Datensicherheit III

Firewalls

Version 1.0

Wintersemester 2002/2003

Technikum Wien

René Pfeiffer

firewalls-11cf9ea664-20030225@email.expiry.luchs.at

Systemadministrator GNU/Linux Manages!

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

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 angegebenen Referenzen zu finden.

1.1 Begriffe

Viele Begriffe der Informationstechnologie kommen ursprünglich aus der englischen Sprache und werden ohne Übersetzung benutzt. Damit keine Verwechslungen oder Mißdeutungen auftreten, sollen die wichtigsten Begriffe und Abkürzungen kurz erläutert werden.

- 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.

- 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.

- 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.

- internet / ein Internet
  Ein Netzwerk aus 2 oder mehr Maschinen.

- Protokoll
  Als Protokoll bezeichnet man eine Sammlung von Konventionen und Regeln, die eine

[1] Faustregel: Ein Client ist eine Maschine, die niemand vermißt.

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
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 1.1: 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.

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.

1.2 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 Internetprotkoll wurde mit dem Ziel entwickelt, mehreren Computersystemen das Teilen von Resourcen zu ermöglichen. Bei 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.

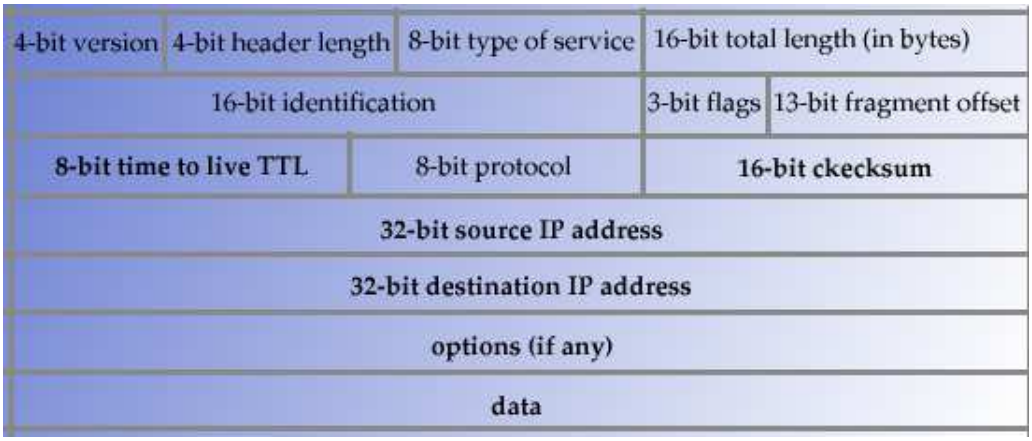

Abbildung 1.1: Aufbau eines Paket Headers im Internet Protocol (IP). Die minimale Länge beträgt 20 Byte ohne Daten.

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. In den meisten Fällen sind die folgenden Schichten völlig ausreichend.

- Applikationsschicht
- Transportschicht
- Vermittlungsschicht
- Übertragungsschicht

1.2.1 Internet Protocol (IP)

Das Internet Protocol (IP) ist die Basis der Internetprotokolle, auch wenn es komisch klingen mag. IP ist ein paketorientiertes und verbindungsloses Protokoll. 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.
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. Der Header 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. Ein ganzes IP Paket hat die in 1.1 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 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.

- 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).

- 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 und eine diesbezügliche Meldung an den Sender des Paketes generieren. Das Paket und seine Daten verschwinden damit aus dem Netzwerk. Es

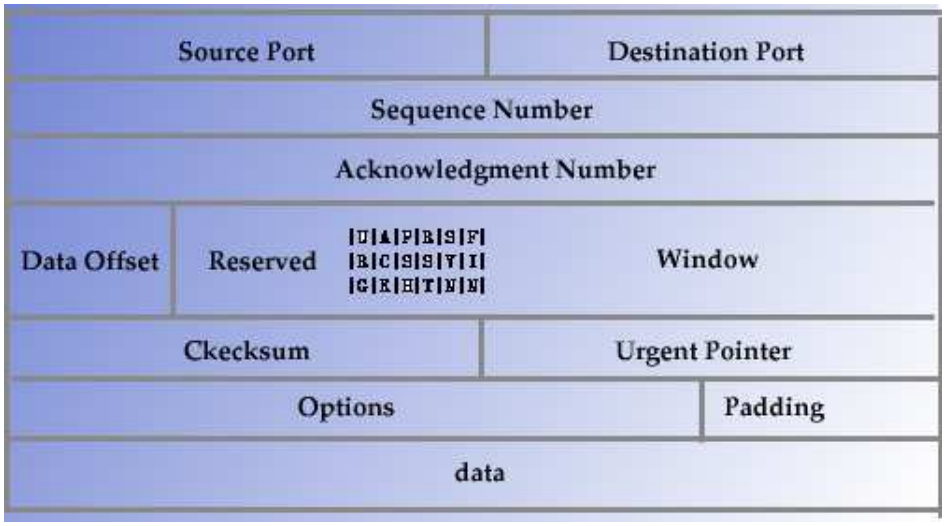

Abbildung 1.2: Aufbau eines Paket Headers im Transmission Control Protocol (TCP). Die minimale Länge beträgt 20 Byte ohne Daten. Der TCP Header ist als Zusatz zum IP Header zu sehen.

gibt folglich keinen Mechanismus, der eine Datenübertragung absichert. Zusätzlich fehlen Informationen, die die Daten an eine bestimmte „Hausnummer“ an der Zieladresse liefern (*Multiplexing*, *Demultiplexing*). Diese beiden Umstände werden mit zwei weiteren Protokollen, die auf IP aufsetzen, angesprochen.

1.2.2 Transmission Control Protocol (TCP)

Das Transmission Control Protocol (TCP) [4] bietet eine verbindungsorientierte Datenübertragung mit Fehlererkennung. 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 1.2 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 und zum Austausch der Initial Sequence Number (ISN) beim Verbindungsaufbau.

- Acknowledgement Number (ACK, 32 Bit)
  Wenn das ACK Kontrollbit gesetzt ist, dann enthält dieses Feld die nächste SEQ Num-

mer, 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 noch nicht auf allen netzwerktauglichen Geräten implementiert. Manche ältere Firewallsysteme blockieren Pakete mit ECN Erweiterungen.
- TCP Window (16 Bit)
  Gibt die Anzahl der Bytes an, die der Sender dieses Pakets gewillt ist zu empfangen.
- 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. **SYNACK 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. 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 etablierten Verbindung ausweist. In Abbildung 1.3 ist dieser Vorgang anhand eines Beispiels dargestellt. Dort besitzt das erste Paket eine SYN, ECN und CWR Kombination, weil eine der beteiligten Maschinen ECN-fähig ist.

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

Abbildung 1.3: 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.

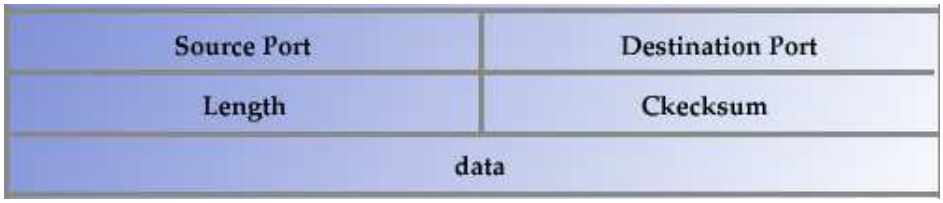

Abbildung 1.4: Aufbau eines Paket Headers im User Datagram Protocol (UDP). Die Länge beträgt 8 Byte ohne Daten. Der UDP Header ist als Zusatz zum IP Header zu sehen.

1.2.3 User Datagram Protocol (UDP)

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.

Abbildung 1.4 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.

