im deutschen Sprachgebrauch nur das Wort Sicherheit. Im angloamerikanischen Sprachgebrauch unterscheidet man zwischen *security* und *safety*. *Safety* meint den Schutz der Umgebung vor einem Objekt. *Security* hingegen meint den Schutz des Objekts vor der Umgebung. Bei der *IT Security* geht es in der Regel um den Schutz digitaler Inhalte jedweder Art vor Umwelteinflüssen und Mißbrauch.

2.1. Begriffe

Viele Begriffe der Informationstechnologie kommen ursprünglich aus der englischen Sprache und werden ohne Übersetzung benutzt. Dieses Skriptum verwendet, wenn immer möglich, die deutschen Begriffe um eine einheitliche Sprache einzuhalten. Damit trotzdem keine Verwechslungen oder Mißdeutungen auftreten, werden hier die wichtigsten Begriffe und Abkürzungen, die in diesem Skript vorkommen, kurz erläutert.

- **Applikation**
  Eine Applikation bezeichnet eine Sammlung von Software, die bestimmte Dienste zur Verfügung stellt oder definierte Aufgaben erfüllt.

- **Client**
  Ein Client ist eine Maschine oder eine Applikation, die sich an einen Server wendet, um einen dort zur Verfügung gestellten Dienst / Service in Anspruch zu nehmen. [1]

- **Dienst / Service**
  Ein Dienst oder Service ist eine Applikation, die Informationen zur weiteren Verarbeitung entgegennimmt oder zur Verfügung stellt. Beispiele für solche Applikationen sind Mailserver, Printserver oder Authentifizierungsserver. Mitunter wird zum Begriff Dienst bzw. Service noch der Bereich angegeben, z.B. Web Service, Windows Service, Proxydienst, etc.

- **Filter**
  Filter ist ein allgemeiner Begriff für eine Inhaltsinspektion. Es ist immer eine Prüfung auf bestimmte Kriterien, die mit einem Ergebnis (z.B. trifft zu, trifft nicht zu) oder einer Aktion (z.B. Transmission wird unterbrochen/forgesetzt) verbunden ist.

- **Firewall**
  Eine Firewall ist eine Sammlung von Maßnahmen, die zum Schutz eines oder mehrerer Netzwerke eingesetzt werden. Elemente einer Firewall trennen vertrauensunwürdige Netzwerke von Netzwerken mit höherem Sicherheitsbedarf. Natürlich kann eine Firewall auch nur aus einem Paketfilter bestehen, wenn die Netzwerktopologie einfach genug ist.

- **Internet / das Internet**
  Das Internet stellt eine riesige Menge aus Netzwerken dar, die miteinander und untereinander verbunden sind (*inter-connected networks*) und über Internetprotokolle Daten austauschen. [2]

[1] Faustregel: Ein Client ist eine Maschine, die niemand vermißt.
[2] Begriffe wie Extranet oder Intranet werden nicht verwendet, da sie keinerlei Aussage über Sicherheitsstandards oder die Vertraulichkeit machen.

- **internet / ein Internet**
  Ein Netzwerk aus zwei oder mehr Maschinen.

- **Mantra**
  Mantras sind Wortfolgen, die man früher in ihrer Kurzform als Paßworte bezeichnet hat. Der Unterschied ist nicht nur semantischer Natur. Paßworte sollten lang genug sein und idealerweise aus mehreren einzelnen Worten bestehen. Man wält daher das Wort Mantra als Bezeichnung, um diesen Punkt zu illustrieren. Im Englischen wird oft stattdessen der Term *passphrase* verwendet.

- **Perimeter**
  Der Perimeter ist der Pfad, der ein Gebiet eingrenzt. Am Perimeter befinden sich Schnittstellen zwischen zwei verschiedenen Gebieten, üblicherweise Netzwerke.

- **Protokoll**
  Als Protokoll bezeichnet man eine Sammlung von Konventionen und Regeln, die eine strukturierte Sprache zum Zweck der Kommunikation verschiedenen Teilnehmer bilden. Dieser Mechanismus ist für das Austauschen von Daten zwischen zwei Punkten unerläßlich.

- **Router**
  Ein Router ist eine Maschine, die eine Verbindungen zwischen verschiedenen Netzwerken darstellt. Router werden daher oft auch als *Gateway* bezeichnet. Die Netzwerke sind meisten auch physikalisch voneinander getrennt.

- **Server**
  Ein Server ist eine Maschine oder eine Applikation, die bestimmte Dienste den Clients im Netzwerk zur Verfügung stellen.

- **Torwächter**
  Torwächter ist kein Fachbegriff, sondern wir in diesem Dokument als Synonym für einen Paketfilter, Proxy oder sonstigen Filter- und Kontrollmechanismus zwischen Netzwerken verwendet.[3]

2.2. Grundkonzeptie von Sicherheitsarchitekturen

Man kann Schutzmaßnahmen in verschiedene Ansätze zerlegen. Die grundlegenden Komponenten sind unabhängig von den eingesetzten Technologien zu sehen. Die technischen Details werden erst in der Implementierung berücksichtigt.

2.2.1. Bedrohungsanalyse

Wer sich zu Tode fürchtet, stirbt auch. Es ist nicht realistisch sich gehen alle Bedrohungs und Gefahren zu wappnen. Bei jedweder Sicherheitsüberlegung muß man sich Klarheit über Risiken, potentielle Schäden und Eintrittswahrscheinlichkeiten verschaffen. Diese Analyse steht am Anfang aller Überlegungen, noch bevor es Implementation gibt.

2.2.2. Trennung

Das Grundkonzept von Sicherheitsarchitekturen ist das Trennen von Bereichen verschiedener Sensitivität und das Einbauen von Schleusen, um den Zugang zu Informationen in Form von Daten zu regeln. Diese Schleusen prüfen die Identität und den Zugriff nach bestimmten Kriterien. Sie können in vielen Formen implementiert werden (Software, Hardware, Prozedur, Mischung, . . .).

Je nach Anforderung und Umfang der abzusichernden Architektur ist eine verschiedene Anzahl von Mechanismen notwendig. Die räumliche und logische Verteilung wird ebenso von den Anordnung der zu schützenden Resourcen bestimmt.

2.2.3. Inspektion

Zugriff auf Daten und transportierte Daten werden inhaltlich überprüft. Dies dient einerseits zum Schutz vor bösartiger Software (Viren, trojanische Pferde), unerwünschten Inhalten und auch Schutz von Daten, die Kontrollpunkte nicht passieren dürfen (sensitive Daten, Schlüssel, . . .).

[3] Sämtliche Ähnlichkeiten mit noch lebenden oder bereits verstorbenen Halbgöttern aus Kinofilmen sind rein zufällig und fast nicht beabsichtigt.

Bei Einsatz von verschlüsselten Protokollen muß darauf geachtet werden, dass eingesetzte Filter die Verschlüsselung terminieren, um Inhalte überprüfen zu können. Der Filter muß dazu zwei verschiedene verschlüsselte Verbindungen unterhalten. Diese Methode funktioniert am besten mit SSL/TLS und einer eigenen Certificate Authority. Dazu müssen alle Clients der eigenen Certificate Authority vertrauen, und der Filter muß mit einem selbst ausgestellten Zertifikat und eigenem Schlüssel ausgestattet sein. Nach außen werden die „normalen“ externen Certificate Authorities und SSL/TLS Verbindungen genutzt.

2.2.4. Protokollierung

Logdaten sind generell für die Überwachung des normalen Betriebs notwendig. Als Sicherheitsmaßnahme verhindern sie zwar keine Regelverletzung oder Kompromittierung, aber sie helfen bei der Verfolgung von Ursachen (forensische Analysen, Erkennen von Anomalien). Ab einer gewissen Größe der Infrastruktur ist es sinnvoll die Logdaten zentral aufzubewahren und zu analysieren.

Die größte Herausforderung bei der Auswertung von Logdaten ist die fehlende Standardisierung der Formate und der Meldungen. Es gibt zwar meistens Klassifikationen für Dringlichkeitsgrade (wie z.B. *critical*, *error*, *warning*, *notice*, ...), allerdings werden diese von den Applikationen selbst eingestuft. Es gilt für eine sinnvolle Bewertung der Logmeldungen herauszufinden welche Klassifikationen pro Applikation relevant für eine Überwachung der Sicherheit sind.

2.3. Die Internetprotokolle

Das Internetprotokoll (IP) [1] wurde von einer Gruppe von Entwicklern der *Defense Advanced Research Projects Agency* (DARPA) erschaffen, die an der Entstehung des ARPAnets beteiligt waren. Das Internetprotokoll wurde mit dem Ziel entwickelt, mehreren Computersystemen das Teilen von Resourcen zu ermöglichen. Beim IP handelt es sich eigentlich um eine ganze Gruppe von Protokollen, wobei durch den Bekanntheitsgrad des Transmission Control Protocols (TCP) und des Begriffs IP das Synonym TCP/IP Verbreitung gefunden hat. Einen Überblick der Internetprotokolle läßt sich in RFC 2600 nachlesen. [2]

Der Fluß von Daten durch Netzwerke wird in bestimmte Ebenen unterteilt, die üblicherweise nach dem *Open Systems Interconnection* (OSI) Modell benannt sind.

7	Applikationsschicht	Benutzerschnittstelle zu Netzwerkdiensten (Dateitransfer, Datenbankzugriff, Terminal, Browser)
6	Darstellungsschicht	Übersetzt Datenformate (z.B. durch Kompression, Umleitungs, etc.)
5	Sitzungsschicht	Etabliert, unterhält und unterbricht eine Datenverbindung; Zwei-Wege-Kommunikation
4	Transportsschicht	Kontrolle des Datenflusses; stellt die sichere Übertragung von Daten ohne Verluste bzw. Duplikate sicher (z.B. TCP)
3	Vermittlungsschicht	Umsetzung von logischen zu physischen Adressen; Finden einer Route zwischen zwei Endpunkten (z.B. IP)
2	Sicherungsschicht	Konvertierung von Paketen in binäre Signale und umgekehrt, Fehlerprüfung (z.B. Netzwerkkarte)
1	Übertragungsschicht	Transport der Daten (z.B. Verkabelung, elektronische Signale)

Tabelle 2.1.: Diese Tabelle zeigt die einzelnen Schichten im OSI Modell. Ebene 7 bezeichnet die Schicht, die am nächsten an den Applikationen ist. Die unterste Ebene kennzeichnet die physische Übertragung der Daten über das Netzwerk.

Den einzelnen Schichten sind in der Regel Protokolle zugeordnet, wodurch erkenntlich wird auf welche Weise die einzelnen Protokolle aufeinander aufgebaut sind und mit welchen Daten die Protokolle direkt zu tun haben. Üblicherweise werden nicht alle 7 Schichten des OSI Modells benötigt (und sind auch nicht implementiert). In den meisten Fällen sind die folgenden Schichten völlig ausreichend.

- Applikationsschicht
- Transportschicht

- Vermittlungsschicht
- Übertragungsschicht

Insbesondere das nun zu diskutierende Internetprotokoll verwendet nur diese vier Schichten. Protokolle, die Applikationen verwenden, können diese vier Schichten ergänzen.

2.3.1. Address Resolution Protocol (ARP)

Zwischen der physikalischen Netzwerkschicht (Leitungs-/Funkprotokoll, Netzwerkgeräte) und der logischen Adressierung liegt das sogenannte *Address Resolution Protocol* (ARP). Es schafft eine Bindung zwischen dem physischen Netzwerkanschluß und den logischen Adressen. Bei Ethernet oder drahtlosen Übertragungen besitzt jede Netzwerkkarte eine Identifikation, die Media Access Control (MAC) Adresse. Diese MAC Adresse hat eine Länge von 48 Bit. Die ersten 24 Bit stellen den Herstellercode dar, die restlichen sind eine Identifikationsnummer. In einem lokalen Netzwerksegment müssen diese Adressen eindeutig sein.

Die Verbindung zur übergeordneten Schicht geschieht durch Austausch der MAC und IP Adressen, die miteinander kommunizieren wollen. Der Sender schickt vor der eigentlichen Datentransmission an alle lokal angeschlossenen Maschinen die Frage „Wer hat die IP Adresse `aa.bb.cc.dd`?". Man erreicht alle lokal angeschlossenen Maschinen über die Broadcast MAC Adresse `ff:ff:ff:ff:ff:ff`. Sofern die Zielmaschine angeschlossen ist, antwortet sie.

```
00:11:09:8b:43:54 -> ff:ff:ff:ff:ff:ff ARP Who has 192.168.45.9?  Tell 192.168.45.42
00:90:27:98:41:bd -> 00:11:09:8b:43:54 ARP 192.168.15.9 is at 00:90:27:98:41:bd
```

Antworten werden betriebssystemspezifisch einen gewissen Zeitraum zwischengespeichert. Man muß bei ARP unbedingt beachten, daß die erste Antwort „gewinnt", aber diese muß natürlich nicht korrekt sein. Angreifer in einem lokalen Netzwerksegment können stetig ARP Antworten senden und so die Datentransmissionen umleiten (*ARP Flooding*). Es gibt mehrere Attacken dieser Art (*ARP Poison Routing*, *ARP poisoning*). Gegenmaßnahmen umfassen das Eintragen statischer MAC Adressen oder Beobachten der Adreßkopplungen. Letzteres wird durch die Werkzeuge ArpON[4] oder arpwatch[5] durchgeführt. Mitunter kann man auch die Datenbank eines DHCP Servers nutzen und die MAC Adressen überwachen und vergleichen. Diese Methoden trägt den Begriff DHCP Snooping. Es gibt Switches, die diese Technologie verwenden.

2.3.2. Internet Protocol (IPv4 & IPv6)

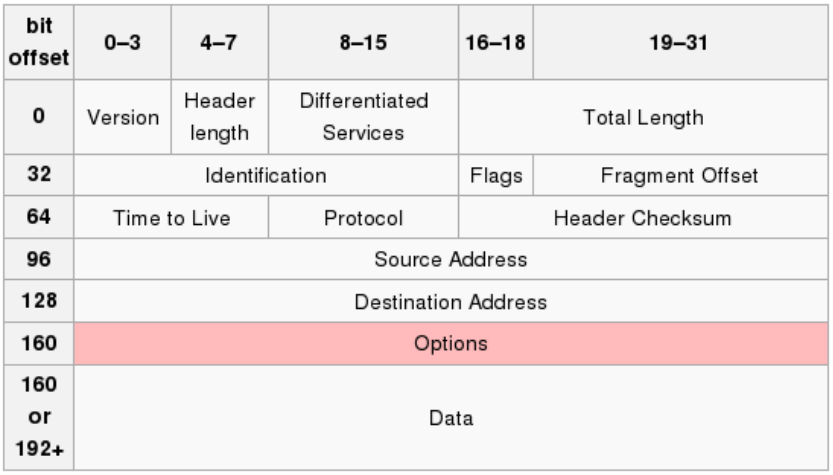
Das Internet Protocol (IP) ist die Basis der Internetprotokolle, auch wenn es komisch klingen mag. IP ist ein paketorientiertes und verbindungsloses Protokoll. Es gehört zur dritten (und vierten) Schicht des OSI Modells. Jedes Datenpaket wird unabhängig betrachtet und als solches transportiert. Es gibt keinen Mechanismus, der das Ankommen eines Paketes garantiert oder überprüft. Es wird jedoch versucht das Paket so gut wie möglich zum Ziel zu bringen, auch wenn der kürzeste Weg nicht zur Verfügung steht. Das ist eine wichtige Designgrundlage von IP.

IP Pakete bestehen aus zwei Teilen. Der erste Teil ist der *Header* oder *Kopf* des Paketes. Dort befinden sich die Informationen über Quelle, Ziel und Optionen, die man dem Paket mitgeben kann. Die eigentlichen Daten werden im Datenteil transportiert. Die maximale Größe eines Paketes sind 65536 Byte (das Längenfeld hat 16 Bit ohne Vorzeichen). Der Kopf besteht aus einem 20-Byte Teil und einem Stück mit variabler Länge, wo verschiedene IP Optionen untergebracht werden können. Der Rest steht den Daten zur Verfügung. Jedes einzelne IP Paket hat die in Abbildung 2.1 dargestellte Form.

Jede Zeile besteht aus einer 32 Bit Zahl.

- IP Version (4 Bit)
  Dieses Feld enthält die Version des Protokolls. Derzeit wird IP Version 4 (IPv4) eingesetzt. Die Nachfolgeversion IPv6 existiert schon und ist bereits in Verwendung.
- Header Länge (4 Bit)
  Dieses Feld gibt die Länge des IP Headers in Vielfachen von 32 Bit an. Die minimale Größe eines korrekten IP Headers sind 5 32 Bit Werte.
- Type of Service (ToS, 8 Bit)
  Der Type of Service ist eine Angabe wie die Daten des Paketes von Routern auf dem Weg zu handhaben sind. Mit dem ToS Wert lassen sich Übertragungseigenschaften der übertragenen Daten bestimmen, sofern die Knotenpunkte, an denen das IP Paket weitergeleitet wird, den ToS auslesen.

[4] http://arpon.sourceforge.net/
[5] http://www-nrg.ee.lbl.gov/



bit offset	0–3	4–7	8–15	16–18	19–31
0	Version	Header length	Differentiated Services	Total Length	
32	Identification			Flags	Fragment Offset
64	Time to Live		Protocol	Header Checksum	
96	Source Address				
128	Destination Address				
160	Options				
160 or 192+	Data				

Abbildung 2.1.: Der Aufbau eines Paket Headers im Internet Protocol (IP). Die minimale Länge beträgt 20 Byte ohne Daten. (Quelle Wikipedia)

- Länge des Paketes (16 Bit)
  Hier steht die Gesamtlänge des Pakets inklusive Header und Daten. Gemessen wird das Paket in Oktets (Bytes).
- Identifikation (16 Bit)
  Dies ist eine Identifikationsnummer für die sogenannte Paketfragmentierung.
- Control Flags (3 Bit)
  Diese Werte bestimmen Eigenschaften des Pakets bei Paketfragmentierung.

Bit 0		reserviert, muß 0 sein
Bit 1 (DF)	0 Fragmentierung möglich	1 nicht fragmentieren
Bit 2 (MF)	0 letztes Fragment	1 Fragmente folgen

  DF steht für *Don't Fragment*, MF steht für *More Fragments*.
- Time To Live (TTL, 8 Bit)
  Das ist die Lebensdauer das Paketes gemessen in Sprüngen von Router zu Router (Hops). Sobald ein Paket eine Lebensdauer von Null hat, wird es verworfen und „stirbt“.
- Protokoll (8 Bit)
  Hier wird das Protokoll benannt zu welchem die Daten im Datenteil des IP Paketes gehören, z.B. 6 für TCP, 17 für UDP, etc. Eine Übersicht über die vergebenen Nummern der Protokolle findet sich in RFC 790. [3]
- Checksumme (16 Bit)
  Dies ist die Checksumme des IP Headers. Da sich beispielsweise die TTL des Paketes dauernd ändert, muß diese stets neu berechnet werden.
- Quell- und Zieladresse (je 32 Bit)
  Die Adressen im Internet Protocol bestehen aus einer 32-Bit Zahl, der sogenannten IP Adresse. Jedes Datenpaket hat eine Quelladresse und eine Zieladresse. IP Adressen werden als ein Tupel von 4 Zahlen dargestellt, z.B. `192.168.10.3` oder `195.230.42.195`. Es ist zulässig Nullwerte auszulassen (`127.1` entspricht `127.0.0.1`).

IP Pakete werden über das Netzwerk von der Quelladresse zur Zieladresse transportiert. Der Weg kann dabei über mehrere Gateways führen, wobei an jedem die Lebensdauer (TTL) um eine Einheit verringert wird. Jeder Router, der ein Paket mit einer TTL von 1 erhält, wird dieses Paket nicht mehr weitergeben, eine diesbezügliche Meldung an den Sender des Paketes generieren und zurückschicken. Das Paket und seine Daten verschwinden damit aus dem Netzwerk. Diese Maßnahme ist notwendig, um Schleifen zu vermeiden. IP kennt keinen Mechanismus, der eine Datenübertragung absichert (verlorene

Pakete werden nicht aufgehoben). Zusätzlich fehlen Informationen, die die Daten an eine bestimmte „Hausnummer“ an der Zieladresse liefern (Konzept des *Multiplexing* und *Demultiplexing*). Diese beiden Umstände werden mit weiteren Protokollen, die auf IP aufsetzen, angesprochen.

Da IPv6 zunehmend Verbreitung findet[6], ist in Abbildung **??** auch der Kopf eines IPv6 Pakets abgebildet. Er unterscheidet sich von einem IPv4 Kopf in wesentlichen Punkten und ist einfacher aufgebaut.

Abbildung 2.2.: Der Aufbau eines Paket Headers im Internet Protocol (IPv6). Die Länge beträgt immer 40 Byte ohne Daten. Der Kopf kann durch sogenannte *Extension Headers* noch erweitert werden. Alle Extension Headers haben eine durch 8 teilbare Länge in Byte. (Quelle Wikipedia)

- IP Version (4 Bit)
  Hier steht der Wert 6 als Markierung für IPv6.
- Traffic Class (8 Bit)
  Dieses Feld markiert die Art des Pakets und kann zur Priorisierung verwendet werden.
- Flow Label (20 Bit)
  Dieses Feld dient ebenfalls einer Priorisierung und kann für Quality of Service (QoS) verwendet werden. Alle Pakete desselben Flows und derselben Sendeadresse werden gleich behandelt. Ein Flow Label von 0 zeigt an, daß die Pakete beliebig behandelt werden können.
- Payload Length (16 Bit)
  Stellt Länge des IPv6-Paketinhaltes (ohne Kopfdatenbereich, aber inklusive der Erweiterungskopfdaten) in Byte dar.
- Next Header (8 Bit)
  Identifiziert den Typ des nächsten Kopfdatenbereiches, welcher entweder Erweiterungskopfdaten ankündigt oder ein Protokoll der nächsthöheren Schicht beschreibt.
- Hop Limit (8 Bit)
  Maximale Zahl von „Hops”, die ein Paket beim Transport durchlaufen darf. Das Feld entspricht dem TTL Feld bei IPv4.
- Source Address (128 Bit) - Absendeadresse.
- Destination Address (128 Bit) - Empfängeradresse.

Die IPv6 Adressen haben einen anderen Aufbau als ihre IPv4 Vorgänger. Die 128 Bit sind in folgende Bereiche eingeteilt (für Unicast).

1. Global Routing Prefix (n Bit, maximal 48 Bit) - verwendet für Routing
2. Subnetz ID (m Bit, maximal 16 Bit) - verwendet für Routing

[6]Die Spezifikation von IPv6 ist älter als 20 Jahre. Die noch freien IPv4 Pools sind ausgeschöpft, daher sollte IPv6 Teil jeder Netzwerkplanung sein.

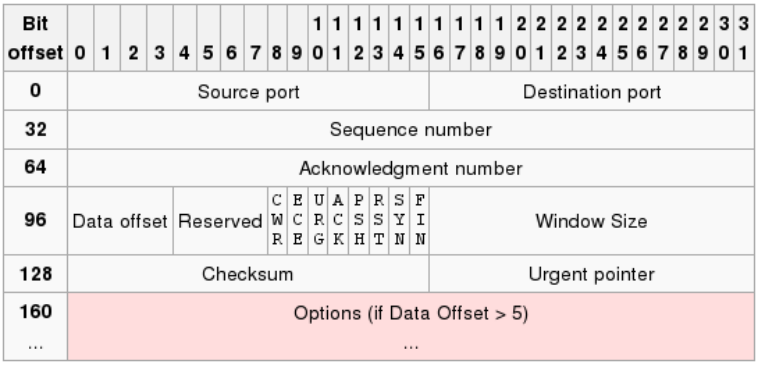


Abbildung 2.3.: Aufbau eines Paket Headers im Transmission Control Protocol (TCP). Die minimale Länge beträgt 20 Byte ohne Daten. Der TCP Header ist als Zusatz zum IP Header zu sehen. (Quelle Wikipedia)

3. Interface ID ($128 - n - m$ Bit, maximal 64 Bit) - verwendet für Host Identifikation; in diese fließt auch die MAC Adresse des Netzwerkgeräts ein

Mögliche Optionen werden in zusätzlichen IPv6 Kopfdaten untergebracht, die aneinander verkettet werden können (durch sogenannte *Extension Headers*). Das Feld *Next Header* ist dafür zuständig. Tabelle 2.2 zeigt mögliche Erweiterungen, die bisher ins RFCs definiert wurden. Der Vorteil dieser Verkettung besteht in der festen Größe eines jeden IPv6 Kopfes (40 Byte). Die vereinfacht die Arbeit von Routern und Filtern stark. Auch die Verarbeitungsgeschwindigkeit steigt (speziell da die Prüfsummen ebenso wegfallen und nicht ständig neu berechnet werden müssen).

Name	Typ	Größe	Beschreibung
Hop-By-Hop Options	0	variabel	Optionen für alle Geräte entlang des Pfads (z.B. für Jumbogramme)
Routing	43	variabel	Routinginformationen für alle Router entlang des Pfads
Fragment	44	64 Bit	Parameter einer Fragmentierung
Authentication Header (AH)	51	variabel	IPsec Integritätssicherung
Encapsulating Security Payload (ESP)	50	variabel	IPsec Verschlüsselung
Destination Options	60	variabel	Informationen für den Empfänger
No Next Header	59	0	Markierung für Ende der Erweiterungen

Tabelle 2.2.: Diese Tabelle zeigt die Erweiterung für verkettete IPv6 Kopfdaten an.

2.3.3. Transmission Control Protocol (TCPv4)

Das *Transmission Control Protocol* (TCP) [4] bietet eine verbindungsorientierte Datenübertragung mit Fehlererkennung und -korrektur. TCP ist als Aufsatz auf das Internet Protocol zu verstehen. Jedes TCP Paket ist ein IP Paket mit Zusatzinformationen, die im TCP Header abgelegt sind. Abbildung 2.3 beschreibt den Aufbau des TCP Headers.

- Quell- und Zielport (je 16 Bit)
  TCP kann Daten nicht nur an eine IP-Adresse liefern, sondern ist auch in der Lage zwischen einzelnen Ports zu differenzieren. Damit ist es möglich mehreren netzwerkfähigen Applikationen auf einer Maschine die vernetzte Kommunikation zur Verfügung zu stellen.

- Sequenznummer (SEQ, 32 Bit)
  Da TCP ein verbindungsorientiertes Protokoll ist, muß es einen Mechanismus geben, der die übertragenen Daten

organisiert und Paketverluste feststellt. Die Sequenznummer dient zum Aufbau einer Verbindung durch Austausch der Initial Sequence Number (ISN) beim Verbindungsaufbau. In Folge gibt die Sequenznummer die Ordnung der Datenpakete vor.

- Acknowledgement Number (ACK, 32 Bit)
  Wenn das ACK Kontrollbit gesetzt ist, dann enthält dieses Feld die nächste SEQ Nummer, die der Absender erwartet. Details zu SEQ und ACK Nummern werden beim TCP Verbindungsaufbau beschrieben.

- Datenoffset (4 Bits)
  Ähnlich wie beim IP Header enthält dieses Feld die Anzahl der 32 Bit Worte des TCP Headers.

- reservierte Bits (4 Bit)

- TCP Kontrollbits (8 Bit)
  Die Kontrollbits steuern den Verbindungsaufbau und die Datenübertragung.

Congestion Window Reduced	CWR	für ECN Mechanismus
ECN-Echo	ECN	für ECN Mechanismus
Urgent	URG	Urgent Pointer Feld beachten
Acknowledgement	ACK	Acknowledgement Nummer beachten
Push	PSH	Push Funktion
Reset	RST	Verbindung zurücksetzen
Synchronize	SYN	Sequenznummer synchronisieren
Finish	FIN	Sender hat keine Daten mehr

  Der *Explicit Congestion Notification* (ECN) Mechanismus [5] ist nicht auf älteren netzwerktauglichen Geräten implementiert. Manche ältere Firewallsysteme blockieren Pakete mit ECN Erweiterungen, weil sie sie für Netzwerkproben halten. Mittlerweile kann man ECN jedoch durchaus auf Servern und Clients aktivieren.[7]

- TCP Window (16 Bit)
  Gibt die Anzahl der Bytes an, die der Sender dieses Pakets gewillt ist zu empfangen. Spätere Erweiterungen setzen die TCP Window Größe auch durch TCP Optionen, um die Begrenzung auf eine 16-Bit-Zahl zu umgehen. Fenstergrößen von mehreren Megabyte sind damit möglich (nützlich für Satellitenverbindungen und WAN-Strecken mit ähnlichen Eigenschaften).

- Checksumme (16 Bits)
  Dieses Feld enthält die Checksumme des ganzen Pakets inklusive Daten.

- Urgent Pointer (16 Bit)
  Mit Hilfe dieses Feldes lassen sich bestimmte Pakete gesondert als wichtig markieren (in Zusammenhang mit gesetztem URG Kontrollbit).

Eine TCP Verbindung kommt durch den Austausch dreier Pakete zustande.

1. **SYN Paket**
   Client sendet ein Paket mit gesetztem SYN Kontrollbit und Initial Sequence Number (ISN) im Sequenznummernfeld.

2. **SYN+ACK Paket**
   Server erhält das Paket und sendet darauf hin ein TCP Paket mit gesetzten SYN und ACK Kontrollbit zurück. Die Acknowledgement Nummer wird auf ISN+1 gesetzt, der Server wählt eine eigene ISN und gibt sie ebenfalls in das Sequenznummernfeld.

3. Der Client reagiert mit einem gesetzten ACK Kontrollbit und der Server-ISN+1.

Durch diese drei Pakete wird die Verbindung aufgebaut. Man nennt diesen Austausch auch den sogenannten „TCP Handshake“. Nach diesem Prozeß können Daten übertragen werden. Ab dem 3. Schritt sind in allen Paketen die ACK Bits gesetzt, welches diese Datensegmente als Teil einer bereits etablierten Verbindung ausweist. In Abbildung 2.4 ist dieser

[7] ECN sorgt im Falle von „Netzwerkverstopfungen” dafür, daß Sender und Empfänger die Transferrate anpassen können. ECN muß dazu von beiden Endpunkten verstanden werden.

Vorgang anhand eines Beispiels dargestellt. Dort besitzt das erste Paket eine SYN, ECN und CWR Kombination, weil eine der beteiligten Maschinen ECN-fähig[8] ist.

```
Source            Destination        Protocol Info
seven.example.net    finagle.example.net   TCP      38774 > smtp [SYN, ECN, CWR] Seq=633500261 Ack=0 Win=5840 Len=0
finagle.example.net  seven.example.net     TCP      smtp > 38774 [SYN, ACK] Seq=765511189 Ack=633500262 Win=5840 Len=0
seven.example.net    finagle.example.net   TCP      38774 > smtp [ACK] Seq=633500262 Ack=765511190 Win=5840 Len=0
finagle.example.net  seven.example.net     SMTP     Response: 220 Leave ESMTP here
seven.example.net    finagle.example.net   TCP      38774 > smtp [ACK] Seq=633500262 Ack=765511212 Win=5840 Len=0
seven.example.net    finagle.example.net   SMTP     Command: QUIT
finagle.example.net  seven.example.net     TCP      smtp > 38774 [ACK] Seq=765511212 Ack=633500268 Win=5840 Len=0
finagle.example.net  seven.example.net     SMTP     Response: 221 2.0.0 finagle.example.net closing connection
seven.example.net    finagle.example.net   TCP      38774 > smtp [ACK] Seq=633500268 Ack=765511260 Win=5840 Len=0
finagle.example.net  seven.example.net     TCP      smtp > 38774 [FIN, ACK] Seq=765511260 Ack=633500268 Win=5840 Len=0
seven.example.net    finagle.example.net   TCP      38774 > smtp [FIN, ACK] Seq=633500268 Ack=765511261 Win=5840 Len=0
finagle.example.net  seven.example.net     TCP      smtp > 38774 [ACK] Seq=765511261 Ack=633500269 Win=5840 Len=0
```

Abbildung 2.4.: Aufbau und Verlauf einer TCP Session zwischen zwei Linux® Maschinen, aufgezeichnet mit `tcpdump`. [6] Die ersten drei Pakete dienen zum Aufbau der Verbindung. Im Anschluß daran werden sogleich die Daten des übergeordneten Protokolls übertragen (in diesem Fall SMTP). Diese Session ist im Anhang detailiert dargestellt.

Man darf dabei nie vergessen, daß der Status der Verbindung (offen, geschlossen, wird aufgebaut, wird abgebaut) lediglich nach Austausch bestimmter Informationen am Server und am Client notiert wird. Der Datentransfer geschieht weiterhin über das verbindungslose IP. Nur der logische Aufsatz simuliert Verbindungen.

TCP schaut einfach aus, hat aber genügend Parameter, um verschiedene Implementationen für ganz bestimmte Netzwerkbedingungen zu ermöglichen. Der Linux® Kern ab der Version 2.6.x unterstützt verschiedene TCP Modelle zur Vermeidung von Netzwerkverstopfungen als Module. Hier sind einige der alternativen Modelle als Beispiel angeführt. Einige dienen speziell zur Optimierung von Datenübertragungen auf Verbindungen hoher Bandbreite und hoher Latenz (sogenannte „fat pipes", wie sie bei Satellitenverbindungen oder Gigabitstrecken über WAN auftreten).

- DataCenter TCP congestion control [7] [8]
- HighSpeed TCP[9]
- Scalable TCP[10]
- TCP BBR[11] (Bottleneck Bandwidth and RTT)[12] [9]
- TCP BIC[13]
- TCP CDG (CAIA Delay-Gradient) [10]
- TCP CUBIC[14]
- TCP Illinois[15]
- TCP Low Priority[16]
- TCP New Vegas[17] [11]
- TCP Reno[18]

[8] Explicit Congestion Notification (ECN) ist eine Erweiterung von TCP, die bei „Verstopfungen"in Netzwerken für eine bessere Datenübertragung sorgt. ECN ist in RFC 2481 und 3168 beschrieben.
[9] http://www.icir.org/floyd/hstcp.html
[10] http://www.deneholme.net/tom/scalable/
[11] https://github.com/google/bbr
[12] Bei Verwendung von BBR muß man die Queuing Discipline auf Fair Queuing (FQ) with Controlled Delay (CoDel) umstellen.
[13] http://netsrv.csc.ncsu.edu/twiki/bin/view/Main/BIC
[14] http://netsrv.csc.ncsu.edu/twiki/bin/view/Main/BIC
[15] http://en.wikipedia.org/wiki/TCP-Illinois
[16] http://www.ece.rice.edu/networks/TCP-LP/
[17] https://dl.acm.org/citation.cfm?id=1098296
[18] http://en.wikipedia.org/wiki/TCP_congestion_avoidance_algorithm

- TCP Vegas[19]
- TCP Veno[20]
- TCP Westwood / Westwood+[21]
- TCP YeAH

Das Kommando `cat /proc/sys/net/ipv4/tcp_congestion_control` zeigt das aktuell aktive Modell an. Durch Hineinschreiben in die Pseudodatei

```
/proc/sys/net/ipv4/tcp_congestion_control
```

kann man das Modell während dem Betrieb wechseln. Es empfiehlt sich vorher noch den Metrik-Cache auszuschalten, damit der Kern dynamisch auf Netzwerksituationen reagieren kann. Dies läßt sich durch Schreiben von `0` in

```
/proc/sys/net/ipv4/tcp_no_metrics_save
```

erreichen. Die Konfigurationen lassen sich in der Datei `/etc/sysctl.conf` dauerhaft konfigurieren.

Möchte man TCP BBR (Bottleneck Bandwidth and RTT) verwenden, so muß man die Queuing Discipline anpassen. Minimal notwendig dafür ist das folgende Kommando:

```
tc qdisc add dev eth0 root fq_codel
```

2.3.4. User Datagram Protocol (UDPv4)

Das *User Datagram Protocol* (UDP) ist ein verbindungsloses Protokoll. Es vermag ebenso wie das IP ein Paket von Quelle zu Ziel zu bringen, wobei allerdings die unter dem TCP eingeführten Port für Quell- und Zieladresse gesetzt werden können. Es baut ebenso wie TCP auf IP auf und besitzt auch eine Integritätsprüfung der Daten, ist aber wesentlich einfacher aufgebaut.

bits	0 - 15	16 - 31
0	Source Port	Destination Port
32	Length	Checksum
64	Data	

Abbildung 2.5.: Aufbau eines Paket Headers im User Datagram Protocol (UDP). Die Länge beträgt 8 Byte ohne Daten. Der UDP Header ist als Zusatz zum IP Header zu sehen. (Quelle Wikipedia)

Abbildung 2.5 stellt den Header dar. Die Felder sind alle schon bei TCP beschrieben. UDP wird hauptsächlich von Protokollen eingesetzt, die eine schnelle Übertragung von kleinen Informationspaketen wünschen. Der Domain Name Service (DNS) zur Namensauflösung ist ein prominentes Beispiel. Auch viele Game Server nutzen UDP, genauso wie höhere Protokolle zu Audio- oder Videostreaming.

[19] http://www.opalsoft.net/qos/TCP-40.htm
[20] http://www.ntu.edu.sg/home/ascpfu/veno/veno.html
[21] http://www.cs.ucla.edu/NRL/hpi/tcpw/

0	Net Unreachable
1	Host Unreachable
2	Protocol Unreachable
3	Port Unreachable
4	Fragmentation Needed and Don't Fragment was Set
5	Source Route Failed
6	Destination Network Unknown
7	Destination Host Unknown
8	Source Host Isolated
9	Communication with Destination Network is Administratively Prohibited
10	Communication with Destination Host is Administratively Prohibited
11	Destination Network Unreachable for Type of Service
12	Destination Host Unreachable for Type of Service
13	Communication Administratively Prohibited [14]
14	Host Precedence Violation [14]
15	Precedence cutoff in effect [14]

Tabelle 2.3.: Hier sind alle möglichen Codes zur ICMP Meldung Typ 3 `Destination Unreachable` aufgelistet.

2.3.5. Internet Control Message Protocol (ICMPv4)

Das *Internet Control Message Protocol* (ICMP) [12, 13, 14] ermöglicht es, Routern oder Zielmaschinen den Sender eines Datenpaketes über Fehler beim Transport zu informieren. Darüber hinaus bietet ICMP die Möglichkeit mit Hilfe von Anfragen und Auskünften den Zustand eines Netzwerkes zu ermitteln und zu beschreiben. Ein ICMP Paket transportiert daher nur Statusmeldungen, besitzt jedoch ebenso wie TCP und UDP einen Header- und einen Datenteil (siehe Abbildung 2.6).

Bits	160-167	168-175	176-183	184-191
160	Type	Code	Checksum	
192	ID		Sequence	

Abbildung 2.6.: Aufbau eines Paket Headers im Internet Control Message Protocol (ICMP). Die Länge beträgt 8 Byte ohne Daten. Der ICMP Header ist als Zusatz zum IP Header zu sehen. (Quelle Wikipedia)

ICMP Meldungen sind nach Typ und Code eingeteilt. Ein Beispiel seien die Meldungen der Familie `Destination Unreachable` (Typ 3). Die einzelnen Codes sind in Tabelle 2.3 aufgelistet. Der Fehlertyp wird durch die einzelnen Fehlercodes genauer spezifiziert.

2.3.6. Internet Control Message Protocol (ICMPv6)

Bei IPv6 wird ICMP für viel mehr verwendet und ist integraler Bestandteil der Netzwerkkonfiguration. Wichtige Aufgaben sind diese:

- Neighbor Discovery Protocol (NDP)[22]
  - Router Solicitation (Typ 133) - Hosts suchen mit diesen Meldungen nach lokalen Routern.

[22] NDP übernimmt die Aufgaben von ARP.

  - Router Advertisement (Typ 134) - Router kündigen ihre Präsenz in lokalen Netzwerken an.
  - Neighbor Solicitation (Typ 135) - Hosts bestimmen damit die Link Layer Adresse ihrer Nachbarn und bestimmen deren Erreichbarkeit.
  - Neighbor Advertisement (Typ 136) - Host antworten damit auf Neighbor Solicitation Nachrichten.
  - Redirect (Typ 137) - Routern informieren mit diesen Meldungen Hosts über einen besseren *first hop*.
- SEcure Neighbor Discovery (SEND) [15] - NDP mit Zertifikaten ohne IPsec Konfiguration.
- Fehlermeldungen
  - Destination Unreachable (Typ 1)
  - Packet Too Big (Typ 2) - IP Fragmentierung wird nur an den Endpunkten durchgeführt. Router fragmentieren keine Pakete. Router melden nur, wenn ein Paket nicht durch den Pfad paßt.
  - Time Exceeded (Typ 3)
  - Parameter Problem (Typ 4)
- Multicast Registrierung / Entdeckung

Die ICMPv6 Meldungen von NDP werden auch für StateLess Address AutoConfiguration (SLAAC) verwendet. [16] Für Filter ist in RFC 4890 (Recommendations for Filtering ICMPv6 Messages in Firewalls) ein Vorschlag speziell für ICMPv6 vorgestellt. [17]

2.3.7. Alternative Transportprotokolle

TCP und UDP sind weit verbreitet. Es gibt mittlerweile allerdings Protokolle, die ähnliche Aufgaben mit anderen Eigenschaften erfüllen.

- Datagram Congestion Control Protocol (DCCP)
  Es wurde in RFC 4340 definiert und beschreibt bidirektionalen Datentransport mit nicht bestätigten Paketen (analog zu UDP). DCCP enthält jedoch als Zusatz Mechanismen, um Verzögerungen auf dem Transmissionsweg zu detektieren.
- Quick UDP Internet Connections (QUIC)
  QUIC ist ein experimentelles Netzwerkprotokoll, entwickelt von Jim Roskind (Google). Es hat zum Ziel gemultiplexte Verbindungen zwischen zwei Endpunkten zu realisieren. SSL/TLS Verschlüsselung ist Teil der Spezifikation. QUIC befindet sich auf dem Weg zur Standardisierung und wird von Applikationen wie Google Chrome oder DNSCrypt verwendet. [18]
- Real-time Transport Protocol (RTP)
  RTP wird oft im Multimediabereich eingesetzt, da es sich sehr gut für die Übertragung von audiovisuellen Streams eignet. [19]
- Stream Control Transmission Protocol (SCTP)
  SCTP wurde in RFC 2960 vorgeschlagen. Es setzt auf IP auf und bietet
  - bestätigte Datentransmissionen ohne Fehler und Duplikate,
  - Fragmentierung als Reaktion auf verschiedene MTU Werte,
  - sequentiellen Nachrichtentransport innerhalb mehrerer Datenströme,
  - Bündeln mehrerer Nachrichten in ein einzelnes Paket und
  - schnelle Konvergenz bei Netzwerkfehlern.

  SCTP ist explizit für Failover-Situationen und Gigabit Ethernet entworfen.
- Transparent Inter Process Communication (TIPC)
  für Interprozeßkommunikation innerhalb von Clustern.

Die Aufzählung ist nicht vollständig. Es gibt Protokolle, die hier nicht aufgeführt sind. Für das Implementieren von Filtern oder Proxies muß man im Zweifelsfalle die Spezifikationen analysieren.

2.3.8. Routing und Transport

Im folgenden sollen nun kurz einige Punkte erläutert werden, die den Transport von Paketen zwischen Netzwerken betreffen und für die folgenden Kapitel wichtig sind. Für eine detailierte Darstellung sei an dieser Stelle auf weiterführende Literatur verwiesen. [24, 25, 26, 27, 28]

IP Adressen und Netzwerke

Eine IPv4 Adresse kann auf mehrere Arten notiert werden. Üblicherweise gibt man Adressen mit ihrer sogenannten *Netzwerkmaske* an, die die Größe des lokalen Adreßpools angibt. Beispielsweise notiert man das Netzwerk `192.168.10.0` mit den Maschinen `192.168.10.1` bis `192.168.10.254` wie folgt:

1. `192.168.10.0/255.255.255.0`
2. `192.168.10.0/24`
3. `192.168.10.0/0b11111111111111111111111100000000`

Schreibweise 1 entspricht der alten Notation nach sogenannten IP Klassen, die nicht mehr verwendet werden. Schreibweise 2 ist nach der vor Jahren eingeführten *Classless Inter-Domain Routing* (CIDR) Notation [30], die die Anzahl der binären 1 Werte in der Netzmaske angibt. Die dritte Schreibweise dient nur zur Veranschaulichung. Die Adresse `192.168.10.255` ist die sogenannte *Broadcast Adresse* des Netzwerkes mit der Netzmaske `255.255.255.0` bzw. `/24`. Pakete mit dieser Bestimmung werden an alle Adressen im Netzwerk geschickt.

IPv6 Adressen sind 128 Bit lang, also viermal größer als IPv4 Adressen. Damit ergibt sich ein viel größerer Adreßraum mit 2^{128} theoretisch möglichen Adressen. Die Notation ist dieselbe wie bei IPv4, wobei die CIDR Notation bevorzugt wird. Nullbits können, ebenso wie bei den IPv4 Adressen, ausgelassen werden. Die folgenden drei Adressen sind daher identisch.

- `2001:cdba:0000:0000:0000:0000:3257:9652`
- `2001:cdba:0:0:0:0:3257:9652`
- `2001:cdba::3257:9652`

Bei IPv6 Anbindungen werden immer ganze Subnetze vergeben. Die kleinste Größe für ein Subnetz ist üblicherweise `/64` zur weiteren Verteilung (manche Anbieter vergeben auch kleinere Subnetze, speziell in Rechenzentren, wo die Systeme dicht gepackt sind). Bei IPv6 haben Hosts immer mehrere Adressen:

interface-local - lokale Adresse nur für ein Netzwerkgerät (läßt sich nur wie Loopback verwenden)

link-local - lokale Adresse für lokales Segment (z.B. Ethernetsegment)

global - globale Adresse für IPv6 Internet

Mit den *Link-Local* Adressen kann man lokal kommunizieren, allerdings hängen sie von der MAC Adresse des Netzwerkgeräts ab und werden ohne Routing betrieben (d.h. sie sind ausschließlich lokal zu verwenden). IPv6 Hosts benötigen nur eine globale Adresse (oder mehrere), welche auch für Paketfilter verwendet wird.

IPv4 Adressen sind in bestimmte Bereiche eingeteilt. Nicht alle Adressen finden sich im Internet. Die folgenden Bereiche sind ausschließlich für den Einsatz in LANs reserviert. [31]

- `192.168.0.0/16`
- `172.16.0.0/12`
- `10.0.0.0/8`

Außerdem gibt es einen speziellen Adreßbereich `224.0.0.0/4` für IPv4 Multicasting, welcher für Datenströme mit einer Quelle und einer großen Anzahl von „Zuhörern“gedacht ist. [32, 33] Dieser Bereich wurde früher oft auch als *Class D* Netzwerk bezeichnet.

- `224.0.0.0/24` - „bekannte“/ „well-known“Multicast Adressen
- `224.0.1.0 - 238.255.255.255`
- `239.0.0.0/8` - lokale Multicast Adressen

Der Bereich `240.0.0.0/4` ist ebenso ein spezieller Bereich, da er von der Internet Corporation for Assigned Names and Numbers (IANA)[23] reserviert ist.

[23] https://www.iana.org/

Darüber hinaus gibt es noch `127.0.0.0/8` für das Loopback Device eines jeden TCP/IP-fähigen Gerätes, welches für lokalen Netzwerkverkehr gedacht ist. Über das Loopback Device können Programme auf ein und derselben Maschine mittels TCP/IP Daten austauschen.

Ein weiterer Adreßbereich ist `169.254.0.0/16`, welcher für den sogenannten *Zeroconf*[24] oder `local-link` Mechanismus vorgesehen sind. Diese Technik stammt aus IPv6 und dient dazu Maschinen, die sich auf einem lokalen Ethernetsegment befinden, zu verbinden. Besagter Adreßbereich wird für eine IPv4-kompatible Version der automatischen Netzwerkadreßkonfiguration verwendet. Maschinen, die keinen DHCP Server finden, nehmen sich meist eine Adresse aus diesem Bereich selbsttätig. Diese `local-link` Adressen sind legitim, man muß sich nur überlegen wo und ob sie auftauchen dürfen, wenn man einen Paketfilter baut.

2012 hat die IANA[25] den Netzblock `100.64.0.0/10` für *Carrier-grade NAT* freigegeben. Adressen aus diesem Netzblock sollen weder in privaten Netzwerken noch im Internet verwendet werden. Sie sind für interne Netzwerke von Carriern (sprich Telekommunikationsfirmen/-organisationen) gedacht, um Daten zu transportieren oder Netzwerke zu verbinden. RFC6598 regelt die Verwendung dieser Adressen. [20]

Generell sollte man alle Adreßbereiche filtern, die nicht verwendet werden (sowohl an Paketfiltern als auch auf Hosts). Man kann dies entweder über Filter oder über *Null Routing* konfigurieren. Auf GNU/Linux Systemen verwendet man die *blackhole* Route.

```
ip route add blackhole 192.168.0.0/16
ip route add blackhole 172.16.0.0/12
ip route add blackhole 10.0.0.0/8
ip route add blackhole 169.254.0.0/16
```

Man kann damit nicht verwendete Netzwerke sperren. Existierende Routen überstimmen *blackhole*, d.h. man kann solche Routen auf allen Systemen setzen ohne verwendete Netzwerke zu blockieren. Das ist keine Firewall, weil eingehende Pakete von diesen Netzwerken trotzdem angenommen werden. Es werden nur alle Antworten am Host verworfen. Generell sollte man solche Konfigurationen auf Routern setzen. Diese Methode nennt sich *Bogon Filtering*. Bogons bezeichnet man auch als *Martians*. Informationen zu diesem Thema findet man in RFC 1918, RFC 5735 und RFC 6598.

> **Wichtig:** Die Listen mit den Bogon/Martian Netzwerken bzw. Quellen sind nicht statisch.

Routing

Der Transport der Pakete verläuft haupsächlich neben den Leitungsstrecken über Router, die entscheiden müssen welchen Weg das Paket nehmen kann. Alle Maschinen innerhalb eines Netzwerksegments können sich direkt sehen (im Falle von Ethernet beispielsweise). Wenn diese Maschinen andere Hosts in anderen Netzen erreichen wollen, so bedarf es eines Gateways, der den Weg zu anderen Netzwerken kennt. Gateways benutzen zu diesem Zweck eine Routing Tabelle , die ihnen den Weg der Pakete anhand deren Zieladresse vorgibt. Ein Beispiel für eine Routing Tabelle sieht wie in Abbildung 2.7 dargestellt aus (die Ausgabe wurde zur besseren Lesbarkeit etwas formatiert).

Um Routingprobleme zu verringern und Schleifen zu vermeiden, besitzt jedes Paket eine Lebensdauer oder Time To Live (TTL). Bei jedem Sprung oder Hop von einem Router zum nächsten wird von der TTL ein Hop abgezogen. Jeder Router, der ein Paket mit TTL von 1 erhält, wird dieses verwerfen und eine ICMP Fehlermeldung vom Typ `Time Exceeded` an den Absender zurückschicken.

ICMP kann auch für Diagnose von Netzwerkproblemen benutzt werden. Oft verwendet man dafür die Kombination von `Echo Request` und `Echo Reply`.

```
[lynx@luchs lynx]$ ping -c 4 192.168.10.11
PING 192.168.10.11 (192.168.10.11) from 192.168.10.3 : 56(84) bytes of data.
64 bytes from 192.168.10.11: icmp_seq=0 ttl=255 time=2.324 msec
64 bytes from 192.168.10.11: icmp_seq=1 ttl=255 time=260 usec
64 bytes from 192.168.10.11: icmp_seq=2 ttl=255 time=275 usec
64 bytes from 192.168.10.11: icmp_seq=3 ttl=255 time=264 usec

--- 192.168.10.11 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.260/0.780/2.324/0.891 ms
```

ICMP Ping kann durch Anfordern von `Echo Reply` testen, ob eine andere Maschine reagiert und wie schnell die Round Trip Time der Pakete ist. Solche Messungen sind jedoch nicht unbedingt maßgeblich für die Performance eines Netzwerkes.

[24] http://www.zeroconf.org/

[25] IANA = Internet Assigned Numbers Authority

```
[root@reorx ~]# ip route
62.116.64.96/27 via 192.168.10.254 dev eth0
192.168.20.0/24 via 192.168.10.254 dev eth0
192.168.50.0/24                    dev eth1    proto kernel  scope link  src 192.168.50.20
192.168.0.0/24  via 192.168.50.21  dev eth1    src 192.168.10.12
192.168.10.0/24                    dev eth0    proto kernel  scope link  src 192.168.10.11
192.168.0.0/16                     dev dummy0  scope link
172.16.0.0/12                      dev dummy0  scope link
10.0.0.0/8                         dev dummy0  scope link
127.0.0.0/8                        dev lo      scope link
default         via 192.168.10.254 dev eth0
```

Abbildung 2.7.: Dies ist die Routing Tabelle eines Gateways, der mehrere Netzwerke verbindet. `eth0` und `eth1` bezeichnen die beiden Netzwerkkarten der Maschine, `lo` ist das Loopback Device und `dummy0` ist ein logisches Gerät, welches alle Pakete, die dorthin gesendet werden, schluckt. Um eine Routing-Entscheidung zu treffen, werden die Routen von oben nach unten mit der Zieladresse des Paketes verglichen. Die `Default` Route bekommt alle Pakete für die keine Route bekannt ist.

