Im Zusammenhang mit der stärkeren Vernetzung von Computern und der besseren Anbindung an das Internet spielt das Thema Sicherheit eine immer größere Rolle. Ein Rechner, der über Modem eine Stunde pro Tag ans Internet angeschlossen ist, wird einen Angreifer wenig interessieren. Wenn der gleiche Rechner aber per DSL nahezu 24 Stunden am Tag im Netz ist, dann wird er schon deutlich interessanter.
Selbst wenn der Rechner selber keine interessanten Daten bietet, so läßt er sich doch als Zombie für Angriffe auf andere Rechner nutzen. Richtig interessant für einen Angriff sind die Server in Computernetzen, da sie in der Regel über entsprechende Ressourcen verfügen.

Leider ist Sicherheit als Zustand im Zusammenhang mit Computern unmöglich. Ständig werden neue Fehler in den Protokollen und der Software entdeckt. Deshalb müssen alle Komponenten ständig aktualisiert werden, um bekannte Probleme zu beseitigen. Leider verlassen sich gerade die Besitzer von Computern mit bequemen grafischen Benutzeroberflächen auf die Versprechungen der Hersteller und unterlassen die notwendigen Updates.

Die Angriffsmöglichkeiten lassen sich folgendermaßen unterteilen.
- Benutzer
- Verbindung
- Rechner

1. Benutzer

Der größte Schwachpunkt in jedem System ist der Benutzer. Er muß sich meist einen Benutzernamen und ein Passwort merken. Die meisten Benutzer sind daher geneigt sehr einfache oder gar triviale Passworte zu verwenden, weil diese dann einfach zu merken sind. Wenn man die Benutzer zwingt bessere Passworte zu verwenden, dann werden diese oft notiert und diese Notizen nicht sicher verwahrt, z.B. unter der Tastatur. Der sichere Umgang mit Passworten ist kein alleiniges Computerthema, sondern tritt auch z.B. im Zusammenhang mit Kreditkarten auf.
Neben diesen menschlichen Schwächen gibt es auch eine Vielzahl von technischen Schwächen. Viele Systeme benötigen Passworte und legen diese „verschlüsselt“ ab. Oft ist diese Verschlüsselung zu schwach, dass sie leicht zu brechen ist. Böse Beispiele sind hier Windows9x und Netscape Messenger.
Generell ist jedes System unsicher, dass Passwörter auf einem Client-Rechner speichert.

1.1 Schwache Passwörter

Passwörter in Unix-Systemen können normalerweise noch nicht einmal die Systemverwalter ermitteln, weil die Passwörter nur verschlüsselt abgelegt sind. Die zugehörige Verschlüsselungsfunktion ist eine Einweg-Funktion, die keine Entschlüsseln vorsieht. Meldet sich ein Benutzer am System an, dann verschlüsselt Unix dieses Passwort und vergleicht es mit der in der Shadow-Datei abgelegten Version. Eine Entschlüsselung ist also nicht notwendig.
Es gibt trotzdem theoretisch ein einfaches Verfahren, die Passwörter zu knacken, man probiert einfach alle Möglichkeiten durch. Der Aufwand hierfür hängt sehr von der Passwortlänge ab, wie die folgende Tabelle zeigt. Diese Tabelle geht davon aus, dass 62 verschiedene Zeichen zur Verfügung stehen, die 26 lateinischen Buchstaben einmal klein, einmal gross und die zehn Ziffern. Weiter geht die Berechnung davon aus, dass man 10 Millionen Kennwörter pro Sekunde überprüfen kann.

Passwortlänge	*Zahl der möglichen Passwörter*	*Zeitbedarf zum Knacken*
1	62	keiner
2	3844	keiner
3	238.328	keiner
4	14.776.336	1,4 Sekunden
5	916.132.832	1,5 Minuten
6	56.800.235.584	1,5 Stunden
7	3.521.614.606.208	4 Tage
8	218.340.105.584.896	8 Monate
9	13.537.086.546.263.552	43 Jahre
10	839.299.365.868.340.224	2660 Jahre

Tabelle 1: Sicherheit in Abhängigkeit von der Passwortlänge

Die Sicherheit eines Passwortes ist aber nicht nur von seiner Länge, sondern auch vom verwendeten Zeichensatz stark abhängig. Die folgende Tabelle geht von einer einheitlichen Passwortlänge von 8 Zeichen aus, wobei wieder 10 Millionen Passwörter pro Sekunde geprüft werden.

Zeichensatz	*Zeichenzahl*	*Zahl der möglichen Passwörter*	*Zeitbedarf zum Knacken*
8-Bit ASCII	256	18.446.744.073.709.551.616	58.500 Jahre
7-Bit ASCII	128	72.057.594.037.927.936	228 Jahre
Buchstaben und Ziffern	62	218.340.105.584.896	8 Monate
Nur Buchstaben	52	53.459.728.531.456	62 Tage
Nur Kleinbuchstaben	26	208.827.064.576	6 Stunden
Wörter aus Wörterbuch	-	ca. 250.000	keiner

Tabelle 2: Sicherheit in Abhängigkeit vom Zeichensatz bei jeweils 8 Zeichen

Da viele Benutzer Passwörter mit deutlich weniger als acht Zeichen benutzen, gibt es eine durchaus realistische Chance, diese Passworte zu knacken. Die Chance erhöht sich noch dadurch, dass man eigentlich nicht alle Kombinationen durchprobieren muss. Viele Leute benutzen Namen, Telefonnummern oder ähnliches, einfach weil diese leichter zu merken sind.
Selbst bei einer Passwortlänge von acht Zeichen kann man daher in wenigen Minuten zum Erfolg kommen, wenn man ein Wörterbuch als Grundlage für Ihre Knackversuche nimmt.
Man kann damit zwar nicht die Passwörter aller Benutzer knacken, aber 50% innerhalb von wenigen Minuten sind ein durchaus realistischer Wert.