1.2.4 Internet Control Message Protocol (ICMP)

Das Internet Control Message Protocol (ICMP) [7, 8, 9] 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

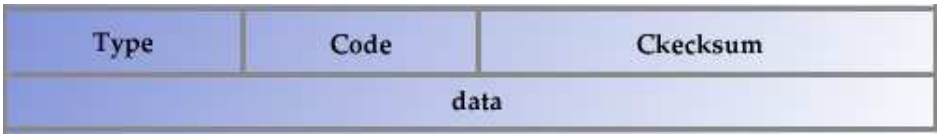

Abbildung 1.5: Aufbau eines Paket Headers im Internet Control Message Protocol (ICMP). Die Länge beträgt 8 Byte ohne Daten. Der ICMP Header ist als Zusatz zum IP Header zu sehen.

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 [9]
14	Host Precedence Violation [9]
15	Precedence cutoff in effect [9]

Tabelle 1.2: Hier sind alle möglichen Codes zur ICMP Meldung Typ 3 `Destination Unreachable` aufgelistet.

Statusmeldungen, besitzt jedoch ebenso wie TCP und UDP einen Header- und einen Datenteil (siehe Abbildung 1.5).

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 1.2 aufgelistet. Der Fehlertyp wird durch die einzelnen Fehlercodes genauer spezifiziert.

1.2.5 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. [10, 11, 12, 13]

IP Adressen und Netzwerke

Eine IP 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 [15], 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.
IP Adressen sind in bestimmte Bereiche eingeteilt. Nicht alle Adressen finden sich im Internet. Die folgenden Bereiche sind ausschließlich für den Einsatz in LANs reserviert. [16]

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

Außerdem gibt es einen speziellen Adreßbereich `224.0.0.0/8` für IP Multicasting, welcher für Datenströme mit einer Quelle und einer großen Anzahl von „Zuhörern“ gedacht ist. [17, 18] Letztlich 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.

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 Netzwerkes können sich direkt sehen. 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 1.6 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

```
[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 1.6: 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.

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.

Network Address Translation (NAT)

Um die Erschöpfung der im Internet direkt erreichbaren IP 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

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

Tabelle 1.3: Maximum Transfer Units (MTU) für gängige Netzwerktechniken. Aufgrund der MTU für X.25 wird oft 576 als Untergrenze für die MTU im Internet angesehen.

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.

Paketfragmentierung

Anfänglich 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 1.3 zeigt verschiedene MTUs im Überblick.

Da nun IP Pakete größer als die MTU sein können, werden alle größeren Pakete an der Quelle in kleinere Pakete aufgeteilt, bevor sie ins Netzwerk gelangen. 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.
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.

1.3 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 zwei Kategorien einteilen:

1. **Attacken über erlaubten Verbindungen zu Services**
   (a) Angriffe auf den Kommandokanal eines Services (FTP, SMTP/ESMTP, etc.)
   (b) datengesteuerte Angriffe („data driven attacks“)
   (c) Angriff von dritten Seite
   (d) falsche Authentifizierung von Clients
2. **Attacken, die den Verbindungsaufbau umgehen**
   (a) Hijacking
   (b) Paketschnüffler
   (c) Einschleusen und Veränderung von Daten
   (d) Replay-Angriff
   (e) Denial of Service

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, etc.

Die Angriffvektoren für Eindringlinge sind vielfältig. Eine Auswahl soll die Aufzählung bieten, jedoch sind das nicht alle Möglichkeiten, die ein kreativer Geist ausnutzen kann.

- **Authentifizierungssysteme**
  Remote Authentication Dial-In User Service[2] (RADIUS), Terminal Access Controller Access Control System[3] (TACACS+), Microsoft Windows Domain Controller oder ähnliche Systeme, die über verteilte Server Zugriff auf Netzwerkresourcen gewährleisten

[2] http://www.cisco.com/warp/public/cc/techno/tity/ipsq/prodlit/555_pp.htm
[3] http://www.cisco.com/warp/public/614/7.html

- **Datenbankserver**
  Injizieren von gefährlichen SQL Befehlen über Front Ends

- **Directory Services**
  Lightweight Directory Access Protocol[4] (LDAP), Sun Yellow Pages / Network Information Service[5] (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

- **File Sharing Protokolle**
  Server Message Block[6] (SMB), Network File System (NFS)

- **File Swapping Programme**
  Programme wie Kazaa, Morpheus, Limewire, AudioGalaxy, Gnutella, 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**
  Yahoo! Messenger, UNIX Talk, ICQ, Microsoft NetMeeting, Internet Relay Chat (IRC), etc.

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

- **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

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

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 (*Defence in Depth*, *mehrschichtige Verteidigung*).

[4] http://www.openldap.org/
[5] http://www.protocols.com/pbook/sun.htm
[6] http://samba.anu.edu.au/cifs/docs/what-is-smb.html

1.4 Schutzmechanismen für vernetzte Systeme

Aufgrund der Vielfalt der eingesetzten Applikationen und Protokollen 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.

- **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.

- **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.

1.4.1 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[7] PGP, GNU Privacy Guard[8] 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 [19, 20] 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.

[7] http://www.pgpI.org/
[8] http://www.gnupg.org/

  - *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)

- **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). [21, 22] 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). [19]

- **Layer 2 Tunneling Protocol (L2TP)**
  L2TP ist eine andere Methode, um Daten in PPP Paketen eingekapselt übertragen zu können. [23] 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. Als Beispiel sei die VTUN[9] Package genannt, die Tunnel über TCP, UDP und Ethernet sowohl verschlüsselt als auch unverschlüsselt.

[9] http://vtun.sourceforge.net/

Kapitel 2

Arten von Paketfiltern

2.1 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 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 Produktes löst noch keine Probleme.
Ausgangspunkt ist ein Paketfilter auf einem Linux System. Die Konfiguration des Filters und des Routings geschieht über Shell Skripte. Schwerpunkt wird der Einsatz des 2.4.x Kerns sein, der mit zwei verschiedenen Paketfiltern ausgestattet ist.

2.2 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. Zulässige Filter 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 (Email Verkehr) unseres Mailservers senden.
- Ein bestimmter Rechner darf eine Verbindung auf Port 22/TCP (Secure Shell) 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.

- ipchains –insert input –interface eth0 –protocol tcp –destination 62.116.64.105 –destination-port 23 –jump DENY

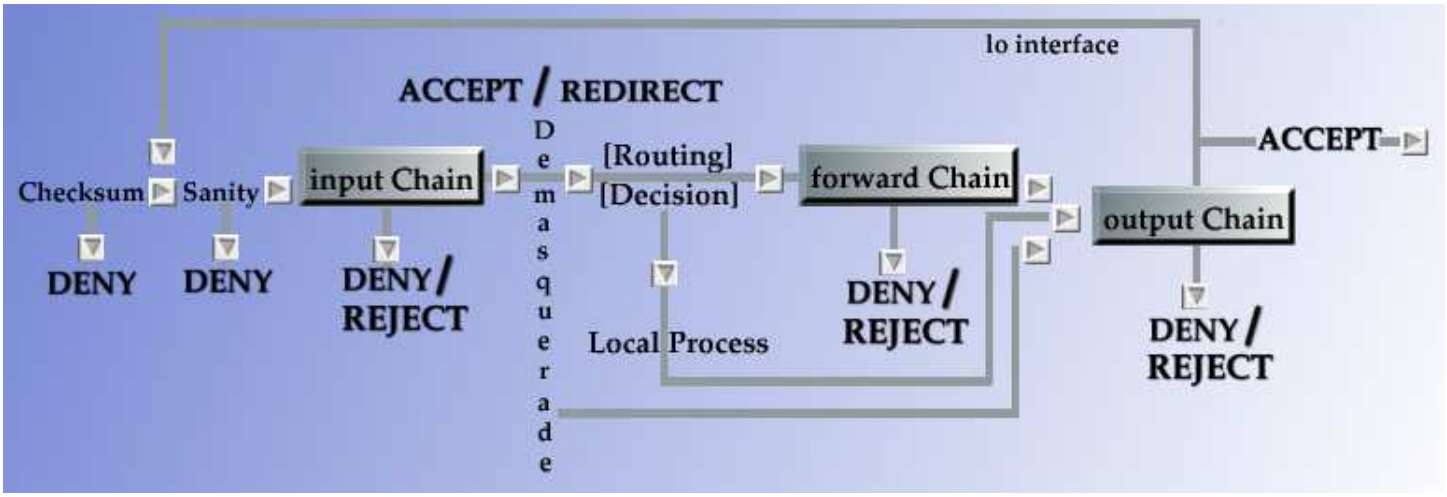

Abbildung 2.1: Das Diagramm zeigt den Fluß der Pakete durch eine Linux Ipchains Firewall. `input` wird von eingehenden Paketen passiert, `forward` betrifft weitergeleitete Pakete und durch `output` müssen alle ausgehenden Pakete.

- ipchains –insert forward –protocol tcp –destination $MAILSERVER –destination-port 25 –jump ACCEPT
- ipchains –insert forward –protocol tcp –source $WHITEHAT –destination $DBSERVER –destination-port 22 –jump ACCEPT
- ipchains –insert forward –protocol tcp –source 10.10.10.0/24 –destination $DNS –destination-port 53 –jump ACCEPT
  ipchains –insert forward –protocol udp –source 10.10.10.0/24 –destination $DNS –destination-port 53 –jump ACCEPT

Die Optionen des Kommandos wurden dabei ausgeschrieben. –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 2.1 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.

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 weietere 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.