Network Address Translation (NAT)

Um die Erschöpfung der im Internet direkt erreichbaren IPv4 Adressen zu vermeiden, werden mittlerweile alle lokalen Netzwerke mit privaten IP Adressen aus den in RFC1918 definierten Adreßbereichen aufgebaut. Damit nun beispielsweise Geräte aus dem LAN trotzdem „direkt“mit Maschinen im Internet Kontakt aufnehmen können, gibt es die Technik des *Network Address Translation (NAT)*. Der Gateway tauscht dabei die Quelladresse gegen seine externe Adresse aus, merkt sich den Zustand und sendet das modifizierte Paket zum Ziel. Alle Antwortpakete erfahren wiederum eine Umwandlung, dieses Mal an die Zieladresse, wenn sie von der Maschine im Internet zurückkommen. Dieses Umlegen von mehreren Adressen auf eine ist ein Spezialfall von NAT, der unter Linux® als *IP Masquerading* bekannt ist. Man kann ebenso mit einer beliebigen Zuordnung $\text{extadr}_n \longleftrightarrow \text{intadr}_m$ arbeiten. IP Masquerading ist der Fall mit $n = 1$ und $m \geq 1$.

Man kann mittels $n, m > 1$ NAT Konfiguration am Gateway Server mit RFC1918 Adressen betreiben. Die nach außen sichtbaren Services werden dann ausschließlich über das NAT an bestimmte IP Adressen gebunden. Dies ist aber *keine* Sicherheitsmaßnahme. NAT wurde aus anderen Gründen erfunden und ist kein Ersatz für Paketfilter.

Der Einsatz von NAT birgt Vor- und Nachteile. Unter den Nachteilen ist die nicht direkte Erreichbarkeit (End-to-End Communication) einer Adresse hinter einer NAT-Barriere. Das kann für bestimmte Netzwerkprotokolle zu einem Problem führen, wenn die direkte Erreichbarkeit von der Applikation verlangt wird. Darüber hinaus lassen sich einzelne Clients hinter NAT nicht oder nur schwer unterscheiden, da alle Anfragen von einer einzigen Adresse oder einem Pool von Adressen stammen.

Wichtig: Network Address Translation (NAT) ist keine Sicherheitsmaßnahme. NAT garantiert nicht, dass die internen Adressen nicht erreichbar sind. Eine solche Garantie ist nur durch Paketfilter oder entsprechendes Routing (sprich Blacklisting) zu gewährleisten. NAT implementiert auch keine zustandsgesteuerten Filter. Es wird lediglich das Umschreiben von Paketen vermerkt. Filter führen zusätzliche Sicherheitstests durch.

Network Address Port Translation (NAPT)

Einige Router, darunter der Linux® Kern, sind in der Lage soagr einzelne Ports auf andere Ports mit gleicher oder verschiedener IP-Adresse umzulegen. Dieses portweise NAT kann dazu verwendet werden, um Pakete für bestimmte Applikationen gezielt auf andere Rechner umzuleiten. Diese Technik wird *Network Address Port Translation* (NAPT) genannt.

Paketfragmentierung

Bei der Besprechung des IP Paketheaders wurde die maximale Größe eines IP Paketes mit 65536 Byte angegeben. Die maximale Größe eines Transportpakets wird vom physischen Datenübertragungmedium bestimmt. Je nach verwendeter Netzwerktechnik gibt es verschiedene vorgegebene Limits, die als Maximum Transfer Unit (MTU) in Bytes angegeben werden. Abbildung 2.4 zeigt verschiedene MTUs im Überblick.

Netzwerktechnik	MTU (Bytes)
16 Mbit/s Token Ring	17914
4 Mbits/s Token Ring	4464
FDDI (Glasfaser)	4352
Fast Ethernet	1500
IEEE 802.3/802.2	1492
X.25	576

Tabelle 2.4.: Maximum Transfer Units (MTU) für gängige Netzwerktechniken. Aufgrund der MTU für X.25 wird oft 576 als Untergrenze für die MTU im Internet angesehen (IPv6 verlangt eine Mindest-MTU von 1260 Byte). Mittlerweile ist das Internet jedoch schon vielerorts 1500 Byte „breit“. Das Tool `tracepath` kann die MTU auf Wegen ermitteln.

Da nun IP Pakete größer als die MTU sein können, werden alle größeren Pakete an der Quelle oder an einem Übergang, wo die MTU wechselt, in kleinere Pakete aufgeteilt, bevor sie im Netzwerk weitertransportiert werden. Dieser Vorgang nennt sich *Paketfragmentierung* . Nur das erste Fragment erhält den vollen IP Header mit weiteren Headern (z.B. für TCP oder UDP), alle folgenden Fragmente besitzen nur einen reinen IP Header, der für das Zusammensetzen notwendig ist. Ein Paket gilt als vollständig übertragen, wenn alle Fragmente übertragen sind. Router nehmen normalerweise keine Kenntnis von den Fragmenten, wenn die MTUs der direkt verbundenen Netze gleich ist. Hat ein Router direkten Kontakt mit Netzen verschiedener MTU Größe, so wird er selbst Fragmente generieren (im Falle von IPv4, bei IPv6 müssen die Endpunkte passende Fragmente generieren).

Die Behandlung von Fragmenten ist für Paketfilter von besonderer Bedeutung, da diese die Filterkriterien erst nach dem Erhalt aller oder des ersten Paketes anwenden können (abhängig von der verwendeten Paketfiltertechnologie). Einige Paketfilter hatten in der Vergangenheit Schwierigkeiten mit der Bearbeitung von Paketfragmenten. Firewallsysteme sollte man unbedingt diesbezüglich testen bevor sie in Produktionsbetrieb gehen. Oft können Fragmente immer noch spürbare Auswirkungen zumindest auf die Performance haben.

Anmerkung: Für Router, die an ADSL angebunden sind, sollte man die MTU anpassen. Durch die PPPoE Verkapselung steigt die Paketgröße auf 1508 Byte, was zu starker Fragmentierung führen kann. Es ist sinnvoll auf Netzwerkkarten, die auf ADSL Verbindungen gehen, die MTU zu verringern[26], z.B. auf 1492 Byte oder weniger. Ein anderes Anwendungsbeispiel sind netzwerkfähige Geräte, die mit drahtlosen Netzwerken wie GSM[27], GPRS[28] oder UMTS[29] zusammenarbeiten.

Gigabit Ethernet kennt sogenannte *Jumbo Frames*, die über 1500 Byte hinausgehen (bis zu 16 kiByte). Leider ist die maximale Größe je nach Netzwerkkarte und Switch verschieden. Man muß daher Jumbo Frames in Abstimmung mit der vorhandenen Hardware konfigurieren sonst kann es zu Verbindungsproblemen kommen (nicht bei TCP, dort wird die maximale Segmentgröße zwischen den Verbindungspunkten beim Aufbau der Verbindung ausgetauscht).

2.3.9. IPv6

IPv6 ist auf Produktionsmaschinen und den damit verbundenen Netzwerken in Verwendung. Es gibt allerdings trotz der ständig angekündigten Einführung immer noch keinen flächendeckenden Einsatz. IPv6 bringt einige Neuerungen mit sich. Beispielsweise gibt es keine Notwendigkeit für NAT mehr. Teilnehmer lokaler Netzwerke bestimmen ihre Adressen automatisch aufgrund der MAC ihrer Ethernetschnittstelle, d.h. man muß in lokalen Netzwerken theoretisch keine IP Adressen mehr vergeben. Alle diese Neuerungen sind für das Filtern oder Behandeln von IPv6-Verkehr transparent. Einzig die Adressen müssen vergeben bzw. angepaßt werden. Die meisten gängigen Systeme und Netzwerkkomponenten können heutzutage mit IPv6 umgehen. Es ist wichtig, daß man daher auf Paketfiltern IPv6 nicht vergißt.

Unterschiede zwischen IPv4 und IPv6

Die wichtigsten Unterschiede zwischen IPv4 und IPv6 lassen sich wie folgt zusammenfassen.

[26] https://www.cisco.com/c/en/us/support/docs/long-reach-ethernet-lre-digital-subscriber-line-xdsl/asymmetric-digital-subscriber-line-adsl/12918-router-mtu.html
[27] GSM = Global System for Mobile Communications
[28] GPRS = General Packet Radio Service
[29] UMTS = Universal Mobile Telecommunications System

- **Größerer Adreßraum** - Mit $2^{128} \approx 3,4 \cdot 10^{38}$ Adressen ist der größere Adreßraum ein unmittelbar erkennbarer Vorteil.[30]
- **Kein NAT** - NAT fällt komplett weg. Dies ist ein immenser Vorteil, der Routing und besonders das Erstellen von Filterregeln extrem vereinfacht.
- **Zustandslose Adreß-Autokonfiguration** - In einem IPv6 Netzwerk ermitteln die Hosts ihre eigene IPv6 Adresse(n) automatisch ohne Adreßkollisionen.
- **Kein Broadcast** mehr - Broadcast-Adressen wurden aufgrund der Größe der Netzwerke nicht mehr vorgesehen.
- **Paketfragmentierung** findet nur noch auf den Endpunkten statt, nicht mehr im Netzwerk. Router und Firewall müssen sich nicht mehr um Fragmente kümmern.
- **Multicast** ist fester Bestandteil der IPv6 Spezifikation und muß daher von allen IPv6-fähigen Hosts unterstützt werden (weil es keine Broadcasts mehr gibt).
- **IPsec** ist fester Bestandteil der IPv6 Spezifikation und muß daher von allen IPv6-fähigen Hosts unterstützt werden.
- **Vereinfachtes Routing** - Der IPv6 Paketkopf ist einfacher als bei IPv4. Selten genutzt Kopffelder sind nun in Optionen abgebildet. Es gibt keine Paketfragmentierung mehr: Hosts müssen die MTU eines Pfades entdecken, selbst die Pakete passend generieren oder auf die minimale IPv6 MTU von 1260 Byte gehen. Die Prüfsummen im Kopf fallen weg, da die Integritätsprüfung auf darunter- oder darüberliegende Schichten verschoben wird. IPv6 router müssen daher keine Prüfsummen berechnen. Sie müssen auf keine TTL Werte mehr berechnen.
- **Mobilität** - IPv6 Router unterstützen mobile Clients, die ihre Router wechseln. Die Unterstützung dafür ist noch gering, daher ist dieser Vorteil derzeit von theoretischer Natur.
- **Jumbogramme** - Die maximale Paketgröße für IPv4 Pakete liegt bei 65535 Byte ($2^{16} - 1$ Byte). Bei IPv6 liegt dieses Limit bei $2^{32} - 1 = 4294967295$ Byte und wird durch die *Jumbo Payload Option* im Kopf angezeigt. Jumbogramme erfordern auch Anpassungen bei TCP und UDP, um die Paketgröße richtig abbilden zu können.

Spezielle IPv6 Adressen sind die folgenden.

- `::/128` markiert das komplette IPv6 Internet.
- `::/0` markiert die Default Route.
- `::1/128` markiert die *Loopback Adresse* eines jeden Hosts.
- `fe80::/10` markiert *Link Local Adressen*, die nur auf einem bestimmten Ethernetsegment gelten (eben dem mit der passenden Link Local Adresse).
- `fec0::/10` markiert standortlokale Adressen (*Site Local Unicast*). Dieser Bereich wird als solcher nicht mehr verwendet.
- `fc00::/7` markiert *Unique Local Adressen* für lokale Datenkommunikation.
- `ff00::/8` markiert Multicast Adressen.
- `0:0:0:0:0:ffff::/96` sind *IPv4 mapped* IPv6 Adressen. Damit kann man IPv4 Adressen im IPv6 Adreßraum verwenden. Die letzten 32 Bit der Adresse entsprechen der kodierten IPv4 Adresse.
- `2000::/3` ist der Adreßraum, der von der IANA vergeben wird. Darin gibt es weitere Unterscheidungen.
  - `2001`-Adressen werden an Internet Provider zur Weitergabe an Kunden vergeben. Dabei sind Adressen aus dem Netzwerk `2001:db8::/32` rein für Dokumentationszwecke reserviert (d.h. sie sollten in Netzen nicht auftauchen).
  - `2002::/16`-Adressen werden für den Tunnelmechanismus *6to4* verwendet.
  - `3ffe::/16` wurde für ein experimentelles IPv6-Netzwerk namens *6Bone* verwendet. Die Adressen wurden zurückgegeben, da das Experiment bereits abgeschlossen ist.

[30] Durch den wesentlich größeren Adreßraum wird das IPv4 Internet langfristig aufgegeben werden. Der IPv4 Adreßraum macht nur einen sehr kleinen Teil des IPv6 Internets aus: $\frac{N_{IPv4}}{N_{IPv6}} \approx 1,26 \cdot 10^{-29}$

Paketfragmentierung bei IPv6

Die Mechanik von Paketfragmentierung wurde bei IPv6 geändert. Bei IPv6 können nur die Endpunkte Pakete fragmentieren. Router und Zwischenstellen (*intermediate nodes*) fragmentieren Pakete nicht, sie müssen sie nur transportieren. Ansonsten ist der Mechanismus ähnlich, d.h. es enthält auch nur das erste Paket die Kopfdaten. [21]

Anbindungen

IPv6 gibt es noch nicht überall „direkt", d.h. so wie öffentliche IPv4 Adressen jetzt (direkt bis zum Modem bzw. Endgerät). Zur Anbindung ist daher mehr als eine Methode vorgesehen.

- direktes Ansprechen über IPv6
- Tunnelmechanismen
  - *6in4* - Senden von IPv6 Paketen in IPv4 Paketen mit Protokollnummer 41 (für IPv6) (RFC 4213).
  - *6to4* - Abbildung des IPv4 Adreßraums in IPv6 und Verbinden über 6to4 Relay Router, welche man durch Verwenden der `2002:c058:6301::` Anycast Adresse findet (RFC 3068).
  - *6over4* - Verbinden von Dual-Stack Hosts über IPv4 Multicast Transmissionen.
  - Teredo-Protokoll [23]
- Proxydienste

Die Tunnelmechanismen basieren auf Verkapselung der IPv6 Pakete. Protokolle werden im IP-Kopf mit der Protokollnummer markiert und können so gefiltert werden. Wenn die Verkapselung in UDP geschieht, dann ist das Filtern schwieriger. In Summe gibt es etwa 15 Methoden, wenn man alle Kombinationen mitrechnet.

Autokonfiguration

Die IPv6 Autokonfiguration erfolgt automatisch durch Generierung einer Link Local Adresse und Suchen eines IPv6-Routers mittels Multicast Paketen. Die Link Local Adresse wird mit Hilfe der MAC-Adresse des Netzwerkgeräts berechnet und sollte daher eindeutig sein. Ein vorhandener IPv6-Router kann dann auf die Neighbor Discovery Protocol (NDP) Anfragen antworten und dem Host den gültige Netzwerkprefix mitteilen. Der Host kann sich dann aus dem verwendeten Netzwerk eine Adresse aussuchen. Eine Adreßkollision wird mit Hilfe der *Duplicate Address Detection* (DAD) verhindert. DAD prüft, ob Adressen doppelt vorkommen. Ein IPv6 Host bekommt dann eine vorläufige Adresse. Mittels *Neighbor Solicitation and Neighbor Advertisement* ICMPv6 Meldungen wird die Eindeutigkeit festgestellt. Ist diese bestätigt, so behält sie der Host. Man benötigt für die Autokonfiguration eigene Software. GNU/Linux Systeme können `radvd` installieren, um NDP Anfragen zu beantworten. Eine Konfigurationsdatei kann dann so aussehen.

```
interface eth0
{
   AdvSendAdvert on;
   AdvHomeAgentFlag on;
   AdvLinkMTU 1480;
   MinRtrAdvInterval 3;
   MaxRtrAdvInterval 10;
   prefix fec0:3617:5648:6848::/64
   {
        AdvOnLink on;
        AdvAutonomous on;
        AdvValidLifetime 86400;
        AdvPreferredLifetime 28800;
        AdvRouterAddr on;
   };
};
```

Da durch diese Autokonfiguration keine Informationen über Host-/Domainnamen, DNS, NTP oder ähnliche Einstellungen verteilt werden, kann man einen DHCPv6-Server verwenden. Dieser führt dann auch Buch über gesehene Hosts und deren Adressen. DHCPv6 ist allerdings optional.

Die Autokonfiguration zeigt damit sehr deutlich, daß eine funktionierende Namensauflösung in jedem Netzwerk absolut notwendig ist! Ein `/64` Subnetz enthält ein Maximum von $\approx 1,8 \times 10^{19}$ Adressen. Portscans sind damit sinnlos, nur eine Namensauflösung kann helfen den richtigen Host in endlicher Zeit zu finden.

2.4. Weitere Layer 2 Protokolle

Es wurde aus der zweiten Schicht bisher nur ARP vorgestellt. Es gibt allerdings auf dieser Schicht in bestimmten Umgebungen noch weitere Protokolle, die für sicherheitstechnische Betrachtungen relevant sind.

- ATA over Ethernet (AoE)
- Cisco Discovery Protocol (CDP)
- IEEE 802.3ad
- Layer 2 Tunneling Protocol (L2TP)
- Link Aggregation Control Protocol (LACP)
- Link Level Discovery Protocol (LLDP)
- Point-to-Point Protocol (PPP)
- Spanning Tree Protocol (IEEE 802.1d, STP)
- Virtual Router Redundancy Protocol (VRRP) [22]

Diese Protokolle erlauben Attacken und Eingriffe in Datentransmissionen. Einige, wie AoE beispielsweise, besitzen keine Authentifizierungsmechanismen und können von Maschinen im selben Segment angesprochen werden.

Oft findet sich auf dieser Schicht keine oder nur eine unzureichende Authentisierung. Viele Protokolle gehen davon aus, dass das lokale Netzwerksegment vertrauenswürdig ist. Systeme müssen daher speziell im Layer 2 entsprechend dem Sicherheitslevel zusammengefaßt bzw. getrennt werden.

2.5. Namens-/Dienstauflösung

Das Kapitel Namensauflösung erscheint an dieser Stelle etwas unpassend, da die dafür notwendigen Protokolle bis in den Layer 7 gehen. Dennoch ist die Namensauflösung ein zentraler Punkt von Netzwerken, da ohne sie Clients und Server keine Resourcen finden. Verbreitete Protokolle sind

- Bonjour (ehemals Rendezvous)[31],
- Domain Name System (DNS),
- Link Local Multicast Name Resolution (LLMNR),
- NetBIOS Namensauflösung (nur für IPv4),
- Service Location Protocol (SLP, srvloc) oder
- Simple Service Discovery Protocol (SSDP).

Die Protokolle nutzen Broadcast, Multicast, HTTP-über-UDP, Multicast DNS (mDNS) und andere Komponenten, um im Netzwerk Clients und Dienst anzukündigen. Microsofts *Universal Plug and Play* (UPnP) fällt teilweise auch in diesen Bereich, macht aber keine Namensauflösung sondern vermittelt nur Applikationsanforderungen.[32]

Bei der Planung von Netzwerken sind Namensauflösung und Dienstankündigung unbedingt zu beachten. Speziell bei IPv6 Netzwerken sind diese Mechanismen unverzichtbar. Filter müssen die Eigenschafter der Protokolle kennen und unter Umständen gezielt solche Dienste erlauben.

[31] https://www.apple.com/support/bonjour/
[32] UPnP kann beispielsweise Löcher in einer SOHO Firewall anfordern.

2.6. Bedrohungen für vernetzte Systeme

Vernetzte Systeme erhöhen die Komplexität und die Abhängigkeit von Computersystemen erheblich. Viele Applikationen verlassen sich auf ständig zur Verfügung stehende Resourcen und auf verläßlich übertragene Daten, die gültig sind. Kombiniert man diesen Umstand mit der Tatsache, daß keine Software ohne Fehler auskommt, so treten die Probleme in den Vordergrund mit denen sich Systemadministratoren und Sicherheitsberater beschäftigen. Ganz grob kann man die Angriffe in die folgenden Kategorien einteilen:

1. **Attacken durch erlaubte Verbindungen zu Services**
   a) Angriffe auf den Kommandokanal eines Services (FTP, SMTP/ESMTP, etc.)
   b) *brute-force* Attacken
   c) datengesteuerte Angriffe („data driven attacks“)
   d) Angriff von dritter Seite
   e) falsche Authentifizierung von Clients
   f) *man-in-the-middle* Attacken (MITM)
   g) Dsitributed Denial of Service (DDoS)
2. **Attacken durch nicht erlaubte Verbindungen zu Services**
   a) Umgehung von Filtern über „Sprungbretter” (andere Protokolle, Tunnel, etc.)
   b) Kommunikation mit Systemen, die nicht beschützt sind
3. **Attacken, die den Verbindungsaufbau umgehen**
   a) Hijacking
   b) Paketschnüffler („sniffer“)
   c) Einschleusen und Veränderung von Daten
   d) Replay-Angriff
   e) Denial of Service (DoS)
4. **Social Engineering**
   Dies umfaßt Attacken, die den menschlichen Faktor ausnutzen. [34]
   a) Umgehung von technischen Maßnahmen durch Beeinflussung von Menschen direkt
   b) ein falscher Klick in Emails oder auf Webseiten
   c) Anschauen oder Ausführen bestimmter Dateiformate
   d) „Phishing“- kurz für *password harvesting fishing*
   e) Spyware

Die Aufzählungen der Möglichkeiten in den Kategorien sind unvollständig und sollen nur als Verdeutlichung dienen. Die Auswirkungen auf vernetzte Systeme zeigen sich auf verschiedene Art und Weise.

- **Erschöpfung von Resourcen**
  Bandbreite, Paketlaufzeiten, Speicherplatz und CPU-Zeit sind beispielsweise kritische Resourcen, deren Verfügbarkeit durch Fehler in oder Angriffe auf Software eingeschränkt werden kann.
- **Mißbrauch von Privilegien**
  Manche Resourcen erlauben den Zugriff nach verschiedenen Sicherheitsstufen. Fehler oder Angriffe können dazu führen, daß Benutzer mit erhöhten Privilegien auf Resourcen zugreifen.
- **Manipulation von Daten**
  Durch Fehler oder Angriffe kann zu Manipulation an Daten kommen, die weitere Applikationen zur Verarbeitung verwenden.
- **Abfangen von Informationen**
  Darunter fällt das Mithören von Paßworten, kryptografischen Schlüsseln, Emails, die Ausführung von Replay Attacken, etc.

- **Einschleusen von Schadsoftware**
  Trojanische Pferde, die Server und Clients zu Robotern („Zombies") machen und diese zu Netzwerken infizierter Clients zusammenfassen („Botnets")

Die Angriffvektoren für Eindringlinge sind vielfältig. Eine Auswahl soll die Aufzählung bieten, jedoch sind das nicht alle Möglichkeiten, die ein kreativer Geist ausnutzen kann. Speziell Webapplikationen bieten eine Reihe von Wegen, die ausgenutzt werden können (wie gut und ob überhaupt, hängt sehr stark von der Qualität des Codes ab).

- **Authentifizierungssysteme**
  Remote Authentication Dial-In User Service[33] (RADIUS), Terminal Access Controller Access Control System[34] (TACACS+), Microsoft Windows Domain Controller, MIT Kerberos[35] oder ähnliche Systeme, die über verteilte Server Zugriff auf Netzwerkresourcen gewährleisten

- **Bootblöcke**
  Bootblöcke enthalten ausführbaren Code und liegen meist frei zugänglich auf Datenträgern.

- **Datenbankserver**
  Injizieren von gefährlichen SQL Befehlen über Front Ends, Attackieren von offenen Ports zu Datenbanken

- **Directory Services**
  Lightweight Directory Access Protocol[36] (LDAP), Sun Yellow Pages / Network Information Service (YP/NIS) oder ähnliche Dienste, die Informationen für Clients zur Verfügung stellen

- **Domain Name Service (DNS)**
  DNS Server (z.B. BIND), Vergiften von DNS Caches mit Falschinformationen, Übernahme eines DNS Servers und Ändern der DNS Informationen

- **Druckersprachen**
  Daten werden zum Drucken oft in bestimmte Formate konvertiert. Einige von ihnen sind de fakto Programmiersprachen (z.B. PostScript®) oder können solche enthalten (z.B. Adobe® Reader). Damit werden Dokumente zu Angriffswerkzeugen.

- **File Sharing Protokolle**
  Server Message Block[37] (SMB), Common Internet File System[38] (CIFS), Network File System[39] (NFS)

- **File Swapping Programme**
  Programme wie Kazaa, Morpheus, Limewire, AudioGalaxy, Gnutella, Soulseek, Bittorrent, etc. bieten vielfältige Möglichkeiten gefährliche Daten auf Client Maschinen zu platzieren

- **FTP Server**
  Ausnutzen von Bugs zwecks Zugriff auf Dateisystem des Servers, Mißbrauch von FTP Server als toter Briefkasten für sogenannte Warez Sites, Deponieren von Angrifftools

- **Instant Messaging und Conferencing Programme**
  AIM, Yahoo! Messenger, UNIX® Talk, ICQ, Jabber, Microsoft NetMeeting, Internet Relay Chat (IRC), Gadu-Gadu, Zephyr, Skype, etc.

- **Mail Transport Agents (MTAs)**
  sämtliche Mailserversoftware; Blockieren von Mail Transport, Ausnutzen von Bugs durch speziell formatierte Emails; Wechselspiel mit installierten Anti-Viren Tools, die ihrerseits wieder Fehler haben können

- **Mail User Agents (MUAs)**
  Ausnutzen von Verknüpfungen zwischen Dateiformaten und Programmen durch bestimmte Dokumente, Einschleusen von gefährlichem Inhalt über Skriptsprachen

[33] https://en.wikipedia.org/wiki/RADIUS
[34] https://datatracker.ietf.org/doc/draft-ietf-opsawg-tacacs/
[35] https://web.mit.edu/Kerberos/
[36] https://www.openldap.org/
[37] https://www.samba.org/cifs/docs/what-is-smb.html
[38] http://ubiqx.org/cifs/
[39] http://nfs.sourceforge.net/

- **Office Programme**
  Implantierung von gefährlichen Makros und Skriptsprachen

- **Peer-to-Peer (P2P) Software**
  Diese Software hat mittlerweile Instant Messaging Funktionalitäten und ist damit komplexer geworden.

- **Voice over Internet/IP (VoIP)**
  VoIP Clients stellen je nach Ausführung eine Mischung aus P2P Software, Instant Messaging/Conferencing, Audio-/Video-Player und VoIP-Signalprotokollclients dar. Dadurch ergeben sich zahlreiche Kombinationen für Angriffe.

- **WWW**
  CGI Skripte, HTTP Server, ebenso Ausnutzen von Bugs zwecks Zugriff auf Dateisystem des Servers

- **Web Browser**
  MS IE, Mozilla Firefox, Opera, Safari, etc. - Web Browser sind mittlerweile sehr komplexe Programme, die mit vielen anderen Applikationen verknüpft sind. Angriff erfolgen über (inkorrekte) URLs, Webseiten oder Add-Ons/Plugins.

In allen Fällen wird man danach suchen, den Fluß der Informationen zu kontrollieren, um Sicherheitsentscheidungen erzwingen zu können. Sicherheitsmaßnahmen sollten möglichst an allen Punkten eines vernetzten Computersystems getroffen werden (Prinzip der *Defence in Depth*, *mehrschichtige Verteidigung*). Dies gilt insbesondere für Systemupgrades und Antivirusmaßnahmen, wobei Paketfilter auch in solchen Fällen eindämmend eingreifen können, sollte es ein Sicherheitsproblem in internen Netzwerken geben. Zusammengefaßt darf der Ausfall einer einzelnen Sicherheitsmaßnahme die Systeme nicht schutzlos lassen. Es müssen immer mehrere Sicherheitsmaßnahmen gleichzeitig wirksam sein, die einander ergänzen.

Das Konzept bedeutet auch, daß sich jedes System bis zu einem gewissen Grad selbst verteidigen muß. Zugriffsbeschränkungen müssen überall konfiguriert sein. Ein System darf nicht in große Gefahr geraten, wenn es mal eine öffentliche IP Adresse bekommt (was bei IPv6 Tunnel mittlerweile sehr leicht passieren kann).

2.7. Schutzmechanismen für vernetzte Systeme

Aufgrund der Vielfalt der eingesetzten Applikationen und Protokolle kommt man sicherheitstechnisch in einem Netzwerk nicht ohne Kontrolle aus. Gateways sind natürliche Engpässe, die alle Pakete bekommen, welche zwischen zwei verschiedenen Netzwerken ausgetauscht werden. Damit kann man Netzwerke mit verschiedenem Sicherheitsbedarf durch Gateways schleusenartig trennen. Es kommen mehrere Methoden in Frage.

- **Paketfilter**
  Paketfilter betrachten die einzelnen IP Pakete und erlauben bzw. verbieten den Transport nach bestimmten Regeln. Unter den Paketfiltern gibt es auch noch Unterschiede, die im nächsten Kapitel genauer beleuchtet werden. Paketfilter arbeiten üblicherweise in den Schichten 2, 3 und 4 des OSI Modells.

- **Proxy Server**
  Proxy Server arbeiten auf Applikationsebene und übersetzen Anfragen und Datentransport. Zu diesem Zweck muß der Proxy speziell für das zu übermittelnde Protokoll geschrieben sein (HTTP/HTTPS, FTP, DNS, ESMTP/SMTP, etc.). Proxy Server lassen in der Regel eine gute Konrolle über Zugriffsregeln zu. Der Nachteil besteht darin, daß nicht immer eine Proxy Software für ein bestimmtes Protokoll zur Verfügung steht. Der große Vorteil von Proxies ist die Überprüfung des Protokolls. Man kann damit oft Datenflüsse eliminieren, die nicht einem bestimmten Format entsprechen, womit viele Angriffe schon abgewehrt werden können. Proxy Server arbeiten in der Schicht 7 des OSI Modells. Sie können sowohl vor Servern (Reverse Proxy) als auch vor Clients (Forward Proxy) eingesetzt werden.

- **Content Filter**
  Es gibt Systeme, die explizit prüfen, ob eine bestimmte Datenübertragung dem richtigen Protokoll entsprich. Manche Systeme verhalten sich nur wie ein Vermittler und prüfen beispielsweise nicht, ob Pakete an einen Webserver auch wirklich HTTP sind oder nicht. Mit einer Überprüfung des Inhalts kann man bis zu einem gewissen Grad sicherstellen, daß kein Eindringling offene Wege zum Durchtunneln von Informationen mißbraucht. Content Filter haben den Nachteil, daß sie wesentlich öfter angepaßt werden müssen, da die Filterregeln komplexer sind. Insbesondere Filter, die auf Signaturen basieren, müssen stetig an Bedrohungen angepaßt werden. Ein Beispiel für solche

Filter sind Anti-Virus-Programme oder bestimmte Anti-Spam-Filter. Content Filter arbeiten, genau wie Proxy Server, in der Schicht 7 des OSI Modells. Bestimmte Paketfilter oder Switches lassen sich jedoch auch mit Mustern „laden", nach denen sie suchen sollen.

Letztlich lassen sich alle Schutzsysteme auf diese Methoden reduzieren.

2.7.1. Verschlüsselung

Zusätzlich zur Zugriffskontrolle ist es oft erforderlich die Daten während des Transports für Dritte unlesbar zu machen. Dies kann einerseits über Verschlüsselung auf Applikationsebene geschehen (z.B. Pretty Good Privacy[40] PGP, GNU Privacy Guard[41] GPG, Secure Socket Layer SSL oder Secure Shell SSH), andererseits ist es möglich auch auf niedrigerer Ebene eine verschlüsselte Übertragung durch sogenannte *Virtual Private Networks* (VPN) durchzuführen. Die Aufgabe von VPNs ist es, eine Datenübermittlung über unsichere Kommunikationswege zu ermöglichen. Daher spricht man auch gelegentlich von VPN Tunneln. Mögliche VPN Implementationen sind:

- **IP Security Protocol (IPsec)**
  IPsec [35, 36] der Internet Engineering Task Force (IETF) stellt eine Reihe von Erweiterungen des IPs dar. Mit IPsec läßt sich jedes der Internetprotokolle übertragen. Die IPsec Erweiterungen stellen Mechanismen zur Authentifizierung, Integritätprüfung, Zugriffskontrolle und Vertraulichkeit transparent zur Verfügung. Zu diesem Zweck dienen die folgenden Protokolle.
  - *Encapsulating Security Payload (ESP)*
    verschlüsselt oder authentifiziert Daten
  - *Authentication Header (AH)*
    authentifiziert Datenpakete
  - *Internet Key Exchange (IKE)*
    dient zum Aushandeln der Verbindungsparameter für ESP und AH (inklusive kryptographischer Schlüssel)

  IPsec ist fester Bestandteil von IPv6. Es wurde nach IPv4 zurückportiert und steht damit dort auch zur Verfügung.

- **Point-to-Point-Tunneling Protocol (PPTP)**
  PPTP transportiert Daten über Pakete des Point-to-Point Protocol (PPP), die in einem IP Paket stecken. Die Einbettung geschieht über das Generic Routing Encapsulation (GRE). [37, 38] Verschlüsselung ist ebenso vorhanden und hängt von der verwendeten PPTP Implementation ab. PPTP ist recht nützlich für Fernzugriffe, jedoch lassen sich ganze Netzwerke nicht gut über PPTP verbinden. Darüber hinaus bietet der Entwurf dieses Protokolls einige Schwächen, da die Aushandlung der PPTP Verbindung einem Mithörer Informationen liefert (z.B. Benutzernamen, evtl. Hash-Wert des Paßwortes). [35] Microsoft® empfiehlt PPTP nicht mehr einzusetzen.[42]

- **Layer 2 Tunneling Protocol (L2TP)**
  L2TP ist eine andere Methode, um Daten in PPP Paketen eingekapselt übertragen zu können. [39] Verschlüsselung muß zusätzlich zu L2TP durch IPsec oder andere Mittel zur Verfügung gestellt werden.

Darüber hinaus gibt es noch weitere Möglichkeiten mit Softwarepaketen VPN Tunnel aufzubauen. Einige Beispiele sind:

- AESOP[43] (historisch)
- CIPE[44] (historisch)
- Linux® FreeS/WAN[45]
- Linux® IPsec HOWTO[46]
- OpenVPN[47]

[40] https://www.openpgp.org/
[41] https://www.gnupg.org/
[42] http://pptpclient.sourceforge.net/protocol-security.phtml
[43] http://kryptology.org/aesop/
[44] http://sites.inka.de/bigred/devel/cipe.html
[45] http://www.freeswan.org/
[46] http://www.ipsec-howto.org/
[47] https://www.openvpn.org/

- tinc[48]
- VTUN[49] (historisch)
- WireGuard[50]

Bei der Wahl einer VPN Technologie sollte man sich die Möglichkeiten und die verwendeten Verschlüsselungsalgorithmen genau anschauen und vergleichen. Insbesondere die Bug History und eventuelle Sicherheitsanalysen sind dazu sehr nützlich. Darüber hinaus erfüllen die vorgestellten Beispiele verschiedene Arten von Interoperabilitätskriterien, welche ebenso eine Rolle spielen.

> **Wichtig:** Eine Verschlüsselung ist *nicht automatisch* eine Sicherheitsmaßnahme! Analog zu Objekten wie Autos oder Wohnungen kommt es immer darauf an, wer die notwendigen Schlüssel besitzt bzw. wie diese Schlüssel aufbewahrt und ausgetauscht werden. Bei nicht oder schlecht dokumentierter oder geplanter Aufbewahrung und Austausch kann dadurch sogar ein Sicherheitsrisiko entstehen. Zertifikate und eine dazu passende Public Key Infrastructure (PKI) können da sehr helfen. Generell ist die Verschlüsselung nur so sicher wie die Verwaltung der Schlüssel.

2.7.2. Public Key Infrastructure (PKI)

Verschlüsselung alleine reicht für eine Absicherung oft nicht. Die Identität von Sender und Empfänger sowie die Integrität der Nachricht selbst müssen auch überprüft werden. Secure Socket Layer (SSL) bzw. dessen Nachfolger Transport Layer Security (TLS) in Kombination mit X.509 Zertifikaten bieten da sehr gute Abhilfe. Bevor man das implementiert, muß man sich aber Gedanken über die Schlüssel- und Zertifikatsverwaltung machen. Man spricht in diesem Zusammenhang oft von einer Public Key Infrastructure (PKI).

Eine PKI regelt Abläufe und schafft damit eine Struktur für den Einsatz von Verschlüsselung.

- Certificate Authority (CA)
  Die Certificate Authority zertifiziert alle eingesetzten kryptografischen Schlüssel und stellt pro Schlüssel ein Zertifikat aus, welches die Integrität des Schlüssels sicherstellt. Das Schlüsselzertifikat enthält auch Auskunft über die zertifizierende CA und besitzt meist eine begrenzte Lebensdauer. Schlüssel mit abgelaufenem Zertifikat werden automatisch unbrauchbar.

  Die CA kann außerdem bereits ausgestellte Zertifikate annullieren. Sie führt dazu eine eindeutige Liste aller erstellten Zertifikate und eine Liste von annullierten Zertifikaten.

  Die CA besitzt einen eigenen kryptografischen Schlüssel, der nicht aus der Hand gegeben werden darf. Die Daten der CA müssen extrem gut gesichert und nur autorisierten Personen zugänglich aufbewahrt werden.

- kryptografischen Schlüssel
  Diese Schlüssel entsprechend Paßworten, sind aber meist zufällig erstellte Codes in binärer Form. Durch Menschen einzugebende Paßworte werden meist mit dem Schlüsselzertifikat verbunden.

- Schlüsselzertifikate
  Diese Zertifikate ermöglichen es festzustellen, ob die CA den Schlüssel zertifiziert hat.

- CA Zertifikat
  Das Zertifikat der CA ist öffentlich. Damit kann man feststellen, ob die Schlüsselzertifikate von der zum CA Zertifikat gehörenden CA ausgestellt wurden.

Alle Schlüssel sind als private schützenswerte Daten anzusehen. Die Zertifikate alleine sind öffentlich(er), da sie ohne dazugehörigen Schlüssel nichts nutzen. Das Vertrauensverhältnis wird durch die CA zentral geregelt. Alle von der CA ausgestellten Zertifikate sollten vertrauenswürdig sein.

CAs können in Hierarchien angeordnet werden, um ausgedehnte Organisationen abbilden zu können. Als Basis dient eine sogenannte *Root CA* an der Wurzel der Hierarchie.

[48] https://tinc-vpn.org/
[49] http://vtun.sourceforge.net/
[50] https://www.wireguard.com/

Dieses Konzept klingt kompliziert. Die Zertifikate ermöglichen jedoch eine gegenseitige Identitätsprüfung der Kommunikationspartner, *wenn* diese konfiguriert ist und eingehalten wird. Bei der Kommunikation zwischen Partnern im „Hoheitsbereich" verschiedener CAs kann es passieren, daß zwischen diesen kein Vertrauensverhältnis besteht und die Identitätsprüfung optional ist. In einem solchen Fall kann ein Angreifer seine eigene CA verwenden und als Mittelsmann agieren. Solche Konstellationen sind bei Kommunikationen über das Internet durchaus gängig, meist immer dort, wo die eigene CA nicht alle Endpunkte zertifiziert hat.

Generell konfiguriert man eine eigene CA für strikt firmeninterne Datenkommunikation (weil die Kontrolle darüber dann auch firmenintern bleibt). Überall dort, wo verschiedene Firmen und Organisationen „aneinanderstoßen", wird man ein Vertrauensverhältnis zuerst etablieren müssen.

2.7.3. Aufbau einer Certificate Authority

Zum Aufbau einer eigenen Root Certificate Authority (Root CA) benötigt man eigentlich nur OpenSSL. Die ganze CA lebt in einem einzigen Verzeichnis, welches natürlich stark abgesichert aufbewahrt werden muß. Ausschließlich autorisiertes Personal darf Zugang zu den Daten der CA haben.

```
# mkdir /secure/root-ca
# cd /secure/root-ca
# chmod 700 .
```

In diesem Verzeichnis legt man nun das `Makefile` ab, welches im Anhang angeführt ist. Zusätzlich benötigt man noch eine OpenSSL Konfigurationsdatei `openssl.conf`, die man einstellen und auch im Verzeichnis der CA speichern muß. Man kann eine Vorlage aus dem OpenSSL Paket verwenden. In der Konfiguration werden Eigenschaften der CA festgelegt. Es ist daher wichtig, daß diese Einstellungen tatsächlich den Wünschen entsprechen bevor man fortfährt. Speziell die folgenden Parameter sind zu kontrollieren.

- `RANDFILE` - gibt die Zufallsquelle an. Sollte wenn möglich `/dev/random` oder ein entsprechender starker Zufallsgenerator sein.
- `default_days` - Gültigkeitsdauer der ausgestellten Zertifikate in Tagen.
- Die Sektion `policy_match` regelt welche Felder im Zertifikat zwingend sind und welche mit Vorgaben übereinstimmen müssen.
- Die Sektion `root_ca_distinguished_name` definiert die Beschreibung der Root CA selbst.
- `nsBaseUrl`, `nsRevocationUrl`, `nsRenewalUrl` und `nsCaPolicyUrl` geben URLs an, die zur CA und Informationen über annullierte Zertifikate führen. Die Angaben sind optional.

Wenn `openssl.conf` und das `Makefile` erstellt ist, dann kann die CA initialisiert werden. Dazu wird ein Schlüssel und ein Zertifikat erzeugt. Per Default gilt das Zertifikat der CA 5 Jahre (Parameter `-days` im `Makefile`). Unter Umständen muß man den Wert anpassen, da eine abgelaufene CA vollständig ersetzt werden muß (dies impliziert den Austausch aller ausgegebenen Zertfikate der alten CA mit von der neuen CA ausgestellten Zertifikaten). Das ist ein beträchtlicher Aufwand.

```
# make init
```

Jetzt ist die Root CA fertig. Im Verzeichnis `private/` befindet sich das Herz der CA, nämlich der private Schlüssel `ca-key.pem`. Die Datei `ca-cert.pem` das öffentliche Zertifikat der CA, mit welchem andere Zertifikate überprüft werden können. Im Verzeichnis `newcerts/` sind alle ausgestellten Zertifikate enthalten. Nun können eigene Zertifikate ausgestellt und Schlüssel erzeugt werden.

1. privaten Schlüssel erstellen
   `openssl genrsa -rand /dev/urandom -out host.key 4096`
2. Zertifikatsanfrage erstellen
   `openssl req -new -rand /dev/urandom -nodes -key host.key -out host.csr`
   Der wichtigste Punkt ist hier die Information *Common Name*, weil das die Identität ist, die zertifiziert wird. Meist handelt es sich dabei um einen Servernamen.
3. Zertifikat(e) erstellen und signieren
   `make sign`

Man erhält danach die Datei `host.key` und `host.cert`. Beide sind im BASE64-kodierten Privacy Enhanced Mail (PEM) Format abgelegt. Möchte man einen Schlüssel annullieren, so führt man folgende Befehle aus.

```
# openssl ca -config openssl.cnf -revoke host.cert
# make gencrl
```

Vor Ausstellen eines neuen Zertifikats mit demselben Namen, muß man das alte annullieren. Dies gilt auch für abgelaufene Zertifikate, da die Root CA über alle je ausgestellte Zertifikate Buch führt. Der Befehl `make gencrl` erzeugt die sogenannte Certificate Revocation List (CRL). Mit ihrer Hilfe können annullierte Zertifikate identizifiert werden.

Auf Clients und Servern verwendet man nun immer diese drei Informationen.

1. `ca-cert.pem` der CA
2. `host.key`
3. `host.cert`

Kommunizieren zwei Stellen mit Zertifikaten, die von unserer Root CA ausgestellt sind, dann können sie ihre gegenseitige Identität überprüfen. Das ist ein fundamentaler Punkt von sicherer Kommunikation. Ohne die Identitätsprüfung sind trotz Verschlüsselung noch Mittelsmannattacken möglich!

3. Arten von Paketfiltern

```
        The seven eyes of Ningauble the Wizard floated back to his hood as he
reported to Fafhrd: ”I have seen much, yet cannot explain all. The Gray
Mouser is exactly twenty-five feet below the deepest cellar in the palace
of Gilpkerio Kistomerces. Even though twenty-four parts in twenty-five of
him are dead, he is alive.
        ”Now about Lankhmar. She’s been invaded, her walls breached
everywhere and desperate fighting is going on in the streets, by a fierce
host which out-numbers Lankhmar’s inhabitants by fifty to one -- and
equipped with all modern weapons. Yet you can save the city.”
        ”How?” demanded Fafhrd.
        Ningauble shrugged.  ”You’re a hero.  You should know.”
                -- Fritz Leiber, ”The Swords of Lankhmar”
```

3.1. Überblick - Wahl der „Waffen”

Es gibt sehr viele Produkte, die zur Sicherung von Netzwerken auf Routern, Servern und anderen netzfähigen Geräten eingesetzt werden. In der Auswahl findet man sowohl verschiedene Hardware als auch Software. Im folgenden beziehen sich die Beispiele ausschließlich auf GNU/Linux® Systeme, die von Haus aus sehr brauchbare Tools für Routing und Paketfilterung mitbringen. Die vorgestellten Prinzipien lassen sich auf beliebige Filtersysteme und Router übertragen, denn die alleinige Wahl eines bestimmten Produktes löst noch keine Probleme per se.

Ausgangspunkt ist ein Paketfilter auf einem GNU/Linux® System. Die Konfiguration des Filters und des Routings geschieht über Shell Skripte. Schwerpunkt wird der Einsatz des 2.4.x bzw. 2.6.x Kerns sein, der mit drei verschiedenen Paketfiltersubsystemen ausgestattet ist.[1]

3.2. Statische Paketfilter

Statische Paketfilter vergleichen sämtliche Pakete, die sie passieren, mit einem Satz von Kriterien und entscheiden anhand dieser Filterliste, was genau mit einem Paket geschieht. Die Kriterien müssen auf das jeweilige IP Paket anwendbar sein. Dabei geschieht dieser Vergleich einzeln und für jedes Paket gleich. Der statische Paketfilter zieht keine Informationen aus anderen Paketen für den Vergleich heran. Zulässige Filterkriterien sind daher

- Der Zugriff von außen auf den Telnet Service auf Port 23/TCP des Servers mit der IP Adresse `62.116.64.105` ist verboten.
- Jeder Rechner darf Daten auf den SMTP Port 25/TCP unseres Mailservers senden.
- Ein bestimmter Rechner darf eine Verbindung auf Port 22/TCP unseres Datenbankrechners öffnen.
- Nur Maschinen aus unserem internen Netzwerk `10.10.10.0/24` dürfen mit unserem DNS Server auf Port 53/TCP und 53/UDP reden.

Ein Beispiel für einen solchen Filter ist die *ipchains Firewall* des Linux® 2.2.x/2.4.x Kerns. Die obigen Paketfilter Regeln sehen dann beispielsweise so aus.

```
1 ipchains --insert input --interface eth0 --protocol tcp --destination $SOURCEIP --destination-port 23 --jump ↩
      DENY
2 ipchains --insert forward --protocol tcp --destination $MAILSERVER --destination-port 25 --jump ACCEPT
3 ipchains --insert forward --protocol tcp --source $WHITEHAT --destination $DBSERVER --destination-port 22 --↩
      jump ACCEPT
4 ipchains --insert forward --protocol tcp --source 10.10.10.0/24 --destination $DNS --destination-port 53 --jump↩
      ACCEPT
5 ipchains --insert forward --protocol udp --source 10.10.10.0/24 --destination $DNS --destination-port 53 --jump↩
      ACCEPT
```

[1] Eigentlich gibt es aus historischen Gründen drei verschiedene Paketfiltersysteme im Linux® Kern. Es wird unbedingt empfohlen sich gleich der neuesten Version namens *Netfilter* oder dessen Nachfolger *nftables* zu widmen.

Die Optionen des Kommandos wurden dabei ausgeschrieben.[2] –insert fügt die Regel in eine sogenannte *Chain* ein. Die Chains input, output und forward stellen dabei Default Chains für eingehenden, ausgehenden und geroutete Pakete dar. Der genaue Fluß ist in Abbildung 3.1 dargestellt. Pakete, die den Filter durchqueren, passieren alle drei Chains. Regeln, die Maschinen hinter einer solchen Firewall schützen, lassen sich beispielsweise in der forward Chain unterbringen.

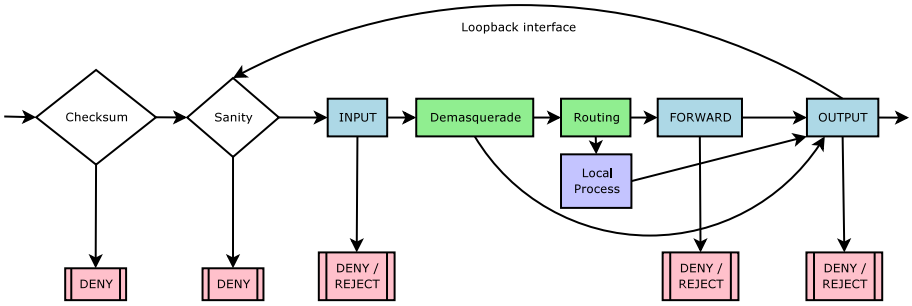


Abbildung 3.1.: Das Diagramm zeigt den Fluß der Pakete durch eine Linux® Ipchains Firewall. `input` wird von eingehenden Paketen passiert, `forward` betrifft weitergeleitete Pakete und durch `output` müssen alle ausgehenden Pakete. Je nach Paket werden daher eine bis alle drei Chains durchlaufen, was das Erstellen von Regeln erschwert.

Die Optionen –protocol, –source, –destination und –destination-port sind Beispiele für Kriterien. Trifft ein Kriterium zu, so wird das Paket mittels der Option –jump einer bestimmten Chain zugewiesen, die das weitere Schicksal des Pakets bestimmt. Es gibt hier wiederum Default Aktionen:

- **ACCEPT**
  Das Paket wird durchgelassen.
- **DENY**
  Das Paket wird verworfen. Der Absender des Paketes bekommt keine Notiz davon.
- **MASQ**
  Dies ist ein spezieller Fall für die forward Chain. Die Quelladresse des Pakets wird umgeschrieben bevor es weitergeleitet wird. Dies ist ein spezieller Fall von Network Address Translation (NAT).
- **REJECT**
  Das Paket wird verworfen, jedoch wird der Absender mit einer ICMP Nachricht davon informiert. Diese Methode ist nützlich für das Testen von Filtern und für das Blocken bestimmter Protokolle ohne Verzögerungen durch Timeouts zu verursachen.

Statische Paketfilter nehmen keine weitergehende Überprüfung des Paketinhalts vor (abgesehen von den Prüfsummen in den Paketen, aber selbst das wird nicht in allen Fällen gemacht) und prüfen keine Zusammenhänge zwischen den einzelnen Paketen. Beispielsweise merkt sich ein solcher Paketfilter aktive TCP Verbindungen nicht, so daß es keine Filterkriterien aufgrund des Zustandes angegeben werden können (fehlendes *connection tracking*). Auch Zusammenhänge zwischen Paketen und darauf folgende ICMP Meldungen werden nicht erkannt.

Statisches Filtern kann dennoch sinnvoll sein, da sich das System wenig bis keine Verbindungsdaten merken muß (NAT erfordert ein *connection tracking* und damit Resourcen wie RAM oder CPU-Zeit). Auf reinen Routern mit wenig Resourcen kann man so ein statisches Filtern durchaus hinzufügen.[3]

3.3. Zustandsgesteuerte Paketfilter

Zustandsgesteuerte Paketfilter führen Informationen über Pakete, die sie passiert haben, mit und vergleichen diese Aufzeichnungen gegebenenfalls mit weiteren Folgepaketen einer Verbindung oder einer Kommunikationssitzung. Damit lassen sich TCP Verbindungen „beobachten“, wodurch man die Injektion von Paketen aus einer dritten Quelle erschwert. Abbildung 3.2 zeigt die Zustandsinformationen einer Linux® Netfilter Firewall.