Hinweis: Schon ein einzelner geknackter Zugang ist ein Sicherheitsrisiko. Wer erst einmal Zugang zum System hat, kann dort nach weiteren Schwachpunkten suchen.

Man sollte daher regelmäßig versuchen, die Passwörter Ihrer Benutzer zu knacken, um wenigsten die unsichersten Kandidaten zu ermahnen.
Beim Knacken und beim Ermahnen der Benutzer kann das Programm *john* helfen, dass bei SuSE im Paket *john* der Serie *sec* zu finden ist.
Nach der Installation liegt das Programm unter */usr/sbin/john* und seine Komponenten unter */var/lib/john/*.
Das Programm kann mit einem Wörterbuch arbeiten, es liefert auch eine englische

Version mit. Man müßten hier erst ein deutsches Wörterbuch erstellen. Hinweise dazu finden sich im Verzeichnis */usr/share/doc/packages/john/*.
Dieser Aufwand ist eigentlich unnötig, meist langt es sogar, mit den Daten in den Benutzerdateien zu arbeiten. Damit kann man die Passwörter knacken, die aus Namen oder Variationen davon bestehen.
Zuerst wechselt man in das Verzeichnis /var/lib/john/.

```
cd  /var/lib/john
```

Nun soll john aus passwd und shadow eine einheitliche Datei montieren, im Beispiel heißt sie passwd.john:

```
unshadow /etc/passwd /etc/shadow  >  passwd.john
```

Mit den Daten aus dieser Datei läßt man john nun arbeiten, es ist erstaunlich, wieviele Passwörter er so ermittelt.

```
john -single passwd.john
```

Mit diesem Befehl nutzt john nur die Benutzerdatenbank als Grundlage, keines der zusätzlich verfügbaren Wörterbücher.
Wenn bereits viele Benutzer vorhanden sind, dann dauert das Knacken schon eine Weile. Wenn man den Fortschritt kontrollieren will, drückt man einmal die Leertaste, worauf john den aktuellen Stand anzeigt.

```
Loaded 1037 passwords with 426 different salts (Standard DES [24/32 4K])
Burak            (bs1002)
laura            (lc1001)
sandra           (kj1002)
laura            (lt1002)
christi          (sw1002)
gast0            (gast)
ahmad-fa         (ak1005)
ann-kath         (ag1005)
wolf-die         (wm1004)
walter           (ja1001)
guesses: 10  time: 0:00:00:05 71%  c/s: 370569  trying: &tc3001& - *j5c*
```

Hier hat john nach knapp 5 Sekunden bereits 10 von etwa 1000 Passwörtern geknackt. Bei dem Datenbestand aus dem Beispiel hatte john nach knapp 2 Minuten bereits mehr als 70 Passwörter geknackt und das im einfachsten Modus.
Man kann john übrigens jederzeit unterbrechen, bei einem Neustart setzt er seine Arbeit an der gleichen Stelle fort. Die bereit geknackten Passwörter hält er in der Datei *john.pot* fest. Falls man erneut alle Passwörter testen will, muss man diese Datei vorher löschen.
Wenn john mit der Arbeit fertig ist, kann er auch eine Mail an alle Benutzer schicken, deren Passwörter er knacken konnte. Dazu finden man im Verzeichnis ein Programm mailer, das man zuerst mit

```
chmod u+x mailer
```

ausführbar machen und dann folgendermaßen aufrufen:

```
./mailer passwd.john
```

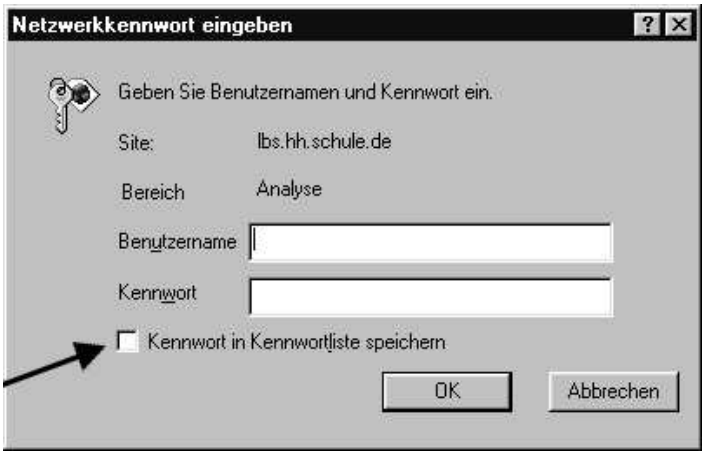
Damit ist dann jeder der nachlässigen Benutzer verwarnt.
Den Text der Mail an die Benutzer kann man in dem Perl-Programm mailer relativ leicht ändern. Im Original handelt es sich um einen englischen Text. Wer das möchte, der kann den Text übersetzen.

```
#!/bin/bash
#
# This file is part of John the Ripper password cracker,
# Copyright (c) 1996-98 by Solar Designer
#

if [ $# -ne 1 ]; then
        echo "Usage: $0 PASSWORD-FILE"
        exit 0
fi

# There's no need to mail users with these shells
SHELLS=-,/bin/false,/dev/null,/bin/sync

# Look for John in the same directory with this script
DIR="`echo "$0" | sed 's,/[^/]*$,,'`"

# Let's start
$DIR/john -show "$1" -shells:$SHELLS | sed -n 's/:.*//p' |
(
        SENT=0

        while read LOGIN; do
                echo Sending mail to "$LOGIN"...
# You'll probably want to edit the message below
                mail -s 'Unsicheres Passwort' "$LOGIN" << EOF
Hallo!

Das Passwort für den Account "$LOGIN" ist unsicher. Bitte umgehend
ändern, sonst mache ich das ;-)
Hinweise zur Auswahl eines besseren Passwortes finden sich unter
http://server/passwort.htm im Intranet.

Gruss,
        Password Checking Robot
        im Auftrag von U. Debacher
EOF

                SENT=$(($SENT+1))
        done

        echo $SENT messages sent
)
```

Wer sich ausführlicher mit der Dokumentation von john beschäftigt, der wird noch mehr Möglichkeiten finden, um weitere Passwörter zu knacken. Eventuell veranlasst einen die Erfahrung ja sogar dazu, die eigenen Passwörter zu ändern.
Man muss sich und auch seinen Benutzern immer wieder klar machen, dass Sicherheit kein Zustand ist, sondern ein anstrengender Prozess. Ein Baustein zu diesem Prozess ist u.a. die Wahl geeigneter Passwörter.