```
[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 2.2: 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.

- **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) 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). Auf reinen Routern mit wenig Resourcen kann man so ein statisches Filtern hinzufügen.

2.3 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 2.2 zeigt die Zustandsinformationen einer Linux Netfilter Firewall.

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, ESTABLISHED, RELATED, INVALID**
  NEW bezeichnet Pakete, die bisher noch mit keiner Verbindung assoziiert sind. ESTABLISHED bezeichnet Pakete einer Verbindung, die schon Pakete in beide Richtungen detektiert hat. RELATED bezeichnet Pakete, die eine neue Verbindung starten, obwohl sie mit einer bestehenden in Zusammenhang stehen.

- **unclean**
  Dieser Filter prüft Pakete auf bestimmte Abweichungen in IP Paketen (Checksummen, überlange IP Optionen, Nullports, zu kleine Header, etc.).

Diese Filter sind eine große Hilfe und erleichtern das Erstellen von Kriterien 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.

2.4 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 auf Port 25 ein SMTP Server lauscht.
Der Nachteil solcher Filtersysteme ist die erhöhte 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 für andere Protokolle.

Kapitel 3

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

3.1 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. 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.
- **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 3.1 zeigt dieses Verhalten schematisch. [24] Damit vermeidet man redundante Regeln und vereinfacht die Notation der Filter. Dies ist ein wichtiger Punkt für die Übersichtlichkeit.

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 kein Filter anspricht. Die Default Policy und die Aktionen, die bei zutreffen jeweils eines Filterkriteriums ausgeführt werden sollen, sind vielfältig und abhängig von der Table. Einige mögliche Aktionen sind:

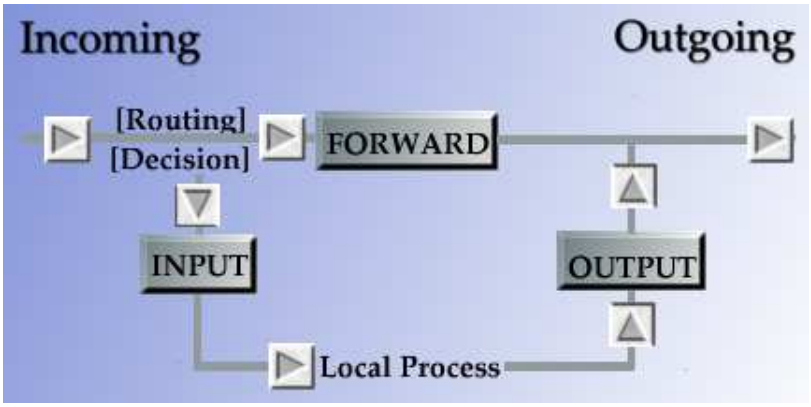

Abbildung 3.1: Das Diagramm zeigt den Fluß der Pakete durch eine Linux Netfilter Firewall. `INPUT` wird von eingehenden Paketen passiert, die für die Firewall selbst bestimmt sind. `FORWARD` betrifft ausschließlich weitergeleitete Pakete und durch `OUPUT` müssen alle ausgehenden Pakete, die die Firewall selbst generiert hat. Durch dieses Design wird vermieden, daß passierende Pakete mehr als eine dieser Chains passieren müssen.

- `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). Optional kann eine spezielle ICMP Meldung (`icmp-net-unreachable`, `icmp-host-unreachable`, `icmp-port-unreachable`, `icmp-proto-unreachable`, `icmp-net-prohibited` oder `icmp-host-prohibited`) generiert werden.

- `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.

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

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

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
```

3.2 Weitere Filterkriterien

Die Kriterien, nach denen der Linux Netfilter Pakete filtern kann, sind sehr vielfältig.

- **IP-Adressen und Ports**
  z.B. für TCP und UDP
- **Protokolle**
  - TCP
  - UDP
  - ICMP
  - weitere Protokolle durch numerische Angabe
    Durchlassen von GRE[1] 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 MAC[2] 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**

[1] **G**eneric **R**outing **E**ncapsulation, Tunnelprotokoll von Cisco System
[2] MAC = **M**edia **A**ccess **C**ontrol

```
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).

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.

3.3 Vorgehensweise

Normalerweise befindet sich der Netfilter Paketfilter im Zustand alles 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. 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. 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 3.2 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`

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.

- `rc.routing`
  Dieses Skript enthält alle Befehle, die benötigt werden, um das Routing korrekt zu setzen. 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.

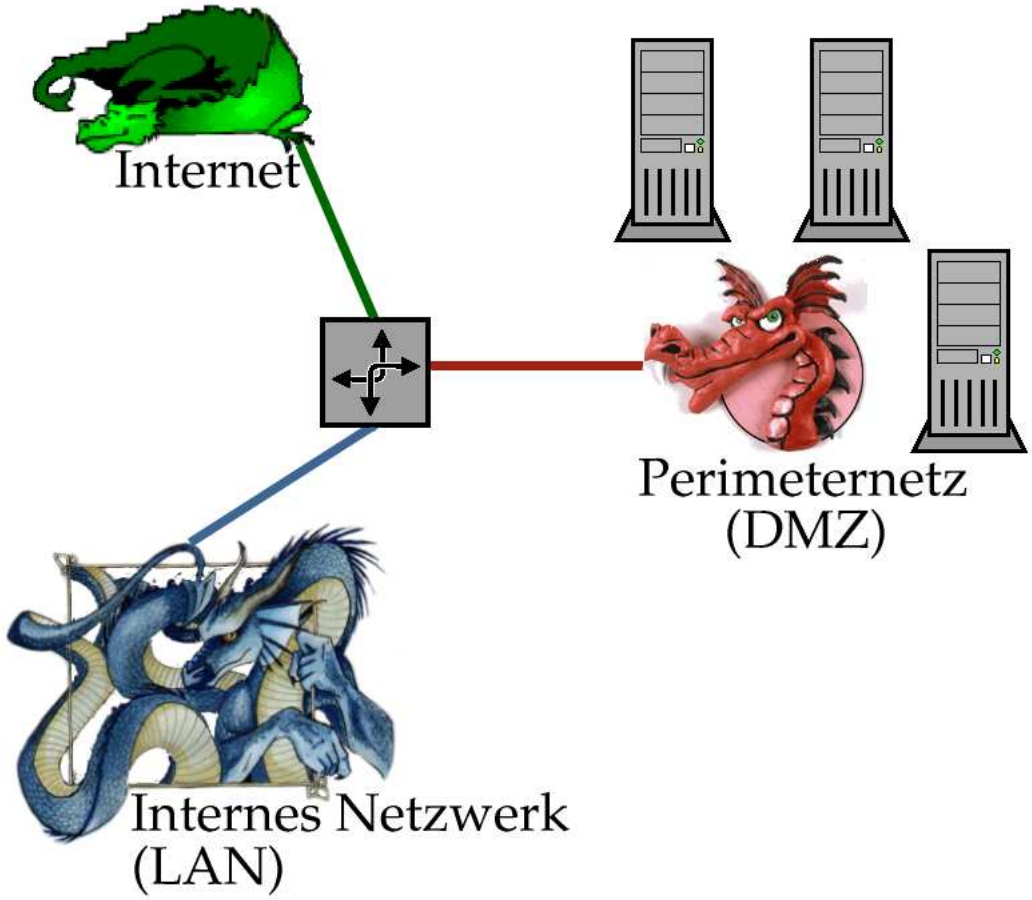

Abbildung 3.2: Hier ist die Übersicht über den Linux Paketfilter. Die Maschine ist mit 3 Netzwerkkarten an die jeweiligen Netze angeschlossen. `eth0` sei mit dem Internet, `eth1` mit dem Perimeternetzwerk und `eth2` mit dem lokalen Netz verbunden.