[2] Das sind die sogenannten *long options*, die in UNIX® Systemen aus Gründen der besseren Lesbarkeit eingeführt wurden.
[3] Natürlich muß man jeglichen Eingriff in Datentransport über Netzwerke immer mit der Performance abwägen.

```
[root@almeida /root]# cat /proc/net/ip_conntrack
tcp      6 326886 ESTABLISHED src=213.33.28.86 dst=192.168.10.252 sport=1455 dport=443 src=192.168.10.252 dst=213.33.28.86 sport=443 dport=1455 [ASSURED] use=1
tcp      6 90 SYN_SENT src=192.168.10.193 dst=192.168.1.107 sport=3286 dport=1234 [UNREPLIED] src=192.168.1.107 dst=192.168.10.193 sport=1234 dport=3286 use=1
tcp      6 403118 ESTABLISHED src=192.168.10.130 dst=213.202.66.7 sport=3467 dport=6346 src=213.202.66.7 dst=192.168.10.130 sport=6346 dport=3467 [ASSURED] use=1
tcp      6 383458 ESTABLISHED src=192.168.10.215 dst=66.35.208.15 sport=2645 dport=80 src=66.35.208.15 dst=192.168.10.215 sport=80 dport=2645 [ASSURED] use=1
tcp      6 141917 ESTABLISHED src=192.168.10.122 dst=24.0.135.80 sport=1200 dport=1214 src=24.0.135.80 dst=192.168.10.122 sport=1214 dport=1200 [ASSURED] use=1
tcp      6 37 SYN_SENT src=192.168.10.23 dst=192.168.0.3 sport=4718 dport=135 [UNREPLIED] src=192.168.0.3 dst=192.168.10.23 sport=135 dport=4718 use=1
tcp      6 431990 ESTABLISHED src=192.168.100.107 dst=192.168.20.28 sport=1040 dport=1723 src=192.168.20.28 dst=192.168.100.107 sport=1723 dport=1040 [ASSURED] use=1
tcp      6 115184 ESTABLISHED src=192.168.100.107 dst=192.168.20.28 sport=1041 dport=1723 src=192.168.20.28 dst=192.168.100.107 sport=1723 dport=1041 [ASSURED] use=1
tcp      6 431960 ESTABLISHED src=192.168.10.193 dst=140.180.158.84 sport=3281 dport=1214 src=140.180.158.84 dst=192.168.10.193 sport=1214 dport=3281 [ASSURED] use=1
udp      17 11 src=192.168.10.254 dst=192.168.10.11 sport=514 dport=514 [UNREPLIED] src=192.168.10.11 dst=192.168.10.254 sport=514 dport=514 use=1
udp      17 11 src=192.168.10.254 dst=192.168.10.13 sport=514 dport=514 [UNREPLIED] src=192.168.10.13 dst=192.168.10.254 sport=514 dport=514 use=1
tcp      6 431999 ESTABLISHED src=195.230.42.195 dst=192.168.10.13 sport=63493 dport=22 src=192.168.10.13 dst=195.230.42.195 sport=22 dport=63493 [ASSURED] use=1
tcp      6 41821 ESTABLISHED src=192.168.10.10 dst=192.168.0.210 sport=139 dport=1105 [UNREPLIED] src=192.168.0.210 dst=192.168.10.10 sport=1105 dport=139 use=1
unknown  47 496 src=192.168.100.107 dst=192.168.20.28 src=192.168.20.28 dst=192.168.100.107 use=1
tcp      6 431999 ESTABLISHED src=192.168.10.13 dst=192.168.10.254 sport=32904 dport=22 src=192.168.10.254 dst=192.168.10.13 sport=22 dport=32904 [ASSURED] use=1
tcp      6 404815 ESTABLISHED src=192.168.10.130 dst=213.6.143.152 sport=3753 dport=1214 src=213.6.143.152 dst=192.168.10.130 sport=1214 dport=3753 [ASSURED] use=1
udp      17 7 src=192.168.10.11 dst=10.10.10.1 sport=25192 dport=53 src=10.10.10.1 dst=192.168.10.11 sport=53 dport=25192 [ASSURED] use=1
tcp      6 388871 ESTABLISHED src=192.168.10.215 dst=66.35.208.15 sport=2064 dport=80 src=66.35.208.15 dst=192.168.10.215 sport=80 dport=2064 [ASSURED] use=1
tcp      6 25 TIME_WAIT src=192.168.10.13 dst=10.10.10.1 sport=32902 dport=3128 src=10.10.10.1 dst=192.168.10.13 sport=3128 dport=32902 [ASSURED] use=1
```

Abbildung 3.2.: Dies ist ein Schnappschuß der Zustandstabelle einer Netfilter Firewall. Man erkennt das Protokoll, den Zustand einer TCP Verbindung, Quelle und Ziel. Einträge mit dem Schlüssel `UNREPLIED` warten noch auf Pakete, `ESTABLISHED` kennzeichnet eine TCP Verbindung, `ASSURED` heißt, daß die Verbindung bzw. das Paket gültig ist. Die Zustandstabelle läßt sich in den Userspace exportieren und mit einem anderen Netfiltersystem synchronisieren.

Zustandsgesteuerte Paketfilter erhöhen die Komplexität einer Firewall und bedeuten eine zunehmende Beanspruchung der Resourcen, da nicht jeder neue Eintrag in die Zustandstabelle aufgrund der Paketlaufzeiten schnell wieder gelöscht werden kann. Sie vereinfachen jedoch das Erstellen von Filterregeln, da der Filtercode eine Reihe von Prüfungen vornimmt, die man dann leichter abfragen kann. Beispielweise kann man nach folgenden Kriterien filtern.

- **NEW**
  NEW bezeichnet Pakete, die bisher noch mit keiner Verbindung assoziiert sind.
- **ESTABLISHED**
  ESTABLISHED bezeichnet Pakete einer Verbindung, die schon Pakete in beide Richtungen detektiert hat.
- **RELATED**
  RELATED bezeichnet Pakete, die eine neue Verbindung starten, obwohl sie mit einer bestehenden in Zusammenhang stehen.
- **INVALID**
  INVALID bezeichnet ungültige Pakete. Vor Verwendung dieser Optione sollte man prüfen, ob dadurch keine legitimen Datentransmissionen aufgehalten werden.
- **unclean**
  Dieser Filter prüft Pakete auf bestimmte Abweichungen in IP Paketen (Checksummen, überlange IP Optionen, Nullports, zu kleine Header, etc.).
- **Tracking von Protokollen**
  FTP, IRC, TFTP, Amanda[4], SANE, SIP, ...

Diese Filter sind eine unschätzbare Hilfe und erleichtern das Erstellen von Filterkriterien sehr. Auch das Mitloggen von Paketen wird dadurch verbessert, denn man kann anhand der Zustandstabelle mitunter feststellen welche Pakete gesperrten Paketen vorausgegangen sind.

```
Sep 10 13:38:56 paladin kernel: IN= OUT=eth1 SRC=62.116.64.100 DST=192.168.10.111 LEN=120 TOS=0x00
  PREC=0xC0 TTL=255 ID=18600 PROTO=ICMP TYPE=11 CODE=0 [SRC=192.168.10.111 DST=213.47.27.79 LEN=92 TO
  S=0x00 PREC=0x00 TTL=1 ID=26183 PROTO=ICMP TYPE=8 CODE=0 ID=512 SEQ=12800 ]
```

Die obige Zeile stammt aus dem Log einer Netfilter Firewall. Aufgezeichnet ist die Emission einer ICMP `Time to Live exceeded in Transit` Fehlermeldung von `62.116.64.100` an `192.168.10.111`. Ausgelöst wurde diese Meldung durch das Paket, dessen Daten in den eckigen Klammern stehen. Das ursprüngliche Paket ging von `192.168.10.111` an `213.47.27.79` und war ein ICMP `Echo Request` (Typ 8, Code 0) mit TTL 1.

Es gibt Tools, die die Zustandstabelle auslesen und in eine besser verständliche Form bringen. Eines dieser Programme ist beispielsweise Conntrack Viewer[5]. Neuerdings kann man auch einen eigenen Dämon namens `conntrackd`

[4] Protokoll einer Backupsoftware
[5] http://cv.intellos.net/

des conntrack-tools[6] Pakets dafür verwenden. Die conntrack-tools erlauben Zugriff auf die Zustandstabelle (*connection tracking table*) und die Erwartungstabelle (*expect table*) des Linuxkerns. Man kann damit auch diese Tabellen zwischen zwei Systemen synchronisieren, um einen *active/active Failover* bei Ausfall eines Systems herbeizuführen.

Bei der Implementation von Filterregeln sollte man **immer** zustandsgesteuertes Filtern (*stateful inspection*) einschalten! Die meisten modernen Paketfilter unterstützen diesen Betriebsmodus. Der Linux® Netfilter verwendet zustandsgesteuertes Filtern **nur** wenn man die Option `-match state` und die dazugehörigen Zustände mit `-state` angibt.

3.3.1. Nftables Paketfiltersystem

Der Linux® Netfilter hat einige Nachteile, die im Nftables Paketfiltersystem verbessert wurden. Seit dem Kern 3.13 fand das Nftables System Eingang in den stabilen Zweig des Linux® Sources. Nftables ist eine virtuelle Maschine, die mittels Bytecode Netzwerkpakete inspiziert. Der Vorteil besteht in der protokollagnostischen Art und Weise, in der Netzwerkverkehr analysiert wird. Generell sind die Vorzüge wie folgt zu verstehen.

- weniger Duplizierung von Code im Linux® Kern
- schnellere Klassifikation von Paketen durch Sets und Maps
- einfachere Behandlung von Dual Stack IPv4/IPv6 Systemen
- bessere Unterstützung von dynamischen Regeln
- Netlink API für Software zum Andocken (wie beim normalen Networking und Netfilter)
- Vermeidung von inkonsistenten Beschreibungen bei der Syntax

Die Beschreibungssprache ist neu, und es wird auch ein neuer Befehl namens `nft` zum Anlegen und Modifizieren der Regeln verwendet. Mittels einer Kompatibilitätsschicht lassen sich aber bestehende Netfilter Skripte übernehmen.

Die wesentlichen Unterschiede zu Netfilter sind die folgenden:

- andere Syntax des `nft` Kommandos, angelehnt an `tcpdump`
- Tables und Chains sind vollständig konfigurierbar
- keine Unterscheidung zwischen Matches und Targets
- pro Regel können mehrere Aktionen konfiguriert werden
- Zähler für Pakete sind optional
- dynamische Updates der Regeln
- Sets, Maps, Dictionaries, Intervalle
- Verkettung von Regeln und Kombinieren mit Maps und Dictionaries
- Unterstützung neuer Protokolle ohne Linux® Kern Updates

3.3.2. BPFilter Paketfiltersystem

Mit dem Linux® Kern 4.18 haben Arbeiten begonnen, ein neues Paketfiltersystem zu implementieren. Es basiert auf dem Berkeley Packet Filter (BPF) Framework. Die Implementation heisst daher BPFilter. Der Blogartikel Why is the kernel community replacing iptables with BPF? hat zur Motivation einige Erklärungen.

[6] http://www.netfilter.org/projects/conntrack-tools/index.html

3.4. Content-basierte Paketfilter

Es gibt noch weitere Filtersysteme, die über die IP Ebene hinaus Prüfungen durchführen. Diese Filter verstehen Protokolle wie etwa HTTP, FTP oder SMTP. Damit läßt sich beispielsweise HTTP zwischen zwei Netzen verbieten, egal auf welchem Port es gesprochen wird. Dies ist sehr nützlich zum Unterbinden von Attacken, die über Protokolltunnel stattfinden, zumal man nicht immer davon ausgehen sollte, daß beispielsweise auf Port 25 ein SMTP Server lauscht.

Der Nachteil solcher Filtersysteme ist die höhere Komplexität, der erhöhte Bedarf an Resourcen und die Anpassung an das Protokoll. Was ist, wenn man ESMTP Transfers haben möchte, der Filter aber nur SMTP unterstützt? Ähnliche Überlegungen gelten analog für andere Protokolle.

Content-basierte Paketfilter sind mittlerweile weit verbreitet. Es gibt sogar ein Layer 7 System für den Linux® Netfilter, welches mittels Patches nachgerüstet werden kann.

4. Aufbau eines Paketfilters

> BOUNDARY, n. In political geography, an imaginary line between two nations, separating the imaginary rights of one from the imaginary rights of the other.
>
> --- The Devil's Dictionary, Ambrose Bierce

4.1. Sinn und Unsinn von Torwächtern

Moderne Netzwerke und Infrastrukturen sind zunehmend durchlässiger geworden. Modems waren früher eine Ausnahme, heute haben viele Laptops eingebaute bzw. per USB angebundene GSM/GRPS/UTMS-Modems. Dazu kommen drahtlose Netzwerke und unzählige andere Sicherheitslücken wie Browser, die Code von Webseiten lokal im LAN ausführen. Dieser Umstand macht Firewalls und Filter nicht sinnlos, allerdings sind sie nur mehr ein Teil des kompletten Sicherheitskonzepts. Es ist immer noch sinnvoll an Übergangspunkten Filter und Kontrollen einzusetzen. Allerdings gilt: Jede Sammlung von Sicherheitsmaßnahmen, die ohne Torwächter in sich zusammenfällt, stellt in sich eine Bedrohung dar und muß vermieden werden!

Man sollte aus diesem Grund bei der Planung von Firewalls und Filtern die Absicherung der Hosts im internen Netzwerk nicht vergessen. Zusätzlich ist es empfehlenswert für alle abzusichernden Datentransmissionen Protokolle einzusetzen, die „sich selbst verteidigen" können. Konzepte dafür werden bei der Absicherung von Hosts und drahtlosen Netzwerken besprochen.

4.2. Unterteilung der Netze durch Filter

Paketfilter sitzen, wie viele andere Sicherheitsmaßnahmen, oft an Grenzen zwischen verschiedenen Netzwerken. Die klassischste Trennung ist die zwischen Internet und lokalem Netzwerk. Wenn man noch Dienste nach außen ins Internet anbieten muß, dann faßt man alle diese Dienste in einem exponierten Netzwerk zusammen. Dieses Netzwerk wird oft „Perimeternetzwerk" oder „Demilitarisierte Zone" (DMZ) genannt.

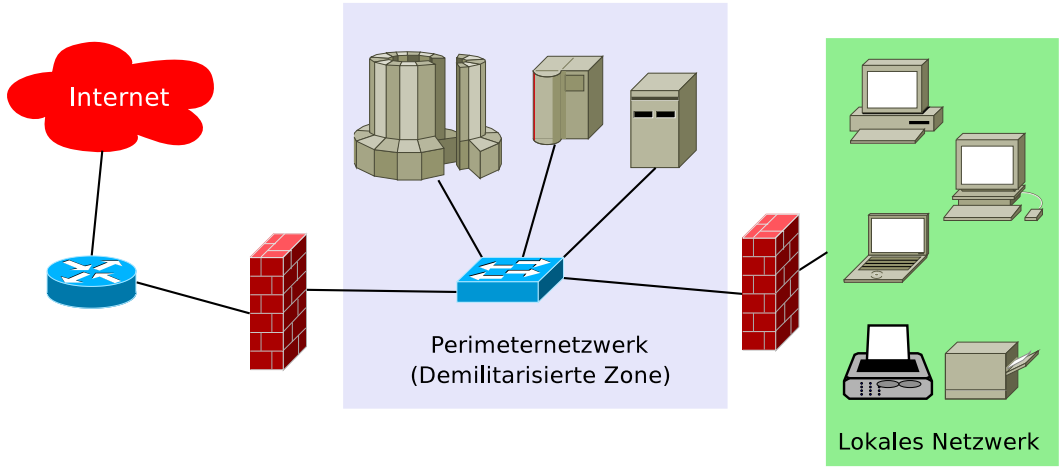


Abbildung 4.1.: Paketfilter eingesetzt als äußerer und innerer Filter; beide Paketfilter schotten das Perimeternetzwerk mit den exponierten Servern vom Internet und das lokale Netzwerk vor den beiden anderen Netzwerken ab.

Zwischen den Netzwerken besteht jeweils ein bestimmtes Vertrauensverhältnis. Klassisch gilt folgendes.

1. **lokales Netzwerk (LAN)** - vertrauenswürdig, abgeschottet, „sicher”
2. **Perimeternetzwerk (DMZ)** - weniger vertrauenswürdig, exponiert, gefährdet
3. **Internet, offenes WLAN** - nicht vertrauenswürdig, gefährlich

Die Nummerierung gibt dabei den Grad der Vertraulichkeit wieder. Für die Kommunikation gibt es nun bestimmte Richtlinien.

- Ein vertrauenswürdigeres Netzwerk darf Verbindungen in weniger vertrauenswürdigeres Netzwerk öffnen.
- Ein weniger vertrauenswürdigeres Netzwerk darf keine Verbindungen in ein vertrauenswürdigeres Netzwerk aufbauen.

Diese Richtlinien gelten immer. Sämtliche Ausnahmen, die in einer Konfiguration notwendig sind, müssen immer explizit angegeben werden. Wie dies genau geschieht, ist abhängig vom jeweiligen Filtersystem. Manche Paketfilter stellen zur Vereinfachung sogenannte „Sicherheitsstufen” (security level) zur Verfügung, wodurch Paketflüsse automatisch geregelt werden. Im Linux® Netfilter muß man diese Sicherheitsstufen durch passende Regeln nachstellen. Produkte, die auf dem Netfilter basieren, bringen meist auch Konfigurationen für Sicherheitsstufen mit.

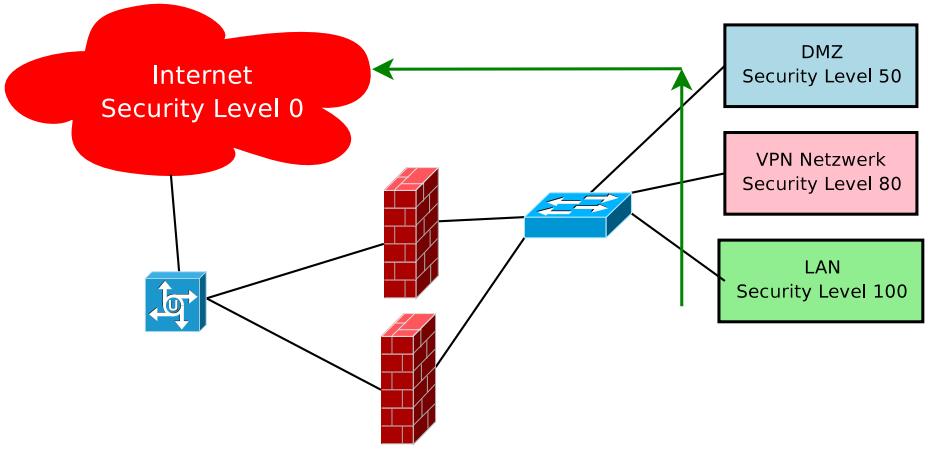


Abbildung 4.2.: Ein Paar von Paketfiltern setzt „security level” ein. Je höher der „security level”, desto vertrauenswürdiger ist das Netzwerk. Im vorliegenden Beispiel hat das LAN den höchsten und das Internet den niedrigsten. Es ergibt sich daher automatisch die Regelung, daß neue Verbindungen nur zu Zielen mit niedrigerem oder gleichen „security level” geöffnet werden dürfen. Der grüne Pfeil macht das deutlich.

4.3. Linux® Netfilter im Überblick

In einem früheren Kapitel wurden schon die Ipchains Filter des Linux® Kerns 2.2.x beschrieben. Der neue Netfilter Code des Linux® Kerns 2.4.x bringt eine Reihe von Änderungen mit sich.[1] Es gibt drei sogenannte *Packet Matching Tables*, die für bestimmte Aufgaben vorgesehen sind.

- **filter**
  Dies ist der Paketfilter bestehend aus den Chains `INPUT`, `FORWARD` und `OUTPUT`.
- **nat**
  In dieser Table kann man mit den drei Chains `PREROUTING`, `OUTPUT` und `POSTROUTING` Pakete verändern bevor und nachdem sie von der Routing Entscheidung des Kerns verteilt werden.

[1] Der Netfilter Code ist ebenso im Linux® Kern 2.6.x vorhanden.

- **mangle**
  Diese Table ist für bestimmte Modifikationen an Paketen vorgesehen und enthält die Chains `PREROUTING` und `OUTPUT`.

Der Fluß der Pakete durch den Netfilter ist im Vergleich mit dem Vorgänger in 2.2.x anders gelöst. Pakete, die weitergeleitet werden, passieren nicht mehr alle drei Chains. Abbildung 4.3 zeigt dieses Verhalten schematisch. [40] Damit vermeidet man redundante Regeln und vereinfacht die Notation der Filter. Dies ist ein wichtiger Punkt für die Übersichtlichkeit.

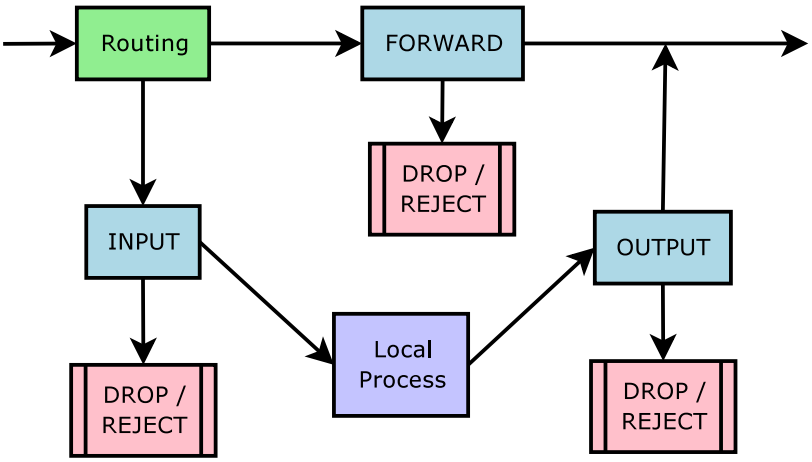


Abbildung 4.3.: Das Diagramm zeigt den Fluß der Pakete durch eine Linux® Netfilter Firewall. `INPUT` wird von eingehenden Paketen passiert, die für die Firewall selbst bestimmt sind. `FORWARD` betrifft ausschließlich weitergeleitete Pakete und durch `OUPUT` müssen alle ausgehenden Pakete, die die Firewall selbst generiert hat. Durch dieses Design wird vermieden, daß passierende Pakete mehr als eine dieser Chains passieren müssen. Aktionen wie Prüfen einer Checksumme oder Plausibilitätschecks sind nicht eingezeichnet, finden jedoch vor Verarbeitung eines Pakets statt.

Die Firewall selbst wird mit Hilfe der Filter in den Chains `INPUT` und `OUTPUT` beschützt. Sämtliche Pakete, die weitergeleitet werden, können durch Filterkriterien in `FORWARD` behandelt werden. Jede der Filter kann eine *Default Policy* haben, die bestimmt, was mit Paketen geschieht, auf die keine Filterregel anspricht. Die Default Policy und die Aktionen, die beim Zutreffen jeweils eines Filterkriteriums ausgeführt werden sollen, sind vielfältig und abhängig von der Table. Einige mögliche Aktionen sind:

- `ACCEPT`
  Das Paket wird unverändert durchgelassen

- `REJECT`
  Das Paket wird abgewiesen und eine entsprechende Reaktion wird an den Sender zurückgeschickt (normalerweise ein TCP RST oder eine ICMP Meldung). Die generierte ICMP Meldung kann explizit auf eine der folgenden Rückmeldungen
  - `icmp-net-unreachable`,
  - `icmp-host-unreachable`,
  - `icmp-port-unreachable`,
  - `icmp-proto-unreachable`,
  - `icmp-net-prohibited` oder
  - `icmp-host-prohibited`

  gesetzt werden. Dies ist notwendig, wenn eine Applikation sofort über geschlossene Ports informiert werden muß.

- `DROP`
  Das Paket wird abgewiesen und nichts wird an den den Sender zurückgeschickt. Das Paket verschwindet einfach. Dieses Verhalten ist sehr nützlich, um automatischen Port-Scannern oder Skripten das Leben zu erschweren. Gebräuchliche Ports, die man nicht haben möchte, sollte man jedoch mit `REJECT` abschirmen, um unnötige Verzögerungen zu vermeiden.

- `LOG`
  Passiert ein Paket den Filter, auf welches die Kriterien zutreffen, so wird das Paket in Form von den wichtigsten IP Header Informationen mitgeloggt.
  Es gibt noch die `ULOG` Option, bei der die Logdaten einer Applikation zur Auswertung übergeben werden.

- `MARK / CONNMARK`
  Dies ist der Vollständigkeit halber erwähnt. Mit dieser Aktion kann man einzelne Pakete oder alle Pakete einer Verbindung markieren, um sie von anderen Filtern speziell behandeln zu lassen. Damit kann man auch in das Routing eingreifen (*policy routing*).

- `MASQUERADE`
  Aktiviert das Maskieren eines bestimmten internen Netzes nach außen. Gedacht für dynamisch zugewiesene IP Adresse des Gateways (z.B. Dialup).

- `DNAT / SNAT / SAME`
  NAT oder NAPT nach Quell- oder Zieladresse.

- `REDIRECT`
  Umleiten eines Paketes auf eine lokale Netzwerkadresse, nützlich für transparente Proxies.

- `TCPMSS`
  Der TCP MSS Wert gibt bei ausgehenden Paketen an, wie groß Pakete sein dürfen, die vom lokalen System angenommen werden. Mit dieser Aktion kann man Probleme beheben, die durch falsche Absprache der MTU entstehen (dies kann durch Blocken von ICMP entstehen).

Es ist ebenso möglich als Aktion auf eine selbstdefinierte Chain weiterzugehen, wo dann weitere Prüfungen durchgeführt werden.

```
iptables --new-chain icmp
iptables --insert icmp   --destination 192.168.10.1 --prococol icmp
iptables --insert INPUT  --destination 192.168.10.1 --protocol icmp --jump icmp
```

4.4. Weitere Filterkriterien

Die Kriterien, nach denen der Linux® Netfilter Pakete filtern kann, sind sehr vielfältig. Hier ist eine Auswahl der wichtigsten Filtermethoden aufgelistet. Eine vollständige Liste findet man bei der Netfilter Dokumentation bzw. in den Linux® Kernel Sourcen.

- **IP-Adressen und Ports**
  z.B. für TCP und UDP; bei ICMP verwendet man Typ und Code

- **Protokolle**
  - TCP, UDP, ICMP
  - weitere Protokolle durch numerische Angabe
    Durchlassen von GRE[2] Paketen konfiguriert man beispielsweise wie folgt:

    ```
    iptables --insert FORWARD --protocol 47 --source 0/0 \
             --destination $GATEWAY --jump ACCEPT
    ```

- **Filtern von IPsec**
  AH bzw. ESP Header

- **Filtern nach ECN, DSCP**

[2] **G**eneric **R**outing **E**ncapsulation, Tunnelprotokoll von Cisco Systems.

- **Filtern nach MAC[3] Adresse**

```
iptables --insert INPUT --protocol tcp --source $SERVER \
         --destination $SELF --destination-port $SSH \
         --match mac --mac-source 00:60:97:11:d9:02 \
         --jump ACCEPT
```

- **Filtern nach lokaler ID**
  - User ID oder Group ID
  - Prozeß-ID
  - Paket generiert von einem bestimmten Kommando
- **Länge des Pakets**

```
iptables --insert INPUT --source 0/0 --destination $SELF \
         --match length --length 20:49152 --jump ACCEPT
```

  Obige Regel läßt Pakete mit der Länge von mindestens 20 Byte und höchstens 49152 Byte passieren.

- **Time To Live (TTL) des Pakets**

```
iptables --insert INPUT --source 0/0 --destination $SELF \
         --match ttt --ttl 254 --jump REJECT
```

  Obige Regel blockiert Pakete mit einer TTL von 254.

- **Type Of Service (TOS) Feld des Pakets**
- **Paketrate**
  Begrenzen des Paketflusses zu oder von bestimmten Adressen

```
iptables --insert INPUT --protocol udp \
         --source 0/0 --destination $SELF --destination-port $DNS \
         --match limit --limit 50/second --jump ACCEPT
```

  Obige Regel erlaubt eingehende UDP Pakete auf den eigenen DNS Port mit einer Rate von **durchschnittlich** 50 Paketen pro Sekunde (kurzzeitige Spitzen sind erlaubt).

- **Pakettyp**
  Multicast oder Broadcast Pakete

```
iptables --append INPUT --match pkttype --pkt-type broadcast --jump LOG
```

- **IP Blacklisting**
  Man kann Kriterien festsetzen, die dazu führen, daß IPs auf eine dynamische, interne Liste gesetzt werden, die man dann in weiteren Regeln verwenden kann.

```
iptables -A FORWARD -m recent --update --seconds 60 -j DROP
iptables -A FORWARD -i eth0 -d 127.0.0.0/8 -m recent --set -j DROP
```

  Obige Regeln setzen eine IP auf die interne Liste, sobald von dort versucht wird Pakete mit Zieladresse `127.0.0.1` über das externe Interface `eth0` einzuschleusen. Alle folgenden Pakete nach dem ersten gefälschten Paket werden für die Dauern von 60 Sekunden geblockt.

[3] MAC = **M**edia **A**ccess **C**ontrol

- **Zeichenketten**
  Man kann gezielt Pakete blocken, die bestimmte Zeichenketten enthalten.

  ```
  iptables --append FORWARD -p tcp --match string --algo bm \
           --string "google" -j REJECT
  ```

  Obiges Beispiel ist nicht sehr sinnvoll. Allerdings eignet sich diese Methode sehr gut zum Blocken von getunnelten Daten, indem man die äußeren Merkmale in Filterregeln faßt.

Der Linux® Netfilter Code wird ständig gepflegt und erweitert. Es gibt noch einige spezielle Anwendungen von Filtereigenschaften, die hier nicht im Detail beschrieben werden können. Für diese Anwendungen und Erweiterung sei auf die Webseite des Netfilter Projektes[4] verwiesen.

4.5. Vorgehensweise

4.5.1. Default Policies - Grundhaltungen

Direkt nach dem Initialisieren, also dem Laden der Kernmodule, befindet sich der Netfilter Paketfilter im Zustand alles ungehindert passieren zu lassen. Für den Aufbau von Filterregeln gibt es zwei prinzipielle Ansätze.

1. **einschränkende Grundhaltung**
   Sämtlicher Paketdurchfluß ist verboten. Die Filterregeln definieren erlaubten Netzwerkverkehr.
2. **freizügige Grundhaltung**
   Sämtlicher Paketdurchfluß ist erlaubt. Die Filterregeln definieren verbotenen Netzwerkverkehr.

Beide Ansätze sind zulässig. Der Ansatz der einschränkenden Grundhaltung wird jedoch oft gewählt, da man bei dieser Variante weniger übersehen kann (wenn es nicht funktioniert, dann fällt es automatisch auf). Übersetzt in ein kleines Shell Skript bedeutet das Setzen der Defaults in der einschränkenden Grundhaltung:

```
#!/bin/sh
#
# Setzen der Default Policy auf DROP für alle Chains

/sbin/iptables --policy INPUT   DROP
/sbin/iptables --policy FORWARD DROP
/sbin/iptables --policy OUTPUT  DROP
```

Der Paketfilter befindet sich nun in einem Zustand, in dem er keine Pakete jedweden Protokolls durchläßt. Ausgehend von dieser Situation kann man nun die Kriterien für zulässige Pakete zusammenstellen und die Regeln formulieren.

4.5.2. Reihenfolge der Regeln

Der Linux® Netfilter geht beim Prüfen von Paketen die Regeln der Chains von „oben“nach „unten“durch (analog zur Routingtabelle). Oben ist dabei immer Regel Nummer 1. Sobald eine Regel zutrifft, wird die entsprechende Aktion durchgeführt. Diese Vorgehensweise wird von vielen Paketfiltern gleichermaßen verwendet. Die beste Methode möglichst wenig falsch zu machen, besteht darin, daß man die Regeln immer an bestehende anhängt (Verwendung von `-append`). Dabei beginnt man mit den speziellen Regeln, die einzelne Maschinen oder Ports betreffen, und arbeitet sich zu den allgemeineren Regeln vor, die ganze Netzwerke oder die Welt betreffen. Zusammengefaßt:

- Immer Verwendung von `-append`.
- Spezielle Regeln für einzelne Maschinen und Ports zuerst.
- Allgemeine Regeln für ganze Netzwerke am Ende.

Die Verwendung von `-append` kommt dem Schreiben eines Skriptes sehr entgegen. Sobald eine Regel angehängt wurde, kann man das, was sie beschreibt, „vergessen“, denn für alle folgenden Regeln gilt es als schon behandelt.

[4] https://www.netfilter.org/

4.5.3. Beispiel eines Paketfilters

Ausgangspunkt ist eine Linux® Maschine mit 3 Netzwerkkarten. Sie soll als Filter und Router dienen. Sie soll an das Internet über eine Standleitung angeschlossen sein und hat direkte physische Verbindung zu den Servern und zum lokalen Netzwerk. Abbildung 4.4 zeigt eine Darstellung der Netzwerke. Für die weitere Betrachtung soll die Belegung der Netzwerkkarten wie folgt gegeben sein:

- externes Netz
  - `eth0`
  - `12.34.56.97`
  - `12.34.56.96/27` wird durchgeroutet
- Perimeternetzwerk
  - `eth1`
  - `12.34.56.96/27`
- lokales Netzwerk
  - `eth2`
  - `10.0.0.1`
  - `10.0.0.0/24`

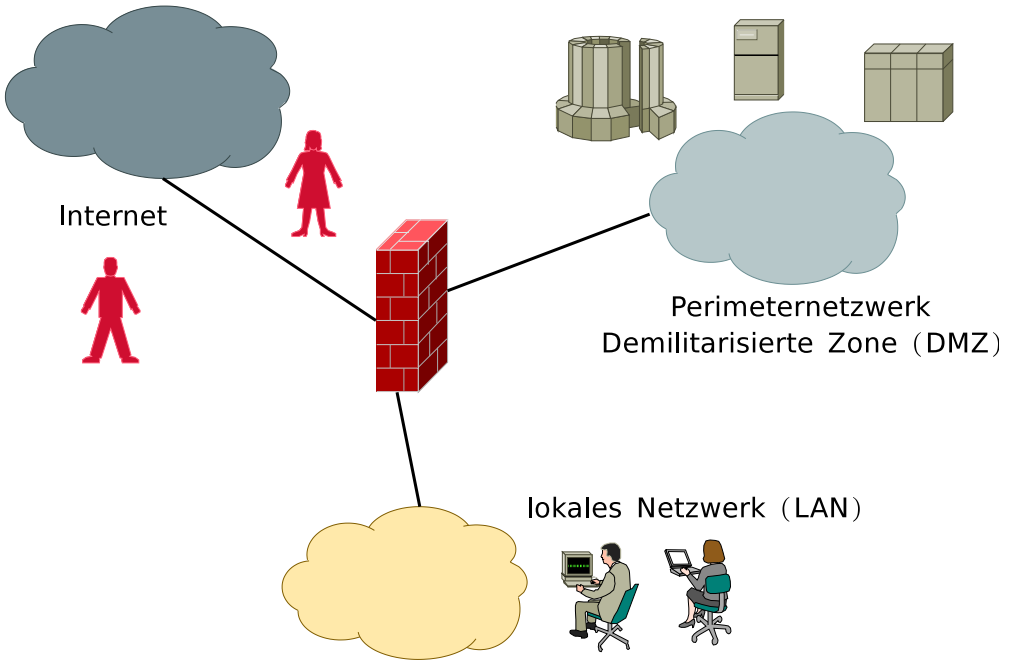


Abbildung 4.4.: Hier ist die Übersicht über den Linux® Paketfilter. Die Maschine ist mit 3 Netzwerkkarten an die jeweiligen Netze angeschlossen. `eth0` sei mit dem Internet, `eth1` mit dem Perimeternetzwerk und `eth2` mit dem lokalen Netz verbunden.

Damit man besser den Überblick behält und die Konfiguration handhabbar bleibt, sollte man drei Teile bilden.

- `rc.definition`
  Hier kommen alle Netzwerke, IP Adressen, Portnummern und dergleichen hinein. Diese Datei wird von den beiden anderen am Anfang aufgerufen, so daß man mit Shell Variablen arbeiten kann. Unter Debian GNU/Linux® Systemen bietet sich das Verzeichnis `/etc/defaults/` als Ablage für die Datei an.

- `rc.routing`
  Dieses Skript enthält alle Befehle, die benötigt werden, um das Routing korrekt zu setzen. Es sollte daher am besten als System V Start-/Stopskript geschrieben sein (oder beim Booten an passender Stelle aufgerufen werden). Darauf soll nicht näher eingegangen werden. Es gibt einige Maßnahmen, die man schon bei den Routing Einstellungen setzen kann. Ein Beispielskript befindet sich im Anhang.

- `rc.firewall`
  Nach dem Setzen der Routing Tabelle sollten die Paketfilterregeln definiert werden. Das Routing sollte erst nach vollständiger Konfiguration aller Filter aktiviert werden.

Wie nun die einzelnen Teile aussehen, bestimmt sich nach den Anforderungen an den Filter. Die Definitionen, die im Skript `rc.definition` gesetzt werden, lassen sich schon jetzt festlegen.

```
#!/bin/sh

# rc.definition - enthält Definitionen für spätere Skripte

# Kommandos

IPTABLES=/sbin/iptables
IP=/sbin/ip

# Paketfilter selbst

EXT_DEV=eth0
DMZ_DEV=eth1
LAN_DEV=eth2

# IP Adressen

EXT_IP="12.34.56.97"
DMZ_IP="12.34.56.98"
LAN_IP="10.0.0.1"

MAILSERVER="12.34.56.105"
WEBSERVER="12.34.56.106"
DNSSERVER="12.34.56.107"
FTPSERVER="12.34.56.108"
DBSERVER="12.34.56.109"

# Netzwerke

DMZ_NET="12.34.56.96/27"
LAN_NET="10.0.0.0/24"

EVERYWHERE="0.0.0.0/0"

# Ports

FTPCMD=20
FTPDATA=21
SSH=22
SMTP=25
DNS=53
HTTP=80
POSTGRES=5432

DYNAMIC_PORTS="1023:65534"
```

Richtung	Quelle	Ziel	Protokoll	Quellport	Zielport	ACK	Bemerkung
In	extern	intern	TCP	> 1023	22	-	Incoming Session Client an Server
Out	intern	extern	TCP	22	> 1023	ja	Incoming Session Server an Client
Out	intern	extern	TCP	> 1023	22	-	Outgoing Session Client an Server
In	extern	intern	TCP	22	> 1023	ja	Outgoing Session Server an Client

Tabelle 4.1.: Die Filtereigenschaften des SSH Protokolls sind hier dargestellt. Die Verbindungen, die mit Incoming Session bezeichnet sind, gelten für SSH Server. Die Spalte mit der Bezeichnung ACK markiert TCP Pakete mit gesetztem ACK Bit, welche zu bereits etablierten Verbindungen gehören (mit „ja“ gekennzeichnet). [35]

Wichtig: Bitte Vorsicht bei den Anführungszeichen bei „cut & paste” aus der PDF-Variante dieses Dokuments. In Shellskripten müssen ausschließlich doppelte Anführungszeichen verwendet werden. Sollte man sich die Skripte auf Systemen mit anderen Betriebssystemen vorschreiben wollen, so ist darauf zu achten, daß die Zeilenendenmarkierungen denen auf UNIX® Maschinen entsprechen. Im Zweifelsfall bitte die Skripte mit Kommandos wie `dos2unix` oder ähnlichen Werkzeugen konvertieren.

Vorbereitungen

Die Maschine, die als Router und Paketfilter dienen soll, muß korrekt installiert und abgesichert sein. Es sollten nur die absolut notwendigen Softwarepakete installiert sein. Es ist unbedingt zu vermeiden, daß nicht benutzte Netzwerkdienste und Programme deaktiviert oder besser gelöscht sind. Was nicht existiert, kann keine Probleme verursachen. [41]
Als nächstes muß man sich über die Protokolle Gedanken machen, die man durchlassen möchte. Angenommen wir haben im Perimeternetzwerk einen Mailserver, einen Webserver, einen FTP Server, einen DNS Server und einen Datenbankserver mit einer PostgreSQL Datenbank[5], wobei jede Maschine noch zusätzlich einen OpenSSH Server[6] für verschlüsselten Fernzugriff besitzt. Damit muß der Paketfilter die folgenden Protokolle durchlassen:

- File Transfer Protocol FTP (Port 20,21 TCP)
- Secure Shell OpenSSH (Port 22 TCP)
- Simple Mail Transfer Protocol SMTP (Port 25 TCP)
- Domain Name Service DNS (Port 53 TCP,UDP)
- Hyper Text Transfer Protocol HTTP (Port 80 TCP)
- PostgreSQL Datenbank (Port 5432 TCP) zu ausgewählten Maschinen

Die meisten Protokolle arbeiten über TCP, so daß man UDP eigentlich nur zum DNS Server durchlassen muß. Hat man Protokolle, wie sie im Audio- und Videobereich für Streaming Applikationen eingesetzt werden, dann muß man sich mitunter auch mit UDP auf anderen Ports auseinandersetzen. Für die einfachen verbindungsorientierten Protokolle (SSH, SMTP, HTTP, PostgreSQL) ergeben sich recht einfache Filterregeln, die man in einen Server- und einen Client-Teil aufteilen kann. Ein Beispiel für die notwendigen Filterkriterien ist in Abbildung 4.1 gegeben. Durch Austausch des Port 22/TCP (für SSH) durch 25/TCP (für SMTP) oder 80/TCP (für HTTP) lassen sich Regeln für andere Layer 7 Protokolle ableiten.

Um nun beispielsweise den SSH Zugriff auf alle Server im Perimeternetzwerk zu erlauben, bedarf es zweier Einträge in die Chain `FORWARD`.

```
$IPTABLES --append FORWARD --protocol tcp \
          --source $EVERYWHERE --source-port $DYNAMIC_PORTS \
          --destination $DMZ_NET --destination-port $SSH \
          --jump ACCEPT
$IPTABLES --append FORWARD --protocol tcp ! --syn \
          --source $DMZ_NET --source-port $SSH \
```

[5] https://www.postgresql.org/
[6] https://www.openssh.org/

```
          --destination $EVERYWHERE --destination-port $DYNAMIC_PORTS \
          --jump ACCEPT
```

Diese Filterregeln führen ein statisches Paketfiltern durch. Alternativ kann man den zustandsgesteuerten Paketfilter aktivieren, welcher alle Pakete auf die Zugehörigkeit zu bereits ausgehandelten Verbindungen prüft.

```
$IPTABLES --append FORWARD --protocol tcp \
          --source $EVERYWHERE --source-port $DYNAMIC_PORTS \
          --destination $DMZ_NET --destination-port $SSH \
          --match state --state NEW,ESTABLISHED \
          --jump ACCEPT
$IPTABLES --append FORWARD --protocol tcp \
          --source $DMZ_NET --source-port $SSH \
          --destination $EVERYWHERE --destination-port $DYNAMIC_PORTS \
          --match state --state ESTABLISHED \
          --jump ACCEPT
```

Wichtig: Bei komplexeren Filtern ist auf die Reihenfolge der Filterregeln zu achten! Darauf sollte man immer aufpassen, denn durch die Reihenfolge können sich mitunter für durchlaufende Pakete andere Ergebnisse ergeben. Es ist daher günstig, wenn man möglichst viel mit `-append` arbeitet und Regeln an Chains anhängt. Dadurch kann man davon ausgehen, daß Pakete von vorherigen Regeln schon abgefangen werden. Idealerweise formuliert man bei dieser Vorgehensweise die allgemeinen Regeln vorher und danach die speziellen für bestimmte Ports.
Der SSH Zugang auf die Paketfiltermaschine selbst wird durch die `INPUT` und `OUTPUT` Chains bestimmt.

```
$IPTABLES --append INPUT --protocol tcp \
          --source $EVERYWHERE --source-port $DYNAMIC_PORTS \
          --destination $EXT_IP --destination-port $SSH \
          --match state --state NEW,ESTABLISHED \
          --jump ACCEPT
$IPTABLES --append OUTPUT --protocol tcp \
          --source $EXT_IP --source-port $SSH \
          --destination $EVERYWHERE --destination-port $DYNAMIC_PORTS \
          --match state --state ESTABLISHED \
          --jump ACCEPT
```

Die Filtereigenschaften von SSH, HTTP, PostgresSQL und SMTP sind gleich, lediglich die Ports sind anders. Viele Protokolle, die auf einem fixen Port am Server angeboten werden, lassen sich so filtern. Dies gilt insbesondere für normale TCP Verbindungen. In unsere Liste gibt es zwei Ausnahmen, nämlich DNS und FTP.

Filtereigenschaften von FTP

Das FTP ist etwas älter und besitzt daher zwei mögliche Betriebsmodi - aktive Transfers und passive Transfers. Der Grund dafür liegt darin, daß FTP Server auf Port 21 nur Befehle empfangen. Die eigentlichen Daten werden entweder direkt vom Server über den Datenport 20 an den Client übermittelt (siehe dazu Abbildung 4.5), oder der Client initiiert eine Verbindung zum FTP Server für die Datenübertragung (Abbildung 4.6).

Aktives FTP kann daher nur zu Clients geschehen, die ungeschützt sind oder durch einen zustandgesteuerten Paketfiltern gehen. Weder ein Router mit NAT noch eine zustandslose Firewall kann aktive FTP Datenübertragungen zu den Empfängern verläßlich und gefahrlos durchschleusen. Beim passiven Datentransfer öffnet der Client die Datenverbindung zum Server nach vorheriger Absprache auf dem FTP Kommandokanal. Dies ist durch eine Firewall hindurch möglich, es hat jedoch zur Folge, daß am FTP Server eine ausreichend großer Portbereich für FTP Datenverbindungen zugänglich sein muß (der wiederum nur durch einen zustandgesteuerten Paketfilter verläßlich geschützt werden kann). Die vollständigen Regel für das Paketfiltern sind in Tabelle 4.2 ersichtlich.

Umgesetzt auf unser Beispiel bedeutet dies, daß wir zunächst den Zugriff auf den Kommandokanal ermöglichen müssen.

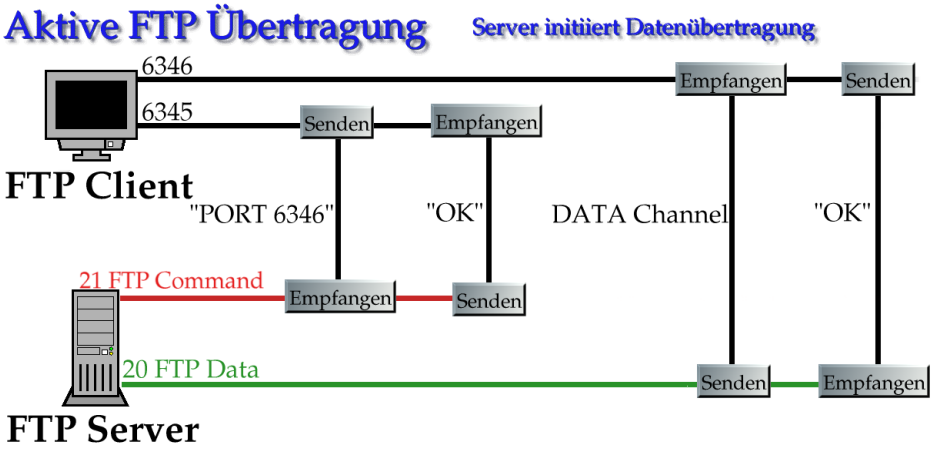


Abbildung 4.5.: Bei aktiven FTP Transfers initiiert der Server die Datenverbindung von Port 20 zu einem beliebigen Port am Client. Um diesen Transfer zuzulassen, müßte die Firewall Verbindungen von externen Servern mit Quelle Port 20 zu den Clients im LAN durchlassen. Dies geht weder durch einen NAT Router noch durch eine Firewall mit zustandslosem Filtern.

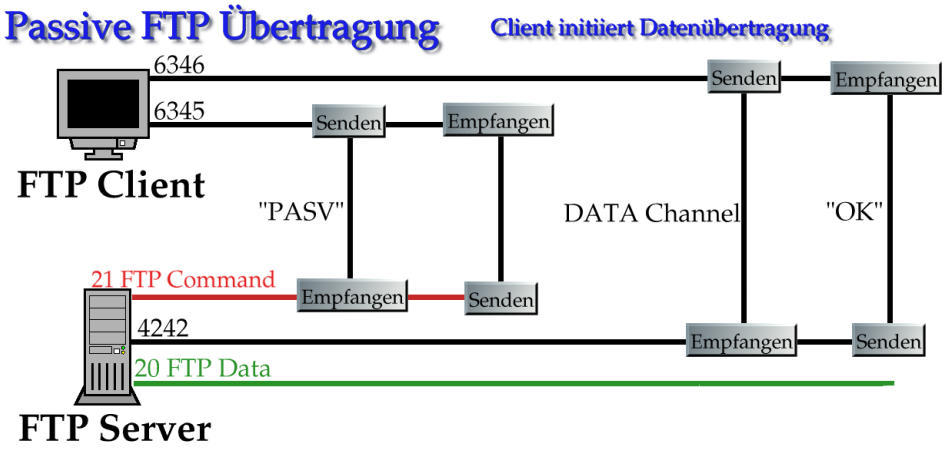


Abbildung 4.6.: Bei passiven FTP Transfers initiiert der Client die Datenübertragung. Dies ist ohne Schwierigkeiten durch eine Firewall hinweg möglich, da der Ursprung der Verbindung im LAN liegt. Die Problematik des Filterns verlagert sich damit auf die Seite des FTP Servers. Auch hier sollte man serverseitige einen zustandgesteuerten Paketfilter verwenden, da man ansonsten einen viel zu großen Portbereich freigeben muß.

Richtung	Quelle	Ziel	Protokoll	Quellport	Zielport	ACK	Bemerkung
In	extern	intern	TCP	> 1023	21	-	Incoming FTP Request
Out	intern	extern	TCP	21	> 1023	ja	Antwort an Incoming FTP Request
Out	intern	extern	TCP	20	> 1023	-	Datenkanalerzeugung Incoming (normaler Modus)
In	extern	intern	TCP	> 1023	20	ja	Datenkanalantworten Incoming (normaler Modus)
In	extern	intern	TCP	> 1023	> 1023	-	Datenkanalerzeugung Incoming (passiver Modus)
Out	intern	extern	TCP	> 1023	> 1023	ja	Datenkanalantworten Incoming (passiver Modus)
Out	intern	extern	TCP	> 1023	21	-	Outgoing FTP Request
In	extern	intern	TCP	21	> 1023	ja	Antwort auf Outgoing FTP Request
In	extern	intern	TCP	20	> 1023	-	Datenkanalerzeugung Outgoing (normaler Modus)
Out	intern	extern	TCP	> 1023	20	ja	Datenkanalantworten Outgoing (normaler Modus)
Out	intern	extern	TCP	> 1023	> 1023	-	Datenkanalerzeugung Outgoing (passiver Modus)
In	extern	intern	TCP	> 1023	> 1023	ja	Datenkanalantworten Outgoing (passiver Modus)

Tabelle 4.2.: Die Filtereigenschaften von FTP sind hier dargestellt. Aufgrund des aktiven und passiven Modus ergeben sich mehrere Filterkriterien. Besondere Aufmerksamkeit muß man den großen Portbereichen für die Datenübertragung schenken, da jenseits Port 1023 auch Services liegen können, die man beschützen muß. Mit statischem Paketfiltern alleine sind die Portbereiche viel zu groß, um sie absichern zu können. [35]

```
# Zugriff auf FTP Kommandokanal Port 21
$IPTABLES --append FORWARD --protocol tcp \
          --source $EVERYWHERE --source-port $DYNAMIC_PORTS \
          --destination $DMZ_NET --destination-port $FTPCMD \
          --match state --state NEW,ESTABLISHED \
          --jump ACCEPT
$IPTABLES --append FORWARD --protocol tcp \
          --source $DMZ_NET --source-port $FTPCMD \
          --destination $EVERYWHERE --destination-port $DYNAMIC_PORTS \
          --match state --state ESTABLISHED \
          --jump ACCEPT
```

Jetzt müssen wir den Datenkanal zugänglich machen und ausgehende Verbindungen erlauben. Es sollen allerdings nur Pakete zu einer Verbindungen passieren dürfen, die zu einer schon etablierten FTP Verbindung gehören. Dies wird mit dem Schlüsselwort `RELATED` erreicht.

```
# FTP Datenkanal Port 20 für aktives FTP
$IPTABLES --append FORWARD --protocol tcp \
          --source $DMZ_NET --source-port $FTPDATA \
          --destination $EVERYWHERE --destination-port $DYNAMIC_PORTS \
          --match state --state ESTABLISHED,RELATED \
          --jump ACCEPT
$IPTABLES --append FORWARD --protocol tcp \
          --source $EVERYWHERE --source-port $DYNAMIC_PORTS \
          --destination $DMZ_NET --destination-port $FTPDATA \
          --match state --state ESTABLISHED \
          --jump ACCEPT

# Datenübertragung bei passivem FTP
$IPTABLES --append FORWARD --protocol tcp \
          --source $EVERYWHERE --source-port $DYNAMIC_PORTS \
          --destination $DMZ_NET --destination-port $DYNAMIC_PORTS \
          --match state --state ESTABLISHED,RELATED \
          --jump ACCEPT
$IPTABLES --append FORWARD --protocol tcp \
          --source $DMZ_NET --source-port $DYNAMIC_PORTS \
          --destination $EVERYWHERE --destination-port $DYNAMIC_PORTS \
          --match state --state ESTABLISHED \
          --jump ACCEPT
```

Das Überwachen der Zugehörigkeit der Datenübertragungen zu aktiven FTP Sessions erledigt der Netfilter Code mit seiner Zustandstabelle. Dazu muß sichergestellt sein, daß das Modul `ip_conntrack_ftp` geladen ist. Wenn man sich nicht sicher ist, so sollte die Zeile

```
/sbin/modprobe ip_conntrack_ftp
```

am Anfang des Firewallskriptes stehen. Linux® Kerne mit neuerem Netfiltersystem müssen das Modul `nf_conntrack_ftp` laden (ab Kernversion 2.6.22).