1.2. Passwörter auf Client-Rechnern

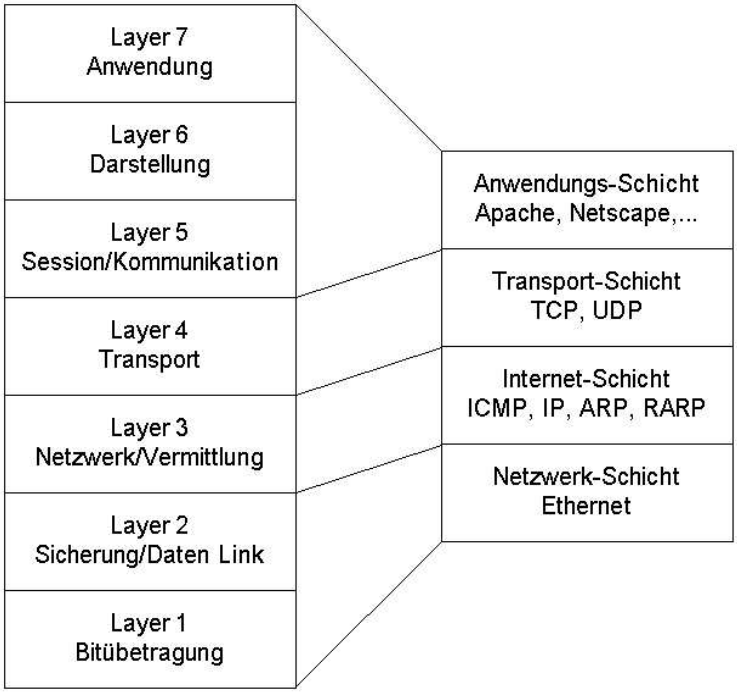
Viele Programme benötigen für Ihre Arbeit einen Benutzernamen und ein Passwort. Jeder E-Mail Client, ein FTP-Programm, sie alle benötigen die Daten des jeweiligen Benutzers. Um diesem die Arbeit möglichst einfach zu machen bieten diese an, die Passworte zu speichern.



Hier ein Dialogfenster, in dem der MS Internet-Explorer anbietet die Anmeldedaten für eine geschützte Website zu speichern.
Kaum jemand macht sich ernsthaft Gedanken darüber was mit diesen Daten geschieht.
Sicherlich wird der Internet-Explorer diese Daten irgendwie in einem nicht lesbaren Format abspeichern.
Sicher könnten die Daten aber nur abgelegt werden, wenn sie über ein zusätzliches Passwort des Benutzers verschlüsselt werden. Was die Software so machen kann ist keine sichere Verschlüsselung, da ja Verfahren und Passwort zum Entschlüsseln dem Programm bekannt sind. Jeder Benutzer, der Zugriff auf den Rechner hat kann diese Daten nutzen.
Einen ähnlichen „Service“ bieten die meisten FTP- und Mailprogramme. Manche Programme legen das Passwort sogar dann ab, wenn der Benutzer dies nicht wünscht.

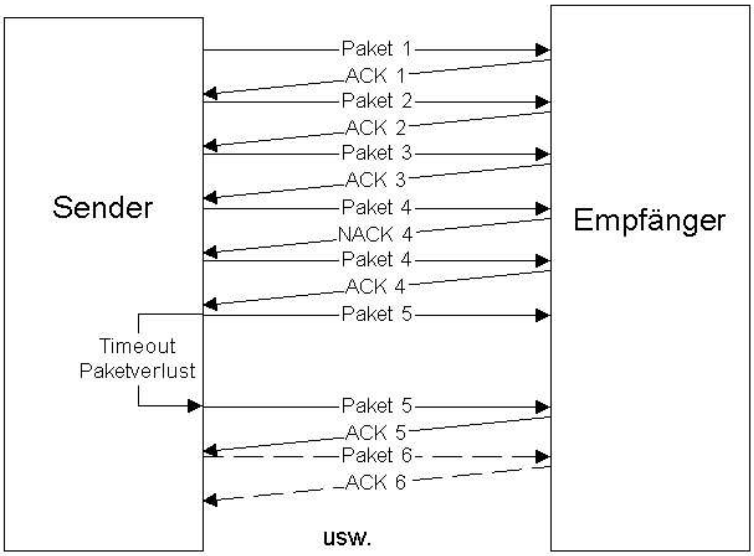
2. Verbindung I

Im Prinzip gibt es für die Vernetzung von Computern eine Vielzahl von sehr unterschiedlichen Protokollen. In den letzten Jahren hat sich aber die Protokollfamilie TCP/IP als Standard durchgesetzt, im Zusammenhang mit dem Siegeszug des Internet.
Jede Datenübertragung ist ein Risiko, wenn die Möglichkeit besteht den Datenstrom zu belauschen. Bei TCP/IP ist das Risiko recht hoch, da es für sehr große, z.T weltweite Netze ausgelegt ist und der Nutzer kaum den Weg seiner Daten beeinflussen kann bzw. ihn meistens noch nicht einmal kennt.
Der Aufbau der TCP/IP-Protokoll-Familie entspricht nicht ganz dem OSI-Modell, es sind hier einige Ebenen zusammengefasst.