- `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

DYNA_PORTS=''1023:65534''
```

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. [25]
Als nächsten 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[3], wobei jede Maschine noch zusätzlich einen OpenSSH Server[4] für verschlüssel-

[3] http://www.postgresql.org/
[4] http://www.openssh.org/

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 3.1: 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). [19]

ten 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 3.1 gegeben.

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

```
$IPTABLES --insert FORWARD --protocol tcp \
          --source $EVERYWHERE --source-port $DYNA_PORTS \
          --destination $DMZ_NET --destination-port $SSH \
          --jump ACCEPT
$IPTABLES --insert FORWARD --protocol tcp ! --syn \
          --source $DMZ_NET --source-port $SSH \
          --destination $EVERYWHERE --destination-port $DYNA_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 --insert FORWARD --protocol tcp \
          --source $EVERYWHERE --source-port $DYNA_PORTS \
          --destination $DMZ_NET --destination-port $SSH \
          --match state --state NEW,ESTABLISHED \
          --jump ACCEPT
$IPTABLES --insert FORWARD --protocol tcp \
          --source $DMZ_NET --source-port $SSH \
          --destination $EVERYWHERE --destination-port $DYNA_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.
Der SSH Zugang auf die Paketfiltermaschine selbst wird durch die `INPUT` und `OUTPUT` Chains bestimmt.

```
$IPTABLES --insert INPUT --protocol tcp \
          --source $EVERYWHERE --source-port $DYNA_PORTS \
          --destination $EXT_IP --destination-port $SSH \
          --match state --state NEW,ESTABLISHED \
          --jump ACCEPT
$IPTABLES --insert OUTPUT --protocol tcp \
          --source $EXT_IP --source-port $SSH \
          --destination $EVERYWHERE --destination-port $DYNA_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. 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 3.3), oder der Client initiiert eine Verbindung zum FTP Server für die Datenübertragung (Abbildung 3.4).

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 3.2 ersichtlich.

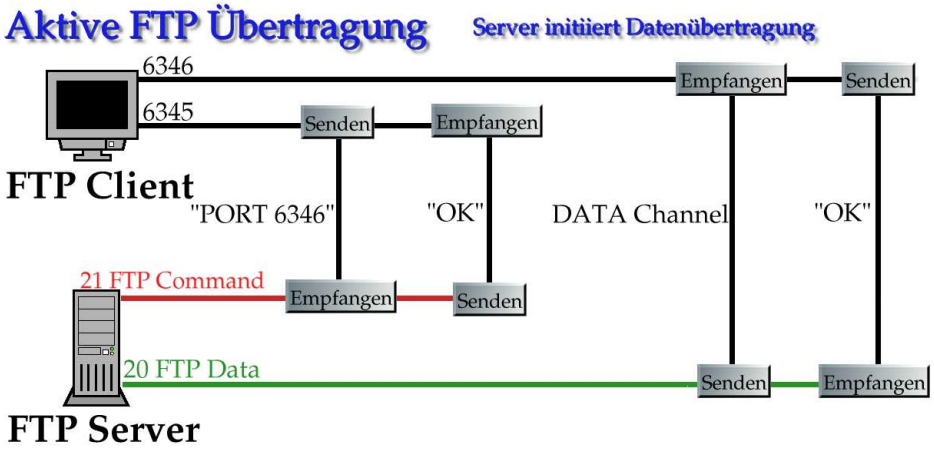

Abbildung 3.3: Bei aktiven FTP Transfers initiiert der Server die Datenverbindung von Port 20 zu einem beliebigen Port am Client. Um diesen Transfer zuzulassen, müßte die Firewall Verbindungen von externen Servern mit Quelle Port 20 zu den Clients im LAN durchlassen. Dies geht weder durch einen NAT Router noch durch eine Firewall mit zustandslosem Filtern.

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

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

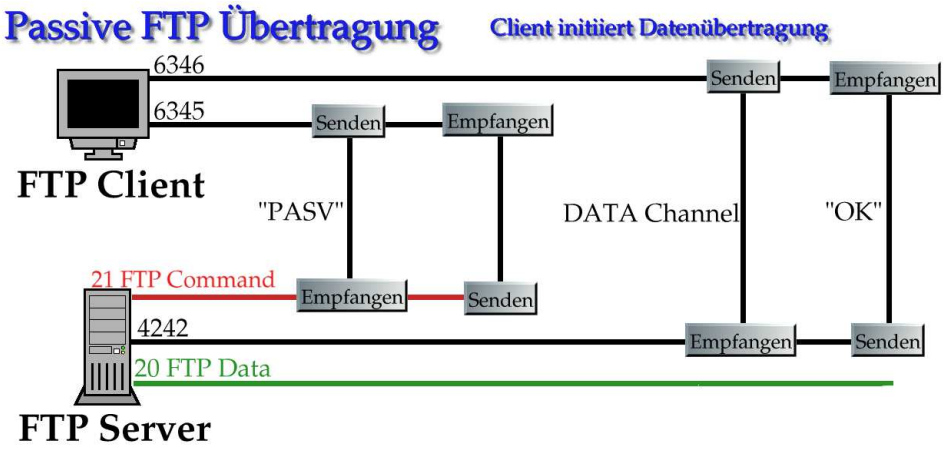

Abbildung 3.4: Bei passiven FTP Transfers initiiert der Client die Datenübertragung. Dies ist ohne Schwierigkeiten durch eine Firewall hinweg möglich, da der Ursprung der Verbindung im LAN liegt. Die Problematik des Filterns verlagert sich damit auf die Seite des FTP Servers. Auch hier sollte man serverseitige einen zustandgesteuerten Paketfilter verwenden, da man ansonsten einen viel zu großen Portbereich freigeben muß.

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

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	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 3.2: 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. [19]

- Zonentransfers werden immer über TCP transportiert.

Eine Übersicht der Filtereigenschaften ist in Tabelle 3.3 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[5] (Berkeley Internet Name Daemon) ab der Version 8.1 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 [6]).

Die einzelnen Filter sehen nun wie folgt aus.

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

[5] http://www.isc.org/products/BIND/

[6] Die `djbdns` DNS Package enthält jeweils eine eigene Software für eine bestimmte Teilaufgabe im DNS Service. [26]

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 3.3: 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. [19]

```
          --source $DNSSERVER --source-port $DNS \
          --destination $EVERYWHERE --destination-port $DYNA_PORTS \
          --match state --state ESTABLISHED --jump ACCEPT

# DNS Anfragen an andere DNS Server
$IPTABLES --insert FORWARD --protocol udp \
          --source $DNSSERVER --source-port $DYNA_PORTS \
          --destination $EVERYWHERE --destination-port $DNS \
          --match state --state NEW --jump ACCEPT
$IPTABLES --insert FORWARD --protocol udp \
          --source $EVERYWHERE --source-port $DNS \
          --destination $DNSSERVER --destination-port $DYNA_PORTS \
          --match state --state NEW --jump ACCEPT
$IPTABLES --insert FORWARD --protocol tcp \
          --source $DNSSERVER --source-port $DYNA_PORTS \
          --destination $EVERYWHERE --destination-port $DNS \
          --match state --state NEW,ESTABLISHED --jump ACCEPT
$IPTABLES --insert FORWARD --protocol tcp \
          --source $EVERYWHERE --source-port $DNS \
          --destination $DNSSERVER --destination-port $DYNA_PORTS \
          --match state --state ESTABLISHED --jump ACCEPT

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

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 --insert FORWARD \
             --source $EVERYWHERE --source-port $DYNA_PORTS \
             --destination $3 --protocol $1 --destination-port $2 \
             --match state --state NEW,ESTABLISHED \
             --jump ACCEPT
   $IPTABLES --insert FORWARD \
             --source $3 --protocol $1 --source-port $2 \
             --destination $EVERYWHERE --destination-port $DYNA_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.