Filtereigenschaften von DNS

Der Domain Name Service (DNS) überträgt Daten mittels UDP und TCP. UDP dient dabei für schnelle *Lookups* (Anfragen) und TCP für die Übertragung größerer Datenmengen (in sogenannten *Zone Transfers*, bei denen die Daten einer kompletten Domain gesendet werden).

- Normalerweise wird UDP benutzt.
- Sollten Daten verlorengehen, so kann ein Lookup über TCP wiederholt werden.
- DNS Pakete mit mehr als 512 Byte Größe werden über TCP transportiert.

Richtung	Quelle	Ziel	Protokoll	Quellport	Zielport	ACK	Bemerkung
In	extern	intern	UDP	> 1023	53	-	eingehende Anfrage externer Client an internen Server
Out	intern	extern	UDP	53	> 1023	-	Antwort interner Server an externen Client
In	extern	intern	TCP	> 1023	53	*	eingehende Anfrage externer Client an internen Server
Out	intern	extern	TCP	53	> 1023	ja	Antwort interner Server an externen Client
Out	intern	extern	UDP	> 1023	53	-	ausgehende Anfrage interner Client an externen Server
In	extern	intern	TCP	53	> 1023	-	Antwort externer Server an internen Client
Out	intern	extern	TCP	> 1023	53	*	ausgehende Anfrage interner Client an externen Server
In	extern	intern	TCP	53	> 1023	ja	Antwort externer Server an internen Client
In	extern	intern	UDP	53	53	-	eingehende Anfrage/Antwort Server $\leftrightarrow$ Server
Out	intern	extern	UDP	53	53	-	ausgehende Anfrage/Antwort Server $\leftrightarrow$ Server
In	extern	intern	TCP	> 1023	53	*	eingehende Anfrage oder Zonentransfer externer Server $\leftrightarrow$ interner Server
Out	intern	extern	TCP	53	> 1023	ja	Antwort (inkl. Zonentransfer) interner Server $\leftrightarrow$ externer Server
Out	intern	extern	TCP	> 1023	53	*	ausgehende Anfrage oder Zonentransfer interner Server $\leftrightarrow$ externer Server
In	extern	intern	TCP	53	> 1023	ja	Antwort (inkl. Zonentransfer) externer Server $\leftrightarrow$ interner Server

Tabelle 4.3.: Die Filtereigenschaften von DNS sind hier dargestellt. Die höhere Komplexität ergibt sich aus der Mischung zwischen UDP und TCP sowohl wie aus zusätzlich möglichen Server $\leftrightarrow$ Server Kommunikationen. [35]

- Zonentransfers[7] werden immer über TCP transportiert.

Eine Übersicht der Filtereigenschaften ist in Tabelle 4.3 ersichtlich. Serverport ist immer Port 53 für UDP und TCP. Manche Server nutzen den Port 53 als Quellport für Anfragen, andere Server können auch einen Port über 1023 verwenden. Der BIND[8] (Berkeley Internet Name Daemon) ab der Version 8.1[9] läßt eine Konfiguration des Ports für Anfragen zu (Option `query-source address * port *`). Durch die Natur der DNS Abfragen kann ein DNS Server sowohl Client als auch Server sein (Holen von Informationen von anderen DNS Servern, sowie Geben von DNS Auskünften an abfragende Clients; dies muß aber nicht bei jeder DNS Server Software so sein [10]).

Die einzelnen Filterregeln sehen nun wie folgt aus. Sie sind in Blöcke geordnet, die jeweils einem Abschnitt in Tabelle 4.3 entsprechen.

```
# DNS Anfragen externer Clients an Server
$IPTABLES --append FORWARD --protocol udp \
          --source $EVERYWHERE --source-port $DYNAMIC_PORTS \
          --destination $DNSSERVER --destination-port $DNS \
          --match state --state NEW --jump ACCEPT
$IPTABLES --append FORWARD --protocol udp \
          --source $DNSSERVER --source-port $DNS \
          --destination $EVERYWHERE --destination-port $DYNAMIC_PORTS \
          --match state --state NEW --jump ACCEPT
$IPTABLES --append FORWARD --protocol tcp \
          --source $EVERYWHERE --source-port $DYNAMIC_PORTS \
          --destination $DNSSERVER --destination-port $DNS \
          --match state --state NEW,ESTABLISHED --jump ACCEPT
$IPTABLES --append FORWARD --protocol tcp \
          --source $DNSSERVER --source-port $DNS \
          --destination $EVERYWHERE --destination-port $DYNAMIC_PORTS \
          --match state --state ESTABLISHED --jump ACCEPT

# DNS Anfragen an andere DNS Server
```

[7] Zonentransfers finden zwischen DNS Servern statt und kopieren alle Daten einer Domain, eben die sogenannte Zone.
[8] https://www.isc.org/products/BIND/
[9] Bitte BIND 8 nicht mehr verwenden! BIND 9 oder eine andere DNS Software ist ein absolutes Muß!
[10] Die `djbdns` DNS Package enthält jeweils eine eigene Software für eine bestimmte Teilaufgabe im DNS Service. [42]

```
$IPTABLES --append FORWARD --protocol udp \
          --source $DNSSERVER --source-port $DYNAMIC_PORTS \
          --destination $EVERYWHERE --destination-port $DNS \
          --match state --state NEW --jump ACCEPT
$IPTABLES --append FORWARD --protocol udp \
          --source $EVERYWHERE --source-port $DNS \
          --destination $DNSSERVER --destination-port $DYNAMIC_PORTS \
          --match state --state NEW --jump ACCEPT
$IPTABLES --append FORWARD --protocol tcp \
          --source $DNSSERVER --source-port $DYNAMIC_PORTS \
          --destination $EVERYWHERE --destination-port $DNS \
          --match state --state NEW,ESTABLISHED --jump ACCEPT
$IPTABLES --append FORWARD --protocol tcp \
          --source $EVERYWHERE --source-port $DNS \
          --destination $DNSSERVER --destination-port $DYNAMIC_PORTS \
          --match state --state ESTABLISHED --jump ACCEPT

# Server - Server
$IPTABLES --append FORWARD --protocol udp \
          --source $EVERYWHERE --source-port $DNS \
          --destination $DNSSERVER --destination-port $DNS \
          --match state --state NEW --jump ACCEPT
$IPTABLES --append FORWARD --protocol udp \
          --source $DNSSERVER --source-port $DNS \
          --destination $EVERYWHERE --destination-port $DNS \
          --match state --state NEW --jump ACCEPT

# DNS Zonentransfers sind schon in den obigen Regeln enthalten
```

Restliche Filterregeln

Die noch ausstehenden Protokolle SMTP/ESMTP, HTTP und PostgreSQL lassen sich analog zum Beispiel des SSH Protokolls filtern. Sie sind gute Beispiele für Standardfilter, da sich viele Client/Server-Protokolle nach diesem Muster filtern lassen können. Man könnte eine Shell Funktion schreiben, die diese Regeln erstellt.

```
# Allow inbound protocol
#
# $1 Protocol (tcp, udp)
# $2 Service (port number)
# $3 Target network or host (internal/DMZ)

allow_inbound()
{
   $IPTABLES --append FORWARD \
             --source $EVERYWHERE --source-port $DYNAMIC_PORTS \
             --destination $3 --protocol $1 --destination-port $2 \
             --match state --state NEW,ESTABLISHED \
             --jump ACCEPT
   $IPTABLES --append FORWARD \
             --source $3 --protocol $1 --source-port $2 \
             --destination $EVERYWHERE --destination-port $DYNAMIC_PORTS \
             --match state --state ESTABLISHED \
             --jump ACCEPT
}
```

Es lassen sich dann die weiteren Filter durch folgende Aufrufe erzeugen.

```
allow_inbound tcp $SMTP $MAILSERVER
```

```
allow_inbound tcp $HTTP $WEBSERVER
allow_inbound tcp $POSTGRES $DBSERVER
```

Damit wird das Shell Skript wesentlich übersichtlicher. Darauf sollte man immer achten, denn Paketfilter sind kompliziert genug. Je übersichtlicher die Konfiguration gestaltet ist, desto geringer ist die Wahrscheinlichkeit für Fehler.

4.5.4. ICMP

ICMP Pakete werden oft bei der Konfiguration von Paketfiltern vernachlässigt. Dies kann ein großer Nachteil sein, denn durch ICMP sind eine ganze Reihe von Proben möglich, die einem potentiellen Angreifer wichtige Informationen über ein Netzwerk geben können. [51] Je nach Sicherheitsstufe sollte man sich überlegen die folgenden ICMP Pakete zu blockieren.

- Eingehend:
  - ICMP Query Requests
    - ICMP Echo Request
    - ICMP Timestamp Request
    - ICMP Address Mask Request
    - ICMP Information Request
  - ICMP Query Replies
  - ICMP Error Messages
- Ausgehend:
  - ICMP Echo Reply (Typ 0)
  - ICMP Destination Unreachable Error Messages (Typ 3 Familie)
  - ICMP „Fragmentation Needed but the DF bit was set“(Type 3 Code 4)
  - ICMP Echo request (Typ 8)
    Dies kann erlaubt werden, wenn der Paketfilter zustandgesteuertes Filtern von ICMP zuläßt.
  - ICMP Time-To-Live Exceeded in Transit (Typ 11 Code 0)
    Verhindert traceroute und inverse Netzwerkerkundung.
  - ICMP Fragment Reassembly Time Exceeded (Typ 11 Code 1)
  - ICMP Parameter Problem
  - ICMP Timestamp Request & Reply
  - ICMP Address Mask Request & Reply

Details zum Ausnutzen dieser Pakete für Informationsbeschaffung sind in [51] beschrieben. Dort sind auch weiterführende Maßnahmen erwähnt, die man mit den Sicherheitsanforderungen an das zu schützende Netzwerk abstimmen muß. Besonderes Augenmerk sollte man auch auf die ICMP Redirect Pakete legen, da durch solche Meldungen Routing-Informationen an Hosts weitergegeben werden können.

Im Prinzip ist das Durchlassen nur derjenigen ICMP Meldungen, die mit Paketfragmentierung zu tun haben, völlig ausreichend. Im Zweifelsfalle beginnt man beim Filtern an diesem Minimum und arbeitet sich mit Ausnahmen weiter.

4.5.5. Mögliche Schlupflöcher

Trotz bester Absichten kann ein Paketfilter Lücken enthalten, die man eigentlich nicht möchte. Es kann beispielsweise durch eine falsche Reihenfolge der Regeln mehr erlaubt sein als geplant war. Weiterhin können Protokolle getunnelt werden, wie es bei IPv6 der Fall ist. IPv6 bietet die Möglichkeit durch IPv6-in-IPv4[11] Tunnel bestehende IPv4 Netzwerke zu durchqueren.

Microsoft® geht seit Windows Vista noch einen Schritt weiter und bietet mit dem Teredo Tunnelprotokoll einen IPv6 Anschluß über UDP an, welcher sogar NAT Router durchquert. Abbildung 4.7 illustriert den Weg der Pakete. Teredo Clients gibt es auch für andere Betriebssysteme (beispielsweise `miredo` für GNU/Linux Systeme). Man muß daher immer davon ausgehen, daß solche Hosts im Netzwerk präsent sind. Tatsächlich nutzen einige Bittorrent Clients absichtlich Protokolle wie Teredo, um Filter oder NAT zu umgehen.

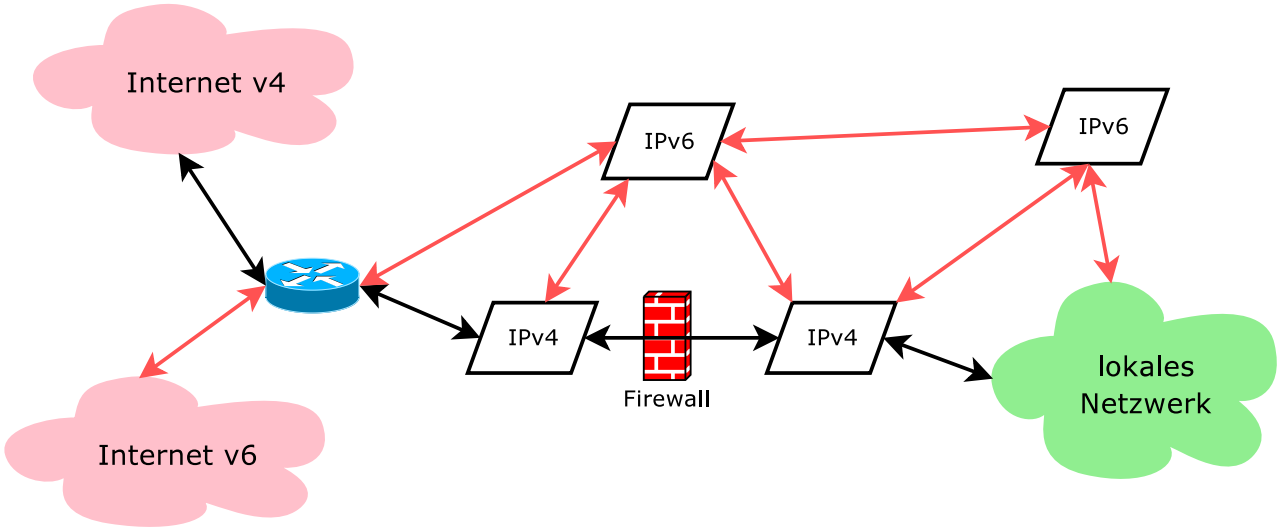


Abbildung 4.7.: Läßt man bestimmte Pakete von Clients nach außen zu, so können plötzlich Clients eine externe IPv6 Adresse erlangen und durch diese erreichbar werden. Protokolle wie beispielsweise der Teredo Tunnel ermöglichen das (Teredo nutzt UDP over IPv4, auch als UDPv4 abgekürzt). Ein ähnliches Problem tritt auf, wenn der Paketfilter zwar IPv4-Regeln hat, aber keine IPv6-Regeln und möglicherweise eine freizügige Grundhaltung. Im Zweifelsfall müssen diese Punkte mit den verwendeten Technologien und Produkten abgeglichen werden. Im Idealfall werden alle Konfigurationen auf solche Löcher periodisch überprüft.

Wichtig: Jede Firewall, die UDP ungehindert/unkontrolliert nach draußen läßt, erlaubt über diesen Mechanismus Tunnelprotokolle wie beispielsweise Teredo. Es ist daher ratsam immer nur exakt die Protokolle freizuschalten, die auch tatsächlich gebraucht werden.

4.5.6. Logging

Ein Filter, welcher nur filtert und keine Zustandsberichte liefert, ist im strengen Sinne eine unbekannte Größe in einem Netzwerk. Oft braucht man Informationen über geblockte Pakete oder bestimmte Pakete, die von außen in das Perimeternetzwerk hineinkommen. Dies kann einerseits dazu dienen, die eigenen Filter zu prüfen, andererseits erhält man auf diese Weise mitunter Frühwarnungen über bevorstehende Probleme.
Netfiler kann mit Hilfe der `LOG` Chain in das Systemlog schreiben (üblicherweise an den lokalen *syslogd*).

```
$IPTABLES --insert FORWARD --destination $DMZ_NET \
          --protocol icmp --icmp-type timestamp-request --jump DROP
$IPTABLES --insert FORWARD --destination $DMZ_NET \
          --protocol icmp --icmp-type timestamp-request \
          --match limit --limit 1/s --jump LOG
```

Mit diesen Regeln werden für das Perimeternetzwerk bestimmte ICMP Time Stamp Requests abgeblockt und gleichzeitig im Systemlog vermerkt. Die Beschränkung der Rate mit Hilfe der Limit Funktion ist ein Schutz vor Paketfluten, die in kurzer Zeit sehr viele Log-Einträge generieren können. Dieser Schutz sollte mit der I/O Performance des Filtersystem abgestimmt sein.
Man kann die Logeinträge zusätzlich mit einem Prefix versehen.

```
$IPTABLES --insert INPUT --protocol tcp \
          --destination $FIREWALL --destination-port 12345 \
          --match limit --limit 1/s \
          --jump LOG --log-prefix Netbus_Backdoor
```

Darüber hinaus können auch TCP oder IP Optionen mitgeloggt werden.

[11] IPv6-in-IPv4 ist Bestandteil der Simple Internet Transition (SIT) Migrationsprotokolle.

```
$IPTABLES --insert INPUT --protocol tcp \
          --destination $FIREWALL --destination-port 31337 \
          --match limit --limit 1/s \
          --jump LOG --log-tcp-options
$IPTABLES --insert INPUT --protocol udp \
          --destination $FIREWALL --destination-port 31337 \
          --match limit --limit 1/s \
          --jump LOG --log-ip-options
```

Logging kann unabhängig geschehen und wird auch oft als Methode verwendet, um Konfigurationen zu überprüfen. Die conntrack-tools bieten eine Möglichkeit direkt aus der Zustandstabelle des Netfilter Statistiken abzufragen.

4.5.7. Generelle Filter

Jeder Paketfilter sollte immer Regeln für folgende Ziele enthalten:

- nicht verwendete lokale Adressen
- nicht verwendete öffentliche Adressen
- Marspakete, sprich solche, die
  - von/zu RFC 1918 und vergleichbaren IPv6 Netzwerken kommen/gehen,
  - von/zu `127.0.0.0/8` und vergleichbaren IPv6 Netzwerken kommen/gehen,
  - als IPv4 Zieladresse `255.255.255.255/32` haben,
  - nicht von/zu „routbare" Adressen kommen/gehen (solche für die keine Route existiert)[12].

4.5.8. Filtern von IPv6

IPv6 bringt neben der Adressierung und Konfiguration auch für Paketfilter Neuerungen mit sich, die in Betracht gezogen werden müssen. Im Anbetracht der bei IPv6 eingesetzten Tunnelmechanismen muß speziell darauf auch geachtet werden. Nur autorisierte Systeme dürfen Tunnel aufbauen und unterhalten, speziell da sie Netzwerke miteinander verbinden.

Generell muß für Systeme dieselbe Sicherheitsrichtlinie gelten, egal ob sie IPv4-, IPv6- oder IPv4- und IPv6-Adressen haben.

Für Filterregeln existieren konkrete Empfehlungen.

- ICMPv6 Filter (RFC 4890)
  Die Neighbour Detection muß unbedingt erhalten bleiben. Dasselbe gilt für Routerankündigungen und die MTU Discovery (Meldung „packet too big"). RFC 4890 ist eine Empfehlung für Firewalls. [13].
- TCP ICMP Attacken[14]
- Einsatz von zustandsgesteuerten Filtersystemen
  Ein einzelner Client mit Teredo Tunnel läßt sich wie folgt einfach „beschützen" (die IPv4 Regeln fehlen, könnten aber genauso aussehen):

  ```
  ip6tables -P OUTPUT  DROP
  ip6tables -P INPUT   DROP
  ip6tables -P FORWARD DROP
  ip6tables -A OUTPUT -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
  ip6tables -A INPUT  -m state --state     ESTABLISHED,RELATED -j ACCEPT
  ```

  Analog könnte eine Firewall dem LAN den „outbound-only" Netzwerkverkehr und die Antworten erlauben.

  ```
  ip6tables -P FORWARD DROP
  ip6tables -A FORWARD -m state -s $LAN --state NEW,ESTABLISHED,RELATED -j ACCEPT
  ip6tables -A FORWARD -m state -d $LAN --state     ESTABLISHED,RELATED -j ACCEPT
  ```

  Diese Beispiele zeigen minimal notwendig Filterregeln, die allerdings nicht ganz vollständig sind.

[12] Pakete von/zu solchen Adressen nennt man *Bogon* und die dazugehörigen Netzwerke *Bogon Space*.
[13] Es gibt eine Umsetzung von Chris Hills in ein Bash Shell-Skript, zu finden auf seiner Webseite.
[14] https://tools.ietf.org/html/draft-ietf-tcpm-icmp-attacks-06

5. Testen eines Paketfilters

> beta test, v: To voluntarily entrust one's data, one's livelihood and one's sanity to hardware or software intended to destroy all three. In earlier days, virgins were often selected to beta test volcanoes.

Unmittelbar nach dem erstmaligen Hineinladen der Filterregeln sollte man testen, ob alle „lebensnotwendigen“Protokolle noch so funktionieren wie sie sollen. [1] Es gibt mehrere Möglichkeiten dies zu tun.

5.1. Beobachten mit Sniffern

Tools wie `tcpdump`[2] [6] oder Wireshark [44] können Pakete in Echtzeit von Netzwerkkarten abgreifen und in eine Datei speichern bzw. darstellen. Man kann damit beispielsweise interne und externe Netzwerkkarte beobachten, um anschließend festzustellen welche Pakete passieren und welche nicht.[3]

```
tcpdump -n -s 0 -i eth0 -w /root/packetdump.log
```

Dieses Kommando zeichnet alle Pakete auf, die die Netzwerkkarte `eth0` sieht. Die Pakete werden vollständig (`-s 0`) erfaßt, es wird kein DNS Lookup der IP Adressen durchgeführt (`-n`), und alle Daten werden in die Datei `/root/packetdump.log` geschrieben (Paket Logs sollten immer unzugänglich für normale Benutzer aufbewahrt werden). Die Daten lassen sich dann mit einem weiteren Aufruf von `tcpdump` oder durch Laden in Ethereal ansehen.

```
tcpdump -r /root/packetdump.log
```

Wireshark hat auch ein eigenes Tool für die Konsole, mit dem sich Netzwerkverkehr auffangen und darstellen läßt. Es nennt sich `tshark`. Die Syntax ist der von `tcpdump` sehr ähnlich. Zusätzlich versteht `tshark` die Logformate mehrerer Netzwerk-Analysatoren (Red Hat 6.1 `tcpdump`, SuSE 6.3 `tcpdump`, Nokia libpcap, Network Associates Sniffer, Sun snoop, Microsoft Network Monitor 1.x). `tethereal` interpretiert die Pakete bei der Ausgabe.

```
[lynx@luchs lynx]$ tshark -i eth0 -s 0 -n
Capturing on eth0
  0.000000 192.168.10.3 -> 192.168.10.255 UDP Source port: 631  Destination port: 631
  0.000094 192.168.10.3 -> 192.168.0.255 UDP Source port: 631  Destination port: 631
  0.000109 192.168.10.3 -> 10.10.10.2   UDP Source port: 631  Destination port: 631
  0.000450 192.168.10.1 -> 192.168.10.3 ICMP Destination unreachable
  1.718436 192.168.10.3 -> 195.230.42.194 ICMP Echo (ping) request
  2.710094 192.168.10.3 -> 195.230.42.194 ICMP Echo (ping) request
  4.994197 00:50:04:0d:c2:67 -> 00:60:97:11:d9:10 ARP Who has 192.168.10.3?  Tell 192.168.10.1
  4.994236 00:60:97:11:d9:10 -> 00:50:04:0d:c2:67 ARP 192.168.10.3 is at 00:60:97:11:d9:10
 22.000018 192.168.10.3 -> 192.168.10.255 UDP Source port: 631  Destination port: 631
 22.000095 192.168.10.3 -> 192.168.0.255 UDP Source port: 631  Destination port: 631
 22.000111 192.168.10.3 -> 10.10.10.2   UDP Source port: 631  Destination port: 631
 22.000401 192.168.10.1 -> 192.168.10.3 ICMP Destination unreachable
 26.999867 00:60:97:11:d9:10 -> 00:50:04:0d:c2:67 ARP Who has 192.168.10.1?  Tell 192.168.10.3
 27.000053 00:50:04:0d:c2:67 -> 00:60:97:11:d9:10 ARP 192.168.10.1 is at 00:50:04:0d:c2:67
```

Diese Art des Beobachtens von Netzwerkverkehr ist eine sehr wichtige Hilfe. Dies wird besonders nützlich, wenn es darum geht Filterregeln für undokumentierte Protokolle aufzustellen. Man kann auch eine Firewall damit aufbauen, indem man sämtlichen Netzverkehr aller erlaubten Datentransferoperationen aufzeichnet und anhand des Ergebnisses die Regeln aufstellt.

In Netzwerken mit Managed Switches gibt es die Möglichkeit den kompletten Netzwerkverkehr auf sogenannte *Mirror Ports* zu kopieren und dort abzugreifen.

Neben den hier vorgestellten Werkzeugen gibt es noch zahlreiche andere, insbesondere für Wireless LANs (beispielsweise Kismet, AirSnort).

[1] Dies gilt insbesondere für die Fernkonfiguration von Paketfiltern, für die man besondere Vorsicht und Vorkehrungen treffen sollte.

[2] Red Hat Linux® hat ein modifiziertes `tcpdump`, welches ein leicht abgeändertes Aufzeichnungsformat hat. Man sollte besser das „Original“von der `tcpdump` Web Site benutzen.

[3] Es gibt für `ipchains` dazu ein Tool namens Mason.

5.2. Generieren von Netzwerkverkehr

Netzwerktransmissionen lassen sich leicht mit sämtlicher netzwerkfähiger Software generieren. Im Prinzip reicht ein simpler Webbrowser oder ein FTP Programm. Manchmal möchte man allerdings den generierten Netzwerkverkehr steuern oder manipulieren. In diesen Fällen kommen spezielle Werkzeuge oder selbstprogrammierte Clients/Server in Frage[4].

5.2.1. hping

`hping`[5] ist ein Programm zum Erstellen von bestimmten IP Paketen eigener Wahl. UDP, TCP und ICMP Pakete können auf diese Weise zu Testzwecken versendet werden. Beispielsweise generiert man 4 TCP SYN Pakete, die einen Verbindungsaufbau bewirken, mit dem folgenden Kommando.

```
[root@luchs hping2]# ./hping2 -c 4 -p 53 -S gilean.luchs.at
HPING gilean.luchs.at (eth0 62.116.64.105): S set, 40 headers + 0 data bytes
len=46 ip=62.116.64.105 flags=SA DF seq=0 ttl=62 id=0 win=5840 rtt=8.5 ms
len=46 ip=62.116.64.105 flags=SA DF seq=1 ttl=62 id=0 win=5840 rtt=7.7 ms
len=46 ip=62.116.64.105 flags=SA DF seq=2 ttl=62 id=0 win=5840 rtt=7.8 ms
len=46 ip=62.116.64.105 flags=SA DF seq=3 ttl=62 id=0 win=5840 rtt=8.1 ms

--- gilean.luchs.at hping statistic ---
4 packets tramitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 7.7/8.0/8.5 ms
[root@luchs hping2]#
```

Die TCP Pakete wurden auf Port 53 geschickt. Man sieht die Antworten der Zielmaschine zusammen mit Round Trip Time (RTT), TTL und den TCP Flags SYN+ACK der Antwortpakete.

5.2.2. isic

ISIC[6] (*IP Stack Integrity Checker*) ist ein kleines Sammlung von Programmen welche IP, TCP, UDP, ICMP und Ethernet Pakete mit zufälligen Inhalt erzeugen und abschicken. Die zufälligen Daten beziehen sich auch auf den Header des Pakets, so daß man mittels ISIC in kurzer Zeit sehr viele ungültige Pakete erzeugen kann. Damit läßt sich die Durchlässigkeit eines Filters für ungültigen Netzwerkverkehr testen.

- „Kontrolliertes“Generieren von IP Paketen mit zufälligem Inhalt
- Quell- und Zieladresse werden festgelegt (nicht unbedingt erforderlich)
- Parameter in den Headern werden zufällig generiert
- Pakete können fragmentiert werden
  Die Angabe eines Prozentsatzes steuert den Anteil der fragmentierten Pakete.
- Streßtest für Firewalls und NIDS (*Network Intrusion Detection Systeme*)
  Der Autor der ISIC Tools konnte zwei kommerzielle Paketfilter damit außer Gefecht setzen.
  - Gauntlet Firewall Denial of Service Attack
    `http://www.securityfocus.com/bid/556`
  - Axent Raptor Denial of Service Vulnerability
    `http://www.securityfocus.com/bid/736`

Das Senden beliebiger TCP Pakete geht wie folgt.

```
tcpsic -s 192.168.10.3,25 -d 192.168.10.1 -D -p 1000
Compiled against Libnet 1.0.1b
Installing Signal Handlers.
Seeding with 16009
No Maximum traffic limiter
Using random destination ports.
Bad IP Version  = 10%           IP Opts Pcnt    = 50%
```

[4] Hier bieten sich Perl, Python oder C/C++ sehr gut an.
[5] http://www.hping.org
[6] http://www.packetfactory.net/Projects/ISIC

```
Frag'd Pcnt      = 30%           Urg Pcnt         = 30%
Bad TCP Cksm     = 10%           TCP Opts Pcnt    = 50%

192.168.10.3 -> 192.168.10.1 tos[254] id[0] ver[4] frag[0]
192.168.10.3 -> 192.168.10.1 tos[28] id[1] ver[4] frag[0]
192.168.10.3 -> 192.168.10.1 tos[159] id[2] ver[4] frag[12315]
192.168.10.3 -> 192.168.10.1 tos[142] id[3] ver[4] frag[20443]
192.168.10.3 -> 192.168.10.1 tos[156] id[4] ver[4] frag[18495]
192.168.10.3 -> 192.168.10.1 tos[86] id[5] ver[4] frag[0]
192.168.10.3 -> 192.168.10.1 tos[41] id[6] ver[4] frag[0]
192.168.10.3 -> 192.168.10.1 tos[216] id[7] ver[4] frag[0]
192.168.10.3 -> 192.168.10.1 tos[15] id[8] ver[4] frag[0]
192.168.10.3 -> 192.168.10.1 tos[87] id[9] ver[4] frag[0]
192.168.10.3 -> 192.168.10.1 tos[25] id[10] ver[4] frag[0]
192.168.10.3 -> 192.168.10.1 tos[216] id[11] ver[4] frag[0]
[...]
```

Ebenso lassen sich beliebige IP Pakete senden.

```
isic -s 192.168.10.3 -d 192.168.10.1 -F50 -V50 -I70 -p 15000
Compiled against Libnet 1.0.1b
Installing Signal Handlers.
Seeding with 9089
No Maximum traffic limiter
Bad IP Version  = 50%  Odd IP Header Length = 70%  Frag'd Pcnt = 50%

 1000 @ 13423.7 pkts/sec and 8267.7 k/s
 2000 @ 12783.8 pkts/sec and 8175.9 k/s
 3000 @ 12138.4 pkts/sec and 7826.7 k/s
 4000 @ 12383.4 pkts/sec and 8118.1 k/s
 5000 @ 13371.5 pkts/sec and 8217.2 k/s
 6000 @ 8743.7 pkts/sec and 5744.8 k/s
 7000 @ 13660.8 pkts/sec and 8572.7 k/s
 8000 @ 14410.9 pkts/sec and 9610.1 k/s
 9000 @ 5824.7 pkts/sec and 3653.3 k/s
 10000 @ 15199.4 pkts/sec and 9477.3 k/s
 11000 @ 12008.7 pkts/sec and 8160.6 k/s
 12000 @ 9317.9 pkts/sec and 5944.5 k/s
 13000 @ 13710.3 pkts/sec and 8685.8 k/s
 14000 @ 12385.7 pkts/sec and 8227.3 k/s

Wrote 15000 packets in 1.38s @ 10837.12 pkts/s
```

Wichtig: Da diese Tools auch Pakete mit falscher IP Adresse als Absender generieren können, sollte man sich sehr sicher sein, daß dieser „illegale“Netzwerkverkehr unter Kontrolle bleibt. Router und Paketfilter, die solche Pakete empfangen, können nämlich ICMP Fehlermeldungen an die fiktiven Absender generieren. Damit können falsche Alarme ausgelöst werden. Der Linux® Netfilter kann dieses Szenario wesentlich besser bewältigen als seine Vorgänger. Trotzdem sollten Tests mit den ISIC Tools daher immer gut kontrolliert und beobachtet werden. Es empfehlen sich alleine schon aus diesen Gründen „anti spoofing” Regeln zur Firewall hinzuzufügen.

5.2.3. nmap

`nmap`[7] ist ein sehr gut entwickelter Portscanner, der eine Vielzahl von Parametern zuläßt und Betriebssysteme anhand ihrer Reaktion auf Paketproben identifiziert. `nmap` beherrscht die folgenden Methoden:

- TCP connect() Scan
- TCP SYN, FIN, Xmas oder NULL Scan

[7] http://www.insecure.org/nmap

  - Ein SYN Scan sendet TCP SYN Pakete auf verschiedene Ports.
  - Ein FIN Scan sendet TCP FIN Pakete, die zu keiner aktiven Verbindung gehören.
  - Ein Xmas Scan sendet TCP Pakete mit gesetztem FIN, URG und PSH Flag.
  - Ein Null Scan sendet TCP Pakete ohne TCP Flags.

- TCP Scanning per FTP Proxy (Bounce Attack)
  RFC 959 erlaubt es zu einem FTP Server zu verbinden und diesen darum zu bitten, eine Datei zu einer *beliebigen* Adresse zu schicken. Damit lassen sich TCP Proben verbergen (unter anderen gefährlichen Dingen).
- SYN/FIN Scannen mit IP Fragmenten
- TCP ACK und Window Scannen
- UDP Scannen (ICMP Port Unreachable)
- ICMP Scannen (Ping Sweep)
- direktes RPC Scannen (nicht über `rpcinfo`)
- OS Identifikation
- Ident[8] Scannen bei TCP connect()

`nmap` besitzt eigene Algorithmen, um die Proben zu optimieren. `nmap` wird versuchen die Probe möglichst rasch durchzuführen. Man darf sich dabei aber nicht ganz auf die Software stützen, denn ein gut konfigurierter Paketfilter ist in der Lage `nmap` aufzuhalten, sofern man die Default Port-Bereich von `nmap` ohne nachdenken übernimmt. Es ist sinnvoll mit einer bestimmten Fragestellung an eine Probe heranzugehen und entsprechende `nmap` Optionen zu wählen.
Wichtig: Man sollte unbedingt nur Maschinen mit vollen `nmap` Scans bearbeiten, sofern man das *Einverständnis des Betreibers hat*! `nmap` erzeugt Megabytes an Logs auf Intrusion Detections Systemen, was mitunter einige interessante Telefonate nach sich ziehen kann.
`nmap` erzeugt sehr gute Übersichten. Eine Linux® Maschine ohne Paketfilterschutz läßt sich beispielsweise sehr schnell scannen und erfassen.

```
# nmap (V. 2.54BETA29) scan initiated Fri Oct 12 11:35:55 2001 as:
# nmap -sT -I -O -P0 -oN /tmp/luchs.txt 127.0.0.1
Interesting ports on localhost.localdomain (127.0.0.1):
(The 1532 ports scanned but not shown below are in state: closed)
Port       State       Service                Owner
22/tcp     open        ssh
23/tcp     open        telnet
25/tcp     open        smtp
53/tcp     open        domain
110/tcp    open        pop-3
111/tcp    open        sunrpc
139/tcp    open        netbios-ssn
143/tcp    open        imap2
389/tcp    open        ldap
587/tcp    open        submission
631/tcp    open        cups
953/tcp    open        rndc
993/tcp    open        imaps
3306/tcp   open        mysql
6000/tcp   open        X11
7100/tcp   open        font-service

No exact OS matches for host (If you know what OS is running on it,
see http://www.insecure.org/cgi-bin/nmap-submit.cgi).
TCP/IP fingerprint:
SInfo(V=2.54BETA29%P=i686-pc-linux-gnu%D=10/12%Time=3BC6B981%O=22%C=1)
TSeq(Class=RI%gcd=1%SI=301444%IPID=Z%TS=100HZ)
TSeq(Class=RI%gcd=1%SI=3014C0%IPID=Z%TS=100HZ)
TSeq(Class=RI%gcd=1%SI=3014B2%IPID=Z%TS=100HZ)
T1(Resp=Y%DF=Y%W=7FFF%ACK=S++%Flags=AS%Ops=MNNTNW)
T2(Resp=N)
T3(Resp=Y%DF=Y%W=7FFF%ACK=S++%Flags=AS%Ops=MNNTNW)
T4(Resp=Y%DF=Y%W=0%ACK=O%Flags=R%Ops=)
T5(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
```

[8] Identification Protocol, RFC1413, http://dungeon.luchs.at/RFC/rfc1413.txt

```
T6(Resp=Y%DF=Y%W=0%ACK=O%Flags=R%Ops=)
T7(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
PU(Resp=Y%DF=N%TOS=C0%IPLEN=164%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%DAT=E)

Uptime 13.416 days (since Sat Sep 29 01:37:39 2001)

# Nmap run completed at Fri Oct 12 11:36:02 2001
# 1 IP address (1 host up) scanned in 7 seconds
```

Hier ist die Signatur des auf der Maschine laufenden Linux® 2.4.10 offenbar zu neu für eine Identifikation. Dasselbe kann man nun auch für UDP durchführen (mit Erkennung von RPC Diensten `-sU`).

```
# nmap (V. 2.54BETA29) scan initiated Fri Oct 12 11:39:47 2001 as:
# nmap -sRU -P0 -oN /tmp/luchs2.txt 127.0.0.1
Interesting ports on localhost.localdomain (127.0.0.1):
(The 1444 ports scanned but not shown below are in state: closed)
Port       State       Service (RPC)
53/udp     open        domain
67/udp     open        bootps
111/udp    open        sunrpc
137/udp    open        netbios-ns
138/udp    open        netbios-dgm
514/udp    open        syslog
600/udp    open        ipcserver
650/udp    open        bwnfs
2049/udp   open        nfs

# Nmap run completed at Fri Oct 12 11:39:53 2001 --
# 1 IP address (1 host up) scanned in 6 seconds
```

Es ist zu beachten, daß `nmap` in beiden Fällen nur eine Auswahl aller möglichen Ports einem Test unterzieht. Ein Vergleich mit dem Tool `rpcinfo` ergibt noch weitere Ports, die `nmap` entgangen sind.

```
[user@localhost user]$ /usr/sbin/rpcinfo -p 127.0.0.1
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100011    1   udp    600  rquotad
    100011    2   udp    600  rquotad
    100005    1   udp  33497  mountd
    100005    1   tcp  40641  mountd
    100005    2   udp  33497  mountd
    100005    2   tcp  40641  mountd
    100005    3   udp  33497  mountd
    100005    3   tcp  40641  mountd
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100021    1   udp  33498  nlockmgr
    100021    3   udp  33498  nlockmgr
    100021    4   udp  33498  nlockmgr
    100024    1   udp  33499  status
    100024    1   tcp  40642  status
```

Weiterhin erfolgt die Identifikation des Services ausschließlich durch die Portnummer. Das bedeutet, daß die Angabe von `domain` oder `syslog` im obigen Beispiels seperat verifiziert werden muß.
Neuere `nmap` Versionen können auch die Version der Applikation aus dem Connect String herauslesen:

```
[root@agamemnon ~]# nmap -sT -sV 192.168.1.2

Starting nmap 3.77 ( http://www.insecure.org/nmap/ ) at 2004-11-23 03:06 CET
Interesting ports on packetsink.foobar.net (192.168.1.2):
(The 1647 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE     VERSION
22/tcp   open  ssh         OpenSSH 3.8.1p1 (protocol 2.0)
25/tcp   open  smtp        Postfix smtpd
53/tcp   open  domain
111/tcp  open  rpcbind     2 (rpc #100000)
139/tcp  open  netbios-ssn Samba smbd (workgroup: FOOBAR)
143/tcp  open  imap        Dovecot imapd
515/tcp  open  printer
631/tcp  open  ipp         CUPS 1.1
804/tcp  open  mountd      1-3 (rpc #100005)
873/tcp  open  rsync       (protocol version 28)
```

```
931/tcp  open  status      1 (rpc #100024)
953/tcp  open  rndc?
993/tcp  open  ssl/imap    Dovecot imapd
2049/tcp open  nfs         2-4 (rpc #100003)
3128/tcp open  http-proxy  Squid webproxy 2.5.STABLE6
8080/tcp open  http-proxy  Squid webproxy 2.5.STABLE6
MAC Address: 00:10:5A:68:E9:95 (3com)

Nmap run completed -- 1 IP address (1 host up) scanned in 24.014 seconds
[root@agamemnon ~]#
```

5.2.4. Seagull

Seagull[9] ist ein Generator für eine ganze Reihe von Protokollen, darunter auch solche aus der Schicht 7.

- Diameter[10] über TCP oder SCTP
- TCAP ITU und ANSI
- XCAP über HTTP in IPv4
- HTTP über IPv4
- H248/Megaco ASCII über UDP, TCP oder SCTP
- RADIUS über IPv4

5.3. Belastungstests und Auditing

Auch wenn Server an einer WAN-Leitung angeschlossen sind, die nicht die Geschwindigkeit der Netzwerkkarte hat, so sollte man durchaus bis zur Grenze des Möglichen gehen und Belastungstests durchführen. Anbindungen von 100 Mbit/s oder 1 Gbit/s sind mittlerweile durch Server Housing in Rechenzentren für wenig Geld machbar. Es ist dann umso wichtiger sicherzustellen, daß das System bis an die Grenzen gehen kann ohne auszufallen. Dies gilt für Paketfiltersysteme und Server gleichermaßen.

5.3.1. Verhalten unter Last

Paketfilter sind oft ein wichtiges Bindeglied zwischen Netzwerken. Ihr Ausfall hat meist Folgen, daher sollte man neben den bisher beschriebenen Methoden auch Belastungstests durchführen. In diesem Falle geht es weniger darum Pakete zu generieren, die aufgehalten werden sollen, sonder möglichst viele Pakete, die der Paketfilter passieren lassen muß. Besonderes Augenmerk ist dabei auf den Durchsatz zu richten. Protokolle wie HTTP, FTP oder SMTP können hohen Datendurchsatz erfordern, den der Filter bewältigen muß. Es gibt mehrere Möglichkeiten dies zu tun.

- `iperf`[11]
  ist ein Client/Server-Tool zum Messen von reinem Datendurchsatz via TCP und UDP. Es überträgt Pakete mit fixer Länge, die mit festgelegten Werten befüllt sind.
- NetPIPE[12]
- `ntop`[13]
  ist ein sehr leistungsfähiger Netzwerkmonitor, der Raten und Netzaktivität wiedergibt.
- `netperf`[14]
  ist ein Tool von Hewlett Packard, mit dem man den Datendurchsatz testen kann.

[9] http://gull.sourceforge.net/
[10] Beschrieben in RFC 3588, Nachfolger von RADIUS.
[11] http://dast.nlanr.net/Projects/Iperf/
[12] http://www.scl.ameslab.gov/netpipe/
[13] http://www.ntop.org/
[14] http://www.netperf.org/netperf/NetperfPage.html

Die üblichen WAN-Geschwindigkeiten an Standleitungen sind derzeit geringer als 100 Mbit/s, deswegen sind interne Paketfilter (z.B. Filter an der Grenze zwischen Perimeternetzwerk und LAN) höheren Belastungen ausgesetzt. Gigabit-Netzwerke in LANs sind ebenfalls nicht mehr so selten wie noch vor einigen Jahren, und üblicherweise liegt ein Paketfilter zwischen LAN und DMZ. Bei Gigabit-Ethernet muß man jedoch auch ohne Filter schon zu Feinabstimmung bei den Netzwerkeinstellungen greifen.

5.3.2. Simulation von Angriffen

Neben den zahlreichen Skripts, die bestimmte Schwachstellen ausnutzen, gibt es eine Reihe von Scannern, die Attacken und weitergehende Tests durchführen. Ein sehr bekannter Open Source Scanner ist Nessus[15] (bis zur Version 2) oder sein Nachfolger OpenVAS[16]. Die Vorzüge von beiden sind:

- Client/Server Architektur
  ⟶ Client und Server kommunizieren verschlüsselt
  ⟶ Einsatz von Nessus Scannern mit Remote Zugriff
- Möglichkeit von Plug-In Modulen
  Vulnerability Datenbank - sehr aktuell
- NASL - Nessus Attack Scripting Language
- gleichzeitiges Testen beliebiger Hosts
  ⟶ abhängig von der Performance des Nessus Servers
  ⟶ konfigurierbar
- Service Erkennung
  Nessus prüft Service bei Connect
  ⟶ Entdecken von Services auf Nicht-Standard-Ports
  ⟶ entsprechendes Wiederholen von Tests
- Knowledge Base beim Scannen
  Plugins teilen die ermittelten Informationen über Hosts
  ⟶ Optimierung der Scans
- übersichtliche Reports
  NSR, ASCII Text, HTML, XML, LaTeX
- unabhängige Developer, breiter Support
  Test-Plugins sind Stunden nach Veröffentlichung eines Problems auf BugTraq, NTBugTraq, Incidents, Vuln-dev, Technotronic, . . . verfügbar; Subskriptionen werden von Tenable Network Security angeboten
- Scannen von Cron Jobs aus
  - Nessus Client läßt sich im Batch Modus einsetzen
  - automatischer Ablauf von periodischen Checks

Darüber hinaus lassen sich konkrete Angriffe mit dem Werkzeug Metasploit[17] durchführen. Metasploit ist eine Sammlung von Software zum Testen und Entwerfen von Angriffen. Das Projekt beschreibt sich selbst so.

> The Metasploit Framework is a development platform for creating security tools and exploits. The framework is used by network security professionals to perform penetration tests, system administrators to verify patch installations, product vendors to perform regression testing, and security researchers world-wide. The framework is written in the Ruby programming language and includes components written in C and assembler.
>
> ...
>
> The framework consists of tools, libraries, modules, and user interfaces. The basic function of the framework is a module launcher, allowing the user to configure an exploit module and launch it at a target system. If the exploit succeeds, the payload is executed on the target and the user is provided with a shell to interact with the payload.

[15] http://www.nessus.org/
[16] http://wald.intevation.org/projects/openvas/
[17] http://www.metasploit.com/

Metasploit ist sowohl für GNU/Linux als auch für Microsoft® Windows® Systeme verfügbar.

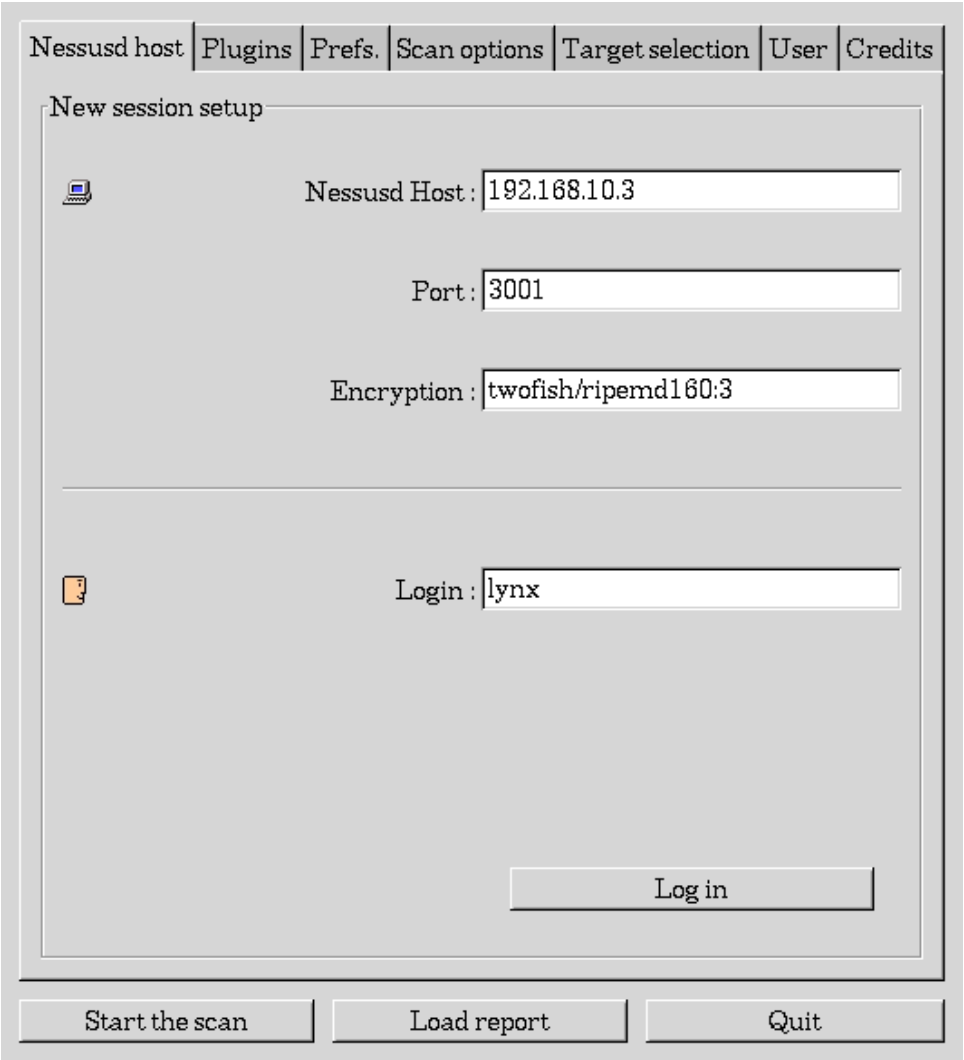


Abbildung 5.1.: Login mit dem Nessus Client. Nessus verfügt über eine eigene Paßwortdatenbank zur Authorisierung. Die Kommunikation zwischen Server und Client ist ebenso verschlüsselt.

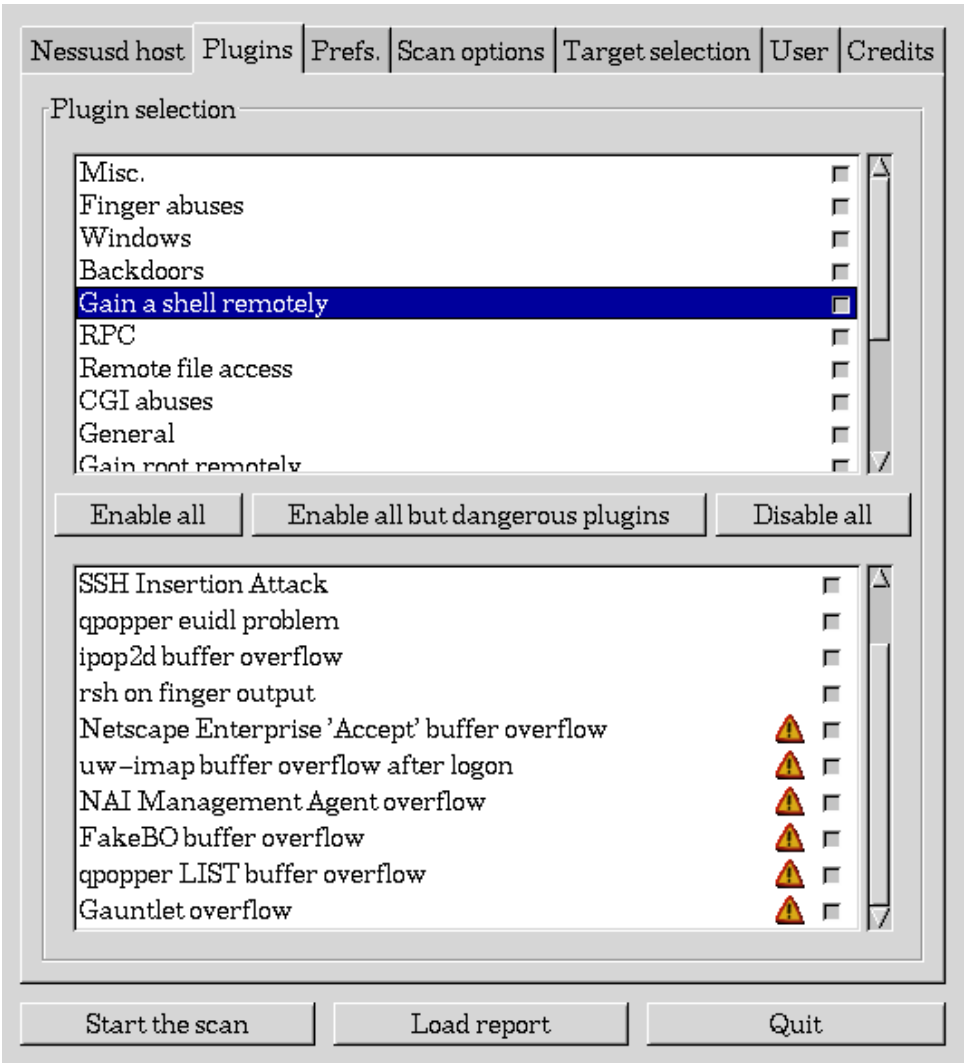


Abbildung 5.2.: Auswählen der zu verwendenen Plugins für den Scan. **Wichtig:** Manche Module können Systeme zum Absturz bringen oder Netzwerk-Equipment wie Router oder Print-HUBs lahmlegen.