Zur eigentlichen Protokoll-Familie gehören:

- TCP Das *Transmission Control Protocol* ist das bekannteste Protokoll auf dieser Ebene. Es setzt auf IP auf und ist verbindungsorientiert. Bevor mit der eigentlichen Datenübertragung begonnen wird, wird zunächst eine Verbindung zum Empfänger aufgebaut. Dann erst werden die Datenpakete abgeschickt und vom Empfänger quittiert. Bleibt diese Empfangsbestätigung aus, so wird das entsprechende Paket erneut versandt. Hierdurch wird sichergestellt, dass die Datenpakete in der richtigen Reihenfolge und vollständig beim Empfänger ankommen. Die Reihenfolge kann beim Versand verändert werden, da IP sich für jedes Paket einen anderen Weg durchs Netz suchen kann, mit eventuell unterschiedlichen Laufzeiten.

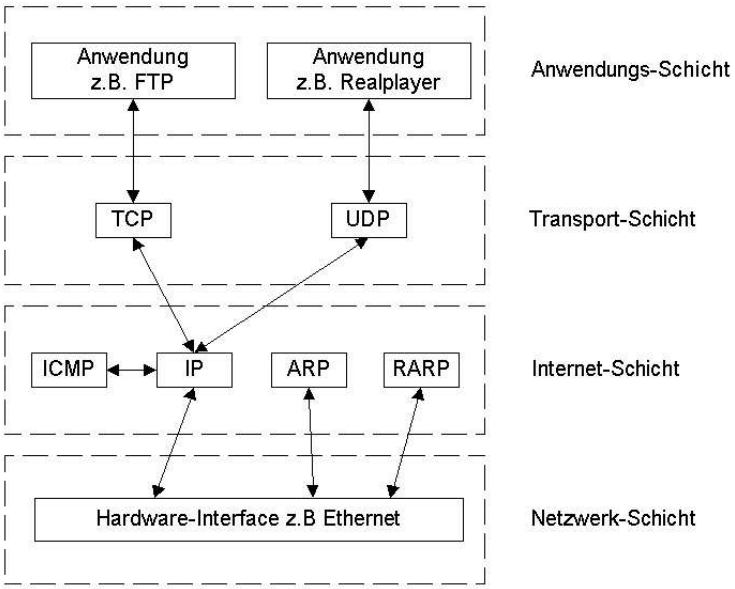


- UDP Beim *User Datagramm Protocol* handelt es sich um ein verbindungsloses Protokoll. Es dienst zum Übertragen von kurzen Nachrichten. Eine Nameserver-Anfrage gehört zu den Dingen, die über UDP abgewickelt werden. Wenn keine Antwort kommt, dann wird einfache eine neue Anfrage gestellt, eventuell an einen anderen Nameserver. Auch Streaming-Video und Netzwerkspiele arbeiten oft mit UDP, dabei geht es vor allem um die höhere Performance. Außerdem ist es hier nicht weiter tragisch bzw. sowieso nicht reparabel, wenn ein Datenpaket verloren ist.

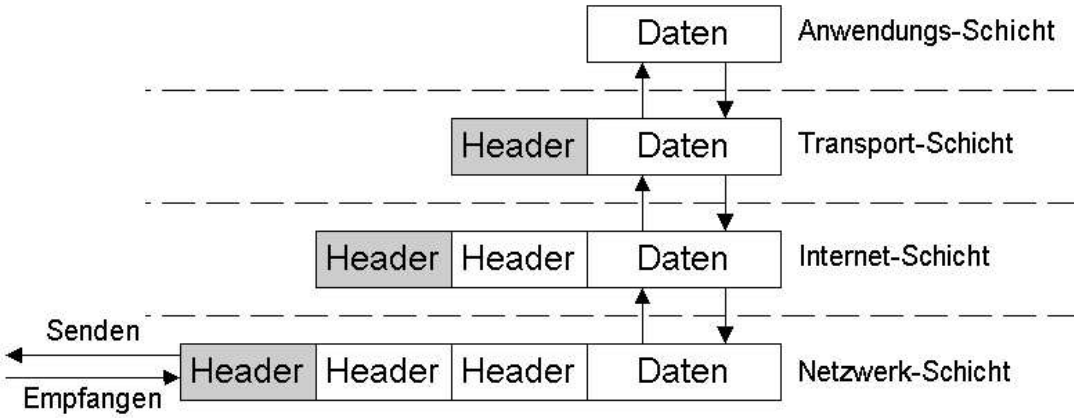
- IP Grundlage ist das Protokoll *Internet Protocol*. Es handelt sich hierbei um ein verbindungsloses Protokoll, das keinerlei Mechanismen zur Sicherung der Datenübertragung enthält.
Zu den Aufgaben von IP gehört die Adressierung der Datenpakete. Dazu dient die IP-Adresse, die aus einer 4Byte Zahl besteht und auf die in einem gesonderten Kapitel eingegangen wird. Eine weitere Aufgabe von IP ist das Aufteilen der Daten in Pakete, die von der darunter liegenden Schicht (z.B. Ethernet) übertragen werden können, sowie das korrekte Zusammensetzen der übertragenen Pakete beim Empfänger.

- ICMP Das *Internet Control Message Protocol* dient zum Transport von Fehler- und Diagnosemeldungen im Netz. Versucht ein Rechner auf einen Port zuzugreifen, der nicht belegt ist, so wird die Fehlermeldung „Port unreachable“ per ICMP zurückgeschickt. Auch Routing-Informationen und der bekannte Ping werden über ICMP weitergeleitet.
- ARP Über das *Address Resolution Protocol* erfolgt die Zuordnung zwischen MAC-Adresse und IP.
- RARP Das *Reverse Address Resolution Protocol* dient dazu zu einer MAC-Adresse die zugehörige IP-Adresse zu ermitteln.

Das Zusammenspiel der einzelnen Protokolle ergibt sich aus der folgenden Abbildung.