3.3.1 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. [29] 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 [29] 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.

3.3.2 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.

```
$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 prüfen.

Kapitel 4

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. Es gibt mehrere Möglichkeiten dies zu tun.

4.1 Beobachten mit Sniffern

Tools wie `tcpdump`[1] [6] oder Ethereal [28] 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.[2]

```
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
```

Ethereal hat auch ein eigenes Tool für die Konsole, mit dem sich Netzwerkverkehr auffangen und darstellen läßt. Es nennt sich `tethereal`. Die Syntax ist der von `tcpdump` sehr ähnlich. Zusätzlich versteht `tethereal` 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.

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

[2] Es gibt für `ipchains` dazu ein Tool namens Mason (http://www.stearns.org/mason/.

```
[lynx@luchs lynx]$ tethereal -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.

4.2 Generieren von Netzwerkverkehr

4.2.1 hping

`hping`[3] 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
```

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.

4.2.2 isic

ISIC[4] (*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 zufälli-

[3] http://www.hping.orgff
[4] http://www.packetfactory.net/Projects/ISICff

gen 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.

```
/usr/local/bin/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%
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.

```
/usr/local/bin/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. Tests mit den ISIC Tools sollten daher immer gut kontrolliert werden.

4.2.3 nmap

`nmap`[5] 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
  - 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.

[5] http://www.insecure.org/nmapff

- 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[6] 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
```