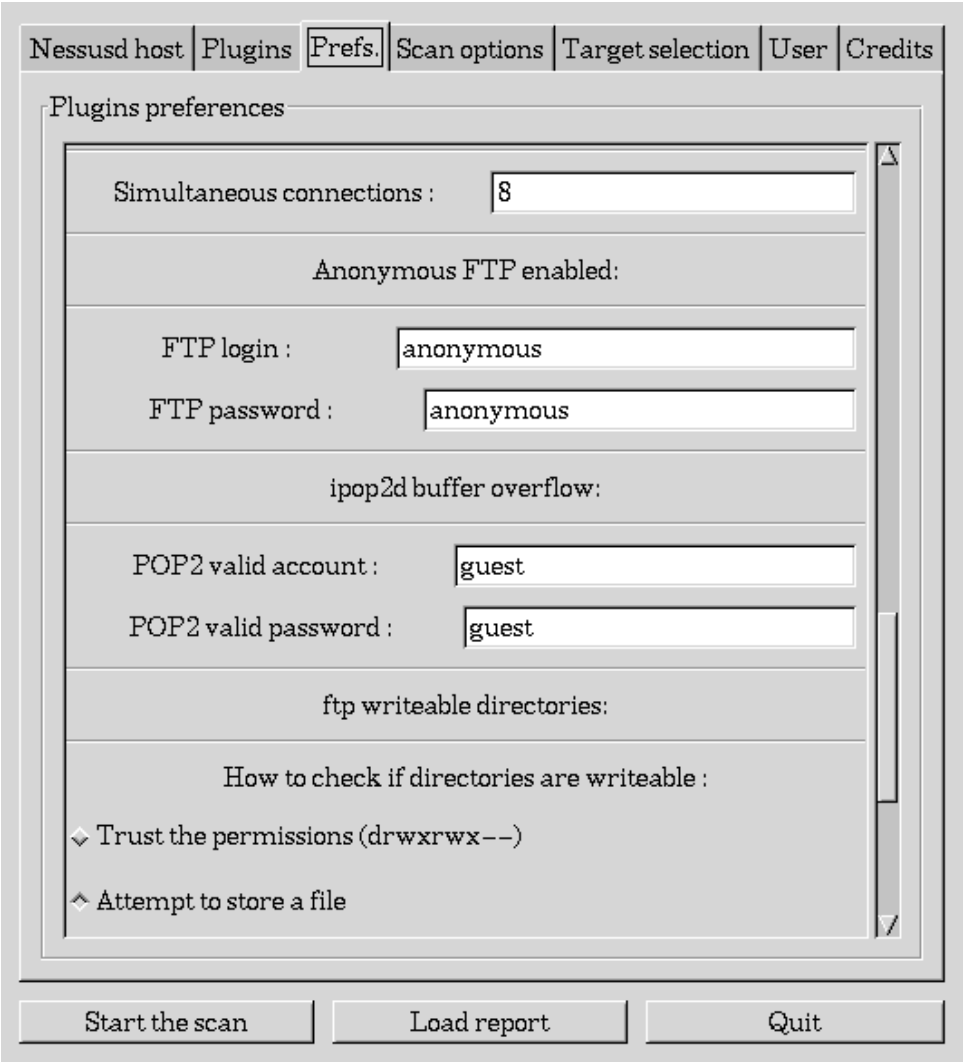


Abbildung 5.3.: Einstellen der Voreinstellungen für die verwendeten Plugin-Module.

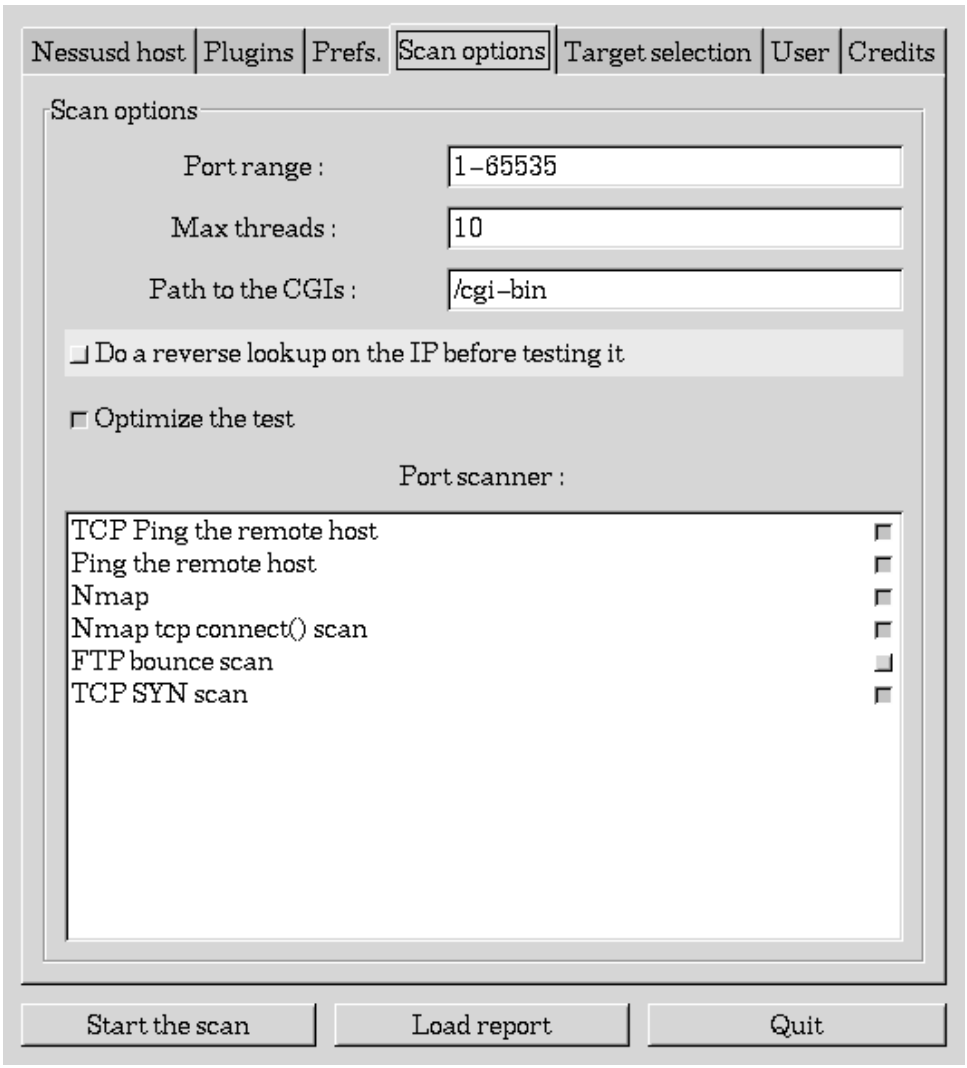


Abbildung 5.4.: Optionen für den bevorstehenden Scan.

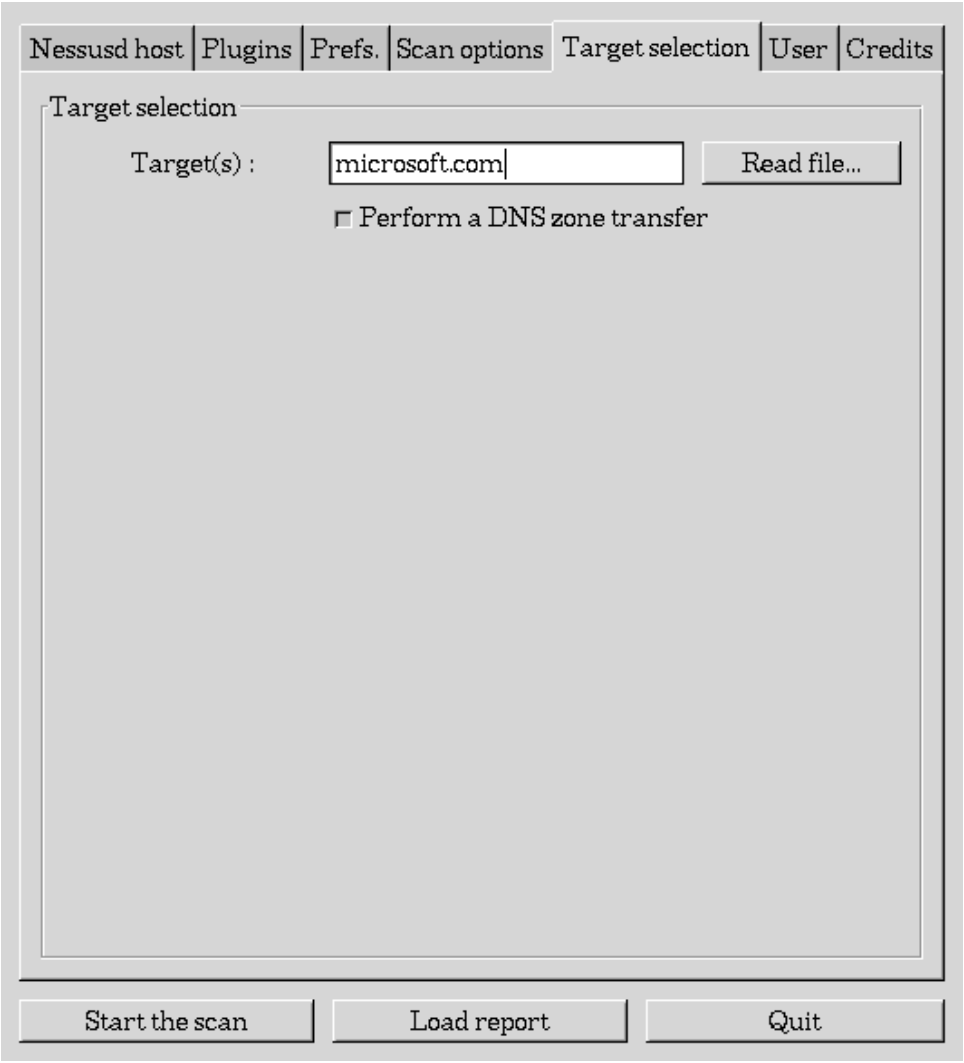


Abbildung 5.5.: Auswahl der Ziele für den bevorstehenden Scan. Es können auch vorbereitete Textfiles und DNS Zone Transfers benutzt werden.

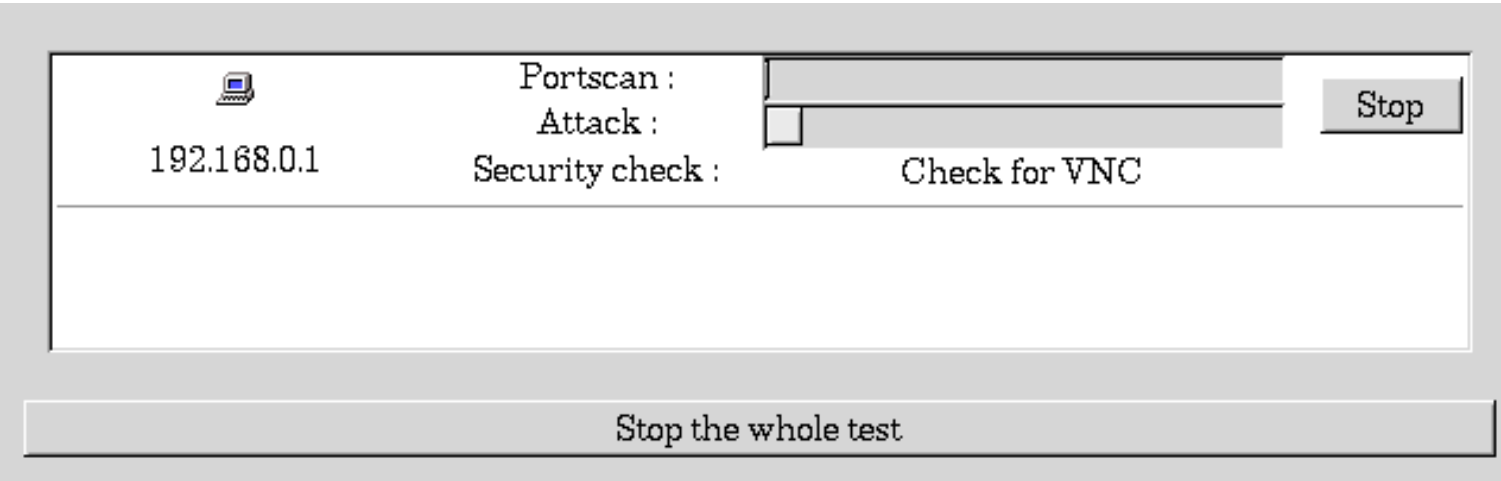


Abbildung 5.6.: Nessus beim Scannen des Ziels. Der Nessus Server kann eine konfigurierbare Anzahl von Prozessen auf mehrere Ziele gleichzeitig loslassen.

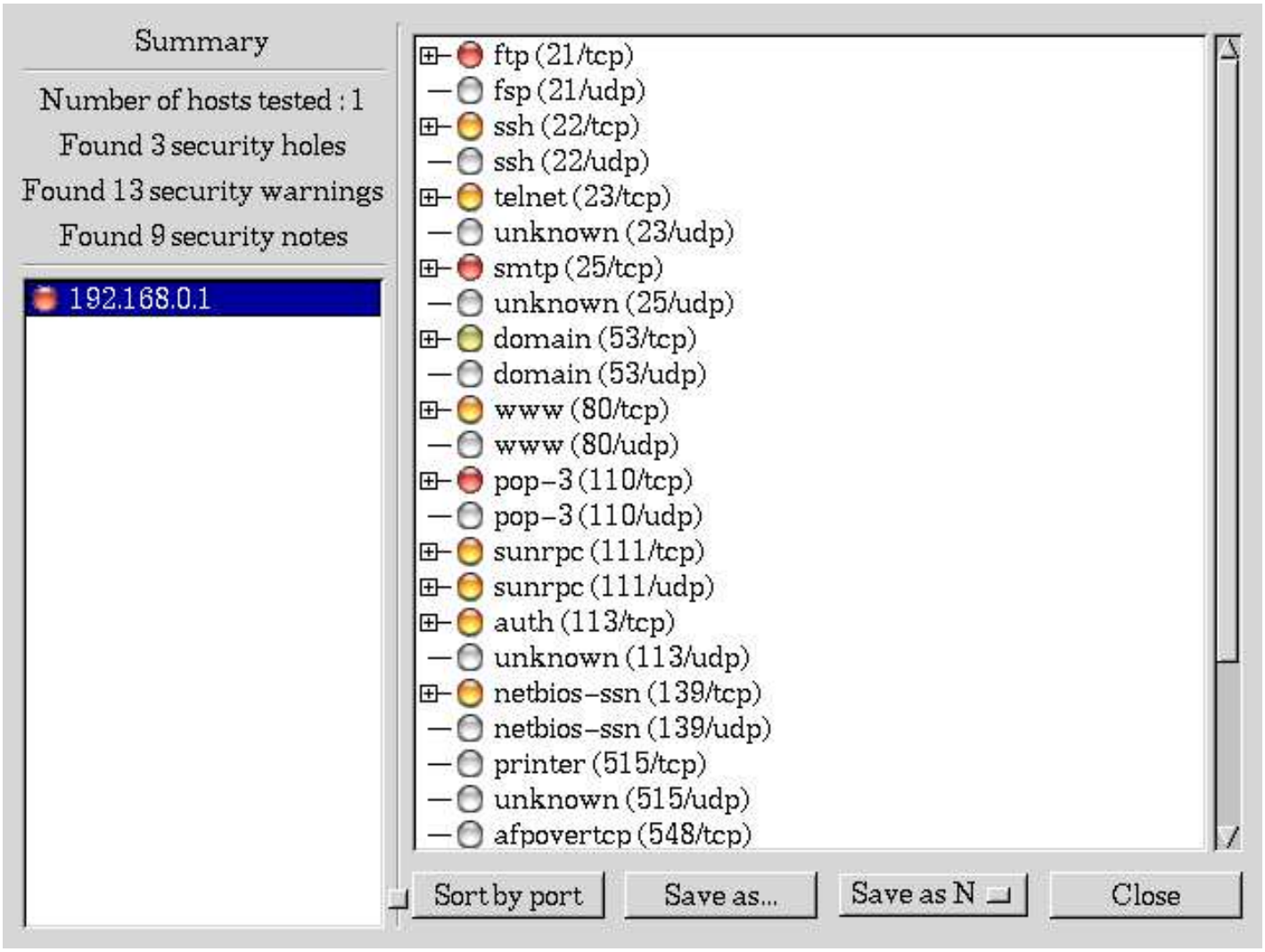


Abbildung 5.7.: Nessus stellt die Ergebnisse eines Scan-Durchlaufs dar. Man kann die Ergebnisse in NSR, HTML mit Grafiken, HTML, ASCII Text und LaTeXabspeichern. Jede gefundene Schwachstelle wird bewertet und mit einem Link zu einer Datenbank mit genauerer Beschreibung versehen.

5.3.3. Einsatz von Live-CDs

Viele der hier vorgestellten Werkzeuge sind einzeln verfügbar und lassen sich installieren. Möchte man schnell auf eine Vielzahl von vorinstallierten Programmen zugreifen, so bieten sich dafür sogenannte Live-CDs an.

- Kali Linux[18] ist speziell für Tests im Sicherheitsbereich vorgesehen. Diese Live-CD stellt alle gängigen Werkzeuge für *Penetration Tests* zur Verfügung.
- grml[19] ist eine Live-CD, die für Systemadministratoren zugeschnitten ist. Der Fokus liegt auf einer Shell-Oberfläche und einer möglichst großen Anzahl von Programmen, die unter anderem auch in den Sicherheitsbereich fallen.

Das Booten einer Live-CD ist oft schneller als das Installieren aller benötigen Werkzeuge. Mit moderner Hardware kann man auch von USB-Sticks oder vom Netzwerk booten. grml bietet die Möglichkeit einer Installation auf Festplatte. Backtrack kann auf Systemen mit ausreichend Speicher komplett ins RAM geladen werden.

[18] https://www.kali.org/
[19] http://www.grml.org/

6. Härten von Serversystemen

```
Among the maxims on Lord Naoshige's wall, there was this one:
"Matters of great concern should be treated lightly."
Master Ittei commented,
"Matters of small concern should be treated seriously."
     -- Hagakure, Yamamoto Tsunetomo
```

6.1. Warum Härten?

Viele Serversysteme werden mit Hilfe von Betriebssystemdistributionen wie Debian Linux®, Red Hat, Novell SuSE, Microsoft® Windows®, FreeBSD, OpenBSD, NetBSD oder ähnlichen installiert. Die Installationssoftware ist dabei auf eine möglichst breite Verwendung zugeschnitten und soll schnell zu einem funktionstüchtigen System führen. Das wirft oft Probleme auf, da die Systeme dann sehr großzügige Berechtigungen besitzen und viel zu viele Softwarepakete installiert sind. Man versteht daher unter dem Begriff „Härten" oder „Hardening" das Vorbereiten des Systems auf den produktiven Einsatz in einer nicht vertrauenswürdigen Umgebung unter Berücksichtigung der Sicherheit.

Das Härten von Clients hat sich ebenso als sinnvolle Gewohnheit ergeben. Das lokale Netzwerk ist längst nicht mehr sicher und kann Spuren von Viren, trojanischen Pferden und im Webbrowser ausgeführter Malware enthalten! Härten sollte man also generell immer und überall, nur beschränkt durch Zeit und Budget.

Die hier aufgeführten Hinweise sind als Zusammenfassung zu verstehen. Man muß für jedes Betriebssystem und jede Applikation eigene Checklisten erstellen und die Konfiguration schrittweise anpassen. Die folgenden Beschreibungen illustrieren die Vorgehensweise. [1]

6.2. Checkliste für Betriebssysteme

Für Betriebssysteme gibt es eine Reihe von Quellen zu diesem Thema.

- Gentoo Security Handbook[2]
- Guides, Checklists & Best Practices[3]
- Hardening Debian[4]
- Minimum Security Standards for Networked Device (University of Berkeley)[5]
- Procedures for Securing Systems[6]
- RUS CERT Absicherung von Betriebssystemen[7]
- Securing and Hardening Red Hat Linux® Production Systems[8]
- Securing Debian Manual[9]
- Security Checklist for GNU/Linux® Systems[10]

[1] Eine gewartete Dokumentation von Serverhärtungsmethoden befindet sich in einem separaten Dokument.
[2] http://www.gentoo.org/doc/en/security/
[3] http://www3.georgetown.edu/security/netadmins/9959.html
[4] https://wiki.debian.org/Hardening
[5] http://security.berkeley.edu/MinStds/
[6] http://www.lbl.gov/cyber/systems/index.html
[7] http://cert.uni-stuttgart.de/bs.php
[8] http://www.puschitz.com/SecuringLinux.shtml
[9] http://www.debian.org/doc/manuals/securing-debian-howto/
[10] http://www.sans.org/score/checklists/linuxchecklist.pdf

- Ubuntu Security Manual[11]
- Windows Server 2003 Security Checklist[12]
- Windows Server 2008 Security Guide[13]
- Windows Server 2003-Sicherheitshandbuch[14]

In jedem Fall sollte man sich eigene Checklisten [45] zulegen und möglichst viele Schritte mit Hilfe von Automatisierungswerkzeugen implementieren. Es bieten sich Shellskripte an. Besser sind Konfigurationswerkzeuge, die direkt zur Systemkonfiguration eingesetzt werden.

6.2.1. Generelle Vorgehensweise zur Absicherung von Betriebssystemen

Im Anhang ist eine allgemeine Checkliste zur Absicherung von Betriebssystemen aufgeführt. Die Absicherung muß mit den jeweiligen Eigenheiten und Möglichkeiten des eingesetzten Betriebssystems abgeglichen werden. Die Methoden in der Ausführung sind plattformabhängig. Die Aussagen der generellen Checkliste sind allgemein gehalten und auch in dieser Art und Weise gültig.

6.2.2. Basisabsicherung eines Debian Serversystems

Debian bietet für den Produktivbetrieb eine stabile Version an. Für die Distribution existieren auch eigene Checklisten zur Absicherung. Die Tips gelten analog für alle GNU/Linux® Distributionen. Das Debian Projekt unterhält auch eigene Anleitungen zum Thema Sicherheit.

- Debian Linux® Security[15]
- Securing Debian Manual[16]

Partitionierung

Bevor noch die minimale Installation beginnt, sollte man sich Gedanken über die Partitionierung des Speicherplatzes machen. Es ist sinnvoll System- und Applikationsdaten zu trennen. Unter UNIX®-artigen Systemen befinden sich die Daten von Benutzern oder Applikationen meist in `/home/` oder `/srv/`. Bestimmte Unterverzeichnisse von `/var/` enthalten teilweise auch solche Daten. Organisatorisch hat sich in Tabelle 6.1 beschriebene Einteilung bewährt. Moderne Installationen kann man auch gemäß der Aufteilung in der Tabelle 6.2. Auf welchen Datenträgern die Partitionierung durchgeführt wird, ist nicht wichtig. Es eignen sich sowohl Partitionen auf einer einzelnen Festplatte oder RAIDx-Stapel bzw. Logical Volumes. Wichtig ist nur die Beziehung zwischen Medium und *mount point*, festgelegt in der Systemdatei `/etc/fstab`.

Der Sinn dahinter ist eine passende Einteilung der Daten und eine Verwendung verschiedener Optionen für die jeweilige Partition. Die meisten Dateisysteme erlauben *mount options* und Quoten pro Benutzer oder Gruppe, die allerdings pro Partition. Zusätzliche setzt die Größe einer Partition harte Limits in punkto Größe. Voll ist voll, egal ob mit oder ohne aktivierte Quoten. Abbildung 6.1 zeigt eine „Livepartitionierung" auf einem Produktivsystem.

Dateisysteme

Die Wahl von Dateisystemen hat große Konsequenzen. Zum einen geht es um Verläßlichkeit im Betrieb, sicher auch um Performance, aber vordergründig spielen Berechtigungen eine wichtige Rolle. Mittlerweile bieten viele Dateisysteme adäquate Möglichkeiten. Eine Auswahl schaut wie folgt aus.

- Btrfs ist ein *copy-on-write* Dateisystem. Es wird in der Zukunft Ext4 ablösen.

[11] https://help.ubuntu.com/community/Security
[12] http://www.lbl.gov/cyber/systems/win-serv03-guide.html
[13] http://www.microsoft.com/downloads/details.aspx?FamilyID=fb8b981f-227c-4af6-a44b-b115696a80ac
[14] http://www.microsoft.com/germany/technet/sicherheit/prodtech/windowsserver2003/w2003hg/s3sgch01.mspx
[15] http://www.debianhelp.co.uk/security.htm
[16] http://www.debian.org/doc/manuals/securing-debian-howto/index.en.html

`/`	300-500 MB	Rootpartition, minimales System
`/boot`	30 MB	Bootpartition mit Kern (optional)
`/usr`	1-5 GB	Anwendungsprogramme
`/opt`	1-5 GB	Anwendungsprogramme (sollte eigentlich in `/usr` integriert sein)
`/usr/local`	1-5 GB	Anwendungsprogramme außerhalb der Distribution (optional)
`/usr/src`	1-2 GB	Ablage für Quellcode installierter Applikationen
`/home`	x GB	Benutzerdaten
`/tmp`	t_t GB	temporäre Daten für alle Benutzer (öffentlich!)
`/srv`	y GB	Applikationsdaten (optional)
`/var`	z GB	variable Daten (Logs, Datenbanken, Caches, etc.)
`/var/tmp`	t_v GB	temporäre Daten für alle Benutzer (öffentlich!)
Swap	variabel	Auslagerungspartition (ca. 2× RAM Größe)

Tabelle 6.1.: Diese Tabelle zeigt ein empfohlenes Partitionslayout eines Servers gemäß gängigen Handbüchern zur Härtung. Einige Partition sind optional, da sie von der Verwendung abhängen. Die Größen der Partitionen, die mit den Variablen t_t, t_v, x, y und z bezeichnet sind, hängen ebenso von den darauf eingesetzten Applikationen ab. Sie sind jedoch erfahrungsgemäß mittlerweile bei 10+ GB anzusiedeln. `/` stellt nach dem Booten alleine ein komplettes System dar. Wenn vorhanden, so wird `/boot` meist nur-lesbar oder gar nicht ins System eingebunden, da der Bootloader blockweise darauf zugreift. Die Aufteilung zusammen mit der Schätzung der Größen stammt aus einem Red Hat 6.x Handbuch und ist auch für modernere Systeme problemlos nach wie vor einsetzbar und empfohlen. Für Microsoft® Windows Systeme gelten analoge Richtlinien, allerdings ist meist die Anzahl der Partitionen geringer. Die Trennung von System und Daten wird dort aber auch sehr empfohlen.

`/`	1-5 GB	Rootpartition, minimales System
`/boot`	400 MB	Bootpartition mit Kern (optional)
`/usr`	$>$ 5 GB	Anwendungsprogramme
`/opt`	$>$ 1-5 GB	Anwendungsprogramme (sollte eigentlich in `/usr` integriert sein)
`/usr/local`	$>$ 1-5 GB	Anwendungsprogramme außerhalb der Distribution (optional)
`/usr/src`	$>$ 1-2 GB	Ablage für Quellcode installierter Applikationen (optional)
`/home`	x GB	Benutzerdaten
`/tmp`	t GB	temporäre Daten für alle Benutzer als `tmpfs` (öffentlich!)
`/srv`	y GB	Applikationsdaten (optional)
`/var`	z GB	variable Daten (Logs, Datenbanken, Caches, etc.)
`/var/tmp`	t GB	temporäre Daten für alle Benutzer als `tmpfs` (öffentlich!)
Swap	variabel	Auslagerungspartition (ca. 2× RAM Größe)

Tabelle 6.2.: Diese Tabelle zeigt ein typisches Partitionslayout eines Servers mit modernen GNU/Linux Distributionen. Für `/tmp` und `/var/tmp` kann man das `tmpfs` Dateisystem verwenden, welches Daten im Hauptspeicher und in der Auslagerung ablegt. Man benötigt dann nur noch die Dimensionierung der Applikationsdaten.

```
lynx@morgion:~$ df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/md0              327M  100M  227M  31% /
tmpfs                 126M  4.0K  126M   1% /dev/shm
/dev/md5               59G  9.0G   50G  16% /home
/dev/md4              3.9G  634M  3.2G  17% /usr
/dev/md3              968M  370M  599M  39% /usr/src
/dev/md2              3.9G  1.9G  2.1G  48% /var
/dev/md1              482M  1.3M  481M   1% /tmp
lynx@morgion:~$ cat /proc/swaps
Filename              Type            Size    Used    Priority
/dev/sda6             partition       248968  38960   -1
/dev/sdb6             partition       248968  23756   -1
lynx@morgion:~$
```

Abbildung 6.1.: Hier ist die Partitionierung eines Webservers gezeigt. Der Apache Server „lebt" im Verzeichnis `/home/www`. `/var` enthält die Webserverlogs, die regelmäßig rotiert und gelöscht werden. `/usr` ist recht großzügig dimensioniert. Die Auslagerungspartitionen erscheinen nicht bei Aufruf von `df`. Sie sind seperat aufgelistet. Man sieht, daß sie auf beide Festplatten im RAID1-Verbund gleichmäßig aufgeteilt sind. Der Swapspace liegt hier nicht auf einem RAID (sollte er aber).

- Ext2 ist das ursprüngliche Dateisystem von Linux®. Es ist von anderen Dateisystemen abgelöst worden, läßt sich aber sehr gut für Partitionen wie `/boot` einsetzen, die sehr wenig E/A Zugriffe haben. Darüber hinaus bietet es sich für USB-Sticks oder Flash Disks an, da es kein *Journaling* kennt.
- Ext3 ist der Nachfolger des Linux® Ext2 Dateisystems. Es unterstützt *Journaling* und wurde von Ext4 abgelöst.
- Ext4 ist der Nachfolger des Linux® Ext3 Dateisystems. Es unterstützt *Journaling* und hat einige Verbesserungen von XFS und anderen Dateisystemen übernommen.
- JFS (IBM)
- ReiserFS 3.x[17]
- XFS (SGI) ist das älteste Journaling Dateisystem für UNIX® Systeme. Es ist sehr performant und bietet eine Reihe von Werkzeugen zur Wartung. Der Code wird stetig aktualisiert.
- ZFS ist ein logischer Volume Manager kombiniert mit einem Dateisystem.

Alle Dateisysteme unterstützen erweiterte Berechtigungen (sogenannte *Extended ACLs*) und Quoten. Es gibt neben diesen Dateisystemen noch einige andere, die aber an dieser Stelle nicht betrachtet werden. Wir nehmen Btrfs, Ext4 und XFS heraus. Gleich bei der Installation sollte man die folgenden Mountoptionen auf bestimmte Verzeichnisse anwenden.

- `noatime` - möglichst nur auf Partitionen, die sehr viel E/A Operationen erwarten[18]
- `nodev` - beispielsweise auf `/tmp`
- `noexec` - beispielsweise auf `/tmp`[19]
- `nosuid` - beispielsweise auf `/tmp`
- `usrquota`, `grpquota`

Details lassen sich in der `mount` Man Page nachlesen.

[17] Der Einsatz von ReiserFS 3 oder 4 ist für Server nicht mehr empfohlen, weil andere Projekte dieses Dateisystem abgelöst haben.
[18] Neuere Linuxkerne und Systeme bieten auch die Option `relatime` an, bei der die *access time* nur bedarfsweise wird (nicht bei jedem Zugriff, nur innerhalb bestimmter Zeitfenster).
[19] Diese Option nicht bei Debian Systemen und Derivaten verwenden.

Installation und Bereinigung

- Minimale Installation durchführen - System für keine bestimmte Rolle installieren.
- Gutes `root` Paßwort wählen.
- Gleich ein unprivilegiertes Benutzerkonto anlegen. **Sämtliche Arbeiten auf Servern werden als unprivilegierter Benutzer durchgeführt!** Das Superuserkonto wird nur verwendet, wenn es nicht anders geht.
- Konfigurationen inspizieren.
  - `/etc/inetd.conf`
  - `/etc/pam.d/login`
  - `/etc/pam.d/passwd`
  - `/etc/security/access.conf`
  - `/etc/security/limits.conf`
- Zugang via SSH beschränken.
  - nur Protokoll Version 2
  - nur bestimmte Konten
  - nur von bestimmten Netzwerkadressen
  - Direkten `root` Login über das Netzwerk deaktivieren.
- Überflüssige Pakete deinstallieren.[20]
- `lpd` und NFS-/RPC-Dienste entfernen (z.B. `portmap`).
- Softwarepakete für Anwendungsgebiete des Servers installieren.
- Konten anlegen (pro Applikation eine eigene Gruppe und einen eigenen Benutzer).
- `sudo` konfigurieren (Konten mit Administratorrechten definieren).
- Alle bekannten Patches installieren.
- Host Firewall und TCP Wrapper konfigurieren.

Nach diesen Schritten geht man meist schon zur Installation der Applikationen für die Produktivumgebung über. Es ist zu empfehlen dem Server noch einen genau zugeschnittenen Linux® Kern für seinen Einsatzbereich zu übersetzen. Je nach Sicherheitspolitik kann man das auf der Maschine oder auf einer eigens dafür vorgesehen anderen Maschine tun (damit der Quellcode nicht am Server selbst liegt).

6.3. Konfigurationsmanagement

Das Verwalten von Konfigurationen jeglicher Art ist ab zwei Systemen bereits reif für eine Automatisierung und Versionskontrolle. Die Chance, dass sich zwei Systeme (desselben Betriebssystems) Softwarekomponenten teilen, ist sehr hoch. Sobald Änderungen ins Spiel kommen, spart die Automatisierung Zeit und schafft Konsistenz. Oft läßt sich dieser Ansatz auch mit einer Versionskontrolle der Konfigurationsdateien verbinden (Motto *configuration is code, code is configuration*). Der einfachste Ansatz ist eine Sammlung von Shell Skripten, die Aufgaben im Bereich der Änderungen an und Neuinstallation von Systemen übernehmen. Dies stellt meist den Anfang für eine Automatisierung dar. Man kann jedoch damit alleine keine Richtlinien an alle Systeme ausgeben und ihre Abweichung von der gewünschten Norm ermitteln. Dazu bedarf es Feedbackmöglichkeiten, die ein System inklusive Verifizierung des Zustands für das Konfigurationsmanagement bereitstellen muß. Vertreter dieser Werkzeuge sind beispielsweise

- Ansible,
- Bcfg2,

[20] Ein Server braucht weder Maus noch graphische Oberfläche. Dabei geht es weniger um Coolness als um Einsparen von möglichst viel Code, der gewartet werden muß. Ein Window Manager, der mit `root` Rechten auf einem Server läuft, ist eine Zeitbombe.

- Cfengine,
- Chef,
- Puppet oder
- Salt.

Es existieren darüber hinaus noch Softwareprodukte für bestimmte Aufgaben (beispielweise nur das Deployment von Webapplikationen oder das Management von Softwarepaketen). Die hier vorgestellten Tools bieten viele Möglichkeiten und haben den Fokus auf Konfigurationsmanagement gemeinsam.

Das Ziel dieser Werkzeuge sind neben der Verteilung von Konfigurationen auch die Überwachung des System-/Softwarezustandes, automatische Korrektur von Abweichungen und Sicherstellung, dass mehrfaches Anwenden von Konfigurationsänderungen idempotent bleibt. [46] [47]

Bcfg2, Cfengine, Puppet und Salt folgen dem Client-Server-Modell und verwenden zentrale Server zur Verteilung der Konfigurationen. Für Ansible ist eine Serverkomponente optional ebenfalls verfügbar. Da sich die Clients periodisch melden, ist automatisch ein Monitoring vorhanden, welches für die Erfassung von Betriebsmetriken ausgenutzt werden kann. Das Design der Tools folgt dabei verschiedenen Ansätzen. Cfengine ist für die Verwaltung großer Installationen mit $>$ 10.000 Systemen gedacht. Chef dient zur Verwaltung von Servern und hat Stärken im Softwaremanagement. Man kann aber alle Frameworks einsetzen. Voraussetzung ist die Unterstützung am Zielsystem.

6.3.1. Konvergenz von Konfigurationszuständen

Ein System ist nach Konfiguration in einem definierten Zustand S_a, der oft auch erwünscht ist. Vorgabe dieses Zustands sind Richtlinien, die festlegen welche Software in welcher Konfiguration auf einem System vorliegt. Weicht ein System vom Zustand S_a ab, so wird diese Abweichung ΔS wieder rückgängig gemacht. Das folgende Beispiel verdeutlicht dies für Cfengine:

```
1 bundle agent test
2 {
3 files:
4   "/tmp/cf_test_file"
5       comment => "Promise that a plain file exists with stated permissions",
6       perms => mog("644", "root", "sys"),
7       create => "true";
8 }
```

Die Konfiguration definiert ein Bundle namens *agent test*, welche die Datei */tmp/cf_test_file* anlegt, sollte sie nicht existieren. Zusätzlich wird festgelegt, dass diese Datei dem Benutzer *root* und der Gruppe *sys* gehören soll. Der Benutzer darf schreiben und lesen, alle anderen dürfen nur lesen (festgelegt durch oktale Bitmaske *644*). Bei jedem Aufruf des Cfengine Clients lokal wird nun geprüft, ob diese Kriterien zutreffen. Ist dies nicht der Fall, so wird eine Korrektur vorgenommen. Man kann daher davon ausgehen, dass nach jedem Aufruf des Cfengine Clients alle Vorgaben der Richtlinie erfüllt sind.

6.3.2. CFEngine

CFEngine besteht aus einem zentral aufgebauten System mit einem oder mehreren Policy Servern an die Clients angebunden sind. Ein Client ist jedes System, welches ein Vertrauensverhältnis mit dem Policy Server hat und von diesem Richtlinien erhält. Die Richtlinien werden auch *business policies* oder *promises* genannt. [48] Die Standardkonfiguration hat zwei Hauptteile: `promises.cf` und `updates.cf`. `promises.cf` enthält die Richtlinien. `updates.cf` regelt die Verteilung der Richtlinien. CFEngine hat bei der Verarbeitung der Konfigurationen einen *fail safe* Modus. Abbildung 6.2 zeigt den Modus schematisch.

Es ist wichtig zu beachten, dass CFEngine eine deklarative Sprache in den Richtlinien verwendet. Man muß CFEngine nicht erklären was es tun soll, sondern man beschreibt stattdessen den gewünschten Zustand. Das schaut sprachlich ausgedrückt so aus.

- Stelle sicher, dass in der Datei `/etc/ssh/sshd_config` die Zeile `UseDNS no` vorkommt.
- Stelle sicher, dass das Konto `mysql` exisitiert / nicht existiert.
- Stelle sicher, dass das der Prozeß `httpd` (nicht) ausgeführt wird.

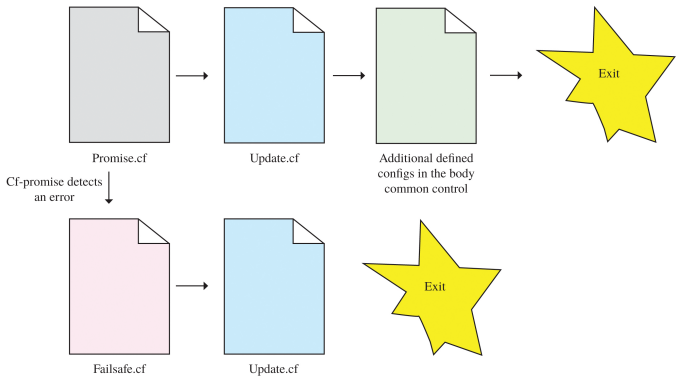


Abbildung 6.2.: Wenn CFEngine beim Analysieren der Konfiguration auf Fehler trifft, so versucht die Software zuerst mit einem reduzierten Satz an Regeln trotzdem den Konfigurationsvorgang am System durchzuführen. Man findet in den Logs dann Hinweise auf den Fehler und den aktiven *fail safe* Modus.

CFEngine wird versuchen diese Vorgaben mit den zur Verfügung stehenden Mitteln zu erreichen (diese Mittel sind eingebaute Funktionen und die CFEngine-eigene Konfigurationsbibliothek). Die Policies lassen sich zusamenfassen und auf eine beliebige Zahl von Systemen ausdehnen.

- Stelle sicher, dass alle Webserver das Apache Paket installiert haben.
- Stelle sicher, dass alle `root` Konten dasselbe Passwort haben.
- Stelle sicher, dass die Parameter `UseDNS` und `PermitRootLogin` in allen *sshd* Konfigurationen außer auf den Systemen *webnode1* und *webnode2* nicht aktiv sind.

Die Änderungen werden von CFEngine durch Aufruf eines Programms namens *cf-agent* durchgeführt. Es gibt noch weitere Komponenten, die Teil des Softwarepakets sind.

cf-agent wertet alle Richtlinien aus und führt die daraus resultierenden Änderungen durch. *cf-agent* wird von den Programmen *cf-execd* und *cf-serverd* gestartet.

cf-execd führt *cf-agent* periodisch aus (etwa alle 5 Minuten, Zeiten sind leicht variabel, um die Policy Server nicht zu überlassen). Dieser Prozess ist der einzige, der für einen Client wichtig ist. Er startet alle anderen Prozesse nach Bedarf (vorausgesetzt ein Bootsrap hat stattgefunden).

cf-serverd implementiert die komplette Serverfunktionalität von CFEngine. Diese Komponente ist nur auf zentralen Policy Servern aktiv. Sämtliche Konfiguration zwischen Server und Clients erfolgt via Port 5308/TCP.

cf-runagent ruft *cf-agent* auf. Man kann damit direkt *cf-agent* auf Clients aufrufen (ohne den Umweg über *cf-execd*).

cf-key wird bei neuen Clients benötigt, um die kryptografischen Schlüssel für die Client-Server-Kommuniktation zu erzeugen.

cf-monitord kann kontinuierlich im Hintergrund laufen, um Systeme zu beobachten. Er sammelt Statistiken und setzt automatisch Klassen (als Flags), damit man in Policies auf Situationen reagieren kann.

cf-hub ist Teil der CFEngine Enterprise Edition und sammelt von allen Clients Statusinformationen.

Mit Hilfe einer eigenen Konfigurationssprache lassen sich beliebige Aufgaben formulieren. Für komplexe Richtlinien lassen sich die Regeln mittels *Bundles* und *Bodies* organisieren. Bundles schauen syntaktisch so aus.

```
1 bundle type name(arguments)
2 {
3     promise_type:
4         class_expression::
5             promise...
6
7 }
```

Der Name eines Bundles ist beliebig und dient zur Identifikation. Der Typ ist einer von den folgenden Begriffen.

- `agent` stellt „ausführbare“ Bundles dar. Mit ihnen lassen sich alle denkbaren Änderungen an einem System vornehmen. Für das Schlüsselwort *promise* lassen sich die folgenden Begriffe angeben.
  - `commands` zwecks Ausführung von Kommandos.
  - `files` zur Manipulation von Dateien.
  - `methods` um andere Bundles vom Typ `agent` aufzurufen.
  - `packages` zur Verwaltung der Softwarepakete (Installation, Abfrage).
  - `processes` zur Verwaltung der laufenden Prozesse.
  - `storage` zur Konfiguration und Abfrage von Dateisystemen.
  - `services` zur Konfiguration von UNIX-Diensten.[21]
  - `database` zur Verwaltung von Datenbanksystemem.
  - `guest_environments` zur Verwaltung von Virtualisierungsumgebungen.
- `common` entsprechenden den `agent` Bundles. Ihre definierten Variablen und Klassen stehen jedoch allen anderen Bundles in der Richtlinie zur Verfügung. Ebenso können diese Bundles von allen CFEngine Komponenten ausgewertet werden. Sie sind für globale Operationen und Variablen gedacht.
- `edit_line` ändert den Inhalt von Dateien.
- `server` steuert den *cf-serverd* Prozess, welcher Dateien an alle CFEngine Systeme verteilt.
- `monitor` überwacht Betriebsparameter (nur in der Enterprise Version von CFEngine verfügbar).

Bodies sind Sammlungen von Attributen mit Werten (*key-value pairs*), die man in der Policy verwenden kann. Die Syntax schaut so aus.

```
body type name(arguments)
{
    attribute1 => value1;
    attribute2 => value2;...

   [class_expression::]
    attributeN => valueN;
}
```

Auch hier gibt es bei den Typen verschiedene Schlüsselworte.

- `control` steuert das Verhalten des CFEngine selbst. Man kann beispielsweise festlegen welche Bundles ausgeführt werden.

```
body common control
{
    inputs => { "tests.cf" };
    bundlesequence => { "test" };
}
```

- `classes` definiert Klassen, die gültig sind je nach Ergebnis eines *promise*.
- `action` ergänzt oder modifiziert bestehende *promise* Definitionen.

```
bundle agent motd
{
    files:
        "/etc/motd"
            edit_line =>
                insert_lines("Access to this system for authorised staff only."),
            action => warn_hourly;
}
body action warn_hourly
{
    action_policy => "warn";  # Produziert nur eine Warnung, keine Aktion wird durchgeführt
    ifelapsed => "60";
}
```

[21] Microsoft® Windows Dienste lassen sich mit der Enterprise Version von CFEngine verwalten.

- `copy_from` kann in `files:` *promises* verwendet werden. Die Direktive regelt wie Dateien von wo nach wo kopiert werden. Man kann In-Transit-Verschlüsselung, Vergleich von Zeitstempeln oder Checksummen zur Überprüfung konfigurieren.
- `depth_search` regelt das Verhalten rekursiver Operationen.

In der Dokumentation von CFEngine finden sich Details zu allen Optionen.

Ganz analog zu objektorientierten Sprachen wie C++ lassen sich definierte *bundles* und *bodies* durch *namespaces* trennen. Damit lassen sich umfangreichere Richtlinien erstellen ohne Namen mit Suffixen oder Präfixen zu versehen.

Cfengine 3 Installation

So gut wie alle GNU/Linux Distributionen stellen Softwarepakete mit der Cfengine Version 3 bereit. Der einfachste Weg, um eine Testinstallation aufzubauen, erfordert zwei Systeme.

1. Installation von CFEngine auf dem Policy Server
2. Bootstrap des Policy Servers (Herstellen des Vertrauensverhältnisses und Initialisierung)
3. Installation von CFEngine auf dem zweiten Host
4. Bootstrap des Hosts und Verbinden mit dem Policy Server

Danach kann man sofort Policies auf dem Policy Server entwerfen und an alle angebundenen Hosts verteilen. Der Bootstrap Prozeß geschieht durch Ausführen dieses Kommandos (der Pfad zum Kommando kann abweichen).

```
1 sudo /var/cfengine/bin/cf-agent --bootstrap <IP Adresse / Hostname des Policy Servers>
```

Der Bootstrap Prozeß stellt sicher, dass die Master-Dateien im Arbeitsverzeichnis liegen (in */var/cfengine/inputs/*), und startet den *cf-execd* Dämon. *cf-execd* stellt sicher, dass der *cf-agent* Prozeß periodisch gestartet wird. *cf-agent* ist die eigentliche Komponente, die die Richtlinien lokal umsetzt. Ganz wichtig an dieser Stelle ist die Angabe des Policy Servers. CFEngine speichert sich den Namen oder die IP Adresse beim Bootstrap Prozeß in die eigene lokale Konfiguration. Ändert sich die Adresse des Policy Servers oder der Name, so kann der CFEngine Client keine Updates erhalten. Es wird dann an der letzten erhaltenen Konfiguration festgehalten.

Per Default akzeptiert CFEngine neue Clients von allen IP(v4) Adressen. Man sollte die Access Lists (ACLs) für die eigenen Clients möglichst strikt konfigurieren (gemäß dem *principle of least privilege*).

Auf Debian Systemen muß man die Datei `/etc/default/cfengine3` noch editieren, um die entsprechenden CFEngine Prozesse zu aktivieren. Auf Clients ist die Zeile `RUN_CFEXECD=1` zu setzen.

Wichtiger Hinweis für IPv6: Auf Debian/Ubuntu Systemen muß man in der Konfigurationsdatei `/etc/gai.conf` die Zeile

```
1 label ::/0          1
```

hinzufügen (sie ist normalerweise auskommentiert). Man muß natürlich die vertrauenswürdigen IPv6 Netzwerke auch in die Konfigurationen übernehmen (Direktiven `TrustKeysFrom`, `AllowConnectionsFrom`, `AllowMultipleConnectionsFrom`)

Erstellen von Richtlinien

CFEngine bringt eine Standardbibliothek von Richtlinien mit. Es ist empfehlenswert diese als Basis zu verwenden und darauf die eigenen Policy Dateien aufzubauen. Auf Debian- und Ubuntu-basierten Systemen befinden sich alle relevanten Dateien im Pfad `/var/lib/cfengine3/masterfiles/`. Ein guter Startpunkt ist die Datei `promises.cf`. Darin wird die CFEngine Bibliothek initialisiert. Möchte man eine Richtlinien einfügen, die die Loginmeldung in `/etc/motd` auf allen Clients überschreibt, so geht man wie folgt vor.

1. Anlegen eines Verzeichnisses für Dateien, die an Clients verteilt werden:
   `mkdir /var/lib/cfengine3/masterfiles/files`
2. Anlegen der Datei `motd` im neu angelegten Verzeichnis.
3. Hinzufügen des Bundles für das Verteilen der Datei.

```
bundle agent edit_motd
{
    vars:
        "motd" string => "/etc/motd";

    files:
        "$(motd)"
            create => "true",
            copy_from => secure_cp( "/var/lib/cfengine3/masterfiles/files/motd", $(policy_server) );
}
```

`$(policy_server)` muß vorher definiert sein und stellt den Namen oder die numerische Adresse des Policyservers dar. Das Makro `secure_cp` ist Teil der CFEngine Bibliothek und kopiert Dateien verschlüsselt zu den Clients.

4. Neustart der CFEngine Prozesse mittels *service cfengine3 restart*.

Die CFEngine Clients holen sich die neue Policy bei der nächsten Überprüfung und führen sie dann lokal aus.

6.3.3. Ansible

Ansible ist wie CFEngine ein mächtiges Konfigurationswerkzeug für verteilte Systeme. Im Gegensatz zu CFEngine verwendet Ansible kein eigenes Protokoll zur Kommunikation. Sämtliche Interaktion zwischen den Systemen geschieht über die Secure Shell (SSH bzw. OpenSSH). Das Paradigma ist hier eine andere Form von Client-Server-Prinzip. Die Ansible Richtlinien (bezeichnet als *Playbooks*) werden von einem System per SSH verteilt. Der Playbook Server kann daher jede beliebige Maschine sein, die SSH Zugang zu anderen Hosts hat. Playbooks sind in YAML[22] geschrieben, lassen sich aufgrund der Textform leicht versionieren und auf jedem beliebigen System einsetzen. Ansible folgt daher einem sehr flexiblen Client-Server-Prinzip wie in Abbildung 6.3 dargestellt.