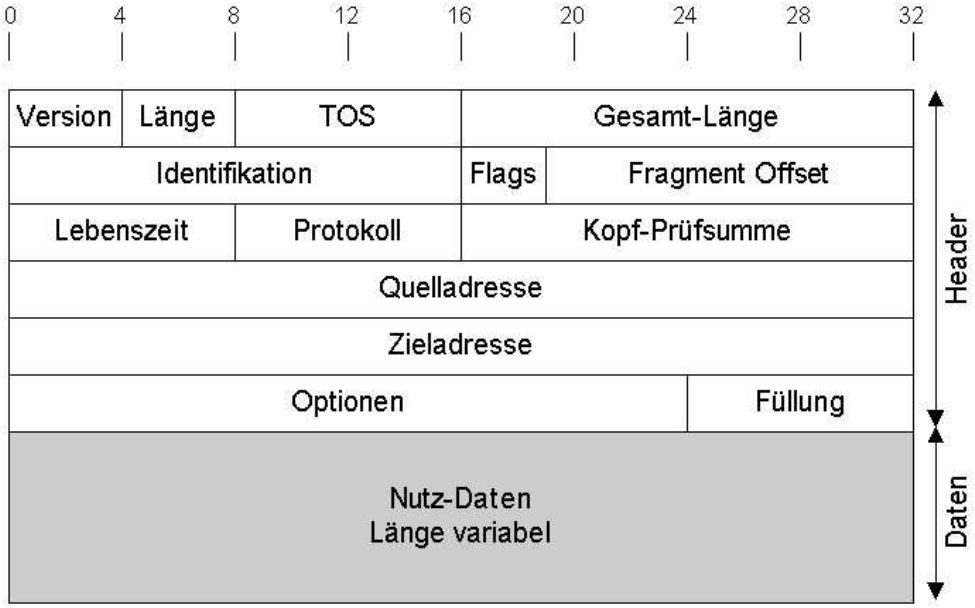


Beim Durchlaufen des Protokollstapels in Richtung Netzwerk-Schicht werden die Datenpakete immer größer, da jedes Protokoll einen spezifischen Header hinzufügt.



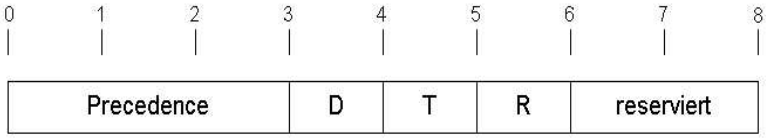
2.1. IP

Das Internet Protokoll (IP) packt den Datenstrom, den es aus der Transportschicht bekommt in neue Pakete, die aber mit den logischen IP-Adressen versehen sind, statt mit den physikalischen Netzadressen. Die interne Struktur eines solchen Datagramms (Paketes) ist in der folgenden Graphik dargestellt.



Die Felder haben folgende Bedeutung:

- *Version* (4Bit)
  4 bei IPv4
  6 bei IPv6
- *Länge des Headers* (4 Bit)
  Länge des IP-Headers in 32 Bit-Worten. Falls keine Optionen gesetzt sind ist der Wert 5, ansonsten entsprechend mehr. Der Wert kann maximal 15 betragen, dann hat der Header eine Länge von 60 Byte.
- *Type of Service* TOS (8 Bit)
  Flags zur Steuerung der Zuverlässigkeit und der Priorität. Hier sind hier verschiedene Kombinationen aus Zuverlässigkeit und Geschwindigkeit möglich. In der Praxis wird dieses Feld aber ignoriert, hat also den Wert 0. Das Feld selbst hat den folgenden Aufbau:



  Precedence (Bits 0-2) gibt die Priorität von 0 (normal) bis 7 (Steuerungspaket) an. Die drei Flags (D,T,R) ermöglichen es dem Absender anzugeben, worauf er bei der Datenübertragung am meisten Wert legt: Verzögerung (Delay - D), Durchsatz (Throughput - T), Zuverlässigkeit (Reliability - R). Die beiden anderen Bits sind reserviert bzw. nicht benutzt.
- *Gesamtlänge* (16 Bit)
  Gesamte Länge des Paketes in Byte, inklusive Header und Daten. Die

Paketgröße ist damit auf 65535 Bytes beschränkt. Das ist weit über der üblichen Paketgröße, 576 Byte muss jedes System verkraften, üblich sind 1500 Byte im Ethernet

- *Identifikation* (16 Bit)
  Jedes Datenpaket trägt eine eindeutige Identifikationsnummer, über die der Empfänger feststellen kann, welche Datenpakete zusammen gehören. Unter Fragmentierung von Datagrammen ist zu verstehen, dass einzelne Datagramme, die durch Gateways in unterschiedliche Netze geroutet werden, dort oft unterschiedlichen Größenbestimmungen unterliegen. IP fragmentiert solche Datagramme und baut sie wieder zusammen. Fragmente eines Datagrammes tragen alle die gleiche Identifikationsnummer.
- *Flags* (3Bit)
  Diese Bits dienen zur Steuerung der Fragmentierung:
  - Bit1 unbenutzt
  - Bit 2 DF Mit dem DF-Bit wird signalisiert, dass das Datengramm nicht fragmentiert werden darf. Auch dann nicht, wenn das Paket dann evtl. nicht mehr weiter transportiert werden kann und verworfen werden muß. Alle Hosts müssen, mindestens Fragemente bzw. Datengramme mit einer Größe von 576 Bytes oder weniger verarbeiten können
  - Bit 3 MF Mit dem MF-Bit wird angezeigt, ob einem IP-Paket weitere Teilpakete nachfolgen. Diese Bit ist bei allen Fragmenten außer dem letzten gesetzt.
- *Fragment Offset* (13 Bit)
  Dieses Feld enthält die Information, an welcher Stelle des Datagrammes das Fragment ursprünglich war. Mit Hilfe dieser Angabe kann der Zielrechner das Datenpaket wieder aus den Fragmenten zusammensetzen. Da dieses Feld nur 13 Bit groß ist, können maximal 8192 Fragmente pro Datengramm erstellt werden. Alle Fragmente, außer dem letzten, müssen ein Vielfaches von 8 Byte als Länge besitzen.
- *Lebenszeit* (8Bit)
  Die Lebenszeit oder Time to live (TTL) gibt eine maximale Lebendauer in Sekunden für ein Datenpaket vor. Bei Routing-Problemen könnte ein Paket sonst endlos im Netz herumwandern. So wird es spätestens nach 255 Stationen verworfen, da jeder Router dieses Feld um mindestens eine Einheit verringert.
- *Protokoll* (8Bit)
  Gibt die Nummer des Protokolls auf der Transportschicht an. Die Zahlen sind auf Unix-Systemen in der Datei /etc/protocols zu finden, u.a.
  1 ICMP
  6 TCP
  17 UDP
- *Kopf-Prüfsumme* (16 Bit)
  Prüfsumme über den Inhalt des Headers, nicht der Daten. Die Prüfsumme muss in jedem Netzknoten neu berechnet werden, das sich jeweils die TTL verringert. Berechnet wird das 1er Komplement über die 16 Bit-Wörter. Ausgangswert ist Null.
- *Quelladresse* (32 Bit)
  IP-Adresse des Absenders