[6]Identification Protocol, RFC1413, http://dungeon.luchs.at/RFC/rfc1413.txtff

```
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=)
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.

4.3 Belastungstests und Auditing

4.3.1 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`[7]
  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.

- `ntop`[8]
  ist ein sehr leistungsfähiger Netzwerkmonitor, der Raten und Netzaktivität wiedergibt.

[7] http://dast.nlanr.net/Projects/Iperf/
[8] http://www.ntop.org/

- `netperf`[9]
  ist ein Tool von Hewlett Packard, mit dem man den Datendurchsatz testen kann.

Die üblichen WAN-Geschwindigkeiten an Standleitung sind derzeit geringer als 100 Mbit/s, deswegen sind interne Paketfilter (z.B. Filter an der Grenze zwischen Perimeternetzwerk und LAN) höheren Belastungen ausgesetzt.

4.3.2 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[10]. Die Vorzüge von Nessus sind:

- Client/Server Architektur
  $\longrightarrow$ Client und Server kommunizieren verschlüsselt
  $\longrightarrow$ 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
  $\longrightarrow$ abhängig von der Performance des Nessus Servers
  $\longrightarrow$ konfigurierbar
- Service Erkennung
  Nessus prüft Service bei Connect
  $\longrightarrow$ Entdecken von Services auf Nicht-Standard-Ports
  $\longrightarrow$ entsprechendes Wiederholen von Tests
- Knowledge Base beim Scannen
  Plugins teilen die ermittelten Informationen über Hosts
  $\longrightarrow$ 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
- Scannen von Cron Jobs aus
  - Nessus Client läßt sich im Batch Modus einsetzen
  - automatischer Ablauf von periodischen Checks