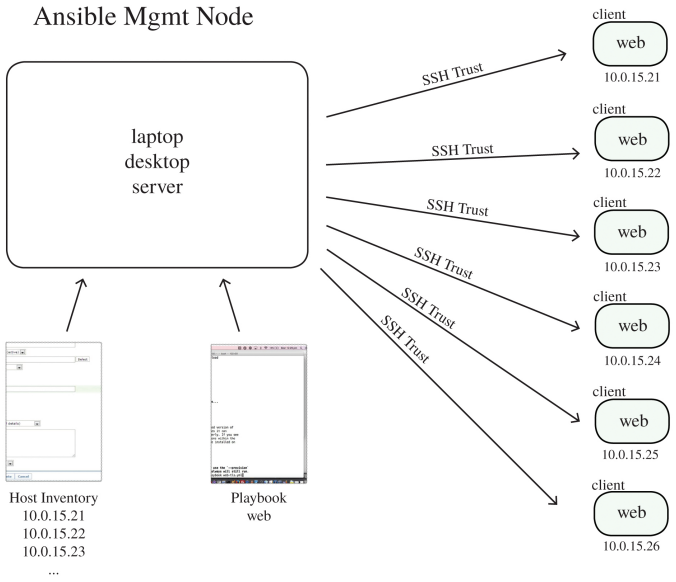


Abbildung 6.3.: Ansible kann von beliebigen Systemen eingesetzt werden, die ein Vertrauensverhältnis zueinander haben (hergestellt durch die Generierung und Verteilung von SSH Schlüsseln).

Neben der Push Architektur kann Ansible auch als Pull System eingesetzt werden. Das Tool `ansible-pull` konfiguriert auf den Nodes eine Kopie von Ansible und kann Playbooks aus einem Quell-Repository installieren.

[22] YAML Ain't Markup Language, siehe `http://yaml.org/`. YAML ist ein Standard, um Daten zu serialisieren.

Es gibt auch den *fireball mode*, in dem die Nodes via AES-verschlüsseltem ØMQ Protokoll kommunizieren. Damit lassen sich in kurzer Zeit sehr viele Systeme ansprechen. Ansible verbindet sich normalerweise per Secure Shell zu den Zielsystemen, was bei >10000 Hosts zu einem Engpaß werden kann. Ziel ist es, eine höchstmögliche Parallelisierung zu erreichen.

Playbooks

Ansible Playbooks sind wesentlich einfacher aufgebaut als CFEngine Policies. Das folgende Beispiel zeigt die Struktur.

```
---
- hosts: firewall      # Zu konfigurierende Server (Firewallsystem)
  remote_user: root    # Konto zur Konfiguration
  roles:               # Welche Aufgaben zur erledigen sind
  - haproxy
  - netfilter

- hosts: webservers    # Webserver
  remote_user: admin
  sudo: yes            # Dem Konto steht sudo zur Verfügung
  roles:
  - apache
  - php
  - wordpress

- hosts: database      # Datenbankserver
  remote_user: root
  roles:
  - postgres

- hosts: all           # Alle Hosts
  remote_user: admin
  sudo: yes
  tasks:
  - group:
    name: manager
    state: present
```

Trotz der Einfachheit lassen sich mit Playbooks komplexe Aufgaben abbilden, auch mit bestimmter Reihenfolge und mit Abhängigkeiten.

Genau wie bei allen anderen Konfigurationsmanagementwerkzeugen muß man sich vor der Erstellung von Playbooks mit der Topologie vertraut machen. Bei Ansible ist das Host Inventory. Man kann es textbasiert unter `/etc/ansible/hosts` anlegen. Hier ist ein Beispiel angeführt:

```
# Ungrouped hosts.

# Ports can be put to the end of the server name
pc1.example.net:223
pc2.example.net
pc3.example.net

# These are our hosts, by groups.
[admin]
192.168.23.42   ansible_ssh_user=admin

[web]
webnode1.example.net    ansible_ssh_user=root
webnode2.example.net    ansible_ssh_user=root
webnode3.example.net    ansible_ssh_user=root
```

Das Format ist das von .INI Dateien. Hosts können in mehreren Gruppen vorkommen. Ansible vereint die Gruppen bei Anwendung der Playbooks. Bei der Benennung von Gruppen kann man Schlüsselworte wie `all`, Regular Expressions, Wildcards und Mengenoperatoren verwenden.

Tasks, wie im Beispiel angeführt, sind Aktionen, die auf eine Auswahl von Systemen angewendet werden.

Module sind Sammlungen von Prozeduren für spezifische Komponenten. Es gibt beispielsweise das Modul `apt` für den Debian / Ubuntu Paketmanager, `yum` für den Red Hat / CentOS Paketmanager, `user` für das Verwalten von Benutzerkonten oder `service` für Dienste. In der Modulsammlung befinden sich auch Module für Amazon EC2/S3, Microsoft® Azure, Datenbanksysteme, Rackspace Public Cloud oder Dateisysteme wie ZFS. Das Kommando `ansible-doc -l` zeigt alle verfügbaren Module und kann eine kurze Hilfe anzeigen.

Roles und Komplexität

Trotz der einfacheren YAML Konfiguration darf man sich nicht täuschen lassen. Es ist nicht empfehlenswert alles in eine Datei zu schreiben. Im ersten Beispiel für ein Playbook kamen Roles vor. Roles sind eine Zusammenfassung von Tasks für bestimmte Aufgaben. Die folgende Dateistruktur stammt aus einem Playbook[23] für einen LAMP Server[24].

```
.
├── group_vars
│   ├── all
│   └── dbservers
├── hosts
├── LICENSE.md
├── README.md
├── roles
│   ├── common
│   │   ├── handlers
│   │   │   └── main.yml
│   │   ├── tasks
│   │   │   └── main.yml
│   │   └── templates
│   │       └── ntp.conf.j2
│   ├── db
│   │   ├── handlers
│   │   │   └── main.yml
│   │   ├── tasks
│   │   │   └── main.yml
│   │   └── templates
│   │       └── my.cnf.j2
│   └── web
│       ├── handlers
│       │   └── main.yml
│       ├── tasks
│       │   ├── copy_code.yml
│       │   ├── install_httpd.yml
│       │   └── main.yml
│       └── templates
│           └── index.php.j2
└── site.yml
```

Im Verzeichnis `roles` befinden sich die Roles für gemeinsame Konfiguration `common`, die Datenbank `db` und den Webserver `web`. Darin befinden sich Unterverzeichnisse mit festen Namen.

tasks enthalten die Logik der Role wie zu installierende Pakete, Dienste, Dateien, etc.

handlers enthalten Aktionen, die bei bestimmten Auslösern aufgerufen werden, z.B. das Neustarten eines Dienstes, Änderungen in einer Konfigurationsdatei.

templates enthält Informationen mit denen man Dateien modifizieren kann; beispielsweise Konfigurationsdateien modifiziert mit Portnummern oder IP Adressen. Templates sind keine YAML Dateien, sondern die eigentlichen Dateien versehen mit Ansible Variablen und Kontrollstrukturen, um die Inhalte anzupassen.

Die Role `db` im Detail besteht aus drei Teilen. Im Tasks-Teil ist die Logik untergebracht:

```
1  ---
2  # This playbook will install mysql and create db user and give permissions.
3
4  - name: Install Mysql package
5    yum: name={{ item }} state=installed
6    with_items:
7     - mysql-server
8     - MySQL-python
9     - libselinux-python
10    - libsemanage-python
11
12 - name: Configure SELinux to start mysql on any port
13   seboolean: name=mysql_connect_any state=true persistent=yes
14   when: sestatus.rc != 0
15
16 - name: Create Mysql configuration file
17   template: src=my.cnf.j2 dest=/etc/my.cnf
18   notify:
19   - restart mysql
20
21 - name: Start Mysql Service
22   service: name=mysqld state=started enabled=yes
```

[23] Quelle: Ansible Examples, `git clone https://github.com/ansible/ansible-examples.git`, Verzeichnis `lamp_simple`.
[24] LAMP = Kombination aus GNU/Linux®, Apache, MySQL/MariaDB und PHP.

```
23
24 - name: insert iptables rule
25   lineinfile: dest=/etc/sysconfig/iptables state=present regexp="{{ mysql_port }}"
26               insertafter="^:OUTPUT " line="-A INPUT -p tcp --dport {{ mysql_port }} -j ACCEPT"
27   notify: restart iptables
28
29 - name: Create Application Database
30   mysql_db: name={{ dbname }} state=present
31
32 - name: Create Application DB User
33   mysql_user: name={{ dbuser }} password={{ upassword }} priv=*.*:ALL host='%' state=present
```

Man sieht, dass alle Pakete installiert werden und die Umgebung für den Betrieb der Datenbank vorbereitet wird. Da jeweils die Firewallregeln und die Datenbankkonfiguration verändert werden, muß sowohl Datenbank selbst als auch die Firewall initialisiert werden. Die Handler Sektion enthält die dafür notwendigen Anweisungen.

```
1 ---
2 # Handler to handle DB tier notifications
3
4 - name: restart mysql
5   service: name=mysqld state=restarted
6
7 - name: restart iptables
8   service: name=iptables state=restarted
```

Es bleibt dann letztlich das Templates Verzeichnis übrig. Darin befindet sich die Vorlage der MySQL/MariaDB Konfigurationsdatei. Als Veränderliche ist der Port markiert.

```
1 [mysqld]
2 datadir=/var/lib/mysql
3 socket=/var/lib/mysql/mysql.sock
4 user=mysql
5 # Disabling symbolic-links is recommended to prevent assorted security risks
6 symbolic-links=0
7 port={{ mysql_port }}
8
9 [mysqld_safe]
10 log-error=/var/log/mysqld.log
11 pid-file=/var/run/mysqld/mysqld.pid
```

6.4. Checkliste für Applikationen

Der „sichere" Weg zur Installation und Konfiguration einzelner Applikationen ist sehr speziell. Man muß sich für jedes Anwendungsgebiet einen Weg mit einer Liste von wichtigen Punkten erarbeiten, die mit den Anforderungen abgestimmt sind. In den folgenden Unterkapiteln sind drei Applikationen als Beispiel aufgeführt, die sich alle auf typischen Webserver finden.

6.4.1. Apache Web Server

Der Apache Webserver kann auf eine lange Entwicklungsgeschichte zurückblicken. Mittlerweile gibt es drei Varianten davon, wobei die letzte stabile Version der 2.4.x Zweig ist. Apache 2.2 unterscheidet sich in der Konfiguration kaum von Apache 2.0. Für den Wechsel auf Version 2.4 gibt es Kompatibilitätsmodule. Eine Installation mit Absicherung kann man wie folgt angehen. [52]

- Installation des Webservers mitsamt allen Daten im Dateisystem als eigener Benutzer in eigener Gruppe, in der sonst niemand Mitglied ist. Insbesondere potentielle CGI-Verzeichnisse und das `DocumentRoot` des Default Virtual Hosts sollten mit konservativen Berechtigungen ausgestattet sein.
- Schreiben und Sichern der Logs in geschützte Bereiche (bevorzugt auch als Kopie auf einen speziellen Logserver).
- Apacheprozeß in eigenem `chroot` Dateisystem einsperren.
- Apache mit mod_security[25] konfigurieren.
- Konfiguration von Defaulteinstellungen reinigen.

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

  - `ServerTokens Prod`
  - `ServerSignature Off`
  - `AllowOverride` einsetzen (`AllowOverride All` vermeiden)
  - `Order Deny,Allow` verwenden und Zugriffe auf Pfade mit passenden `Allow` und `Deny` Regeln steuern
  - `Options Indexes` vermeiden
  - `Options SymLinksIfOwnerMatch` statt `Options FollowSymLinks`

- Webserver mit geeigneten Mitteln überwachen.
- Deaktivieren der HTTP TRACE Methode, kann durch `mod_security` oder `mod_rewrite` geschehen. Mit Hilfe von `mod_rewrite` läßt sich das so sperren.

```
RewriteEngine On
RewriteCond %{REQUEST_METHOD} ^TRACE
RewriteRule .* - [F]
```

  Neuere Versionen erlauben das Abschalten durch Angabe von `TraceEnable Off` in der globalen Konfiguration.

Ganz wichtig sind Berechtigungen im Dateisystem und Beschränkungen durch das Betriebssystem (wie z.B. Speicher oder CPU-Zeit).

6.4.2. Webapplikationen

Da Webapplikationen verschieden aufgebaut sein können, ist es ratsam den Webbrowsern Hinweise bezüglich Sicherheitsinformationen zu geben. Dazu wurden in den letzten Jahren eine Reihe von zusätzlichen HTTP/HTTPS Kopfzeilen eingeführt, die Clients helfen gefährliche Informationen zu ignorieren. Die folgenden Kopfzeilen stellen eine Zusammenfassung des Artikels Hardening your HTTP response headers dar.

Content Security Policy

Webseiten bestehen aus mehreren Komponenten, die vom Client nachgeladen werden. Fonts, Style Sheets oder JavaScript wird oft durch zusätzliche Requests eingebaut. Der Content Security Policy (CSP) Header kann Clients Informationen über die korrekten Quellen geben.

- `Content-Security-Policy: script-src 'self'` zeigt an, dass Skripte nur vom Host der Webseite kommen dürfen.
- `Content-Security-Policy: default-src https://www.technikum-wien.at:*` erlaubt das Nachladen von Inhalten nur per HTTPS von der angegebenen Domain.

Eine Einführung in CSP findet sich auf der OWASP Webseite oder in einem Scott Helmes Artikel. Der Standard ist durch das W3C definiert.

HTTP Strict Transport Security

Viele Webseiten leiten Clients per HTTP Status Code 301 oder 302 von HTTP auf HTTPS um. Mit Hilfe der HTTP Strict Transport Security Kopfzeile kann man einem Browser mitteilen, dass HTTPS gleich von Anfang an verwendet werden soll. HSTS ist in RFC 6797 spezifiziert.

```
Strict-Transport-Security: "max-age=31536000; includeSubdomains" always;
```

HTTP Public Key Pinning

SSL/TLS Zertifikate werden von Zertifizierungsstellen ausgestellt. Solange diese Certificate Authorities als vertrauenswürdig eingestuft werden, vertraut der Client ihnen. Problematisch wird es, wenn eine Certificate Authority kompromittiert wird. HTTP Public Key Pinning (HPKP) erlaubt ein Whitelisting von kryptographischen Identitäten, denen der Webbrowser vertrauen soll. HPKP ist in RFC 7469 spezifiziert.

X-Frame-Options

Die X-Frame-Options (XFO) Kopfzeile soll den Client vor Clickjacking Attacken schützen. Beispielsweise kann man Frames per Same Origin Policy einschränken.

```
X-Frame-Options: "SAMEORIGIN"
```

XFO ist in RFC 7034 spezifiziert.

X-Xss-Protection

Die X-Xss-Protection konfiguriert den eingebauten Schutz vor Cross Site Scripting im Internet Explorer, Chrome und Safari Webbrowser.

```
X-Xss-Protection: "1; mode=block"
```

X-Content-Type-Options

Die X-Content-Type-Options hat nur eine Option namens `nosniff`. Ist diese gesetzt, so werden Internet Explorer und Chrome daran gehindet Inhalte auf ihren MIME Typ zu analysieren. Der Server schickt den Typ ohnehin meist mit, aber diese Webbrowser versuchen eine Erkennung des Inhaltstyps trotzdem.

6.4.3. MySQL

MySQL hat eine ausführliche Dokumentation, in der alle Aspekte der Software nachgeschlagen werden können. Insbesondere gibt es eigene Sicherheitsrichtlinien[26] für den Betrieb.

- Installation des Datenbankprozesses mitsamt allen Daten im Dateisystem als eigener Benutzer mit eigener Gruppe, in der sonst niemand Mitglied ist. Zugriff auf das Verzeichnis mit den Datenbanken hat alleine der MySQL Benutzer.
- Zugriff für den Datenbankadministrator `root` darf sowohl über TCP/IP als auch lokal über UNIX® Sockets nur mit Paßwort möglich sein.
- Vor Anlegen weitere Benutzer muß das MySQL-eigene Privilegiensystem gut verstanden sein.
- `SHOW GRANTS` zeigt Privilegien an, die man überprüfen kann.
- Sämtliche in den Datenbanken gespeicherten Paßworte müssen mit geeigneten Hash-Funktionen unkenntlich gemacht werden.
- MySQL darf niemals ohne Firewall betrieben werden.

6.4.4. PHP

PHP[27] ist eine sehr verbreitete Skriptsprache, die auf Apache Webservern eingesetzt wird. Sie hat einen reichhaltigen Funktionsumfang und birgt daher viele Fallen für Systemadministratoren. PHP kann entweder als externes CGI für jeden Skriptaufruf gestartet werden oder als Modul in den Apacheprozeß integriert werden. Unabhängig von der Installationsmethode wird der PHP Interpreter immer zusätzlich als ausführbare Datei miterzeugt. Wir konzentrieren uns nun auf PHP eingebunden als Modul.

Quellcodeübersetzen mit ./configure und die Grundkonfiguration

PHP läßt sich sehr gut aus dem Quellcode installieren. Man sollte sich vorher überlegen, ob man die „mainstream" Variante oder eine speziell vorbereitete Version wie Hardened-PHP[28]. Um ein funktionstüchtiges Apachemodul für den Apache 2.0/2.2 zu erhalten, geht man wie folgt vor. [29]

[26] http://dev.mysql.com/doc/refman/5.0/en/security-guidelines.html
[27] http://www.php.net/
[28] http://www.hardened-php.net/
[29] PHP kann mit einer Vielzahl von Optionen konfiguriert werden, welche hier nun nicht in allen Details behandelt werden können.

```
$ ./configure --with-apxs2=/usr/local/apache2/bin/apxs
$ make
# make install
```

Danach muß man die PHP Bibliothek in der Apache Konfiguration eintragen.

```
# Load the PHP module
LoadModule php5_module libexeb/libphp5.o
# Activate the module
AddModule mod_php5.c

# File extensions to be used by the PHP module
AddType application/x-httpd-php .phtml
AddType application/x-httpd-php .php3
AddType application/x-httpd-php .php4
AddType application/x-httpd-php .php5
AddType application/x-httpd-php .php
AddType application/x-httpd-php .inc
AddType application/x-httpd-php .class
AddType application/x-httpd-php .module
```

Die eingetragenen Dateiendungen versuchen möglichst alle PHP-Skripte mitsamt potentiellen Include-Dateien zu erfassen und zum PHP Modul zu leiten. Das verhindert die Herausgabe von PHP-Skripten durch den Apache als Klartext. Natürlich muß man ebenso die Suchreihenfolge für die Indexseite anpassen.

```
DirectoryIndex index.php index.php5 index.htm index.html index.html.var
```

Jetzt muß man unbedingt die PHP-eigene Konfigurationsdatei namens `php.ini` in den Pfad `/usr/local/lib/php/` (abhängig von der `configure` Option `-prefix`) kopieren. Die Quellcodedistribution von PHP enthält Dateien mit sinnvollen Defaults (beispielsweise die Datei `php.ini-recommended`). Die Datei muß unbedingt an der richtigen Stelle stehen, sonst hat sie keinen Einfluß auf PHP! Man kann mittels eines Testskripts und der Funktion `phpinfo()` überprüfen, ob die richtige `php.ini` verwendet wird.

PEAR - PHP Extension and Application Repository

PHP läßt sich ähnlich wie Perl durch das sogenannte PHP Extension and Application Repository (PEAR)[30] um Module erweitern. Mittels `php -m` läßt sich anzeigen welche Module dem Interpreter zur Verfügung stehen. Dies ist immer zu überprüfen und jegliche Anfrage nach Installation zusätzlicher Module ist unbedingt zu prüfen.

PHP Konfiguration

PHP biete derart viele Funktionen, daß die absolut „richtige" Konfiguration sehr zeitaufwendig sein kann. Ein guter Einstiegspunkt ist die bereits erwähnte `php.ini-recommended` Datei als Grundkonfiguration. Darüber hinaus sollte man auf folgendes achten.

- `register_globals = off`
- `allow_url_fopen = off`
- `enable_dl = off`
- `expose_php = off`
- Deaktivieren bestimmer Funktionen mittels `disable_functions`
  - `openlog, apache_child_terminate, apache_get_modules, apache_get_version, apache_getenv, apache_note, apache_setenv, virtual,...`
  - `proc_open , popen, disk_free_space, diskfreespace, set_time_limit, leak, tmpfile, exec, system, shell_exec, passthru,...`

[30] http://pear.php.net/

Welche sind genau abschalten lassen, muß man mit den zu verwendenden Webapplikationen klären.

- `open_basedir = /var/www/`
- maximales Logging und Umleiten der Fehlermeldungen
  - `error_reporting = E_ALL`
  - `log_errors = on`
  - `error_log = /var/www/logs/php/php_error_log`
  - `display_errors = off`
  - `display_startup_errors = off`

  Man darf nicht vergessen die Schreibberechtigungen für den Webserver auf der Logdatei `/var/www/logs/php/php_error_log` zu vergeben. Auf Produktionsservern dürfen Fehler nur im Log erscheinen und nicht an Clients weitergegeben werden.

Limits

Wenn der PHP Interpreter mit der Option `-enable-memory-limit` übersetzt wurde (was unbedingt zu empfehlen ist), dann läßt sich der maximal zulässige Speicherbedarf eines laufenden PHP-Skriptes begrenzen.

- `memory_limit = 8M`
- `post_max_size = 8M`
- `max_input_time = 60`
- `max_execution_time = 30`

Es kann sein, daß bestimmte Webapplikationen mit diesen Grenzen nicht arbeiten können. Beispielsweise verlangen einige Content Management Systeme (CMS) höhere Limits. Dies ist abzustimmen und niemals pauschal hoch anzusetzen.

Heraufladen von Dateien

Idealerweise schaltet man diese Options mittels `file_uploads = off` aus. Wenn dies nicht möglich ist, so sollte man eine Begrenzung setzen und das Verzeichnis für temporäre Uploaddateien ändern.

- `upload_max_filesize = 2M`
- `upload_tmp_dir = /var/www/tmp`

Session Sicherheit

Session Cookies werden verwendet, um die zustandslosen HTTP Anfragen zu benutzergebundenen Sessions/Sitzungen zuzuordnen. PHP generiert dazu serverseitig IDs und legt sie im allgemein zugänglichen `/tmp/` Verzeichnis ab. Damit kann jeder lokale Benutzer am Webserver Sessiondaten übernehmen. Man begrenzt daher die Session-IDs auf ein nur für den Apacheprozeß zugänglichen Bereich. Zusätzlich kann man den HTTP Referrer auf gültige Strings überprüfen, um bestimmten Cross-Site-Scripting Attacken vorzubeugen.

- `session.save_path = /var/www/sessions`
- `session.referer_check = herebeyourdomain.net`

Es empfiehlt sich zusätzliche Maßnahmen zur Sessionkontrolle[31] zu konfigurieren. Einige davon müssen durch die Webapplikation ausgeführt werden.

[31] http://web.luchs.at/tinyget.php?c=0EA71HR

Safe Mode

Bis zur Version 6.0.0 von PHP kann man noch den sogenannten *Safe Mode* einsetzen, um dem PHP Interpreter zu zusätzliche Prüfungen bei Zugriff auf das Dateisystem zu zwingen. Der Safe Mode läßt sich mit `safe_mode = on` einschalten und hat mehrere Parameter, die den Zugriff auf Dateien, Umgebungsvariablen und ausführbare Programme beschränken und regeln.

Ab der Version 6.0 müssen PHP Applikationen und System selbst Maßnahmen zur Absicherung einsetzen. Man muß dann die Berechtigungen derart konfigurieren, so daß der Apache Server nur an die erlaubten Kommandos kommen kann (z.B. durch Einschränkung auf die `bin/` und `sbin/` Verzeichnisse oder Installation in ein `chroot` bzw. Jail).

PHP als CGI

Auf Mehrbenutzersystemen muß man oft die einzelnen PHP Skripte der Benutzer voneinander abschotten. Dazu muß man PHP als CGI implementieren und pro Benutzer ein eigenes Konto einrichten. Der Apache bietet dafür die Möglichkeit über die `suEXEC` Option CGI Skripte mit verschiedenen Benutzern auszuführen. Man kann dann pro Benutzer eine eigene PHP Konfiguration definieren. Darüber hinaus können dann die Berechtigungen so gewählt werden, daß die PHP Skripte nur Zugriff auf die eigenen Daten haben.

6.4.5. Postfix Mail Transport Agent

Der Postfix Mail Transport Agent[32] ist zwar von Haus aus „sehr sicher" konfiguriert, aber man muß auch dort auf bestimmte Umstände achten. Postfix liest seine Hauptkonfiguration aus den Dateien

- `/etc/postfix/main.cf` und
- `/etc/postfix/master.cf`.

Die Datei `master.cf` konfiguriert Subprozesse, unter anderem auch Möglichkeiten in den Mailtransport einzugreifen. In `main.cf` befinden sich die serverweiten Einstellungen. Bei MTAs muß vor Inbetriebnahme die Rolle klar definiert sein. Soll der MTA nur lokaler Mailserver sein? Dürfen andere E-Mails über ihn verschicken oder nicht? Möchte man Verschlüsselung mit Zertifikaten (TLS) haben oder nicht? All das wird in der `main.cf` konfiguriert. Die wichtigsten Parameter, die man prüfen sollte, sind:

- `smtpd_banner` - definiert den Grußstring für SMTP/ESMTP. Man entfernt üblicherweise alle Hinweise auf Hostnamen, Domain, Softwareversion und Betriebssystem.
- `mynetworks` - gibt vertrauenswürdige Netzwerke an, die über diesen Server E-Mails verschicken dürfen.
- `inet_interfaces` - gibt die Netzwerkgeräte an, an die Postfix binden soll. Für rein lokalen Einsatz kann hier das Loopback Device `lo` angegeben werden.
- `smtpd_recipient_restrictions` - Einschränkungen für die Annahme von Daten via SMTP/ESMTP. Hier konfiguriert man Blacklisten, Protokollprüfungen, Mustern von Sendern bzw. Empfängern, Greylisting und anderes.
- `disable_vrfy_command = yes` - deaktiviert das VRFY Kommando. Mittels VRFY lassen sich mitunter gültige Benutzerkonten erraten.

Die meisten Distributionen sind zwar nach Installation nur auf lokale Verwendung des Postfix konfiguriert, jedoch schadet eine Überprüfung nicht. Für rein lokale Verwendung sollte der Port 25/TCP geschlossen sein.

[32] Als Mail Transport Agent (MTA) bezeichnet man die umgangssprachlich mit „Mailserver" titulierte Applikation, die SMTP/ESMTP spricht.

7. Härten von Netzwerken

```
As security or firewall administrators, we've got basically
the same concerns [as plumbers]: the size of the pipe, the
contents of the pipe, making sure the correct traffic is in
the correct pipes, and keeping the pipes from splitting and
leaking all over the place. Of course, like plumbers, when
the pipes do leak, we're the ones responsible for cleaning
up the mess, and we're the ones who come up smelling awful...
     -- Marcus J. Ranum
```

7.1. Härten von Netzwerken?

Eigentlich sollte dieses Kapitel vor dem Part mit den Paketfiltern angeordnet sein, denn wenn man den Paketfilter konfiguriert, ist es schon längst zu spät. Warum? Weil die Verkabelung und die Switches schon in Betrieb sind (meistens jedenfalls); speziell bei drahtlosen Netzwerken wird kaum auf die physische Ausdehnung geachtet. Dieser Umstand ist aber normal, denn man hat nur ganz selten die Chance komplett neu anzufangen und alles „richtig" zu machen (Ausnahme sind Umzüge oder Erweiterungen des Büro-/Produktionsraums). Aber auch ohne Neuanfang sollte man immer versuchen die Sicherheits des Netzwerks stetig zu hinterfragen oder zu verbessern. Schließlich verlassen sich alle auf das Netzwerk:

> The wire protocol guys don't worry about security because that's really a network protocol problem. The network protocol guys don't worry about it because, really, it's an application problem. The application guys don't worry about it because, after all, they can just use the IP address and trust the network.
>
> --- Arnold G. Reinhold

Man kann Netzwerke genauso härten wie Server und Hosts. Da Netzwerkkomponenten ebenso zahlreich wie vielfältig sind, gibt es in diesem Abschnitt nur ein paar grundlegende Checklisten, die möglichst allgemein gehalten sind. Manche Hersteller stellen diesbezüglich eigene Dokumentation oder sogar Kurse zur Verfügung. Im Zweifelsfalle sollte man die Produktbeschreibung konsultieren.

Wichtig: Prinzipiell gilt durch alle Schichten des OSI Modells, daß jedwede Managementfunktion eines Netzwerkgeräts nur für die (Sicherheits)Administration zugänglich sein darf. Wenn der Zugang nicht geregelt werden kann, so sind andere Maßnahmen zu treffen, die diese Einschränkung erfüllen

7.2. Zugang zu Layer 1

Netzwerkkomponenten müssen störungsfrei untergebracht werden. Das schließt Störungen durch Dritte ein. Dritte können Bagger, Bohrmaschinen, Sägen, Blitzschlag, Wassereinbruch, Techniker oder gar Angreifer sein. Netzwerkkomponenten müssen so gut wie möglich von Manipulation oder Schaden bewahrt werden. Vergraben, Einziehen in Kabelkanäle und Wegschließen sind daher eine gute Idee. Es kann auch nicht schaden der Verlauf der Kabel periodisch zu überprüfen und die Gebäude-/Grundstückspläne einzubeziehen. Speziell in Büroräumen mit mehreren Parteien haben mitunter zuviele Personen Zugang zu Netzwerkkomponenten.

Der Layer 1 ist ganz wichtig bei dem Einsatz von drahtlosen Netzwerken. Die Antennenform und -abstrahlung sowie die Positionierung der Access Points kann man gezielt für eine bessere Abschirmung ausnutzen. Die Frequenzwahl (2,4 GHz, 5 GHz, . . .) und die Sendeleistung spielt ebenfalls eine große Rolle.

7.3. Zugang zu Layer 2

Eine weit verbreitete Praxis in Netzwerken ist der freie Zugang zum Layer 2. Sobald man eine Patchdose mit Ethernetzugang findet, kann man sie verwenden. Das muß nicht so sein. Mit passenden Switches läßt sich der Zugang nur

auf registrierte Clients beschränken, die sich gegenüber dem Switch ausweisen müssen. Die Maßnahme nennt sich *port-basiertes Network Admission Control (NAC)*. IEEE 802.1X ein Standard für NAC. Die Aufgaben von NAC umfassen

- eindeutige Identifizierung von Benutzerinnen und Geräten,
- Einhaltung von Sicherheitsrichtlinien,
- Quarantäne von nicht registrierten oder konformen Clients sowie
- Verwaltung von Nutzergruppen.

Die Bedingungen für den Einsatz von 802.1X sind dafür vorbereitete Switche sowie ein im Netzwerk vorhandener Authentifizierungsserver mit RADIUS[1] oder DIAMETER[2] Unterstützung. Zusätzlich muß der Client eine Softwarekomponente, den *Supplicant*, installiert haben. Diese verwendet das Extensible Authentication Protocol (EAP) als Authentifizierungsprotokoll. EAP bietet eine generische Unterstützung bei der Authentifizierung und wird bei NAC oft als Standard verwendet. Abbildung 7.1 zeigt die Reihenfolge der Schritte, um Zugang zum Netzwerk zu erlangen.

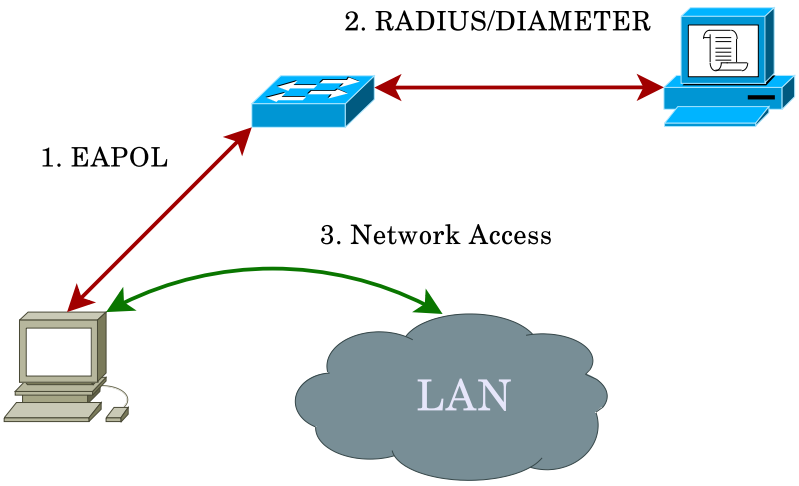


Abbildung 7.1.: Der Client (*Supplicant*) wendet sich an den Switch und schickt EAP Daten verpackt in EAPOL Paketen (1). Der Switch fragt beim Authentifizierungsserver via RADIUS/DIAMETER an, ob der Supplicant gültig ist (2). Bei erfolgreicher Authentifizierung darf der Client auf das Netzwerk zugreifen (3).

Die Verwendung von 802.1X geschieht oft nur bei „idealen Bedingungen", da alle Komponenten 802.1X-fähig sein müssen und die Registrierung von Clients notwendig ist. 802.1X ist allerdings die einzige Möglichkeit eine starke Authentifizierung möglichst früh mit Hilfe der aktiven Netzwerkkomponenten durchzusetzen.

Eine einfachere Variante stellt die Konfiguration von *port-based security* an den Switches und das Freischalten von MAC-Adressen mit den damit verbundenen Problemen (MAC Spoofing). Manche Switches erlauben das Korrelieren von DHCP Transaktionen mit Clients und das Sperren von nachträglich an anderen Ports erscheinenden, gefälschten MAC Adresse (DHCP Snooping). Zusätzlich läßt sich die Anzahl der MAC-IP-Adreßkopplungen pro Port festlegen, um das Einführen zusätzlicher Switches zu verhindern.

7.4. Trennen auf Layer 2

Eine unmittelbare Trennung auf dem Layer 2 läßt sich durch Einführen von physisch getrennten Ethernetsegmenten herbeiführen. Man nehme einen eigenen Switch für das LAN und einen weiteren für die DMZ, wobei beide Switches

[1] RADIUS = Remote Authentication Dial In User Service, ein Protokoll für zentralisierte Authentifizierung, Autorisierung und zum Accounting (AAA)

[2] Diameter ist ein Authentifizierungs-, Autorisierungs- und Accountingprotokoll; Nachfolgeprotokoll von RADIUS

nicht miteinander verbunden werden dürfen. Da man Switches nicht immer mit genau passenden Ports kaufen kann, hat man sich eine Trennung von Ethernetsegmenten auf Basis von Software einfallen lassen - Virtual LANs (VLANs).

Das VLAN Format ist mittlerweile durch IEEE 802.1Q geregelt, welches das Markieren der Ethernetframes beschreibt.[3] Ein IEEE 802.1Q Ethernetframe-Kopf enthält eine 4 Byte lange Markierung.

- 3 Bit Prioritätsfeld
- 1 Bit Canonical Format Indicator (CFI)
- 12 Bit VLAN Identifikation (VLAN-ID)

Die VLAN-IDs können $2^{12} = 4096$ verschiedene Werte annehmen. Das VLAN mit der ID 1 ist dabei immer das *Management VLAN*, welches Zugriff auf die Managementfunktionen von Netzwerkgeräten hat. Im Management VLAN dürfen daher keine Clients und keine Hosts sein, die im normalen Betrieb sind. Die maximale Größe für Ethernetframes mit VLAN Tag ist mit der Markierung daher 1522 Byte statt der für Ethernet typischen 1518 Byte.Prinzipiell lassen sich nun damit zwei Arten von VLANs aufbauen.

- statische VLANs mit fester Zuordnung zwischen Ethernet-Port und VLAN-ID
- dynamische VLANs mit Zuordnung von MAC-Adresse, Protokoll oder Benutzername und VLAN-ID[4]

Switches bestimmter Hersteller können VLAN Frames gemeinsam über Datenverbindungen transportieren und dabei die VLAN-IDs erhalten. Bei Cisco geschieht dies über das *VLAN Trunking Protocol (VTP)*. Die VLAN Tags der Frames werden beim Transport über die Trunks erhalten und bei der Ausgabe an passende Ports bzw. Clients entfernt. Der Umgang mit Tagged Frames ist bei Verwendung angepaßter Netzwerktreiber möglich.

Bei Einsatz von dynamischen VLANs ist unbedingt darauf zu achten, daß kein normaler Zugangsport auf VTP oder ähnliche Protokoll reagieren kann. Ein Angreifen kann so VLANs manipulieren oder Netzwerkverkehr abgreifen.

7.5. Trennen auf Layer 3

Prinzipiell wird die Trennung auf Layer 3 und darüber von Firewalls oder Routern geregelt. Es gibt allerdings auch Switches, die bis zum Layer 4 gehen, um Kopfdaten zweckes Weiterleitung oder VLAN-Zugehörigkeit zu überprüfen. Das erhöht die Komplexität des Switches, kann aber bei der Sicherheitsadministration sehr hilfreich sein.

7.5.1. NAT ist keine Sicherheitsmaßnahme!

NAT wurde aus mehreren Gründen erschaffen. Vordergründig sollte es das Umnummerieren von Netzwerken erleichtern und ein gewisses Maß an Privatsphäre schaffen (was aber nicht das Designziel war, nur ein Nebeneffekt). Im RFC 1631 von 1994 steht jedoch in der Sektion Sicherheit:

```
Unfortunately, NAT reduces the number of options for providing security.
[...]
On the other hand, NAT itself can be seen as providing a kind of privacy mechanism.
```

Im RFC 2663 betitelt mit *IP Network Address Translator (NAT) Terminology and Considerations* liest man folgendes:

```
9.0 Security considerations

Many people view traditional NAT router as a one-way (session)
traffic filter, restricting sessions from external hosts into their
machines.
[...]
The combination of NAT functionality, ALGs and firewalls will provide
a transparent working environment for a private networking domain.
```

Dort steht in klaren Worten, daß *the combination of NAT functionality, ALGs and firewalls* für die Sicherheit verantwortlich ist. Ohne NAT sind es nur die Application Layer Gateways und Firewalls, die man aber sowieso immer hat. NAT ist daher eigentlich überflüssig, was sich spätestens beim Einsatz von IPv6 herausstellt. Dort wollte man vor kurzem wieder NAT einführen. Das Internet Architecture Board (IAB) hatte dazu diese Meinung.

```
As discussed in [RFC4864] „Local Network Protection", Section 2.2, the act
of translation does not provide security in itself. The stateful filtering
function can provide the same level of protection without requiring a
translation function.
```

Bitte merken und nie wieder vergessen!

[3] Vor 802.1Q waren proprietäre Protokolle wie Ciscos Inter-Switch Link (ISL) oder 3coms Virtual LAN Trunk (VLT) in Verwendung.
[4] Die Möglichkeiten der Zuordnung hängen von den Fähigkeiten des Switches ab.

7.5.2. Netzwerkzugang mittels IPsec

In bestimmten Szenarien läßt sich eine Zugangskontrolle auch über eine durchgehende IPsec Konfiguration implementieren. Auf diese Weise sind IPsec-konfigurierte Maschinen „unter sich“. Da alle IPv6-fähigen Systeme IPsec unterstützen müssen, ist diese Option realistisch. Das einzige Problem sind Protokolle wie beispielsweise DHCP oder Autoconf, da die IPsec Konfiguration von Anfang an aktiv sein muß. Es bieten sich statische Adressen mit all ihren Nachteilen an.

Dehnt man die IPsec Konfiguration auf den Gateway aus, so kann man unauthorisierte Gastsysteme vom Netzwerk und dem Internetzugriff ausschließen. Authorisierte Systeme müssen vorkonfiguriert werden (sei es mit festen Paßworten oder X.509 Zertifikaten).

7.6. Drahtlose Netzwerke

Seit dem Knacken von WEP sind drahtlose Netzwerke auf dem Radar von Sicherheitsverantwortlichen. Wenn man es genau nimmt, dann sind drahtlose Netzwerke sehr einfach abzusichern. Das Kürzel WEP steht für *Wired Equivalent Privacy*. Die Motivation für WEP war das Problem der Luftschnittstelle. Bei verkabelten Ethernetports gibt es (hoffentlich) eine gewisse Zutrittskontrolle, die bei Funkverbindungen fehlt. WEP sollte daher *einen zu Verkabelung äquivalenten Schutz* bieten. Strenggenommen ist das auch gelungen, denn so gut wie niemand benutzt starke Verschlüsselung und Authentifizierung über Kupferkabel.[5] Wo liegt also das eigentliche Problem?

7.6.1. Begriffsverwirrung

Leider gibt es im Bereich der drahtlosen Netzwerke eine babylonische Begriffsverwirrung. Der Grund liegt im Drängen der Wi-Fi Alliance um Standards, die von der IEEE erst nach Markteinführung der Produkte finalisiert wurden. Die Wi-Fi Alliance hat Standards unter einem anderen Namen verwendet, um nicht warten zu müssen bis die Standards vollständig fertig spezifiziert waren.

WEP - Wired Equivalent Privacy (1997)
: WEP sollte die Luftschnittstelle sichern und nur Zugang durch zugelassene Clients ermöglichen. WEP setzt statische Schlüssel mit der RC4 Stromverschlüsselung und CRC32 als Message Authentication Code (MAC) ein.

WPA - Wi-Fi Protected Access (1999)
: WPA wurde als „Quick Fix” für Schwachstellen in WEP von der Wi-Fi Alliance herausgebracht. WPA verwendet immer noch die RC4 Stromverschlüsselung mit dem Temporal Key Integrity Protocol (TKIP). Der Schlüssel wechselt periodisch, so daß nicht mehr alle Pakete mit demselben Schlüssel verschlüsselt werden. Zusätzlich wurde der Integritätscheck für gesendete und empfangene Paket verbessert, um Injektionsattacken abzuwehren. Die Maßnahme wurde als Zwischenschritt eingeführt, um bereits ausgelieferte Netzwerkkarten und Access Points mittels der Firmware nachrüsten zu können.

WPA2 - Wi-Fi Protected Access 2 (2004)
: WPA2 erweitert WPA um Counter-Mode/CBC-Mac Protocol (CCMP) und AES Verschlüsselung. Viele ältere Geräte können diese Optionen nicht unterstützen und nicht nachgerüstet werden. Der 256 Bit lange AES Schlüssel wird durch Kombination eines Mantras (64 Bit Hexdazimalzahlen oder 8 bis 63 druckbare ASCII-Zeichen), der Password-Based Key Derivation Function (PBKDF2), der SSID als Salz und 4096 Iterationen HMAC-SHA1 gebildet.

IEEE 802.11i-2004 stellt einer Erweiterung des IEEE 802.11 Standard im Hinblick auf Sicherheit dar. 802.11i-2004 löst 802.11i-1999 ab. WPA implementiert nur einen Teil von 802.11i, WPA2 implementiert 802.11i-2004. 802.11i verwendet AES Verschlüsselung. Damit man vollends den Überblick verliert, nennt die Wi-Fi Alliance 802.11i WPA2 oder Robust Security Network (RSN).

Moderne Netzwerke sollten ausschließlich WPA2 / 802.11i mit CCMP und AES unterstützen. In allen Fällen wo man noch ältere Netzwergeräte unterstützen muß, wird die Verwendung eines zweiten Netzwerks mit WEP oder WPA empfohlen. Es ist ganz wichtig nicht zuviele Betriebsmodi in ein einziges Gerät zu verpacken, da sonst der kleine gemeinsame Nenner zum Tragen kommen kann.

Für WEP existieren mindestens 17 verschiedene Angriffsmethoden. Ein WEP Schlüssel läßt sich durch einen Angreifer, der regulären Datenverkehr „sehen” kann, binnen 40.000 bis 60.000 Datenpaketen knacken (entspricht ca. 15 bis 45 Minuten, je nach Aufkommen von Datenpaketen).

[5] Das ist allerdings problemlos möglich, wenn man die passende 802.1X Infrastruktur hat.

WPA, WPA2 und 802.11i im „pre-shared key" (PSK) oder *Personal Mode* lassen sich durch Anwenden von Wörterbuchattacken oder Durchprobieren von Schlüsseln angreifen. Generell ist der Nachteil bei festen Mantras der statische Schlüssel. Die Verwendung von X.509 Zertifikaten ist immer bevorzugt. Es ist sehr empfehlenswert die Verwendung der Schlüssel bei WPA, WPA2 und 802.11i im Detail nachzuvollziehen, da im Betrieb mehrere Schlüssel im Einsatz sind, die aus dem Mantra abgeleitet werden. [56]

7.6.2. Trennung

Der wichtigste Punkt ist wieder die Trennung. Drahtlose Netzwerke müssen von verkabelten Netzwerken abgeschirmt werden. Der einzige Übergang, wenn ein solcher existiert, muß durch passende Maßnahmen geregelt werden. Darüber hinaus müssen die Datentransmissionen vor Dritten geschützt werden. Eine gängige Praxis umfaßt die folgenden Maßnahmen.

1. WPA2 / 802.11i als äußere Hülle (möglichst ohne Unterstützung der alten 802.11b und 802.11g Protokolle)
2. Torwächter als Zutrittskontrolle und Filter
3. optional Sondensysteme, die zusätzliche Access Points oder Clients aufspüren können
4. VPN über IPsec, OpenVPN™ oder SSL/TLS als innerer Schutz

Im ersten Schritt ist der bereits besprochene 802.1X Modus die beste Wahl (wird auch oft als *WPA Enterprise Mode* bezeichnet), weil dabei die Access Points keinerlei Authentifizierungsinformationen enthalten und nur als Relay dienen.

Eine gut abgesicherte, drahtlose Infrastruktur darf den ganzen Schutz nicht auf einer einzigen Maßnahme aufbauen. Spätestens jetzt sollte klar sein warum wir ständig über Sicherheitsarchitektur reden. Man kann Systeme nicht nur an einer Stelle absichern.

8. Webapplikationen

```
First we thought the PC was a calculator. Then we found out
how to turn numbers into letters with ASCII - and we thought
it was a typewriter. Then we discovered graphics, and we
thought it was a television. With the World Wide Web, we've
realized it's a brochure.
     -- Douglas Adams
```

8.1. Das W³ und die Systemadministration

Wenn nun die Firewall steht, die Content Filter aktiv sind und die Webserver samt aktiven Skriptkomponenten darauf gehärtet sind, dann bleibt noch viel zu tun. Sehr viele Attacken, die in den letzten Jahren gängig geworden sind, benötigen nur die Ports 80 oder 443. Beide sind so gut wie immer offen oder zumindest über Proxies zugänglich. Beide verbinden zwei kritische Komponenten, welche nach wie vor Sicherheitslücken werden können - nämlich Webapplikation und Webbrowser.

Sowohl Webapplikationen als auch Webbrowser sind Plattformen, die Programmiersprachen unterstützen. Zusätzlich bieten sie prinzipiell Zugriff zum lokalen System, sei es Server oder Client. Verbunden sind beide über das zustandslose HTTP/HTTPS. Diese Kombination ist ein fruchtbarer Boden für Probleme, und daher wird in diesem Kapitel nochmals ausdrücklich darauf hingewiesen. [53]

8.2. Angriffe und Schwachstellen

Es folgt eine unvollständige Aufzählung von möglichen Angriffen und Schwachstellen.

8.2.1. Vorbereitungen

Jede Webapplikation hat ihre Eigenheiten. Das fängt bei der URL an, geht über Cookies, Session IDs und Pfade mit Parametern zu den einzelnen Skripten am Server. Jeder Angreifer muß diese Informationen ausforschen. Die einfachste Methode dafür ist die Verwendungen eines protokollierenden Proxies (z.B. SPIKE, BurpSuite oder OWASP Zed Attack Proxy) auf der lokalen Maschine und das „Durchsurfen" der Applikation. Danach stehen durch den Proxy Informationen über die Struktur der Webapplikation zur Verfügung. Zusätzlich erfolgt eine Identifikation der Plattform (Webserver, Skriptsprachen, Versionen, etc.), um die nächsten Schritte vorzubereiten.

Suchmaschinen

Suchmaschinen sind sehr ergiebige Mittel, um an bestimmte Informationen heranzukommen. Google kann beispielsweise nach bestimmten Textfragmenten in einer URL suchen oder ganz gezielt bestimmte Dokumentenformate in den Suchergebnissen präsentieren.

- `inurl:phpmyadmin`
- `allinurl:lebenslauf.doc`
- `password login filetype:xls`
- `eggdrop filetype:user user`
- `Sony SNC-RZ30 Network Cameras`
- `PHP Shell (unprotected)`
- `ext:ppk ssh key -github.com -gitlab`

- `inurl:/wp-content/ai1wm-backups + wpress`

Es gibt noch viele bessere Beispiele in der Google Hacking Database[1].

8.2.2. Plattformen

Plattformen lassen sich mitunter direkt angreifen, wenn die Software bekannte Fehler hat.

8.2.3. Authentisierung

Im W³ sind Logins sehr beliebt. Sie dienen meist dazu über das zustandslose HTTP/HTTPS Informationen zu generieren, um einzelne Sessions mit Clients identifizieren zu können. Manche Logins dienen nur dazu bestimmte Bereiche eines Webservers schwerer zugänglich zu machen. HTTP selbst bietet eine Reihe von Authentisierungsmethoden an, die dazu verwendet werden könnnen.

Basic HTTP AUTH

Basic HTTP AUTH[2] übermittelt Benutzername und Paßwort im Klartext. Die Informationen sind im HTTP Header abgelegt und können während dem Login bzw. danach bei jedem Seitenaufruf mitgelesen werden. Es bietet sich hier die Verwendung von HTTPS an.

HTTP Digest Access Authentication

Digest Access Authentication[3] versucht die Logindaten in eine andere Form zu bringen.

1. Client setzt normalen HTTP Request ohne Login an den Server ab.
2. Server antwortet mit Code 401 und übermittelt den *authentication realm* als String und eine zufällig generierte Zahl *nonce*, die nur einmal verwendet wird.
3. Clientsoftware zeigt den *authentication realm* an und bittet um Benutzername sowie Paßwort.
4. Aus der Eingabe generiert der Client einen Hash, der *authentication realm*, Paßwort und Benutzernamen enthält.
5. Client schickt die kodierten Daten an den Server.
6. Server bestätigt oder weist Client zurück, wenn die Eingabe falsch war.

Ein Zeitstempel wird in die Generierung des *nonce* Wertes eingebaut, um Replay Attacken zu erschweren. Mittlerweile unterstützen allen gängigen Webbrowser Digest Access Authentication, daher ist diese Methode bei der Konfiguration von Zugangsbeschränkungen unbedingt vorzuziehen. [4]

CAPTCHA

CAPTCHA steht für „*Completely Automated Public Turing test to tell Computers and Humans Apart*". Die Idee dahinter ist die Kodierung von Informationen in Bildern, die nur ein Mensch dekodieren kann. Oft handelt es sich dabei um verzerrte Texte, die mit Software nicht gelesen werden sollen. Das Prinzip funktioniert relativ gut, es hat jedoch Nachteile. Beispielsweise sind CAPTCHAs nicht immer behindertengerecht. Blinde Computerbenutzer können keine Bilder sehen. Ein weiterer Nachteil besteht in der nach wie vor vorhandenen Möglichkeit die Informationen dekodieren zu können. Es gibt Algorithmen zur Bilderkennung, die mitunter doch an den Text im Bild kommen können. Ein Bespiel dafür ist EZ-Gimpy[5], welches bei den CAPTCHAs der Yahoo! Registrierung eine Trefferquote von bis zu 92% hatte. Abbildung 8.1 zeigt ein Beispiel für eine mit Gimpy generierte CAPTCHA Grafik mit mehreren Worten.

CAPTCHA können umgangen werden, wenn der Algorithmus zur Verzerrung nicht gut genug ist. Darüber hinaus werden für die Umgehung große Gruppen von Angestellten in Billiglohnländern eingesetzt, deren Dienste vermietet werden. CAPTCHA kann daher nur ein kleiner Teil des Sicherheitskonzepts sein.

[1] https://www.exploit-db.com/google-hacking-database/
[2] RFC 2617
[3] RFC 2617
[4] Eine Ausnahme stellt die Anbindung an einen LDAP Server dar, denn das passende Apache Modul dafür kann HTTP Digest Access Authentication nicht verwenden. Man muß dann SSL/TLS und Basic HTTP AUTH einsetzen.
[5] http://web.luchs.at/tinyget.php?c=Fqrb

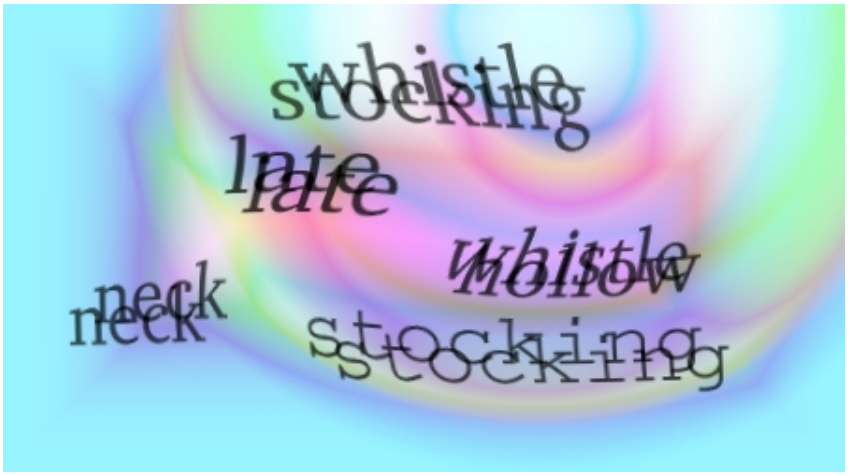


Abbildung 8.1.: So sieht eine CAPTCHA Grafik aus, die mit Gimpy generiert wurde. Hier sind mehrere Worte gezeigt. Am Client müssen ein oder mehrere Worte eingegeben werden.