- *Zieladresse* (32 Bit)
  IP-Adresse des Empfängers
- *Optionen und Füllung* (variable)
  Über dieses Feld ist der IP-Header erweiterbar gehalten worden. Das Feld besitzt eine variable Länge und wird durch die Füllung auf ein Vielfaches von 4 Byte aufgefüllt. Die bisher üblichen Optionen haben meist mit Debugging und Routenüberprüfung zu tun.

2.2 ARP/RARP

Bekanntlich kommunizieren Rechner im TCP/IP Netz über ihre IP-Adressen. Auf der Netzwerkebene erfolgt die Kommunikation aber über die *Media Access Control* (MAC) Adresse der Netzwerkkarte. Über das ARP-Protokoll erfolgt die Zuordnung zwischen IP- und MAC-Adresse.
Will z.B. der Rechner mit der IP-Adresse 192.168.1.31 den Rechner mit der IP-Adresse 192.168.1.1 erreichen, so benötigt er dessen MAC-Adresse. Dazu gibt er ein ARP-Paket per Broadcast ins Netz.

```
ARP: ----- ARP/RARP frame -----
             ARP:
             ARP: Hardware Typ = 1 (Ethernet)
             ARP: Protokoll Typ = 0x0800 (IP)
             ARP: Länge der Hardware addresse = 6 bytes
             ARP: Länge der Protokoll Addresse = 4 bytes
             ARP: Opcode 1 (ARP request)
             ARP: Hardware Addresse Absender = 0000E2187CBD
             ARP: Protokoll Addresse = [192.168.1.31]
             ARP: Hardware Addresse Ziel   = 000000000000
             ARP: Protokoll Addresse Ziel = [192.168.1.1]
             ARP:
             ARP: 18 Bytes Füllung
             ARP:
```

Der angesprochene Rechner schickt dann, sofern erreichbar, ein Antwortpaket der folgenden Art.

```
ARP: ----- ARP/RARP frame -----
             ARP:
             ARP: Hardware Typ = 1 (Ethernet)
             ARP: Protokoll Typ = 0x0800 (IP)
             ARP: Länge der Hardware addresse = 6 bytes
             ARP: Länge der Protokoll Addresse = 4 bytes
             ARP: Opcode 2 (ARP reply)
             ARP: Hardware Addresse Absender = 0048541B5973
             ARP: Protokoll Addresse = [192.168.1.1]
             ARP: Hardware Addresse Ziel   = 0000E2187CBD
             ARP: Protokoll Addresse Ziel = [192.168.1.31]
             ARP:
             ARP: 18 Bytes Füllung
             ARP:
```

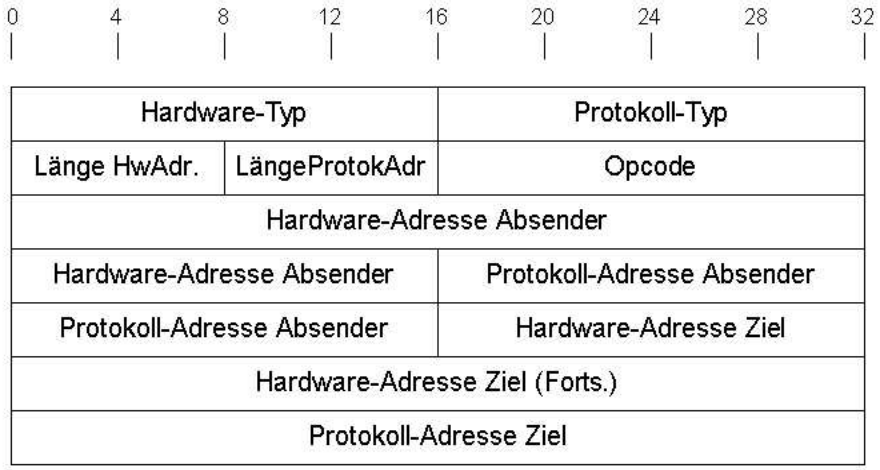
Die Antwort nimmt der Fragende Rechner in seine Arp-Tabelle mit auf, so dass er bei weiteren Datenübertragungen nicht erneut nachfragen muss. Der momentane Inhalt der

Arptabelle läßt sich auf den meisten Systemen mittels

```
arp -a
```

abfragen. Für die Einträge in dieser Tabelle gilt eine gewisse Lebendauer, danach werden die Einträge gelöscht, um Veränderungen im Netz erkennen zu können.