[9] http://www.netperf.org/netperf/NetperfPage.html
[10] http://www.nessus.org/

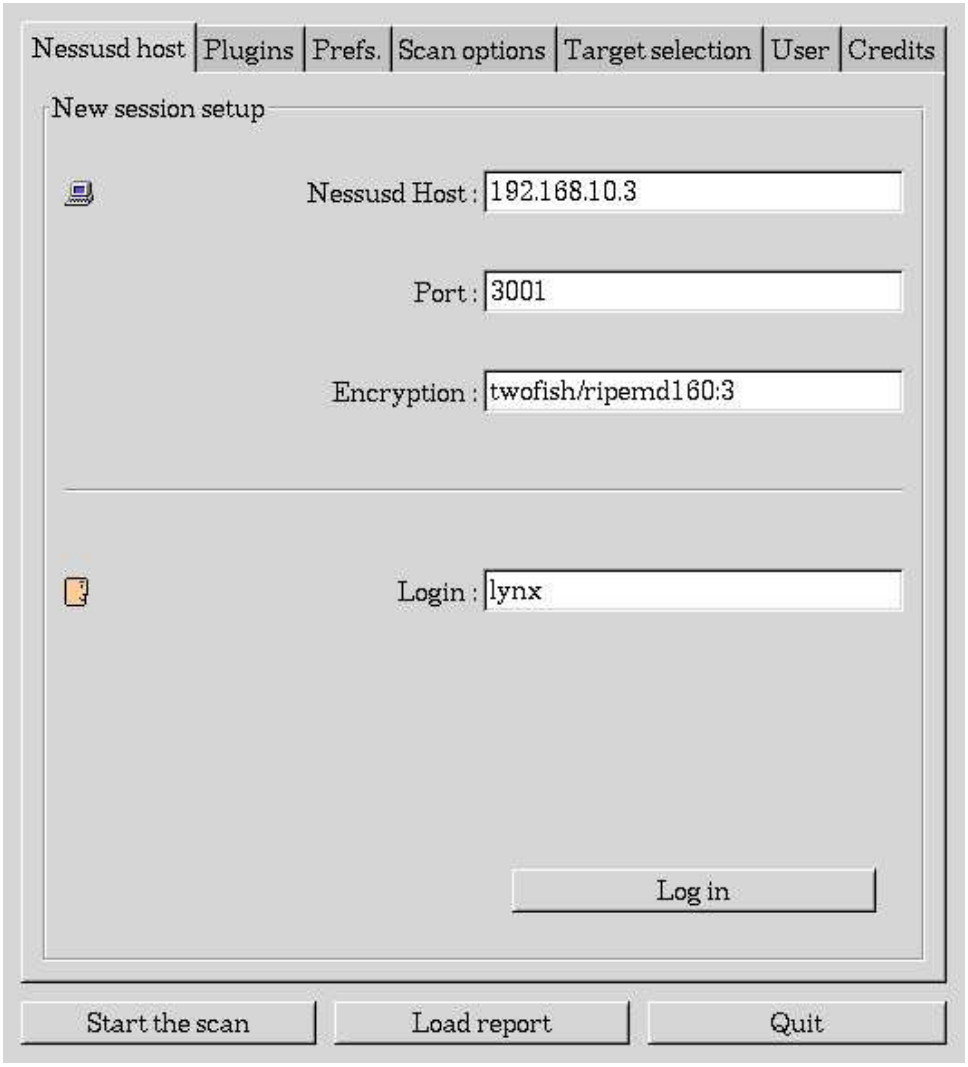

Abbildung 4.1: Login mit dem Nessus Client. Nessus verfügt über eine eigene Paßwortdatenbank zur Authorisierung. Die Kommunikation zwischen Server und Client ist ebenso verschlüsselt.

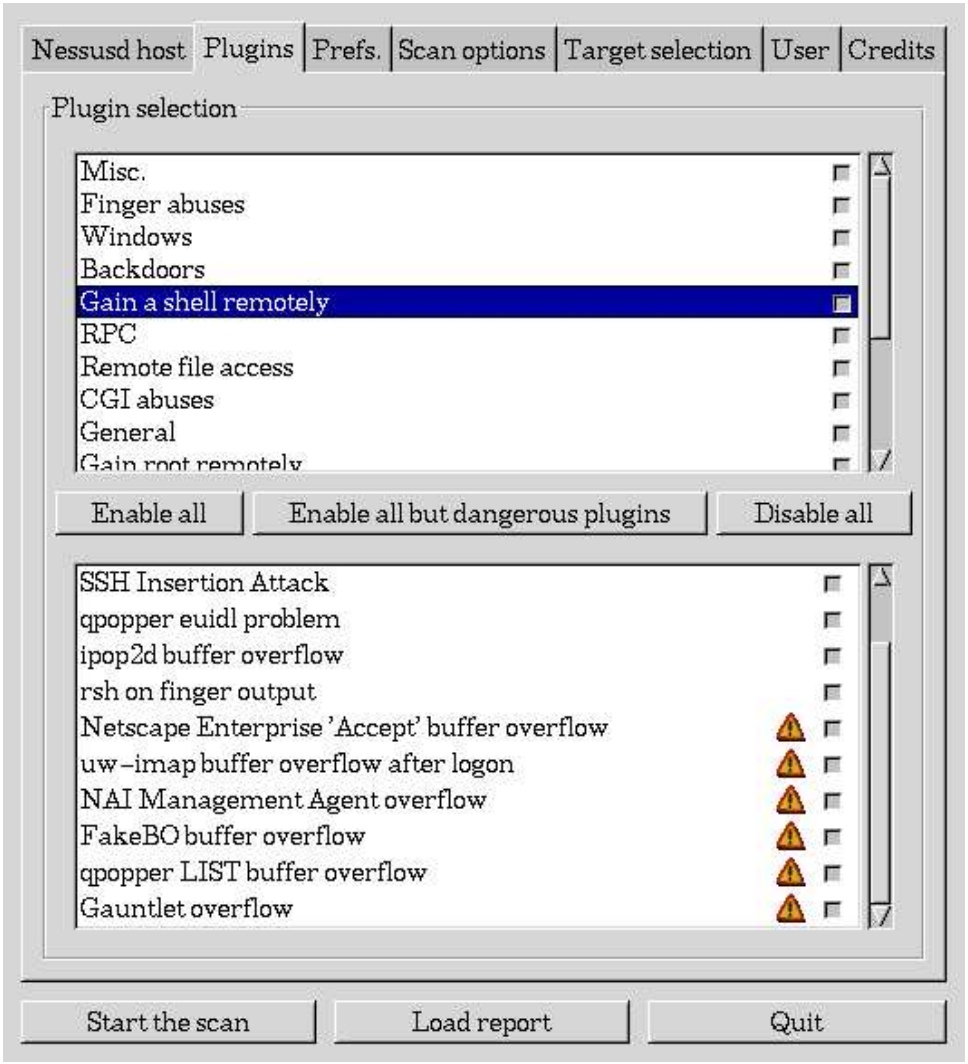

Abbildung 4.2: Auswählen der zu verwendenen Plugins für den Scan. **Wichtig:** Manche Module können Systeme zum Absturz bringen oder Netzwerk-Equipment wie Router oder Print-HUBs lahmlegen.

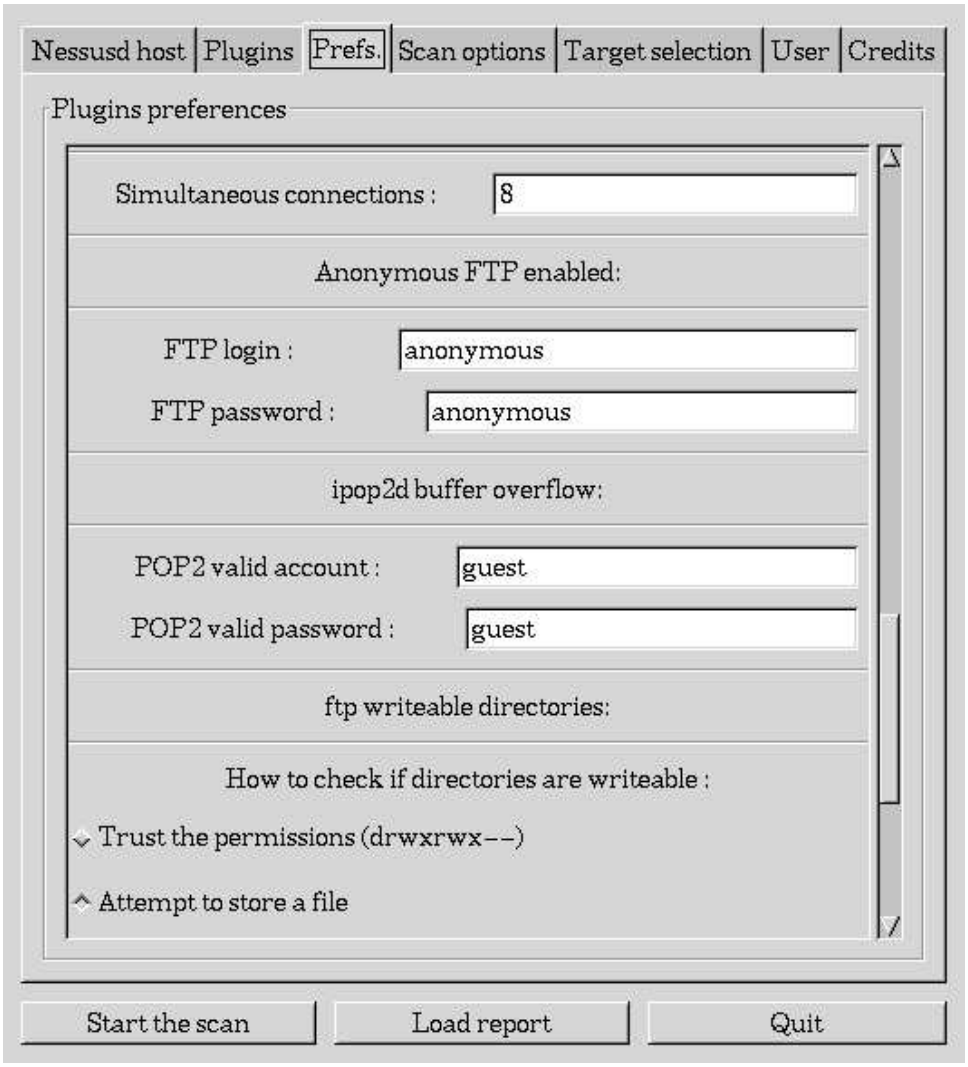

Abbildung 4.3: Einstellen der Voreinstellungen für die verwendeten Plugin-Module.

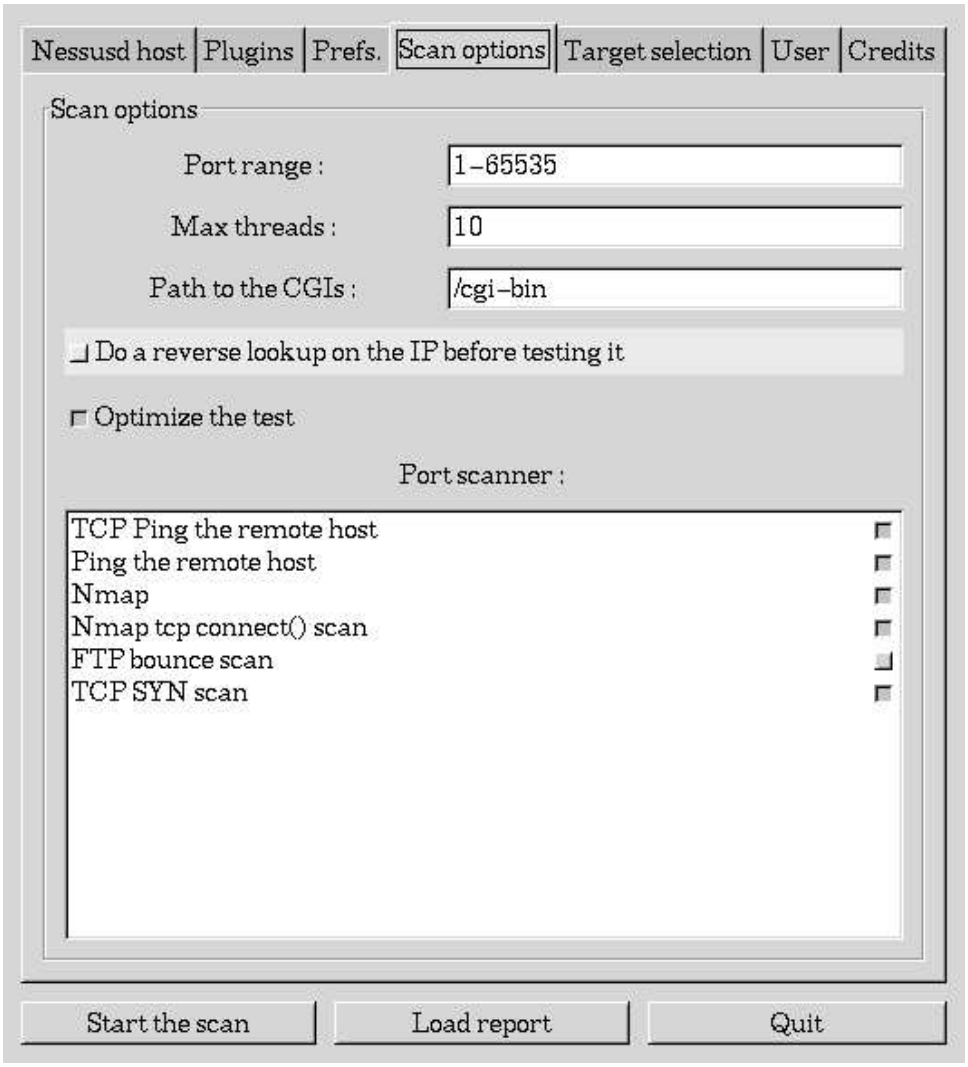

Abbildung 4.4: Optionen für den bevorstehenden Scan.

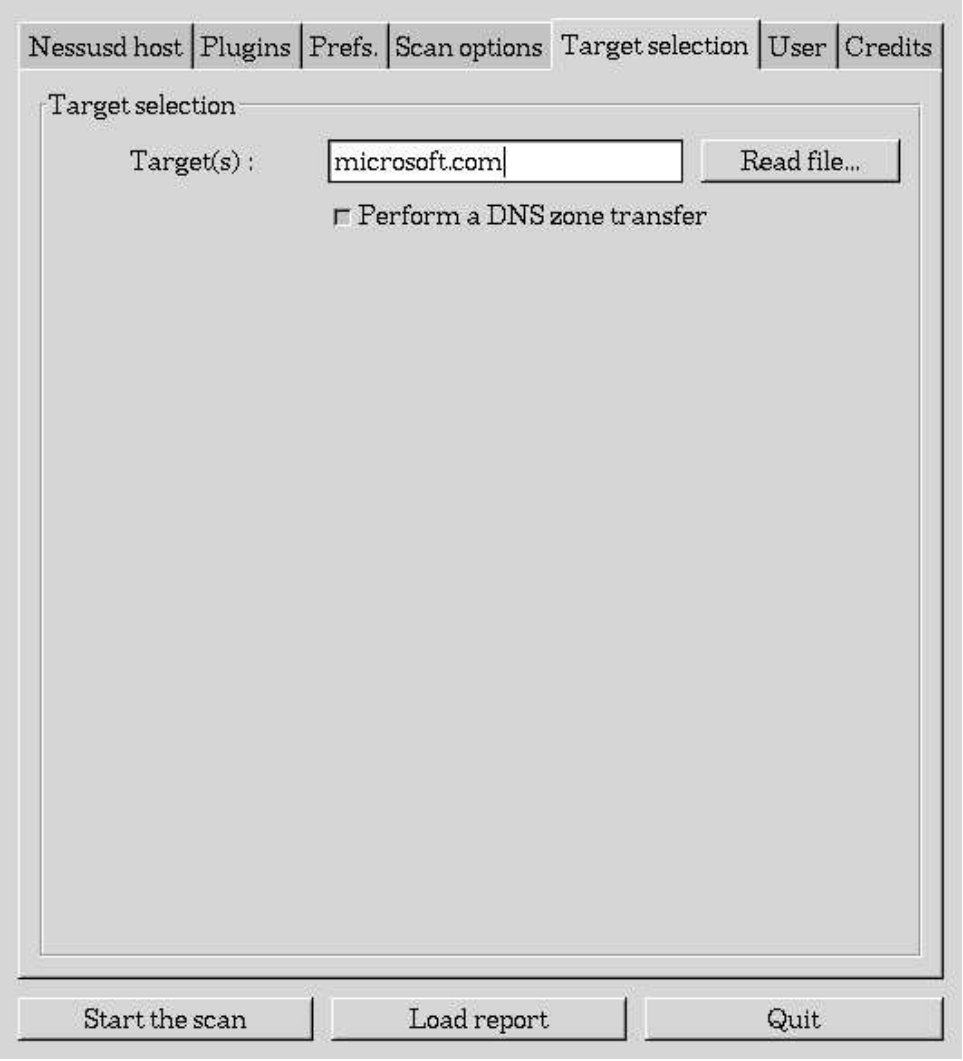

Abbildung 4.5: Auswahl der Ziele für den bevorstehenden Scan. Es können auch vorbereitete Textfiles und DNS Zone Transfers benutzt werden.

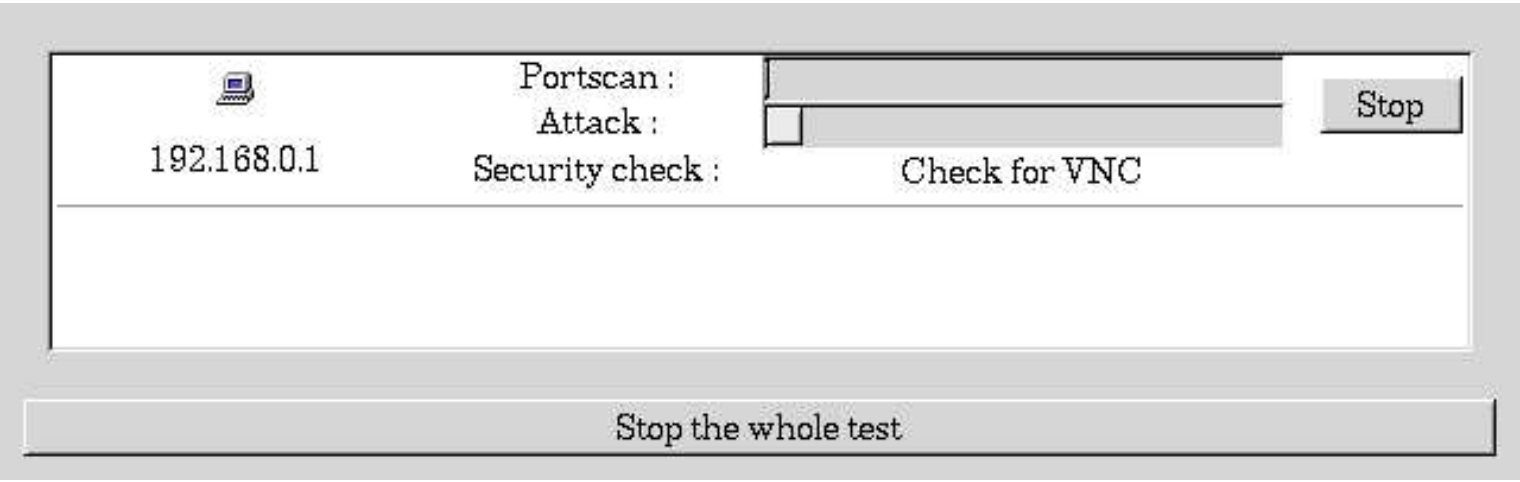

Abbildung 4.6: Nessus beim Scannen des Ziels. Der Nessus Server kann eine konfigurierbare Anzahl von Prozessen auf mehrere Ziele gleichzeitig loslassen.

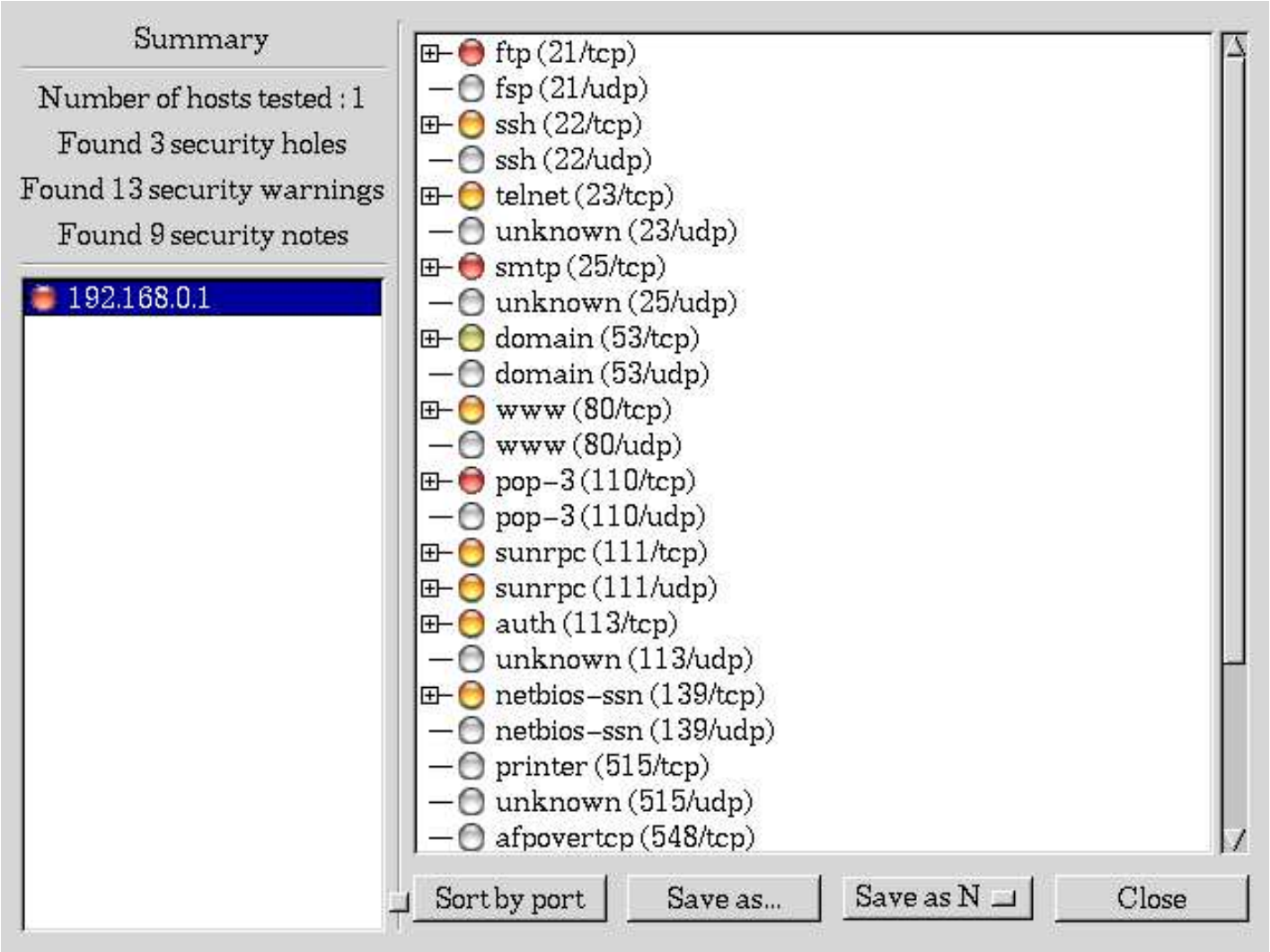

Abbildung 4.7: Nessus stellt die Ergebnisse eines Scan-Durchlaufs dar. Man kann die Ergebnisse in NSR, HTML mit Grafiken, HTML, ASCII Text und LaTeXabspeichern. Jede gefundene Schwachstelle wird bewertet und mit einem Link zu einer Datenbank mit genauerer Beschreibung versehen.

Anhang A

Beispiele aus dem Netz

A.1 Session einer TCP Verbindung

Der folgende Dump einer TCP Session wurde mit dem Netzwerkanalysator Ethereal[1] durchgeführt. 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] http://www.ethereal.com/