8.2.4. Autorisierung

Durch eine Autorisierung stellt ein System fest, ob ein Client eine bestimmte Operation durchführen darf. Autorisierung erfolgt während der Benutzung einer Applikation stetig und auch noch nach der Authentisierung. Bei Webapplikationen werden oft Session IDs und Cookies gepaart mit Parametern verwendet.

Session Hijacking

Meist reicht die Kenntnis einer Session ID für die Übernahme einer Session aus. Werden diese IDs nach einem vorhersagbaren Muster generiert, so ist die Übernahme trivial. Gelingt es einem Angreifer die ID über andere Mittel wie beispielsweise Sniffing oder Auslesen der Cookies am Browser zu ermitteln, so kann ebenso eine Session übernommen werden. Eine Methode für lokales Auslesen sind *Cross-Site Scripting* (CSS) Attacken oder Rogue Proxies.

Session Fixation

Man kann auch versuchen die Session ID auf einen fixen Wert zu setzen. Die kann geschehen, wenn die Webapplikation die ID aus Parametern eines HTTP Requests übernimmt.

1. Versenden des Links `http://unsafe/?SID=FIXEDSID` an ein Opfer.
2. Opfer klickt auf Link und loggt sich ein.
3. Angreifer kennt die ID.

Das kann auch mit servergenerierten Session IDs funktionieren. Eine weitere Methode ist das Setzen von IDs durch *Cross-Site Cooking*

1. Versenden des Links `http://attack/` an ein Opfer.
2. Opfer klickt auf Link, und Webseite setzt die ID auf `FIXEDSID` für die Domain `trusted`.
3. Versenden der Aufforderung sich bei `http://trusted/` einzuloggen an Opfer.
4. Angreifer kennt die ID.

Übliche Gegenmaßnahmen sind die folgenden.

- Keine Session IDs aus `GET`/`POST` Requests übernehmen.
- Verwenden von SSL/TLS.

- Regenerieren der ID bei jedem Request oder in bestimmten Perioden.
- Ausschließlich servergenerierte IDs akzeptieren.
- Logout Funktion anbieten (und diese nur mit POST implementieren).
- IDs mit einer Lebensdauer versehen.
- Referrer prüfen und bei Unstimmigkeit die Session löschen.
- ID mit anderen Daten aus dem HTTP Request gegenprüfen.
- Verschlüsseln von Daten in Cookies

Cross-Site Request Forgery (CSRF)

Cross-Site Request Forgery (CSRF) nennt man auch *One-Click Attack* oder *Session Riding*. Es bezeichnet das Absetzen von unauthorisierten Kommandos an Webseiten von Clients, denen die Webapplikation vertraut. Alle Attacken haben folgendes gemeinsam.

- Die Webseite verläßt sich auf die Identität der Benutzerin.
- Ausnutzen dieses Vertrauensverältnisses.
- Der Browser des Benutzers wird verwendet, um Anfragen an den Zielserver zu schicken.
- Abgesetzte HTTP Anfragen haben Nebenwirkungen.

Alle Webseiten, die Aktionen aufgrund von Aktionen durch vertrauenswürdige Benutzerinnen ausführen, aber die für diese bestimmte Aktion keine Autorisierung verlangen. Typische Aktionen sind solche, die eine HTTP POST Anfrage zur Folge haben.

Gegenmaßnahmen umfassen:

- Begrenzung der Lebensdauer von Cookies zur Authentisierung.
- Überprüfen des HTTP Referer Kopfes.
- Sicherstellen, daß keine `crossdomain.xml` Datei anderen Webseiten über den Adobe® Flash Player Zugriff erteilt.
- Alle Formulare mit Security Tokens versehen, die pro ausgegebenem Formular eindeutig sind (Tokens werden auch in der Session pro Benutzer gespeichert und nach dem Request verglichen). Die angreifende Webseite kann dieses Token dann nur raten. Diese Maßnahme läßt sich auf HTTP GET Aktionen erweitern. Im Prinzip darf die Information eines Cookie alleine nicht zur Autorisierung herangezogen werden.

8.2.5. Input Validation

Die Überprüfung von Daten, die mit Webclients ausgetauscht werden, ist ein weites Feld. Schwachstellen fangen bei der Kodierung der URL, und damit der Parameter, an und gehen bis hin zu HTML und SQL Injection. Es ist daher wichtig, daß jede Komponente am Client und speziell am Server übertragene Daten bereinigt. Das schließt ein Filtern von unerwünschten Daten und ein Prüfen auf erlaubte Formate mit ein.

Bei Applikationen, die mit Datenpuffern arbeiten, ist besondere Vorsicht geboten. Die Puffergrößen dürfen nicht mit übergroßen Daten verletzt werden (*Buffer Overflow*). Ob solche Überläufe stattfinden können und welche Effekte sie haben, hängt von der verwendeten Programmiersprache und vom Betriebssystem ab.

Daten prüfen: Textfelder

- „Da kann der Benutzer einen Text eingeben."
- Bedeutung?
  Name? Familienname? Vorname? Adresse? Ort? Kommentar? Beschreibung? Artikel? Buch? Notiz? Land? Einkaufliste? Medikamentenbeschreibung? . . .
- Kodierung bzw. Transformation?
  ASCII-7? ISO-8859-1? ISO-8859-15? UTF-8? UTF-7? UCS-2? UCS-4? EBDIC? Morsezeichen? Wer legt das fest? Was passiert, wenn sich der Server aber nicht der Client daran hält? Was passiert im umgekehrten Fall? Und wenn der Benutzer sich nicht daran hält? Wer prüft die Kodierung? Welche Kodierung ist Default? Wie wird umgewandelt? . . .
- Länge?
  Gibt es ein Maximum? ∞ paßt auf keinen Datenträger. 8 Byte oder 8 Zeichen? 8 Zeichen sind bis zu 48 Byte in UTF-8. Was passiert, wenn eine maximale Länge überschritten wird? . . .
- Ziffern und Zeichen
  Welche sind erlaubt? Welche sind nicht erlaubt? Welche verträgt die Datenbank ohne Mißverständnisse? Welche müssen umgewandelt werden? . . .
- Datenstrom durch Applikation
  Weiß jede Komponente was sie erwartet und kann sie mit allen denkbaren Datenformaten umgehen? Brauche ich eine Normalisierung der Daten? Wenn ja, an welcher Stelle oder an welchen Stellen? Wenn nein, warum nicht? . . .
- **Kurzfassung: Plain Text ist ausgestorben!**
  Bitte allen Betroffenen weitersagen!

Lektüre zum Thema:
Buchstabensuppe[6]
The Absolute Minimum Every Software Developer Absolutely, Positively Must Know About Unicode and Character Sets (No Excuses!)[7]

Daten prüfen: URLs

- „Bitte geben Sie eine URL ein."
- generische Form laut RFC 3986
  `<scheme name> : <hierarchical part> [ ? <query> ] [ # <fragment> ]`
- Protokolle
  `http://`, `https://`, `ldap://`, `news://`, `gopher://`, `mms://`, `rtsp://`, `ftp://`, `sip://`, ...
- Blick auf minimale und maximale Form
  - `a://a` ist das theoretische Minimum
  - Maximum für Internet Explorer ist 2083 Zeichen[8]
  - Firefox 1.5.x zeigt nur 65535 Zeichen an, nimmt aber mehr als 100000 Zeichen[9]
  - Safari und Opera nehmen mehr als 80000 bzw. 190000 Zeichen
  - Apache Webserver 2.0 gibt nach mehr als 8000 Zeichen die Fehlermeldung „414 Request-URI Too Large" zurück
  - Microsoft IIS erlaubt maximal 16384 Zeichen (konfigurierbar)
  - Perls `HTTP::Daemon` erlaubt 16384 Zeichen (im Source änderbar)

[6] http://www.hjp.at/publ/buchstabensuppe/
[7] http://www.joelonsoftware.com/articles/Unicode.html
[8] http://support.microsoft.com/kb/208427
[9] http://www.boutell.com/newfaq/misc/urllength.html

- Gültigkeit
  Servernamen im DNS „auflösbar"? URL „aufrufbar"? Statuscodes? . . .
- URLs bestehen aus Text
  Siehe vorheriges Unterkapitel . . .

W3C Naming and Addressing[10]

8.2.6. Datenbanken und Storage

Der überwiegende Teil der dynamischen Webapplikationen kommt nicht mehr ohne Datenbanken aus. Die dargestellten Inhalte hängen von den Aktionen der Clients ab. Das bedeutet, daß Daten aus dem HTTP Request in der einen oder anderen Form in einer Datenbankabfrage landen. Damit eröffnen sich einem Angreifer Möglichkeiten indirekt das Backend zu manipulieren. Man spricht von *SQL Injection*, wenn man durch Variieren von Parametern SQL Kommandos verändern kann. Andere Angriffe sind ebenso möglich.

- Erraten von Tabellen durch sukzessives Verändern von Parametern und Beobachten der Fehlermeldungen.
- Einfügen von Attackskripten und Schreiben ins Dateisystem.
- Aufzählen von gültigen Benutzernamen.

Gegenmaßnahmen umfassen:

- Minimale Rechte für alle Anwendungen, die Zugriff auf Datenbanken habe.
- Filtern aller Daten, die in Datenbankabfragen landen.
- Verschlüsseln von Daten in der Datenbank.[11]

8.2.7. Web Services

Web Services spielen bei heutigen Webapplikationen eine zunehmende Rolle. Gemäß dem W3C sind Web Services wie folgt definiert.

```
A software system designed to support interoperable Machine to Machine
interaction over a network.
```

Bei der Übertragung von Daten und Kommandos kommen die folgenden Technologien zum Einsatz.

- Service Oriented Architecture Protocol (SOAP)
  SOAP ist ein in XML formulierter Container für Nachrichten, die darunterliegende Protokolle ansprechen. Es ist primär für HTTP und HTTPS konzipiert, allerdings existieren auch Bindungen für SMTP und Extensible Messaging and Presence Protocol (XMPP). SOAP kann als Nachfolger von XML-RPC gesehen werden.
- Web Services Description Language (WSDL)
  Dies ist ebenso ein Nachrichtenformat in XML, welches Interfaces für Services definiert. Daraus wird Code für Client/Server Interaktion generiert. Zusätzlich können über WDSL Konfigurationen ausgetauscht werden.
- Universal Description, Discovery, and Integration (UDDI)
  Dieses Protokoll dient zur Beschreibung und Detektierung von Web Services, sei es zum Zeitpunkt der Generierung oder zur Laufzeit. Über dieses Protokoll können andere Applikationen Web Services finden.

Abbildung 8.2 zeigt das Zusammenspiel der Protokolle in einem Überblick. Web Services sind aus mehreren Komponenten aufgebaut, die teilweise ihre API direkt aus dem Quellcode automatisch generieren. Die Komplexität läßt Raum für Fehler, und XML läßt sich ähnlich manipulieren wie HTML oder andere Datenformate. XML Parser sowie die Applikation müssen nicht wohlgeformte Nachrichten abfangen und entsprechend reagieren. Es gibt die Möglichkeit XML Nachrichten mit Security Tokens[12] zu versehen. Das ist aber optional und wird nicht immer genutzt. Service Broker bieten sich noch dazu sehr gut für die Aufklärung an, wenn sie direkt zugänglich sind.

[10] http://www.w3.org/Addressing/
[11] Verschlüsselung in Datenbanken ist in einigen Ländern rechtlich verpflichtend.
[12] http://www.ibm.com/developerworks/library/specification/ws-secure/

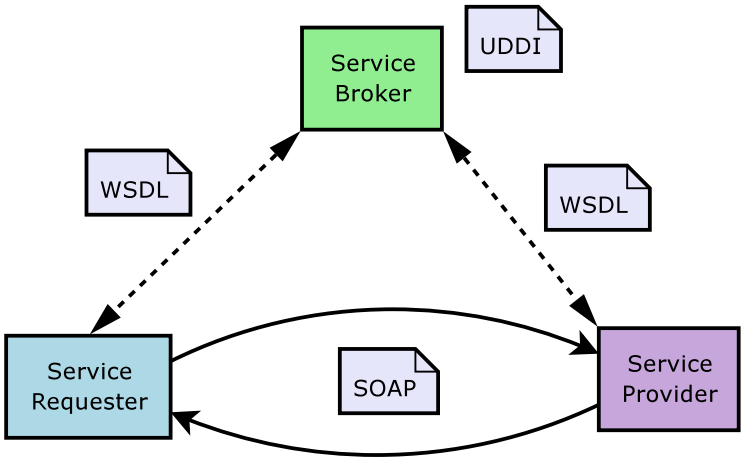


Abbildung 8.2.: Web Services bestehen aus mehreren Komponenten. Man hat einen Service Broker über den man aktive Services suchen kann. Ein Service Requester kann mittels in SOAP eingeschlossenen Nachrichten einen Service Provider um Bearbeitung oder Informationen bitten. Alle Nachrichten sind in XML kodiert.

8.2.8. Managementzugänge

Managementzugänge sind de fakto Hintertüren, wenn sie nicht richtig abgesichert sind. Sie werden zur Administration genutzt, sei es von Systemadministratoren oder von Redakteuren der Websysteme. Sie können völlig verschieden aussehen. Die Spanne reicht von Telnet oder SSH bis hin zu administrativen Webseiten (eigene Seiten, FrontPage, WebDAV) oder proprietären Netzwerkprotokollen. Alle diese Zugänge müssen unbedingt von Paketfiltern beschützt oder von Monitoringsystemen überwacht werden. Man sollte auch unbedingt mehrere falsche Logins hintereinander detektieren und Maßnahmen setzen, um *Password Brute Forcing* zu verhindern.

8.2.9. Webclients

Im W³ gilt dasselbe wie im wirklichen Leben - man sollte nie alles glauben, was man sieht. Wenn ein Benutzer eine Webseite ansurft, die genauso ausschaut wie das Original, dann können daraus immense Probleme erwachsen. Plugins machen Angreifern zusätzlich das Fälschen von Webseiten einfacher. Das Absichern der Webclients gehört daher zu einem vollständigen Sicherheitskonzept dazu.

ActiveX

ActiveX ist ein clientseitiges Plugin für den Microsoft® Internet Explorer. Es hat eine lange Geschichte von sicherheitskritischen Schwachstellen und ermöglichst es einem Webserver die Kontrolle über einen Client zu bekommen. Es können selbst ActiveX Objekte angegriffen werden, die nicht für den Gebrauch im Internet Explorer gedacht sind.

Java™

Java™ ist eine von Sun Microsystems entwickelte Plattform, deren Objekte in einem Webclient eingesetzt werden können. Java™ bietet zwar für Applikationen eine „Sandkiste" zum Spielen an, aber dies ist als Eindämmung nicht immer ausreichend. Digital signierte Java™ Programme können beispielsweise teilweise aus dieser beschränkten Umgebung ausbrechen.

Cross-Site Scripting (CSS/XSS)

Cross-Site Scripting bezieht sich auf Attacken, die clientbasiertes JavaScript ausnutzen und Skripte in einem nicht dafür gedachten Kontext ausführen. Die meisten JavaScript Implementationen haben ähnlich wie Java™ eine „Sandkiste" für ausgeführte Skripte. Man unterscheidet verschiedene Typen.

- Typ 0 - DOM-basiert oder lokal
  Ein JavaScript Code modifiziert die eigene HTML Seite und zwingt den Browser zusätzliches HTML erneut auszuwerten und dadurch potentiellen neuen JavaScript Code auszuführen. Man versucht damit oft Code in der lokalen Zone des Browsers auszuführen, um mehr Privilegien zu erlangen.

- Typ 1 - temporär oder reflektiert
  Dieser Typ wird häufig verwendet. Es handelt sich dabei um JavaScript Code, welcher über HTTP Requests an eine Webapplikation geschickt wird und durch ungenügendes Filtern bei der Ausgabe einer neuen Webseite ausgeführt wird. Das betrifft oberflächlich nur den Webclient selbst, allerdings können gepaart mit Social Engineering daraus ernste Sicherheitsprobleme entstehen.

- Typ 2 - persistent, gespeichert oder CSS 2. Ordnung
  Der Typ 2 wird auch als *HTML Injection* bezeichnet. Hier wird der JavaScript Code über einen Request zunächst am Server abgespeichert (in einer Datenbank oder am Dateisystem). Ein zweiter Request führt zur Ausführung des Codes. Klassisches Beispiel für betroffene Applikationen sind Gästebücher oder Forensysteme, teilweise auch Webmail.

Cross-Zone und Cross-Domain Schwachstellen

Webclients versuchen Skripte von verschiedenen Domains gegeneinander zu isolieren. Primär kommt dabei die Netscape Same Origin Policy[13] bzw. URL Security Zones[14] zur Anwendung. Es gibt eine Reihe von Attacken, die diese Grenzen durchbrechen und dann beispielsweise Cookieinformationen auslesen oder Code mit erweiterten Privilegien ausführen.

Bösartige Skripte, aktive Inhalte und HTML

Ein Webserver kann auch ganz ohne Ausnutzung von Schwachstellen einem Webclient bösartige Skripte in Form von aktiven Inhalten schicken. Bei aktiven Inhalten handelt es sich nicht nur um Code, sondern um jegliche Inhalte, die am Client durch zusätzliche Plugins bearbeitet werden. Dazu gehören beispielsweise

- Adobe® Flash,
- Adobe® PDF,
- Adobe® PostScrip® 3™,
- Audioformate,
- Dokumentenformate generell,
- Microsoft® Silverlight™,
- Skriptsprachen oder
- Videoformate.

Spoofing

Mit Spoofing im Zusammenhang mit Webseiten bezeichnet mal das Nachbilder einer anderen Webseite zur Täuschung des Benutzers. Dabei beschränkt man sich nicht nur auf den Inhalt, sondern greift auch in die Oberfläche des Webclients ein (z.B. durch Fälschen des Titels, URL-Leiste, TLS-Anzeige, etc.). Die Methode des „Phishing" nutzt diese Techniken intensiv. Beispiele für gefälschte Seiten findet man beispielsweise bei Websense® Security Labs™[15].

[13] http://www.mozilla.org/projects/security/components/same-origin.html
[14] http://msdn.microsoft.com/workshop/security/szone/overview/overview.asp
[15] http://www.websense.com/securitylabs/alerts/

9. „Cloud“ Infrastruktur

```
For every complex problem there is an answer that is clear, simple, and wrong.
 -- H. L. Mencken
```

9.1. Definition der „Cloud“

Der Begriff „Cloud“ oder „Cloud Computing“ hat keine exakte technische Definition. Es handelt sich vielmehr um einen Sammelbegriff von ausgelagerter Infrastruktur, wobei es auch sogenannte „private Clouds“ gibt, die Teil der eigenen Infrastruktur ist. Zusätzlich zum Begriff „Cloud“ gibt es noch Bezeichnungen wie „Software as a Service (SaaS)“, „Infrastructure as a Service (IaaS)“, „Storage as a service (STaaS)“, „Security as a service (SECaaS)“, „Data as a service (DaaS)“, „Test environment as a service (TEaaS)“, „Desktop as a service (DaaS)“ oder „API as a service (APIaaS)“.

Prinzipiell kann man „Cloud Computing“ als eine Kombination von ausgelagerten Diensten sehen. Virtualisierung ist dabei eine fundamentale Komponente, auf der die Auslagerung ansetzt. In den meisten Organisationen oder Firmen gibt es mittlerweile eine Mischung von „Cloud Computing“ und „Classic Computing“ abhängig von den Diensten (z.B. E-Mail Konten, Spamfilter oder Webspace wird oft ausgelagert). Trotz der Auslagerung ist jeder „Cloud Computing“ Dienst Teil der eigenen Infrastruktur und muß in die Sicherheitsbetrachtung eingebaut werden, völlig unabhängig ob man eine Kontrolle auf den „Cloud“ Anbieter ausüben kann oder nicht.

9.2. Virtualisierung

Virtualisierungstechnologien gibt es seit 1964. Damals wurde Virtualisierung auf den IBM Mainframes eingesetzt. Nach 2000 wurde Virtualisierung auf PCs durch die Verfügbarkeit von VMware® und Xen™ „wiederentdeckt“. Gängige Virtualisierungstechnologien sind im Moment die folgenden.

- Kernel-based Virtual Machine (KVM)[1] mit QEMU[2]
- Microsoft Hyper-V[3]
- OpenVZ[4]
- Virtuozzo[5]
- VMware®[6]
- Xen™[7]

Die Virtualisierungen unterscheiden sich in der Art und Weise wie sie aufgesetzt werden. Manche bringen den Hypervisor[8] direkt auf der Hardware zum Einsatz, andere verwenden ein Host-OS mit Hypervisor darin, um die virtualisierten Gäste zu steuern. Prinzipiell unterscheidet man zwischen dem *native mode* (Abbildung 9.1) und dem *hosted mode* (Abbildung 9.2).

[1] http://www.linux-kvm.org/
[2] http://www.qemu.org/
[3] https://en.wikipedia.org/wiki/Hyper-V
[4] http://openvz.org/
[5] http://www.parallels.com/
[6] http://www.vmwareinc.com/
[7] http://www.xen.org/
[8] Der Hypervisor ist die Komponente, die die Gastsysteme kontrolliert und gegeneinander abschirmt.

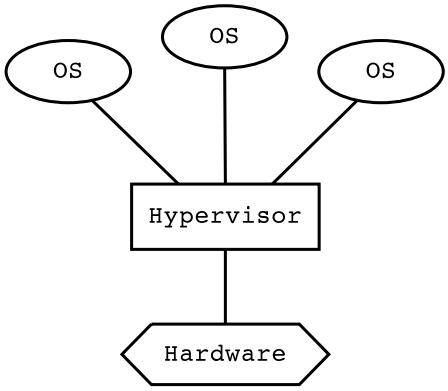


Abbildung 9.1.: Bei Virtualisierung im *native mode* läuft der Hypervisor direkt auf der Hardware des Virtualisierungshosts.

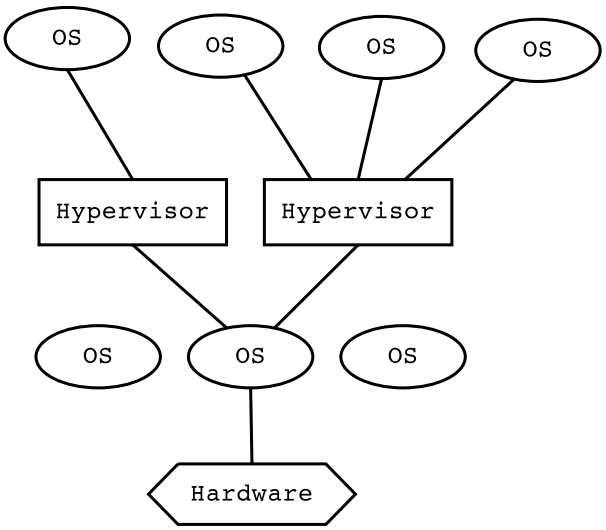


Abbildung 9.2.: Bei Virtualisierung im *hosted mode* läuft ein Betriebssystem direkt auf der Hardware des Virtualisierungshosts. Darin wird der Hypervisor gestartet, welcher als Kern- oder Userland-Prozeß läuft. Der Hypervisor verwaltet dann die Gastsysteme. Bei dieser Konfiguration können auch mehrere (verschiedene) Hypervisors aktiv sein.

9.3. Sicherheit

Für die Absicherung von „Cloud Computing“ Diensten gelten dieselben Strategien wie in den vorherigen Kapiteln. Man muß genauso Trennen, Privilegien einführen, Zugänge absichern; wobei man es sehr oft mit Weboberflächen, -services und -applikationen zu tun haben wird.

9.3.1. Managementzugang

Der kritischste Punkt sind die Managementzugänge. Bei ausgelagerten Diensten lassen sich diese oft nicht einschränken. Meist besteht der Zugang aus einem Login an einer Weboberfläche. Der Login ist in der Regel an ein E-Mail-Konto oder seltener an eine Mobilfunknummer gekoppelt, um eine Möglichkeit zu haben Paßworte zurückzusetzen und Bestätigungen für Aktionen durchzuführen. Das Absichern dieser alternativen Konten bzw. des Mobilfunktelefons ist dann ein wesentlicher Bestandteil der Sicherheitsmaßnahmen.

9.3.2. Virtualisierung

Trotz Hypervisor kann es passieren, dass durch Fehler in der Software oder andere Umstände die Gastsysteme nicht vollständig gegeneinander isoliert sind. Es gibt Angriffe gegen Hypervisors, die die Barrieren durchbrechen können und so ein kompromittierter Gast andere ebenso angreifen kann. Änliche Attacken gibt es vom Gast auf das Hostsystem selbst. Es gelten daher bestimmte Grundlagen für die Sicherheitsbetrachtung.

- Der Virtualisierungshost muß gehärtet werden. Zugang zum Virtualisierungshost eröffnet meist den Zugang zu allen Gastsystemen.
- Virtualisierungshosts dürfen keine Gäste verschiedener Sicherheitsstufen beinhalten. Beispielsweise dürfen Gastsysteme nur im Perimternetzwerk exklusive oder nur im lokalen Netzwerk präsent sein. Das bedeutet, dass man pro Netzwerk mit unterschiedlicher Sicherheitsstufe je einen Virtualisierungshost verwenden muß.
- Attacken gegen die Schicht 2 sind relevant, weil bei Virtualisierung oft Software-Switches eingesetzt werden. Eine Netzwerkhärtung muß daher Virtualisierung besonders betrachten.

9.3.3. Minimalisierung der Systeme

Bisher mußte man ein Betriebsystem als Basis für einen Webserver nehmen. Bestimmte Technologien reduzieren das Betriebsystem in Cloud-Umgebungen auf ein Minimum. Es gibt Ansätze beispielsweise den Webserver direkt auf einem Kernel mit den minimal notwendigen Softwarebibliotheken laufen zu lassen. Solche Methoden verringern die Angriffsoberfläche.

9.3.4. Speichern von Daten

Alle „Cloud“ Services speichern Daten, genauso wie „echte“ Services. Es ist daher unbedingt zu beachten wer Zugriff auf diese Daten hat. Es spielt dabei keine Rolle was in den Verträgen oder Service Level Agreements (SLAs) festgeschrieben ist, denn es geht um technische Sicherheitsmaßnahmen. Es ist denkbar, dass vertrauliche Daten entweder nur verschlüsselt[9] oder gar nicht auf fremder Infrastruktur gespeichert werden. Bei nicht sensitiven Daten ist das Auslagern leichter.

[9] Es ist darauf zu achten, dass die Schlüssel **nicht** beim „Cloud“ Service Anbieter liegen.

A. Nützliche Kommandos und Werkzeuge

Beim Aufbau von Netzwerken und Paketfiltern im Labor kann man sich der folgenden Kommandos bei der Überprüfung der Konfiguration bedienen.

A.1. Layer 2

Wenn man sich ganz sicher ist, daß alle Kabel richtig verkabelt wurden, dann kann man sich den Layer 2 gründlich anschauen und überprüfen.

A.1.1. mii-diag

`mii-diag` zeigt den Status des Media Independent Interface (MII) Bus an.[1] Eine korrekt angeschlossene Netzwerkkarte zeigt beispielsweise folgende Statusmeldungen via MII an.

```
agamemnon:~# mii-diag eth0
Basic registers of MII PHY #32:  1140 796d 001c c910 01e1 c5e1 000d 2001.
 The autonegotiated capability is 01e0.
The autonegotiated media type is 100baseTx-FD.
 Basic mode control register 0x1140: Auto-negotiation enabled.
 You have link beat, and everything is working OK.
 Your link partner advertised c5e1: Flow-control 100baseTx-FD 100baseTx
   10baseT-FD 10baseT, w/ 802.3X flow control.
   End of basic transceiver information.

agamemnon:~#
```

Damit läßt sich überprüfen, ob an der Netzwerkkarte ein Linkpartner angeschlossen ist und welches Medienformat verwendet wird. Im obigen Fall wurde `100baseTx-FD`, also 100 Mbit/s Full-Duplex, ausgehandelt. `mii-diag` funktioniert nur mit Treibern, die die MII Funktionalität unterstützen.

A.1.2. mii-tool

`mii-tool` zeigt auch den Status des Media Independent Interface (MII) Bus an. Die Ausgabe eignet sich besser zum Skripten.

A.1.3. ARP Cache

Die Funktionalität des Address Resolution Protocol (ARP) läßt sich mit Hilfe des ARP Caches feststellen.

```
agamemnon:~# ip neighbor
fe80::2c0:f0ff:fe4c:8182 dev eth0 lladdr 00:c0:f0:4c:81:82 router nud stale
fe80::2c0:f0ff:fe4c:8182 dev eth2 lladdr 00:c0:f0:4c:81:82 router nud stale
192.168.15.1 dev eth0 lladdr 00:c0:f0:4c:81:82 nud stale
192.168.15.220 dev eth0 lladdr 00:c0:f0:4d:89:2e nud stale
192.168.15.2 dev eth0 lladdr 00:40:f4:ee:32:6d nud stale
agamemnon:~#
```

Das System ist an ein Netzwerk angeschlossen und sieht einige seiner Nachbarn. Eventuell muß man vor Einblick in den ARP Cache IP Pakete verschicken, damit eine ARP Auflösung provoziert wird. Das Kommando `arp` leistet ähnliche Dienste wie `ip`.

```
agamemnon:~# arp -n -a
? (192.168.15.1) at 00:C0:F0:4C:81:82 [ether] on eth0
? (192.168.15.220) at 00:C0:F0:4D:89:2E [ether] on eth0
? (192.168.15.2) at 00:40:F4:EE:32:6D [ether] on eth0
agamemnon:~#
```

[1] IEEE 802.3u

Man sollte sich gleich angewöhnen die Kommandos mit dem Schalter `-n` aufzurufen, um die DNS Auflösung zu verhindern. Oft hat man im Labor keinen DNS Cache zur Verfügung, und „erleidet" dann große Verzögerungen aufgrund unbeantworteter DNS-Anfragen.

A.2. Layer 3/4

Wenn im Layer 2 keine Fehler gefunden wurden und dort alles funktioniert, so kann man sich den nächsten beiden Schichten widmen.

A.2.1. netstat

`netstat` zeigt aktive Verbindungen und besetzte Ports an. `netstat -pant` zeigt aktive TCP Sockets.

```
laptop:~# netstat -pant
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address       Foreign Address    State       PID/Program name
tcp        0      0 0.0.0.0:25          0.0.0.0:*          LISTEN      4190/master
tcp        0      0 127.0.0.1:2947      0.0.0.0:*          LISTEN      3924/gpsd
tcp        0     12 192.168.15.30:47416 192.168.15.2:9421  ESTABLISHED 11687/mfsmount
tcp        0      0 10.83.60.34:40241   173.194.70.109:993 ESTABLISHED 20170/evolution
tcp        0      0 10.83.60.34:46301   173.194.70.108:993 ESTABLISHED 20170/evolution
tcp        0      0 10.83.60.34:40759   173.194.70.109:993 ESTABLISHED 20170/evolution
tcp        0      0 10.83.60.34:46300   173.194.70.108:993 ESTABLISHED 20170/evolution
tcp        0      0 10.83.60.34:40757   173.194.70.109:993 ESTABLISHED 20170/evolution
tcp        0      0 10.83.60.34:46305   173.194.70.108:993 ESTABLISHED 20170/evolution
tcp        0      0 10.83.60.34:46297   173.194.70.108:993 ESTABLISHED 20170/evolution
tcp        0      0 10.83.60.34:46304   173.194.70.108:993 ESTABLISHED 20170/evolution
tcp        0      0 10.83.60.34:40758   173.194.70.109:993 ESTABLISHED 20170/evolution
tcp        0      0 10.83.60.34:46299   173.194.70.108:993 ESTABLISHED 20170/evolution
tcp        0      0 10.83.60.34:46298   173.194.70.108:993 ESTABLISHED 20170/evolution
tcp        0      0 10.83.60.34:40760   173.194.70.109:993 ESTABLISHED 20170/evolution
tcp        0      0 10.83.60.34:40240   173.194.70.109:993 ESTABLISHED 20170/evolution
tcp        0      0 192.168.15.30:36072 192.168.15.242:143 ESTABLISHED 12060/mutt
tcp        0      0 192.168.15.30:38532 192.168.15.2:445   ESTABLISHED -
tcp6       0      0 :::25               :::*               LISTEN      4190/master
tcp6       0      0 ::1:2947            :::*               LISTEN      3924/gpsd
laptop:~#
```

Analog zeigt `netstat -panu` aktive UDP Ports an.

A.2.2. tcptraceroute

Das normale `traceroute` funktioniert durch das Versenden von UDP Paketen. Diese werden meistens von Paketfiltern geblockt. `traceroute` gibt es daher auch als TCP-Variante. Klarerweise muß man dort auch einen Port angeben.

```
agamemnon:~# tcptraceroute -n 84.113.64.47 25
Selected device eth0, address 192.168.15.242, port 52409 for outgoing packets
Tracing the path to 84.113.64.47 on TCP port 25 (smtp), 30 hops max
 1  192.168.15.1  0.548 ms  0.318 ms  0.309 ms
 2  213.129.239.201  1.378 ms  1.246 ms  1.602 ms
 3  213.129.243.1  13.051 ms  4.480 ms  4.452 ms
 4  213.129.224.130  4.929 ms  4.660 ms  4.739 ms
 5  213.129.224.211  5.222 ms  5.611 ms  7.162 ms
 6  193.203.0.23  5.707 ms  5.348 ms  5.401 ms
 7  213.46.173.5  5.477 ms  5.400 ms  5.598 ms
 8  213.46.173.130  5.714 ms  7.614 ms  10.911 ms
 9  213.46.173.110  5.611 ms  12.674 ms  6.184 ms
10  212.17.99.18  5.610 ms  6.929 ms  6.052 ms
11  213.47.218.180  6.208 ms  6.122 ms  6.235 ms
12  213.47.218.210  10.312 ms  6.210 ms  6.161 ms
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
19  * * *
20  * * *
```

```
21  * * *
22  * * *
23  * * *
24  * * *
25  * * *
26  * * *
27  * * *
28  * * *
29  * * *
30  * * *
Destination not reached
agamemnon:~#
```

A.2.3. Telnet

Mittels `telnet` kann man zu jedem TCP Port eine Verbindung aufbauen. Damit kann man prinzipiell prüfen, ob ein bestimmter Port erreichbar ist.

A.2.4. mtr

`mtr` kombiniert die Eigenschaften von `ping` und `traceroute`. Es kann stetig ICMP Echo Request Pakete abschicken und Statistiken zu jedem Hop zwischen Quelle und Ziel anzeigen. Obwohl es ICMP verwendet, ist es sehr brauchbar um Netzwerkprobleme aufzuspüren (speziell Paketverlust).

A.3. Layer 7

Wenn das Netzwerk steht und alles richtig verbunden ist, dann kann man Applikationen selbst testen. Üblicherweise verwendet man dazu geeignete Clients, die das zu prüfende Layer 7 Protokoll sprechen.

- DNS
  `dig`, `host`, `nslookup`, `dnswalk`, `dnsdoctor`
- FTP
  `ftp`, `ncftp`, `lftp`
- HTTP/HTTPS
  jeder Webbrowser (Firefox, Microsoft® Internet Explorer, Apple Safari, `lynx`, `elinks`, `w3m`), die Perl libwww Tools (`GET`, `HEAD`), der Squid Proxy, der Paros Proxy, der Spike Proxy, der ratproxy, die Burp Suite
- SMTP/ESMTP
  `telnet`, jeder (E)SMTP-fähige Mailclient, Frameworks für das Injizieren von Viren und Spam
- SSH
  OpenSSH, `putty`, `sshcan`
- SSL/TLS
  OpenSSL, `ssldump`, `stunnel`, `sslscan`
- VPN Protokolle
  `ikescan` (für IPsec)

Für weitere Inspirationen kann man die Paketliste von Backtrack[2] konsultieren.

[2] http://wiki.remote-exploit.org/index.php/Tools

B. Konfigurationen

B.1. Makefile für Root CA

Das folgende Makefile stammt von der Webseite `sial.org` und ist im Artikel OpenSSL Certificate Authority Setup beschrieben.

```
# $Id: Makefile,v 1.3 2004/01/31 19:44:34 jmates Exp $
#
# Automates the setup of a custom Certificate Authority and provides
# routines for signing and revocation of certificates. To use, first
# customize the commands in this file and the settings in openssl.cnf,
# then run:
#
# make init
#
# Then, copy in certificate signing requests, and ensure their suffix is
# .csr before signing them with the following command:
#
# make sign
#
# To revoke a key, name the certificate file with the cert option
# as shown below:
#
# make revoke cert=foo.cert
#
# This will revoke the certificate and call gencrl; the revocation list
# will then need to be copied somehow to the various systems that use
# your CA cert.

requests = *.csr

sign: ${requests}

# remove -batch option if want chance to not certify a particular request
${requests}: FORCE
@openssl ca -batch -config openssl.cnf -in $@ -out ${@:.csr=.cert}
@[ -f ${@:.csr=.cert} ] && rm $@

revoke:
@test $${cert:?"usage: make revoke cert=certificate"}
@openssl ca -config openssl.cnf -revoke $(cert)
@$(MAKE) gencrl

gencrl:
@openssl ca -config openssl.cnf -gencrl -out ca-crl.pem

clean:
-rm ${requests}

# creates required supporting files, CA key and certificate
init:
@test ! -f serial
@mkdir crl newcerts private
@chmod go-rwx private
@echo '01' > serial
@touch index
@openssl req -nodes -config openssl.cnf -days 4015 -x509 -newkey rsa -out ca-cert.pem -outform PEM

help:
@echo make sign
@echo '  - signs all *.csr files in this directory'
@echo
@echo make revoke cert=filename
```

```
@echo '  - revokes certificate in named file and calls gencrl'
@echo
@echo make gencrl
@echo '  - updates Certificate Revocation List (CRL)'
@echo
@echo make clean
@echo '  - removes all *.csr files in this directory'
@echo
@echo make init
@echo '  - required initial setup command for new CA'

# for legacy make support
FORCE:
```

C. Beispiele aus dem Netz

Die hier wiedergegebenen Beispiele sind recht alt und wurden nicht mit aktuellen Werkzeugen durchgeführt. Es ist empfohlen die Beispiele mit aktuellen Versionen und eigenen Setups in verschiedenen Netzwerken zu wiederholen, um ein Gefühl für Netzwerktransmissionen zu entwickeln.

C.1. Session einer TCP Verbindung

Der folgende Dump einer TCP Session wurde mit dem Netzwerkanalysator Ethereal durchgeführt. Mittlerweile verwendet man dafür Wireshark[1]. Man sieht die Session für das Kommando `telnet 192.168.10.11 25`, welches von der Maschine `192.168.10.3` initiiert wurde. `luchs` ist ECN-fähig, `trinity` nicht.

```
Frame 3 (74 on wire, 74 captured)
    Arrival Time: Aug 19, 2001 23:49:45.9288
    Time delta from previous packet: 0.000000 seconds
    Time relative to first packet: 0.000162 seconds
    Frame Number: 3
    Packet Length: 74 bytes
    Capture Length: 74 bytes
Ethernet II
    Destination: 00:40:c7:99:a6:df (trinity.luchs.at)
    Source: 00:60:97:11:d9:10 (luchs.luchs.at)
    Type: IP (0x0800)
Internet Protocol
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 60
    Identification: 0x105d
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (0x06)
    Header checksum: 0x9500 (correct)
    Source: luchs.luchs.at (192.168.10.3)
    Destination: trinity.luchs.at (192.168.10.11)
Transmission Control Protocol, Src Port: 38774 (38774), Dst Port: smtp (25), Seq: 633500261, Ack: 0
    Source port: 38774 (38774)
    Destination port: smtp (25)
    Sequence number: 633500261
    Header length: 40 bytes
    Flags: 0x00c2 (SYN, ECN, CWR)
        1... .... = Congestion Window Reduced (CWR): Set
        .1.. .... = ECN-Echo: Set
        ..0. .... = Urgent: Not set
        ...0 .... = Acknowledgment: Not set
        .... 0... = Push: Not set
        .... .0.. = Reset: Not set
        .... ..1. = Syn: Set
        .... ...0 = Fin: Not set
    Window size: 5840
    Checksum: 0x0695 (correct)
    Options: (20 bytes)
        Maximum segment size: 1460 bytes
        SACK permitted
        Time stamp: tsval 43278904, tsecr 0
        NOP
        Window scale: 0 bytes

Frame 4 (66 on wire, 66 captured)
    Arrival Time: Aug 19, 2001 23:49:45.9290
    Time delta from previous packet: 0.000236 seconds
    Time relative to first packet: 0.000398 seconds
    Frame Number: 4
```

[1] https://www.wireshark.org/

```
    Packet Length: 66 bytes
    Capture Length: 66 bytes
Ethernet II
    Destination: 00:60:97:11:d9:10 (luchs.luchs.at)
    Source: 00:40:c7:99:a6:df (trinity.luchs.at)
    Type: IP (0x0800)
Internet Protocol
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 52
    Identification: 0x0000
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (0x06)
    Header checksum: 0xa565 (correct)
    Source: trinity.luchs.at (192.168.10.11)
    Destination: luchs.luchs.at (192.168.10.3)
Transmission Control Protocol, Src Port: smtp (25), Dst Port: 38774 (38774), Seq: 765511189, Ack: 633500262
    Source port: smtp (25)
    Destination port: 38774 (38774)
    Sequence number: 765511189
    Acknowledgement number: 633500262
    Header length: 32 bytes
    Flags: 0x0012 (SYN, ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 0... = Push: Not set
        .... .0.. = Reset: Not set
        .... ..1. = Syn: Set
        .... ...0 = Fin: Not set
    Window size: 5840
    Checksum: 0x9f6b (correct)
    Options: (12 bytes)
        Maximum segment size: 1460 bytes
        NOP
        NOP
        SACK permitted
        NOP
        Window scale: 0 bytes

Frame 5 (54 on wire, 54 captured)
    Arrival Time: Aug 19, 2001 23:49:45.9291
    Time delta from previous packet: 0.000048 seconds
    Time relative to first packet: 0.000446 seconds
    Frame Number: 5
    Packet Length: 54 bytes
    Capture Length: 54 bytes
Ethernet II
    Destination: 00:40:c7:99:a6:df (trinity.luchs.at)
    Source: 00:60:97:11:d9:10 (luchs.luchs.at)
    Type: IP (0x0800)
Internet Protocol
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 40
    Identification: 0x105e
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (0x06)
    Header checksum: 0x9513 (correct)
    Source: luchs.luchs.at (192.168.10.3)
    Destination: trinity.luchs.at (192.168.10.11)
Transmission Control Protocol, Src Port: 38774 (38774), Dst Port: smtp (25), Seq: 633500262, Ack: 765511190
    Source port: 38774 (38774)
    Destination port: smtp (25)
    Sequence number: 633500262
    Acknowledgement number: 765511190
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
```

```
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 0... = Push: Not set
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...0 = Fin: Not set
    Window size: 5840
    Checksum: 0xe036 (correct)

Frame 8 (76 on wire, 76 captured)
    Arrival Time: Aug 19, 2001 23:50:04.2866
    Time delta from previous packet: 18.357537 seconds
    Time relative to first packet: 18.357983 seconds
    Frame Number: 8
    Packet Length: 76 bytes
    Capture Length: 76 bytes
Ethernet II
    Destination: 00:60:97:11:d9:10 (luchs.luchs.at)
    Source: 00:40:c7:99:a6:df (trinity.luchs.at)
    Type: IP (0x0800)
Internet Protocol
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 62
    Identification: 0x64c6
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (0x06)
    Header checksum: 0x4095 (correct)
    Source: trinity.luchs.at (192.168.10.11)
    Destination: luchs.luchs.at (192.168.10.3)
Transmission Control Protocol, Src Port: smtp (25), Dst Port: 38774 (38774), Seq: 765511190, Ack: 633500262
    Source port: smtp (25)
    Destination port: 38774 (38774)
    Sequence number: 765511190
    Next sequence number: 765511212
    Acknowledgement number: 633500262
    Header length: 20 bytes
    Flags: 0x0018 (PSH, ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 1... = Push: Set
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...0 = Fin: Not set
    Window size: 5840
    Checksum: 0xa02d (correct)
Simple Mail Transfer Protocol
    Response: 220
    Parameter: Leave ESMTP here

Frame 9 (54 on wire, 54 captured)
    Arrival Time: Aug 19, 2001 23:50:04.2867
    Time delta from previous packet: 0.000062 seconds
    Time relative to first packet: 18.358045 seconds
    Frame Number: 9
    Packet Length: 54 bytes
    Capture Length: 54 bytes
Ethernet II
    Destination: 00:40:c7:99:a6:df (trinity.luchs.at)
    Source: 00:60:97:11:d9:10 (luchs.luchs.at)
    Type: IP (0x0800)
Internet Protocol
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 40
    Identification: 0x105f
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
```

```
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (0x06)
    Header checksum: 0x9512 (correct)
    Source: luchs.luchs.at (192.168.10.3)
    Destination: trinity.luchs.at (192.168.10.11)
Transmission Control Protocol, Src Port: 38774 (38774), Dst Port: smtp (25), Seq: 633500262, Ack: 765511212
    Source port: 38774 (38774)
    Destination port: smtp (25)
    Sequence number: 633500262
    Acknowledgement number: 765511212
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 0... = Push: Not set
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...0 = Fin: Not set
    Window size: 5840
    Checksum: 0xe020 (correct)

Frame 10 (60 on wire, 60 captured)
    Arrival Time: Aug 19, 2001 23:50:16.7268
    Time delta from previous packet: 12.440181 seconds
    Time relative to first packet: 30.798226 seconds
    Frame Number: 10
    Packet Length: 60 bytes
    Capture Length: 60 bytes
Ethernet II
    Destination: 00:40:c7:99:a6:df (trinity.luchs.at)
    Source: 00:60:97:11:d9:10 (luchs.luchs.at)
    Type: IP (0x0800)
Internet Protocol
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 46
    Identification: 0x1060
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (0x06)
    Header checksum: 0x950b (correct)
    Source: luchs.luchs.at (192.168.10.3)
    Destination: trinity.luchs.at (192.168.10.11)
Transmission Control Protocol, Src Port: 38774 (38774), Dst Port: smtp (25), Seq: 633500262, Ack: 765511212
    Source port: 38774 (38774)
    Destination port: smtp (25)
    Sequence number: 633500262
    Next sequence number: 633500268
    Acknowledgement number: 765511212
    Header length: 20 bytes
    Flags: 0x0018 (PSH, ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 1... = Push: Set
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...0 = Fin: Not set
    Window size: 5840
    Checksum: 0x385f (correct)
Simple Mail Transfer Protocol
    Command: QUIT

Frame 11 (60 on wire, 60 captured)
    Arrival Time: Aug 19, 2001 23:50:16.7271
    Time delta from previous packet: 0.000242 seconds
    Time relative to first packet: 30.798468 seconds
    Frame Number: 11
    Packet Length: 60 bytes
    Capture Length: 60 bytes
Ethernet II
    Destination: 00:60:97:11:d9:10 (luchs.luchs.at)
    Source: 00:40:c7:99:a6:df (trinity.luchs.at)
    Type: IP (0x0800)
```

```
    Trailer: 000000000000
Internet Protocol
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 40
    Identification: 0x64c7
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (0x06)
    Header checksum: 0x40aa (correct)
    Source: trinity.luchs.at (192.168.10.11)
    Destination: luchs.luchs.at (192.168.10.3)
Transmission Control Protocol, Src Port: smtp (25), Dst Port: 38774 (38774), Seq: 765511212, Ack: 633500268
    Source port: smtp (25)
    Destination port: 38774 (38774)
    Sequence number: 765511212
    Acknowledgement number: 633500268
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 0... = Push: Not set
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...0 = Fin: Not set
    Window size: 5840
    Checksum: 0xe01a (correct)

Frame 12 (102 on wire, 102 captured)
    Arrival Time: Aug 19, 2001 23:50:16.7273
    Time delta from previous packet: 0.000255 seconds
    Time relative to first packet: 30.798723 seconds
    Frame Number: 12
    Packet Length: 102 bytes
    Capture Length: 102 bytes
Ethernet II
    Destination: 00:60:97:11:d9:10 (luchs.luchs.at)
    Source: 00:40:c7:99:a6:df (trinity.luchs.at)
    Type: IP (0x0800)
Internet Protocol
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 88
    Identification: 0x64c8
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (0x06)
    Header checksum: 0x4079 (correct)
    Source: trinity.luchs.at (192.168.10.11)
    Destination: luchs.luchs.at (192.168.10.3)
Transmission Control Protocol, Src Port: smtp (25), Dst Port: 38774 (38774), Seq: 765511212, Ack: 633500268
    Source port: smtp (25)
    Destination port: 38774 (38774)
    Sequence number: 765511212
    Next sequence number: 765511260
    Acknowledgement number: 633500268
    Header length: 20 bytes
    Flags: 0x0018 (PSH, ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 1... = Push: Set
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...0 = Fin: Not set
    Window size: 5840
    Checksum: 0x7f2d (correct)
Simple Mail Transfer Protocol
```

```
    Response: 221
    Parameter: 2.0.0 trinity.luchs.at closing connection

Frame 13 (54 on wire, 54 captured)
    Arrival Time: Aug 19, 2001 23:50:16.7274
    Time delta from previous packet: 0.000028 seconds
    Time relative to first packet: 30.798751 seconds
    Frame Number: 13
    Packet Length: 54 bytes
    Capture Length: 54 bytes
Ethernet II
    Destination: 00:40:c7:99:a6:df (trinity.luchs.at)
    Source: 00:60:97:11:d9:10 (luchs.luchs.at)
    Type: IP (0x0800)
Internet Protocol
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 40
    Identification: 0x1061
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (0x06)
    Header checksum: 0x9510 (correct)
    Source: luchs.luchs.at (192.168.10.3)
    Destination: trinity.luchs.at (192.168.10.11)
Transmission Control Protocol, Src Port: 38774 (38774), Dst Port: smtp (25), Seq: 633500268, Ack: 765511260
    Source port: 38774 (38774)
    Destination port: smtp (25)
    Sequence number: 633500268
    Acknowledgement number: 765511260
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 0... = Push: Not set
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...0 = Fin: Not set
    Window size: 5840
    Checksum: 0xdfea (correct)

Frame 14 (60 on wire, 60 captured)
    Arrival Time: Aug 19, 2001 23:50:16.7276
    Time delta from previous packet: 0.000225 seconds
    Time relative to first packet: 30.798976 seconds
    Frame Number: 14
    Packet Length: 60 bytes
    Capture Length: 60 bytes
Ethernet II
    Destination: 00:60:97:11:d9:10 (luchs.luchs.at)
    Source: 00:40:c7:99:a6:df (trinity.luchs.at)
    Type: IP (0x0800)
    Trailer: 000000000000
Internet Protocol
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 40
    Identification: 0x64c9
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (0x06)
    Header checksum: 0x40a8 (correct)
    Source: trinity.luchs.at (192.168.10.11)
    Destination: luchs.luchs.at (192.168.10.3)
Transmission Control Protocol, Src Port: smtp (25), Dst Port: 38774 (38774), Seq: 765511260, Ack: 633500268
    Source port: smtp (25)
    Destination port: 38774 (38774)
    Sequence number: 765511260
    Acknowledgement number: 633500268
```

```
    Header length: 20 bytes
    Flags: 0x0011 (FIN, ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 0... = Push: Not set
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...1 = Fin: Set
    Window size: 5840
    Checksum: 0xdfe9 (correct)

Frame 15 (54 on wire, 54 captured)
    Arrival Time: Aug 19, 2001 23:50:16.7282
    Time delta from previous packet: 0.000597 seconds
    Time relative to first packet: 30.799573 seconds
    Frame Number: 15
    Packet Length: 54 bytes
    Capture Length: 54 bytes
Ethernet II
    Destination: 00:40:c7:99:a6:df (trinity.luchs.at)
    Source: 00:60:97:11:d9:10 (luchs.luchs.at)
    Type: IP (0x0800)
Internet Protocol
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 40
    Identification: 0x1062
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (0x06)
    Header checksum: 0x950f (correct)
    Source: luchs.luchs.at (192.168.10.3)
    Destination: trinity.luchs.at (192.168.10.11)
Transmission Control Protocol, Src Port: 38774 (38774), Dst Port: smtp (25), Seq: 633500268, Ack: 765511261
    Source port: 38774 (38774)
    Destination port: smtp (25)
    Sequence number: 633500268
    Acknowledgement number: 765511261
    Header length: 20 bytes
    Flags: 0x0011 (FIN, ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 0... = Push: Not set
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...1 = Fin: Set
    Window size: 5840
    Checksum: 0xdfe8 (correct)

Frame 16 (60 on wire, 60 captured)
    Arrival Time: Aug 19, 2001 23:50:16.7284
    Time delta from previous packet: 0.000181 seconds
    Time relative to first packet: 30.799754 seconds
    Frame Number: 16
    Packet Length: 60 bytes
    Capture Length: 60 bytes
Ethernet II
    Destination: 00:60:97:11:d9:10 (luchs.luchs.at)
    Source: 00:40:c7:99:a6:df (trinity.luchs.at)
    Type: IP (0x0800)
    Trailer: 000000000000
Internet Protocol
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 40
    Identification: 0x64ca
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
```

```
    Time to live: 64
    Protocol: TCP (0x06)
    Header checksum: 0x40a7 (correct)
    Source: trinity.luchs.at (192.168.10.11)
    Destination: luchs.luchs.at (192.168.10.3)
Transmission Control Protocol, Src Port: smtp (25), Dst Port: 38774 (38774), Seq: 765511261, Ack: 633500269
    Source port: smtp (25)
    Destination port: 38774 (38774)
    Sequence number: 765511261
    Acknowledgement number: 633500269
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 0... = Push: Not set
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...0 = Fin: Not set
    Window size: 5840
    Checksum: 0xdfe8 (correct)
```

C.2. Beispiel für FTP Connection Tracking

Dieses Skript illustriert das FTP Connection Tracking mit dem Netfilter Code. Die Maschine `Anubis` besitzt einen FTP Server, der für aktives und passives FTP zugänglich gemacht werden soll. Weiterhin darf `Anubis` beliebige Verbindungen zu anderen Hosts aufbauen. Alles andere dringt nicht durch.
Anmerkung: Es ist wichtig einen Linux® Kern mit Version 2.4.4 oder neuer einzusetzen, da der FTP Connection Tracking Code bis Version 2.4.3 einen Bug hat. [43]

```
#!/bin/sh

# Script to illustrate FTP connection tracking
# Sun 23-Sep-2001 20:00:17 CEST <pfeiffer@luchs.at>

# Definitions

IPTABLES=/usr/local/sbin/iptables

LO_NET="127.0.0.0/8"            # Loopback network
LAN1_NET="192.168.00.0/24"      # LAN 10 Mbit/s
LAN2_NET="192.168.10.0/24"      # LAN 100 Mbit/s

RFC1918_A="10.0.0.0/8"          # RFC1918 private networks
RFC1918_B="172.16.0.0/12"
RFC1918_C="192.168.0.0/16"
MULTICAST="224.0.0.0/8"         # Multicast range

EVERYWHERE="0/0"                # The World(TM)

ETH_DEV="eth0"
LO_DEV="lo"

ANUBIS=`/sbin/ifconfig $ETH_DEV | grep inet | awk '{ print $2; }' | sed -e 's/addr://'`

# Set default policy

$IPTABLES --policy INPUT   DROP
$IPTABLES --policy OUTPUT  DROP
$IPTABLES --policy FORWARD DROP

# Allow local nets

$IPTABLES --insert INPUT  --in-interface $LO_DEV  --source $LO_NET --destination $LO_NET --jump ACCEPT
$IPTABLES --insert OUTPUT --out-interface $LO_DEV --source $LO_NET --destination $LO_NET --jump ACCEPT

# Allow FTP command channel connections

$IPTABLES --insert INPUT  --protocol tcp --source $EVERYWHERE --destination $ANUBIS --destination-port 21 \
          --match state --state NEW,ESTABLISHED \
```

```
          --jump ACCEPT
$IPTABLES --insert OUTPUT --protocol tcp --source $ANUBIS --source-port 21 --destination $EVERYWHERE \
          --match state --state ESTABLISHED \
          --jump ACCEPT

# Allow active FTP

$IPTABLES --insert INPUT  --protocol tcp --source $EVERYWHERE --destination $ANUBIS --destination-port 20 \
          --match state --state ESTABLISHED \
          --jump ACCEPT
$IPTABLES --insert OUTPUT --protocol tcp --source $ANUBIS --source-port 20 --destination $EVERYWHERE \
          --match state --state ESTABLISHED,RELATED \
          --jump ACCEPT

# Allow passive FTP

$IPTABLES --insert INPUT  --protocol tcp \
          --source $EVERYWHERE --source-port 1024: --destination $ANUBIS --destination-port 1024: \
          --match state --state ESTABLISHED,RELATED \
          --jump ACCEPT
$IPTABLES --insert OUTPUT --protocol tcp \
          --source $ANUBIS --source-port 1024: --destination $EVERYWHERE --destination-port 1024: \
          --match state --state ESTABLISHED \
          --jump ACCEPT

# Allow all outgoing connections
$IPTABLES --insert OUTPUT --source $ANUBIS --destination $EVERYWHERE \
          --match state --state NEW,ESTABLISHED \
          --jump ACCEPT
$IPTABLES --insert INPUT  --source $EVERYWHERE --destination $ANUBIS \
          --match state --state ESTABLISHED \
          --jump ACCEPT
```

Eine Zustandstabelle mit aktiven Verbindungen sieht dann beispielsweise so aus.

```
[root@anubis scripts]# cat /proc/net/ip_conntrack
tcp      6 103 TIME_WAIT src=192.168.0.141 dst=192.168.10.3 sport=20 dport=34300
                         src=192.168.10.3 dst=192.168.0.141 sport=34300 dport=20 [ASSURED] use=1
tcp      6 39 TIME_WAIT src=192.168.10.3 dst=192.168.0.141 sport=34296 dport=21
                        src=192.168.0.141 dst=192.168.10.3 sport=21 dport=34296 [ASSURED] use=2
tcp      6 431983 ESTABLISHED src=192.168.10.3 dst=192.168.0.141 sport=34299 dport=21
                              src=192.168.0.141 dst=192.168.10.3 sport=21 dport=34299 [ASSURED] use=2
tcp      6 2 TIME_WAIT src=192.168.10.3 dst=192.168.0.141 sport=34298 dport=4542
                       src=192.168.0.141 dst=192.168.10.3 sport=4542 dport=34298 [ASSURED] use=1
```

Die Maschine `Anubis` hat die IP Adresse `192.168.0.141`, von `192.168.10.3` hat ein Client eine aktive Datenübertragung durchgeführt. Der Zustand `TIME_WAIT` kennzeichnet, daß die Übertragung bereits abgeschlossen ist. Das Schlüsselwort `ASSURED` zeigt an, daß die Verbindung vom Netfilter kontrolliert und erlaubt wurde.