Ein ARP-Paket hat folgenden Aufbau:



Die Felder haben folgende Bedeutung:

- *Harware-Typ* (16 Bit) u.a.
  - 1 Ethernet
  - 4 Token Ring
  - 7 Arcnet
  - 17 HDLC
  - 31 Ipsec Tunnel

- *Protokoll-Typ* (16 Bit)
  0x0800 IP

- *Länge der Harware Adresse* (8 Bit)
  6 Byte bei Ethernet

- *Länge der Protokoll Adresse* (8Bit)
  4 Byte bei IPv4

- *Opcode (16 Byte)* u.a.
  - 1 ARP Request
  - 2 ARP Reply
  - 3 RARP Request
  - 4 RARP Reply

- *Hardware Adresse Absender*
  hier steht bei Ethernet die MAC-Adresse mit einer Länge von 6 Byte

- *Protokoll Adresse Absender*
  bei IPv4 steht hier die IP Adresse mit einer Länge von 4 Byte

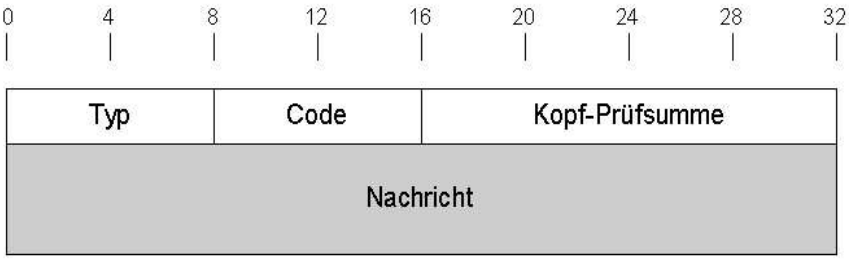
- *Hardware Adresse Ziel*
  hier steht bei Ethernet die MAC-Adresse mit einer Länge von 6 Byte

- *Protokoll Adresse Ziel*
  bei IPv4 steht hier die IP Adresse mit einer Länge von 4 Byte

Zusätzlich wird das Datenpaket noch mit 18 Byte aufgefüllt für den Versand per IP.

2.3 ICMP

ICMP dient zum Austausch von technischen Meldungen, die die eigentliche Internet-Schicht nicht erreichen müssen. Über ICMP können Netzwerkknoten Routing-Informationen austauschen. Auch kann z.B. ein Gateway einen Absender darüber informieren, dass der Zielrechner nicht erreichbar ist.



Die Felder haben folgende Bedutung:

- *Typ* (8Bit)
  Spezifiziert das Format der Nachricht
  - 0 Echo Reply (Ping)
    Antwort auf ein Echo Request
  - 3 Destination unreachable (Ziel nicht erreichbar), diese Nachricht wird z.B. versandt, wenn ein Netzwerk, Host, Protokoll oder Port nicht erreichbar ist, ein Paket nicht fragmentiert werden kann, weil das DF-Bit gesetz ist
  - 5 Redirect
    Das Paket hat zum aktuellen Netzknoten nicht den geschicktesten Weg genommen. Der Absender wird über einen geschickteren Weg informiert.
  - 8 Echo Request (Ping)
    Paket, das den Empfäger dazu auffordern soll ein Antwortpaket zu schicken. Damit kann die Erreichbarkeit des Zielrechners getestet werden
  - 11 Time exceeded
    Mitteilung an den Absender, dass die TTL seines Paketes 0 erreicht hat und das Paket wurde daher verworfen wurde.
  - 13 Timestamp Request
    Ähnelt dem Echo Request, nur dass hier zusätzliche Zeitinformationen übermittelt werden. Damit lässt sich die Netzlast feststellen.
  - 14 Timestamp Reply
    Antwort auf ein Timestamp Request

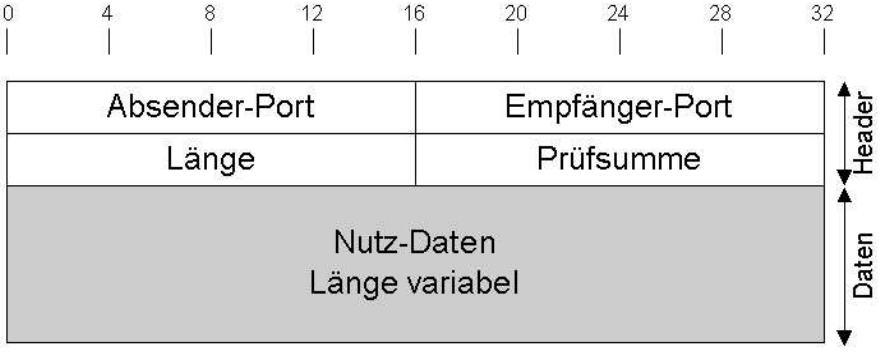
- *Code* (8Bit)
  Zusätzliche Informationen für die Nachricht. Bei den meisten Nachrichtentypen ist der Wert dieses Feldes 0. Bei manchen Nachrichtentypen stehen hier folgende Zusatzinformationen:
  Typ 3 (Destination Unreachable) Codes:
  0 Net Unreachable
  1 Host Unreachable
  2 Protocol Unreachable
  3 Port Unreachable
  4 Fragmentation Needed and Don't Fragment was Set
  5 Source Route Failed
  6 Destination Network Unknown
  7 Destination Host Unknown
  Typ 5 (Redirect) Codes:
  0 Redirect Datagram for the Network (or subnet)
  1 Redirect Datagram for the Host
  2 Redirect Datagram for the Type of Service and Network
  3 Redirect Datagram for the Type of Service and Host
  Typ 11 (Time Exceeded) Codes:
  0 Time to Live exceeded in Transit
  1 Fragment Reassembly Time Exceeded

- *Kopf-Prüfsummer* (16 Bit)
  Prüfsumme über den Header, im 1er Komplement über 16 Bit Worte

- *Nachricht* (variable Länge)
  Spezifische Informationen je nach Nachrichten-Typ. Ggf. auch zum Auffüllen des Paketes auf die Mindestlänge genutzt.

2.4 UDP

Als verbindungsloses Protokoll benötigt UDP keinerlei Bestätigung oder gar eine Sequenzverwaltung. UDP-Pakete werden in der Reihenfolge abgearbeitet, in der sie eintreffen. Verlorene Pakete werden nicht erneut angefordert, was z.B. bei einem Videostream oder einem Online-Spiel ja auch keinen Sinn machen würde.
Bei diesem Protokoll geht es nur darum die Ports zu adressieren und für die Konsistenz des Datenpaketes selber zu sorgen. Hierzu dient eine Längenangabe und eine Prüfsumme.