```
        .... ..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
    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)
```

A.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. [27]

```
#!/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.

A.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_forward` 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

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 $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 beforehand
#
# 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
```

A.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

A.4.1 Sichern der Maschine

- minimale Installation des Systems
- Beheben aller bekannten Systemfehler
- Verwenden bzw. Aufstellen einer Checkliste
- Sichern der System-Logs

A.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

A.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

A.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

A.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

A.5 nmap Proben

A.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
```

A.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
```

A.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
```

A.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
```

A.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.

Anhang B

Protokolle und Ports

B.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 using Kryptonet key management	[Oberg]
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.

B.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/

Anhang C

GNU Free Documentation License

Version 1.1, March 2000

Preamble

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.
This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.
We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

C.1 Applicability and Definitions

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".
A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

C.2 Verbatim Copying

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

C.3 Copying in Quantity

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.
If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.
If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.
It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

C.4 Modifications

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
- State on the Title page the name of the publisher of the Modified Version, as the publisher.
- Preserve all the copyright notices of the Document.

- Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- Include an unaltered copy of this License.
- Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.
- Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.
You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties – for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.
You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or

through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.
The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

C.5 Combining Documents

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.
The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.
In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

C.6 Collections of Documents

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.
You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

C.7 Aggregation With Independent Works

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.
If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts

may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

C.8 Translation

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

C.9 Termination

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

C.10 Future Revisions of This License

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyl
Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License ör any later versionäpplies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

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 Ethernetpaketen, http://www.tcpdump.org/

[7] J. Postel, RFC792, *Internet Control Message Protocol*, *DARPA Internet Program*, USC/Information Sciences Institute, September 1981.

[8] R. Braden, RFC1122, *Requirements for Internet Hosts - Communication Layers*, USC/Information Sciences Institute, Oktober 1989.

[9] F. Baker, RFC1812, *Requirements for IP Version 4 Routers*, Cisco Systems, Juni 1995.

[10] Craig Hunt, *TCP/IP Network Administration*, 2nd Edition, O'Reilly & Associates, Inc., Dezember 1997.

[11] W. Richard Stevens, *TCP/IP Illustrated, Volume 1, The Protocols*, Addison-Wesley, 1994.

[12] Gary R. Wright & W. Richard Stevens, *TCP/IP Illustrated, Volume 2, The Implementation*, Addison-Wesley, 1995.

[13] Adolfo Rodriguez et al., *TCP/IP Tutorial and Technical Overview*, IBM Redbooks, ISBN 0738421650, IBM Form Number GG24-3376-06, August 2001.

[14] 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.

[15] Y. Rekhter, RFC1817, *CIDR and Classful Routing*, Cisco Systems, August 1995.

[16] 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.

[17] S. Deering, RFC1112, *Host Extensions for IP Multicasting*, Stanford University, August 1989.

[18] R. Finlayson, RFC2588, *IP Multicast and Firewalls*, `live.com`, Mai 1999.

[19] Elizabeth D. Zwicky, Simon Cooper & D. Brent Chapman, *Building Internet Firewalls*, 2nd Edition, O'Reilly & Associates, Inc., 2000.

[20] FreeS/WAN Project, IPsec RFC collection, http://www.freeswan.org/freeswan_trees/freeswan-1.91/doc/rfc.html

[21] S. Hanks, D. Farinacci, P. Traina, *Generic Routing Encapsulation (GRE)*, Cisco Systems, Oktober 1994.

[22] S. Hanks, D. Farinacci, P. Traina, *Generic Routing Encapsulation over IPv4 networks*, Cisco Systems, Oktober 1994.

[23] 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.

[24] R. Russell, *Linux 2.4 Packet Filtering HOWTO*, http://netfilter.samba.org/unreliable-guides/

[25] K. Seifried, *Linux Administrator's Security Guide*, http://www.linuxdoc.org/LDP/lasg/

[26] D. J. Bernstein, *djbdns* DNS Server Software und Tools, http://cr.yp.to/

[27] 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

[28] G. Combs, *Ethereal Network Protocol Analyzer*, http://www.ethereal.com/

[29] O. Arkin, *ICMP Usage In Scanning*, http://www.sys-security.com/html/papers.html Sys-Security Group, Juni 2001.

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