C.3. Routing Skript

Das ist ein Beispiel für ein Routing Skript. Es stammt von einer Firewall, die als äußerer Paketfilter vor der DMZ fungiert. Hinter der DMZ ist eine weitere interne Firewall, die die internen Netzwerke zusätzlich abschirmt. In diesem Skript werden alle Routing Parameter gesetzt. Es ist kein übliches Start-/Stop-Skript, da davon ausgegangen wird, daß der Router den Run Level nicht ändert. Es kann aber durchaus auf ein System V Start-/Stop-Skript erweitert werden. Die verwendeten Shell Variablen sind als Beispiel zu verstehen. $IP bezeichnet das Kommando `ip` aus der iproute2 Package.
Dieses Skript aktiviert das Routing nicht. Dies sollte nach Aktivieren der Paketfilterregeln durch `echo 1 > /proc/sys/net/ipv4/ip` geschehen, damit die zu beschützenden Netzwerke nicht temporär während des Ladens der Filterregeln zugänglich sind.

```
#!/bin/sh

# Routing script (edited for educational purpose)
# René Pfeiffer <firewalls-78cd55b6cd-20020401@email.expiry.luchs.at>

# Get include file

. /etc/rc.d/rc.fireconf

# Activate RP filter on all interfaces
```

```
# Turn on source address verification in kernel
#
# (a cleaner way to set these things is to use
# /etc/sysctl.conf)

if [ -e /proc/sys/net/ipv4/conf/all/rp_filter ]; then
  for IF in /proc/sys/net/ipv4/conf/*/rp_filter; do
    echo 1 > $IF
  done
fi

# ------------- Linux Kernel

# Disallow broadcast pings and normals pings
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts 2> /dev/null
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all 2> /dev/null

# Change dynamically used ports for outgoing packets
echo "$IP_LOCAL_LO $IP_LOCAL_HI" > /proc/sys/net/ipv4/ip_local_port_range

# IP settings
echo 72     > /proc/sys/net/ipv4/ip_default_ttl
echo 524288 > /proc/sys/net/ipv4/ipfrag_high_thresh # 262144
echo 196608 > /proc/sys/net/ipv4/ipfrag_low_thresh # 196608

# TCP settings (sample, usually not needed to modify)
echo 7000 > /proc/sys/net/ipv4/tcp_keepalive_time
echo 8    > /proc/sys/net/ipv4/tcp_keepalive_probes

# Security settings
echo 0 > /proc/sys/net/ipv4/conf/all/accept_redirects 2> /dev/null
echo 1 > /proc/sys/net/ipv4/conf/all/accept_source_route 2> /dev/null
echo 1 > /proc/sys/net/ipv4/conf/all/log_martians 2> /dev/null

# Congestion window settings (sample)
echo 153600 > /proc/sys/net/core/rmem_default
echo 524288 > /proc/sys/net/core/rmem_default
echo 153600 > /proc/sys/net/core/wmem_default
echo 524288 > /proc/sys/net/core/wmem_default

# ------------- Routing

# Set default route

$IP route add default via $UPSTREAM_ROUTER

# Activate dummy device for blackholing traffic

$IP link set $BLACKHOLE up
$IP addr add $DUMMY dev $BLACKHOLE

# Blackhole private and multicast range

$IP route add $RFC1918_A dev $BLACKHOLE
$IP route add $RFC1918_B dev $BLACKHOLE
$IP route add $RFC1918_C dev $BLACKHOLE
$IP route add $LOCALLINK dev $BLACKHOLE
$IP route add $MULTICAST dev $BLACKHOLE

# Route certain FTP servers via ADSL for speedy access

$IP route add $GDTU_STORE via $ADSL
$IP route add $ADSL_DNS1  via $ADSL
$IP route add $ADSL_DNS2  via $ADSL

# Internal networks

$IP link  set $DMZ_DEV up
$IP addr  add $EXTERNAL_IP     dev $DMZ_DEV
$IP route add $INTERNAL_ROUTER dev $DMZ_DEV
```

```
$IP route add $LAN_NET via $INTERNAL_ROUTER dev $DMZ_DEV
$IP route add $TEC_NET via $INTERNAL_ROUTER dev $DMZ_DEV

# Source routing for DMZ
# (alternative routing table for DMZ machines, all others use
# main routing table)
#
# This needs to be done before we can use ”dmz_out” in
# combination with the ip command
#
# echo 200 dmz_out >> /etc/iproute2/rt_tables
#

$IP route add $AKIS dev $ISP_DEV

$IP route add $DMZ2NET dev $DMZ_DEV
$IP route add $DMZ3NET dev $DMZ_DEV
$IP route add $GILEAN  dev $DMZ_DEV

$IP rule add from $DMZ2NET      table dmz_out
$IP rule add from $DMZ3NET      table dmz_out
$IP rule add from $EXTERNAL_ISP table dmz_out

$IP route add $LAN_NET via $INTERNAL_ROUTER dev $DMZ_DEV table dmz_out
$IP route add $TEC_NET via $INTERNAL_ROUTER dev $DMZ_DEV table dmz_out
$IP route add default  via $AKIS dev $ISP_DEV src $EXTERNAL_IP table dmz_out

# ”COMMIT” for routers

$IP route flush cache

# IP Port Forwarding
#

$IPTABLES --table nat --insert PREROUTING --protocol tcp \
          --destination $FIREWALL_EX --destination-port $HTTP \
          --jump DNAT --to $EXCHANGE:$HTTP
$IPTABLES --table nat --insert PREROUTING --protocol tcp \
          --destination $FIREWALL_EX --destination-port $HTTPS \
          --jump DNAT --to $EXCHANGE:$HTTPS
$IPTABLES --table nat --insert PREROUTING --protocol udp \
          --destination $FIREWALL_EX --destination-port $HTTPS \
          --jump DNAT --to $EXCHANGE:$HTTPS
```

C.4. Einsatz und Aufbau von Bastion Hosts

Firewalls, Router und alle Server, die öffentlich Dienste anbieten, sollten mit der notwendigen Sorgfalt installiert und in Betrieb genommen werden.

Wichtig: keine Benutzer-Accounts am Bastion Host!

Gründe für dieses Verhalten:

- Schwachstellen der Accounts selber
- Schwächen im Support dieser Accounts
- verringerte Stabilität und Verläßlichkeit des Hosts
- unabsichtliches Untergraben der Hosts Security durch User
- erhöhte Schwierigkeit Attacken zu entdecken

Aufbau eines Bastion Hosts

1. Sichern der Maschine
2. Deaktivieren aller unnötigen Services

3. Installieren oder ändern der benötigten Services
4. Rekonfigurieren der Maschine von Development in den Einsatzzustand
5. Testen und Prüfen der Sicherheit (Auditing)
6. *Zum Schluß:* Verbinden des Hosts mit dem Einsatznetzwerk

C.4.1. Sichern der Maschine

- minimale Installation des Systems
- Beheben aller bekannten Systemfehler
- Verwenden bzw. Aufstellen einer Checkliste
- Sichern der System-Logs

C.4.2. Deaktivieren aller unnötigen Services

- Säubern der Init-Skripts
- Säubern der `inetd` Services
- unnötige Services am Bastion Host sind
  - NFS und Verwandte
    nfsd, mountd, statd, lockd, automount, rquotad, amd
  - andere RPC Prozesse
    ypserv, ypbind, ypupdated, rexd, walld
  - Bootprotokolle
    tftpd, bootd, bootpd
  - BSD „r“Kommandos
    rshd, rlogind, rexecd (alle anderen „r“Kommandos laufen ohne diese nicht)
  - routed
    in der Regel wird man am Bastion Host nicht dynamisch routen
  - fingerd
  - ftpd
    anonyme FTP Server sollten nur mit besonders dafür geeigneter Software betrieben werden
  - uucp, rwhod, lpd (Printing)
  - IP Routing und Forwarding

C.4.3. Installieren oder ändern der benötigten Services

- Ersetzen von Standardsoftware mit spezialisierter Software
  Beispiel: Austausch von `inetd` durch `xinetd`, Einsatz der Secure Shell statt Telnet, Einsatz eines „sicheren“FTP Servers
- Einsatz des TCP Wrappers
  $\longrightarrow$ Authentifizierung durch IP Adressen; einfache, aber nützlicher Kontrollmechanismus
- Installieren von überwachungssoftware
  evtl. alle Aufgaben des Promiscuous Logging an einen seperaten Guardian Host weitergeben

C.4.4. Rekonfigurieren der Maschine für Production Environment

- Konfigurieren des Kernels
  - keine Verwendung von Kernel Modulen
  - keine Netzwerkfilesysteme (Serverkomponente NFS, CODA, SMB/NetBIOS, AppleTalk)
  - keine Sniffing Tools (libpcap, Berkeley Packet Filter)
  - keine experimentellen Treiber
  - nicht mehrere Kernel Images auf der Maschine halten
- Entfernen aller unwichtigen Programme
  - Bastion Host ist *keine* Arbeitsumgebung
  - Entfernen der Entwicklungsumgebung
    Compiler, Header Files, Build Tools
  - Software für Graphical User Interfaces
  - Programme mit *setguid/setuid* Fähigkeiten

  Alternativ: Ersetzen dieser Programme mit Alarmierungstools, die nach Aufruf eine Notiz an die Administratoren senden
- Filesysteme als Read Only konfigurieren
  alternativ bei BSD basierten Systemem das *immutable* Flag setzen
- Backup des ganzen Systems
- Bilden und Archivieren von Checksummen aller statischen Files

C.4.5. Testen und Prüfen der Sicherheit (Auditing)

- Prüfen gegen bekannte Sicherheitsprobleme
- Scannen des Hosts mit verschiedenen Tools
- Ermitteln von Checksummen für alle Binaries
  erleichtert das Erkennen von veränderten Programmen durch Administratoren; sehr nützlich auch das Aufzeichnen von Inodes für bestimmte Programme
- Security Auditing sollte regelmässig wiederholt werden

Letzter Schritt: Verbinden mit dem Netzwerk

C.5. nmap Proben

C.5.1. Standard TCP connect() Scan mit Version-ID-Patch

```
Nmap (V. nmap) scan initiated 2.53 as:
nmap -sT -sV -sR -r -O -I -oN /root/output_sT ein.host.xy
Interesting ports on some.host.xy (a.b.c.d):
(The 1512 ports scanned but not shown below are in state: closed)
Port       State       Service (RPC)       Owner       Protocol     Version
21/tcp     open        ftp                 0           FTP          ProFTPD 1.2.0pre1
22/tcp     open        ssh                 0           SSH          1.5-1.2.26
25/tcp     open        smtp                0           SMTP
53/tcp     open        domain              0
110/tcp    open        pop-3               0           POP3         QPOP 2.53
111/tcp    open        sunrpc (rpcbind V2) 1
113/tcp    open        auth                65535
515/tcp    open        printer             0
724/tcp    open        (bwnfsd V1)         0
767/tcp    open        phonebook (mountd V1-2)0
2049/tcp   open        nfs (nfs V2)        0
```

```
TCP Sequence Prediction: Class=truly random
                         Difficulty=9999999 (Good luck!)
Remote operating system guess: Linux 2.0.35-38

# Nmap run completed at Sun May 28 19:00:57 2000 --
# 1 IP address (1 host up) scanned in 116 seconds
```

Anmerkung: Der Version-ID-Patch ist nun Teil der `nmap` Standarddistribution.

C.5.2. TCP connect() Scan - mit Portauswahl

```
nmap -sT -sR -I -p 21,22,23,25,110,111,113,143,80,2049,3128,4000 -O 212.17.78.195

Starting nmap V. 2.53 by fyodor@insecure.org ( www.insecure.org/nmap/ )
Interesting ports on TK212017078195.teleweb.at (212.17.78.195):
(The 1 port scanned but not shown below is in state: closed)
Port       State       Service (RPC)           Owner
21/tcp     open        ftp
22/tcp     open        ssh
23/tcp     filtered    telnet
25/tcp     open        smtp
80/tcp     open        http
110/tcp    filtered    pop-3
111/tcp    filtered    sunrpc
143/tcp    filtered    imap2
2049/tcp   filtered    nfs
3128/tcp   filtered    squid-http
4000/tcp   filtered    unknown

TCP Sequence Prediction: Class=random positive increments
                         Difficulty=2596192 (Good luck!)
Remote OS guesses: Linux 2.1.122 - 2.2.14, Linux kernel 2.2.13

Nmap run completed -- 1 IP address (1 host up) scanned in 9 seconds
```

C.5.3. ICMP Ping Sweep

```
nmap -sP 10.2.2.*

Starting nmap V. 2.30BETA21 by fyodor@insecure.org ( www.insecure.org/nmap/ )
Host subdomains.somewhere.lan (10.2.2.1) appears to be up.
Host prank.somewhere.lan (10.2.2.2) appears to be up.
Host dazzled.somewhere.lan (10.2.2.3) appears to be up.
Host deregulated.somewhere.lan (10.2.2.10) appears to be up.
Host appliers.somewhere.lan (10.2.2.11) appears to be up.
Host rotary.somewhere.lan (10.2.2.12) appears to be up.
Host telegraphic.somewhere.lan (10.2.2.15) appears to be up.
Host swept.somewhere.lan (10.2.2.19) appears to be up.
Host sitting.somewhere.lan (10.2.2.20) appears to be up.
Host functionals.somewhere.lan (10.2.2.22) appears to be up.
Host Freetown.somewhere.lan (10.2.2.27) appears to be up.
Host followed.somewhere.lan (10.2.2.38) appears to be up.
Host yanked.somewhere.lan (10.2.2.40) appears to be up.
Host contender.somewhere.lan (10.2.2.42) appears to be up.
Host robberies.somewhere.lan (10.2.2.43) appears to be up.
Host parrots.somewhere.lan (10.2.2.48) appears to be up.
Host ingestion.somewhere.lan (10.2.2.51) appears to be up.
Host load.somewhere.lan (10.2.2.59) appears to be up.
Host outskirts.somewhere.lan (10.2.2.61) appears to be up.
Host designs.somewhere.lan (10.2.2.68) appears to be up.
Host winded.somewhere.lan (10.2.2.200) appears to be up.
Host bucolic.somewhere.lan (10.2.2.201) appears to be up.
Host besmirched.somewhere.lan (10.2.2.202) appears to be up.
Host crumbled.somewhere.lan (10.2.2.203) appears to be up.
Host requester.somewhere.lan (10.2.2.205) appears to be up.
Host Scot.somewhere.lan (10.2.2.250) appears to be up.
Nmap run completed -- 256 IP addresses (26 hosts up) scanned in 6 seconds
```

C.5.4. Ansicht einer Linux 2.2.19 Firewall

```
# nmap (V. 2.54BETA7) scan initiated Tue Nov 21 15:43:59 2000 as:
  nmap -oN /tmp/shaytan -sT -sR -I -O shaytan.extern.tv
Insufficient responses for TCP sequencing (3), OS detection may be less accurate
Interesting ports on shaytan.extern.tv (221.67.3.99):
(The 1527 ports scanned but not shown below are in state: filtered)
Port       State       Service (RPC)           Owner
25/tcp     open        smtp                    root
113/tcp    open        auth                    nobody
222/tcp    open        rsh-spx                 root
223/tcp    open        cdc
47557/tcp  closed      dbbrowse
54320/tcp  closed      bo2k
65301/tcp  closed      pcanywhere

No exact OS matches for host (If you know what OS is running on it,
see http://www.insecure.org/cgi-bin/nmap-submit.cgi).

TCP/IP fingerprint:
SInfo(V=2.54BETA7%P=i586-pc-linux-gnu%D=11/21%Time=3A1A8B37%O=25%C=47557)
TSeq(Class=RI%gcd=1%SI=11AB56)
TSeq(Class=RI%gcd=1%SI=2F9258)
T1(Resp=Y%DF=Y%W=7F53%ACK=S++%Flags=AS%Ops=MENNTNW)
T1(Resp=Y%DF=Y%W=7F53%ACK=S++%Flags=AS%Ops=MENNTNW)
T2(Resp=Y%DF=Y%W=100%ACK=O%Flags=BRSF%Ops=)
T2(Resp=Y%DF=N%W=0%ACK=O%Flags=BPR%Ops=)
T2(Resp=Y%DF=N%W=0%ACK=O%Flags=BS%Ops=)
T3(Resp=Y%DF=Y%W=7F53%ACK=S++%Flags=AS%Ops=MENNTNW)
T3(Resp=Y%DF=Y%W=7F53%ACK=S++%Flags=AS%Ops=MENNTNW)
T4(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)
T4(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)
T5(Resp=Y%DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T5(Resp=Y%DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=N)
T6(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)
T7(Resp=N)
T7(Resp=N)
T7(Resp=Y%DF=N%W=0%ACK=S%Flags=AR%Ops=)
PU(Resp=N)
PU(Resp=Y%DF=N%TOS=C0%IPLEN=164%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%DAT=E)
PU(Resp=N)

# Nmap run completed at Tue Nov 21 15:48:23 2000
# 1 IP address (1 host up) scanned in 264 seconds
```

C.5.5. Linux® 2.2.19 Paketfilter von der LAN Seite gesehen

```
# nmap (V. 2.54BETA29) scan initiated Fri Oct 12 11:14:52 2001 as:
# nmap -sTUR -P0 -oN /tmp/fw_innen.txt -I -O fw.lan.seit.te
Interesting ports on fw.lan.seit.te (fw.lan.seit.te):
(The 2993 ports scanned but not shown below are in state: closed)
Port       State       Service (RPC)           Owner
25/tcp     open        smtp
53/tcp     open        domain
53/udp     open        domain
222/tcp    open        rsh-spx
514/udp    open        syslog
800/udp    open        mdbs_daemon
953/tcp    open        rndc
45000/udp  open        ciscopop

Remote operating system guess: Linux 2.1.19 - 2.2.17
Uptime 46.892 days (since Sun Aug 26 14:14:53 2001)

# Nmap run completed at Fri Oct 12 11:39:48 2001 --
# 1 IP address (1 host up) scanned in 1496 seconds
```

Einige der Services sind nicht richtig bezeichnet, da es sich um Ports handelt, die für VPN Tunnel benutzt werden. `nmap` prüft nicht welches Protokoll an einem Port vorhanden ist.

D. Protokolle und Ports

D.1. Protokolle

Eine komplette Liste befindet sich unter http://www.iana.org/assignments/protocol-numbers

```
PROTOCOL NUMBERS

(last updated 2001 June 28)

In the Internet Protocol version 4 (IPv4) [RFC791] there is a field,
called "Protocol", to identify the next level protocol.  This is an 8
bit field.  In Internet Protocol version 6 (IPv6) [RFC1883] this field
is called the "Next Header" field.

Assigned Internet Protocol Numbers

Decimal    Keyword     Protocol                          References
-------    -------     --------                          ----------
     0     HOPOPT      IPv6 Hop-by-Hop Option             [RFC1883]
     1     ICMP        Internet Control Message            [RFC792]
     2     IGMP        Internet Group Management          [RFC1112]
     3     GGP         Gateway-to-Gateway                  [RFC823]
     4     IP          IP in IP (encapsulation)           [RFC2003]
     5     ST          Stream                    [RFC1190,RFC1819]
     6     TCP         Transmission Control                [RFC793]
     7     CBT         CBT                              [Ballardie]
     8     EGP         Exterior Gateway Protocol     [RFC888,DLM1]
     9     IGP         any private interior gateway          [IANA]
                       (used by Cisco for their IGRP)
    10     BBN-RCC-MON BBN RCC Monitoring                     [SGC]
    11     NVP-II      Network Voice Protocol          [RFC741,SC3]
    12     PUP         PUP                              [PUP,XEROX]
    13     ARGUS       ARGUS                                 [RWS4]
    14     EMCON       EMCON                                  [BN7]
    15     XNET        Cross Net Debugger             [IEN158,JFH2]
    16     CHAOS       Chaos                                 [NC3]
    17     UDP         User Datagram                   [RFC768,JBP]
    18     MUX         Multiplexing                     [IEN90,JBP]
    19     DCN-MEAS    DCN Measurement Subsystems            [DLM1]
    20     HMP         Host Monitoring                 [RFC869,RH6]
    21     PRM         Packet Radio Measurement              [ZSU]
    22     XNS-IDP     XEROX NS IDP                [ETHERNET,XEROX]
    23     TRUNK-1     Trunk-1                               [BWB6]
    24     TRUNK-2     Trunk-2                               [BWB6]
    25     LEAF-1      Leaf-1                                [BWB6]
    26     LEAF-2      Leaf-2                                [BWB6]
    27     RDP         Reliable Data Protocol          [RFC908,RH6]
    28     IRTP        Internet Reliable Transaction   [RFC938,TXM]
    29     ISO-TP4     ISO Transport Protocol Class 4  [RFC905,RC77]
    30     NETBLT      Bulk Data Transfer Protocol     [RFC969,DDC1]
    31     MFE-NSP     MFE Network Services Protocol   [MFENET,BCH2]
    32     MERIT-INP   MERIT Internodal Protocol              [HWB]
    33     SEP         Sequential Exchange Protocol         [JC120]
    34     3PC         Third Party Connect Protocol          [SAF3]
    35     IDPR        Inter-Domain Policy Routing Protocol  [MXS1]
    36     XTP         XTP                                   [GXC]
    37     DDP         Datagram Delivery Protocol            [WXC]
    38     IDPR-CMTP   IDPR Control Message Transport Proto  [MXS1]
    39     TP++        TP++ Transport Protocol               [DXF]
    40     IL          IL Transport Protocol              [Presotto]
    41     IPv6        Ipv6                               [Deering]
    42     SDRP        Source Demand Routing Protocol        [DXE1]
    43     IPv6-Route  Routing Header for IPv6            [Deering]
```

```
 44     IPv6-Frag   Fragment Header for IPv6          [Deering]
 45     IDRP        Inter-Domain Routing Protocol   [Sue Hares]
 46     RSVP        Reservation Protocol           [Bob Braden]
 47     GRE         General Routing Encapsulation     [Tony Li]
 48     MHRP        Mobile Host Routing Protocol[David Johnson]
 49     BNA         BNA                          [Gary Salamon]
 50     ESP         Encap Security Payload for IPv6   [RFC1827]
 51     AH          Authentication Header for IPv6    [RFC1826]
 52     I-NLSP      Integrated Net Layer Security  TUBA [GLENN]
 53     SWIPE       IP with Encryption                   [JI6]
 54     NARP        NBMA Address Resolution Protocol  [RFC1735]
 55     MOBILE      IP Mobility                       [Perkins]
 56     TLSP        Transport Layer Security Protocol   [Oberg]
                    using Kryptonet key management
 57     SKIP        SKIP                              [Markson]
 58     IPv6-ICMP   ICMP for IPv6                     [RFC1883]
 59     IPv6-NoNxt  No Next Header for IPv6           [RFC1883]
 60     IPv6-Opts   Destination Options for IPv6      [RFC1883]
 61                 any host internal protocol           [IANA]
 62     CFTP        CFTP                           [CFTP,HCF2]
 63                 any local network                    [IANA]
 64     SAT-EXPAK   SATNET and Backroom EXPAK             [SHB]
 65     KRYPTOLAN   Kryptolan                            [PXL1]
 66     RVD         MIT Remote Virtual Disk Protocol      [MBG]
 67     IPPC        Internet Pluribus Packet Core         [SHB]
 68                 any distributed file system          [IANA]
 69     SAT-MON     SATNET Monitoring                     [SHB]
 70     VISA        VISA Protocol                        [GXT1]
 71     IPCV        Internet Packet Core Utility          [SHB]
 72     CPNX        Computer Protocol Network Executive  [DXM2]
 73     CPHB        Computer Protocol Heart Beat         [DXM2]
 74     WSN         Wang Span Network                     [VXD]
 75     PVP         Packet Video Protocol                 [SC3]
 76     BR-SAT-MON  Backroom SATNET Monitoring            [SHB]
 77     SUN-ND      SUN ND PROTOCOL-Temporary             [WM3]
 78     WB-MON      WIDEBAND Monitoring                   [SHB]
 79     WB-EXPAK    WIDEBAND EXPAK                        [SHB]
 80     ISO-IP      ISO Internet Protocol                 [MTR]
 81     VMTP        VMTP                                 [DRC3]
 82     SECURE-VMTP SECURE-VMTP                          [DRC3]
 83     VINES       VINES                                 [BXH]
 84     TTP         TTP                                   [JXS]
 85     NSFNET-IGP  NSFNET-IGP                            [HWB]
 86     DGP         Dissimilar Gateway Protocol     [DGP,ML109]
 87     TCF         TCF                                  [GAL5]
 88     EIGRP       EIGRP                          [CISCO,GXS]
 89     OSPFIGP     OSPFIGP                       [RFC1583,JTM4]
 90     Sprite-RPC  Sprite RPC Protocol            [SPRITE,BXW]
 91     LARP        Locus Address Resolution Protocol     [BXH]
 92     MTP         Multicast Transport Protocol          [SXA]
 93     AX.25       AX.25 Frames                         [BK29]
 94     IPIP        IP-within-IP Encapsulation Protocol   [JI6]
 95     MICP        Mobile Internetworking Control Pro.   [JI6]
 96     SCC-SP      Semaphore Communications Sec. Pro.    [HXH]
 97     ETHERIP     Ethernet-within-IP Encapsulation     [RDH1]
 98     ENCAP       Encapsulation Header         [RFC1241,RXB3]
 99                 any private encryption scheme        [IANA]
100     GMTP        GMTP                                 [RXB5]
101     IFMP        Ipsilon Flow Management Protocol   [Hinden]
102     PNNI        PNNI over IP                        [Callon]
103     PIM         Protocol Independent Multicast  [Farinacci]
104     ARIS        ARIS                              [Feldman]
105     SCPS        SCPS    [Durst]
106     QNX        QNX   [Hunter]
107    A/N        Active Networks                     [Braden]
108    IPComp      IP Payload Compression Protocol   [RFC2393]
109    SNP        Sitara Networks Protocol          [Sridhar]
110    Compaq-Peer Compaq Peer Protocol                [Volpe]
111    IPX-in-IP   IPX in IP         [Lee]
112     VRRP        Virtual Router Redundancy Protocol [Hinden]
113     PGM         PGM Reliable Transport Protocol  [Speakman]
```

114		any 0-hop protocol	[IANA]
115	L2TP	Layer Two Tunneling Protocol	[Aboba]
116	DDX	D-II Data Exchange (DDX)	[Worley]
117	IATP	Interactive Agent Transfer Protocol	[Murphy]
118	STP	Schedule Transfer Protocol	[JMP]
119	SRP	SpectraLink Radio Protocol	[Hamilton]
120	UTI	UTI	[Lothberg]
121	SMP	Simple Message Protocol	[Ekblad]
122	SM	SM	[Crowcroft]
123	PTP	Performance Transparency Protocol	[Welzl]
124	ISIS over IPv4		[Przygienda]
125	FIRE		[Partridge]
126	CRTP	Combat Radio Transport Protocol	[Sautter]
127	CRUDP	Combat Radio User Datagram	[Sautter]
128	SSCOPMCE		[Waber]
129	IPLT		[Hollbach]
130	SPS	Secure Packet Shield	[McIntosh]
131	PIPE	Private IP Encapsulation within IP	[Petri]
132	SCTP	Stream Control Transmission Protocol	[Stewart]
133	FC	Fibre Channel	[Rajagopal]
134	RSVP-E2E-IGNORE		[RFCXXXX]
135-254		Unassigned	[IANA]
255		Reserved	[IANA]

REFERENCES

[CFTP] Forsdick, H., "CFTP", Network Message, Bolt Beranek and Newman, January 1982.

[CISCO] Cisco Systems, "Gateway Server Reference Manual", Manual Revision B, January 10, 1988.

[DDN] Feinler, E., Editor, "DDN Protocol Handbook", Network Information Center, SRI International, December 1985.

[DGP] M/A-COM Government Systems, "Dissimilar Gateway Protocol Specification, Draft Version", Contract no. CS901145, November 16, 1987.

[ETHERNET] "The Ethernet, A Local Area Network: Data Link Layer and Physical Layer Specification", AA-K759B-TK, Digital Equipment Corporation, Maynard, MA. Also as: "The Ethernet - A Local Area Network", Version 1.0, Digital Equipment Corporation, Intel Corporation, Xerox Corporation, September 1980. And: "The Ethernet, A Local Area Network: Data Link Layer and Physical Layer Specifications", Digital, Intel and Xerox, November 1982. And: XEROX, "The Ethernet, A Local Area Network: Data Link Layer and Physical Layer Specification", X3T51/80-50, Xerox Corporation, Stamford, CT., October 1980.

[IEN90] Cohen, D. and J. Postel, "Multiplexing Protocol", IEN 90, USC/Information Sciences Institute, May 1979.

[IEN119] Forgie, J., "ST - A Proposed Internet Stream Protocol", IEN 119, MIT Lincoln Laboratory, September 1979.

[IEN158] Haverty, J., "XNET Formats for Internet Protocol Version 4", IEN 158, October 1980.

[MFENET] Shuttleworth, B., "A Documentary of MFENet, a National Computer Network", UCRL-52317, Lawrence Livermore Labs, Livermore, California, June 1977.

[PUP] Boggs, D., J. Shoch, E. Taft, and R. Metcalfe, "PUP: An Internetwork Architecture", XEROX Palo Alto Research Center, CSL-79-10, July 1979; also in IEEE Transactions on Communication, Volume COM-28, Number 4, April 1980.

[SPRITE] Welch, B., "The Sprite Remote Procedure Call System",

Technical Report, UCB/Computer Science Dept., 86/302, University of California at Berkeley, June 1986.

[RFC741] Cohen, D., "Specifications for the Network Voice Protocol", RFC 741, ISI/RR 7539, USC/Information Sciences Institute, March 1976.

[RFC768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, USC/Information Sciences Institute, August 1980.

[RFC791] Postel, J., "Internet Protocol - DARPA Internet Prnogram Protocol Specification", STD 5, RFC 791, DARPA, September 1981.

[RFC792] Postel, J., "Internet Control Message Protocol - DARPA Internet Program Protocol Specification", STD 5, RFC 792, USC/Information Sciences Institute, September 1981.

[RFC793] Postel, J., "Transmission Control Protocol - DARPA Internet Program Protocol Specification", STD 7, RFC 793, USC/Information Sciences Institute, September 1981.

[RFC823] Hinden, R., and A. Sheltzer, "The DARPA Internet Gateway", RFC 823, BBN, September 1982.

[RFC869] Hinden, R., "A Host Monitoring Protocol", RFC 869, Bolt Beranek and Newman, December 1983.

[RFC888] Seamonson, L., and E. Rosen, "STUB" Exterior Gateway Protocol", RFC 888, BBN Communications Corporation, January 1984.

[RFC905] International Standards Organization, "ISO Transport Protocol Specification - ISO DP 8073", RFC 905, April 1984.

[RFC908] Velten, D., R. Hinden, and J. Sax, "Reliable Data Protocol", RFC 908, BBN Communications Corporation, July 1984.

[RFC938] Miller, T., "Internet Reliable Transaction Protocol", RFC 938, ACC, February 1985.

[RFC969] Clark, D., M. Lambert, and L. Zhang, "NETBLT: A Bulk Data Transfer Protocol", RFC 969, MIT Laboratory for Computer Science, December 1985.

[RFC1112] Deering, S., "Host Extensions for IP Multicasting", STD 5, RFC 1112, Stanford University, August 1989.

[RFC1190] Topolcic, C., Editor, "Experimental Internet Stream Protocol, Version 2 (ST-II)", RFC 1190, CIP Working Group, October 1990.

[RFC1241] Woodburn, W., and D. Mills, " A Scheme for an Internet Encapsulation Protocol: Version 1", RFC 1241, SAIC, University of Delaware, July 1991.

[RFC1583] Moy, J., "The OSPF Specification", RFC 1583, Proteon, March 1994.

[RFC1735] Heinanen, J., and R. Govindan, " NBMA Address Resolution Protocol (NARP)", RFC 1735, Telecom Finland and USC/ISI, December 1994.

[RFC1819] L. Delgrossi, L. Berger, and ST2 Working Group, "Internet Stream Protocol Version 2 (ST2) Protocol Specification - Version ST2+", RFC 1819, August 1995.

[RFC1826] Atkinson, R., "IP Authentication Header", RFC 1826, Naval Research Laboratory, August 1995.

[RFC1827] Atkinson, R., "IP Encapsulating Security Payload (ESP)", RFC

1827, Naval Research Laboratory, August 1995.

[RFC1883] Deering, S., and R. Hinden, ”Internet Protocol, Version 6 (IPv6) Specification”, RFC 1883, Xerox PARC, Ipsilon Networks, December 1995.

[RFC2003] Perkins, C., ”IP Encapsulation within IP”, RFC 2003, IBM, September 1996.

[RFC2393] Shacham, A., and R. Monsour, R. Pereira, M. Thomas, ”IP Payload Compression Protocol (IPComp)”, RFC 2393, Cisco, Hi/fn, TimeStep, AltaVista Internt, December 1998.

[RFCXXXX] F. Baker, C. Iturralde, F. Le Faucheur, B. Davie, ”Aggregation of RSVP for IPv4 and IPv6 Reservations”, RFC XXXX, Month Year.

D.1.1. Ports

Jedes Linux® System hat in `/etc/services` eine rudimentäre Portliste. Eine wesentlich vollständigere Liste findet man unter den folgenden Quellen.

- http://www.iana.org/assignments/port-numbers
- http://www.seifried.org/security/ports/

Literaturverzeichnis

[1] J. Postel, RFC791, *Internet Protocol*, *DARPA Internet Program*, USC/Information Sciences Institute, September 1981.

[2] J. Reynolds, RFC2600, *Internet Official Protocol Standards*, Network Working Group Internet Engineering Task Force, März 2000.

[3] J. Postel, RFC790, *Assigned Numbers*, *DARPA Internet Program*, USC/Information Sciences Institute, September 1981.

[4] J. Postel, RFC793, *Transmission Control Protocol*, *DARPA Internet Program*, USC/Information Sciences Institute, September 1981.

[5] K. Ramakrishnan, S. Floyd, RFC2481, *A Proposal to add Explicit Congestion Notification (ECN) to IP*, AT&T Labs Research, LBNL, Januar 1999.

[6] Tools für das Abgreifen von Netzwerkverkehr: `tcpdump`, wireshark

[7] Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, Murari Sridharan, *Data Center TCP (DCTCP)*, Data Center Networks session Proc. ACM SIGCOMM, New Delhi, 2010.

[8] Mohammad Alizadeh, Adel Javanmard, Balaji Prabhakar, *Analysis of DCTCP: Stability, Convergence, and Fairness*, Proc. ACM SIGMETRICS, San Jose, 2011.

[9] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, Van Jacobson, *BBR: Congestion-Based Congestion Control*, ACM Queue, Vol. 14 No. 5, September-October 2016.

[10] D.A. Hayes und G. Armitage, *Revisiting TCP congestion control using delay gradients*, IFIP Networking, Seiten 328-341. Springer, 2011.

[11] J. Sing, B. Soh, *Proceeding NCA '05 Proceedings of the Fourth IEEE International Symposium on Network Computing and Applications*, S. 73-82, IEEE Computer Society Washington, DC, USA, 2005.

[12] J. Postel, RFC792, *Internet Control Message Protocol*, *DARPA Internet Program*, USC/Information Sciences Institute, September 1981.

[13] R. Braden, RFC1122, *Requirements for Internet Hosts - Communication Layers*, USC/Information Sciences Institute, Oktober 1989.

[14] F. Baker, RFC1812, *Requirements for IP Version 4 Routers*, Cisco Systems, Juni 1995.

[15] J. Arkko, J. Kempf, B. Zill, P. Nikander, RFC3971, *SEcure Neighbor Discovery (SEND)*, März 2005.

[16] S. Thomson, T. Narten, RFC2462, *IPv6 Stateless Address Autoconfiguration*, Dezember 1998.

[17] E. Davies, J. Mohacsi, RFC4890, *Recommendations for Filtering ICMPv6 Messages in Firewalls*, May 2007.

[18] J. Roskind, *Multiplexed Stream Transport Over UDP*, April 2012.

[19] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, RFC3550, *RTP: A Transport Protocol for Real-Time Applications*, July 2003.

[20] J. Weil, V. Kuarsingh, C. Donley, C. Liljenstolpe, M. Azinger, RFC6598, *IANA-Reserved IPv4 Prefix for Shared Address Space*, April 2012.

[21] Bill Cerveny, *IPv6 Fragmentation*, July 25, 2011.

[22] S. Nadas, *Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6*, March 2010.

[23] C. Huitema, RFC4380, *Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)*, Februar 2006.

[24] Craig Hunt, *TCP/IP Network Administration*, 2nd Edition, O'Reilly & Associates, Inc., Dezember 1997.

[25] W. Richard Stevens, *TCP/IP Illustrated, Volume 1, The Protocols*, Addison-Wesley, 1994.

[26] Gary R. Wright & W. Richard Stevens, *TCP/IP Illustrated, Volume 2, The Implementation*, Addison-Wesley, 1995.

[27] Adolfo Rodriguez et al., *TCP/IP Tutorial and Technical Overview*, IBM Redbooks, ISBN 0738421650, IBM Form Number GG24-3376-06, August 2001.

[28] Bert Hubert, et. al., *Linux® Advanced Routing & Traffic Control*, http://www.lartc.org/.

[29] Y. Rekhter, C. Topolcic, RFC1520, *Exchanging Routing Information Across Provider Boundaries in the CIDR Environment*, T.J. Watson Research Center, IBM Corp., CNRI, September 1993.

[30] Y. Rekhter, RFC1817, *CIDR and Classful Routing*, Cisco Systems, August 1995.

[31] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, E. Lear, RFC1918, *Address Allocation for Private Internets*, Cisco Systems, Chrysler Corp., RIPE NCC, Silicon Graphics, Inc., Februar 1996.

[32] S. Deering, RFC1112, *Host Extensions for IP Multicasting*, Stanford University, August 1989.

[33] R. Finlayson, RFC2588, *IP Multicast and Firewalls*, `live.com`, Mai 1999.

[34] Kevin D. Mitnick, William L. Simon, Steve Wozniak, *The Art of Deception: Controlling the Human Element of Security*, John Wiley & Sons, 2002.

[35] Elizabeth D. Zwicky, Simon Cooper & D. Brent Chapman, *Building Internet Firewalls*, 2nd Edition, O'Reilly & Associates, Inc., 2000.

[36] FreeS/WAN Project, IPsec RFC collection, http://www.freeswan.org/freeswan_trees/freeswan-1.91/doc/rfc.html.

[37] S. Hanks, D. Farinacci, P. Traina, *Generic Routing Encapsulation (GRE)*, Cisco Systems, Oktober 1994.

[38] S. Hanks, D. Farinacci, P. Traina, *Generic Routing Encapsulation over IPv4 networks*, Cisco Systems, Oktober 1994.

[39] W. Townsley, A. Valencia, A. Rubens, G. Pall, G. Zorn, B. Palter, *Layer Two Tunneling Protocol "L2TP``*, Cisco Systems, Ascend Communications, Microsoft Corporation, Redback Networks, August 1999.

[40] R. Russell, *Linux® 2.4 Packet Filtering HOWTO*, http://netfilter.samba.org/unreliable-guides/.

[41] K. Seifried, *Linux® Administrator's Security Guide*, http://www.linuxdoc.org/LDP/lasg/.

[42] D. J. Bernstein, *djbdns* DNS Server Software und Tools, http://cr.yp.to/.

[43] C. L. Mattos, *Security flaw in Linux® 2.4 IPTables using FTP PORT*, Tempest Security Technologies - Advisory #01 / 2001, http://netfilter.samba.org/security-fix/index.html.

[44] G. Combs, *Wireshark Network Protocol Analyzer*, http://www.wireshark.org/.

[45] Atul Gawande, *The Checklist Manifesto - How to Get Things Right*, Henry Holt and Co., 2010.

[46] M. Burgess. „Cfengine: a system configuration engine.“ University of Oslo report 1993. PDF

[47] M. Burgess. „Configurable immunity for evolving human-computer systems.“ Science of Computer Programming **51** 2004, S. 197-213. PDF

[48] Zamboni, Diego. „Learning CFEngine 3“. O'Reilly Media. 2012. ISBN-10: 1-4493-1220-9, ISBN-13: 978-1-4493-1220-6

[49] Shah, Gourav. „Ansible Playbook Essentials“. Packt Publishing. 2015. ISBN-13: 978-1-7843-9829-3

[50] Bentley, Walter. „OpenStack Administration with Ansible“. Packt Publishing. 2016. ISBN-13: 978-1-7858-8461-0

[51] O. Arkin, *ICMP Usage In Scanning*, http://www.sys-security.com/html/papers.html, Sys-Security Group, Juni 2001.

[52] I. Ristic, *Apache Security*, O'Reilly, ISBN 0596007248, März 2005.

[53] Joel Scambray, Mike Shema & Caleb Sima, *Hacking Exposed™ Web Applications*, 1. Auflage, ISBN 0-07-226299-0, McGraw-Hill, 2006.

[54] A. Vladimirov, *Hacking Exposed™ Cisco® Networks*, 1. Auflage, ISBN 0-07-225917-5, McGraw-Hill, 2006.

[55] Peter van der Linden, *Deep C Secrets*, Prentice Hall, ISBN-10 0131774298, ISBN-13 9780131774292, Juni 1994.

[56] Bundesamt für Sicherheit in der Informationstechnik (BSI), *Technische Richtlinie Sicheres WLAN*, SecuMedia Verlags GmbH, ISBN 978-3-922746-70-6, 2005.

[57] Ross J. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, ISBN 9780470068526, Wiley Publishing, 2008.

[58] Bruce Schneier, *Beyond Fear*, ISBN 0-387-02620-7, Copernicus Books, September 2003.

Wichtig: Bitte beim Nachschlagen von RFCs beachten, daß viele der hier angeführten RFCs durch spätere erweitert bzw. ersetzt wurden. Beispielsweise definiert RFC 793 nicht mehr alleinig das heutzutage eingesetzte TCP. Unter dem RFC Index[1] kann man online RFCs nachschlagen.

[1] http://www.faqs.org/rfcs/rfc-titles.html

Index

802.11i, 98, 99
802.1Q, 97
802.1X, 96

active/active Failover, 42
AH, 35
Ansible, 86
AoE, 31
Apache, 89
ARP, 16
ARP Flooding, 16
ARP Poisoning, 16

Berkeley Packet Filter, 42
Body, 83
Bogon, 26
Bogon Filtering, 26
Bonjour, 31
Botnet, 33
BPF, 42
BPFilter, 42
Bundle, 83

CA, 36
CAPTCHA, 102
Carrier-grade NAT, 26
CCMP, 98
CDP, 31
Certificate Authority, 36
conntrack-tools, 42
conntrackd, 42
Content Security Policy, 90
Cross-Site Request Forgery, 104
CSP, 90
CSRF, 104

DAD, 30
Dateisysteme, 78
DCCP, 24
DDoS, 32
Demilitarisierte Zone, 45
DHCP Snooping, 16, 96
DHCPv6, 30
DIAMETER, 96
Digest Access Authentication, 102
DMZ, 45
DNS, 57
DoS, 32

EAP, 96
ECN, 20
ESP, 35
ESTABLISHED, 41

Fireball, 87
FTP, 54

grml, 76

Hardening, 77
HPKP, 90
HSTS, 90
HTTP AUTH, 102
HTTP TRACE, 90
Hypervisor, 109
Härten, 77

ICMP, 23
ICMP Codes, 59
ICMPv4, 23
ICMPv6, 23
IKE, 35
Input Validation, 104
Internet Control Message Protocol, 23
Internet Protocol, 16
Internetprotokoll, 15
INVALID, 41
IP, 15
IPsec, 35
IPv4 Adresse, 25
IPv6, 28, 59
IPv6 Adresse, 25
IPv6 Router, 30
IPv6-in-IPv4, 59
IPv6-over-IPv4, 59

Kali Linux, 76

L2TP, 31, 35
LCAP, 31
LLDP, 31
Logging, 60

MAC, 16
Martian, 26
Maximum Transfer Unit, 27
Media Access Control, 16
Metasploit, 69

mod_security, 89
Mount Options, 80
MySQL, 91

NAC, 96
NAPT, 27
NAT, 27, 97
NDP, 23, 30
Neighbor Discovery Protocol, 23
Nessus, 69
Netfilter, 46
Network Address Port Translation, 27
Network Address Translation, 27
NEW, 41
nft, 42
Nftables, 42
nmap, 65
Null Routing, 26

OpenVAS, 69
OpenVPN, 35
OSI Modell, 15

Paketfragmentierung, 28
Partitionierung, 78
Perimeternetzwerk, 45
PHP, 91
PHP als CGI, 94
PKI, 36
PPP, 31
PPTP, 35
Proxy Server, 34

QUIC, 24

RADIUS, 96
radvd, 30
Real-time Transport Protocol, 24
RELATED, 41
RFC 6598, 26
Routing Tabelle, 26
RSN, 98
RTP, 24

SCTP, 24
SEcure Neighbor Discovery, 24
security level, 46
SEND, 24
Session Fixation, 103
Session Hijacking, 103
Sicherheitsstufen, 46
SLP, 31
Sniffer, 63
SOAP, 106
SSDP, 31
SSH, 53
STP, 31
Suchmaschinen, 102

suEXEC, 94

Teredo Tunnel, 60
tethereal, 63
Time To Live, 26
TIPC, 24
TKIP, 98
Transmission Control Protocol, 19
TTL, 17

UDDI, 106
UDP over IPv4, 60
UDPv4, 60
User Datagram Protocol, 22

Verschlüsselung, 35
Virtualisierung, 109
VLAN, 97
VoIP, 34
VRRP, 31
VTP, 97

Web Services, 106
WEP, 99
WireGuard, 36
WPA, 99
WPA2, 99
WSDL, 106

X-Frame-Options, 91
XFO, 91
XML, 106

Zertifikat, 36
Zombie, 33

ØMQ, 87