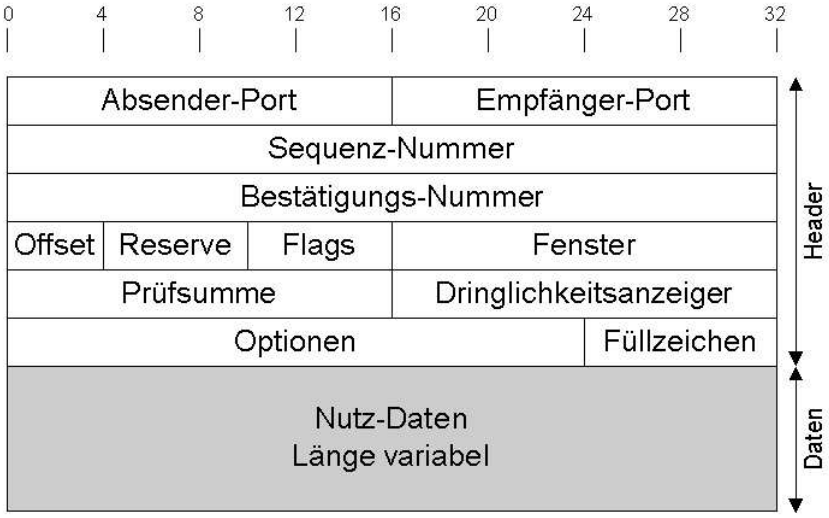


Aufbau einer UDP-Message:

- Absender-Port (16 Bit)
  Port für das Portokoll auf Sender-Seite. Kann auch auf 0 gesetzt sein.
- Empfänger-Port (16 Bit)
  Port für das Portokoll auf Empfänger-Seite. Kann auch auf 0 gesetzt sein.
- Länge (16 Bit)
  Länge des UDP-Headers und der Daten. Die Mindestlänge ist 8
- Prüfsumme (16 Bit)
  Prüfsumme im 1er-Komplement über die Daten im Header

2.5 TCP

Dieses Protokoll muss sich nicht um die Adressierung der Daten kümmern, sondern nur um die Sicherung. Der einzige Adressteil innerhalb dieses Protokolls ist die Portnummer, der eine Zuordnung zu Prozessen innerhalb der jeweiligen Rechner erlaubt.
Ein TCP-Segment hat folgenden Aufbau:



Die Felder haben folgende Bedeutung:

- *Absender-Port* (16 Bit)
  Port für das Portokoll auf Sender-Seite
- *Empfänger-Port* (16 Bit)
  Port für das Portokoll auf Empfänger-Seite
- *Sequenz-Nummer* (32 Bit)
  s.u.
- *Bestätigungs-Nummer* (32 Bit)
  Die Sequenznummer und die Bestätigungsnummer sind zwei 32-Bit-Zahlen, die die Stellung der Daten des Segments innerhalb ausgetauschten Datenstroms angeben. Die Sequenznummer gilt in Senderichtung, die Bestätigungsnummer für Empfangsquittungen. Jeder der beiden TCP-Verbindungspartner generiert beim Verbindungsaufbau eine Sequenznummer, die sich während der gesamnten Verbindung nicht wiederholen darf (bei einem Zahlenraum von 2^32 sicherlich kein Problem). Diese Nummern werden beim Verbindungsaufbau

ausgetauscht und gegenseitig quittiert. Bei der Datenübertragung wird die Sequenznummer vom Absender jeweils um die Anzahl der bereits gesendeten Bytes erhöht. Mit der Quittungsnummer gibt der Empfänger bei gesetztem ACK-Bit an, bis zu welchem Byte er die Daten bereits korrekt empfangen hat.

- *Offset* (4 Bit)
  Länge des Headers in 32 Bit Worten, kennzeichnet den Anfang des Datenbereiches. Notwendig, da der Header durch das Options-Feld eine variable Länge besitzt.
- *Reserve* (6 Bit)
  Zur Zeit nicht genutzt, muss jeweils auf 0 stehen.
- *Flags* (6 Bit)
  Mit den sechs 1-Bit-Flaggen wird u.a. der Verbindungsablauf gesteuert.
  - URG wird das Flag Urgent pointer valid auf 1 gesetzt, so bedeutet dies, dass der Urgent Pointer (Dringend-Zeiger) verwendet wird.
  - ACK Das Acknowledgment number valid-Bit wird gesetzt, um anzugeben, daß die Bestätigungsnummer im Feld Acknowledgement Number gültig ist. Ist das Bit auf 0 gesetzt, enthält das TCP-Segment keine Bestätigung, das Feld Acknowledgement Number wird ignoriert.
  - PSH Ist das Push-Bit gesetzt, so werden die Daten in dem entsprechenden Segment sofort bei Ankunft der adressierten Anwendung bereitgestellt ohne sie zu puffern.
  - RST Das Reset Connection-Bit dient dazu eine Verbindung zurückzusetzen, falls ein Fehler bei Übertragung aufgetreten ist.
  - SYN Das Synchronize Sequenze Numbers-Bit wird verwendet, um Verbindungen aufzubauen. Zusammen mit der Acknowledgement Number und dem ACK-Bit wird die Verbindung im Form eines Dreiwege-Handshake aufgebaut (siehe oben).
  - FIN Das End of Data-Bit dient zum Beenden einer Verbindung. Ist das Bit gesetzt, gibt dies an, dass der Sender keine weiteren Daten zu Übertragen hat. Das Segment mit gesetztem FIN-Bit muß quittiert werden.
- *Fenster* (16 Bit)
  Gibt an, wieviele Daten-Bytes der Absender dieses Segmentes in der Lage ist zu akzeptieren. Die angegebene Anzahl von Bytes kann der Empfänger dieser Information senden, ohne auf eine Quittung warten zu müssen.
- *Prüfsumme* (16 Bit)
  Prüfsumme für den Protokollkopf.
- *Dringlichkeitsanzeiger* (16 Bit)
  Wenn das URG-Flag gesetzt ist, dann ergibt dieser Wert zusammen mit der Sequenznummer einen Zeiger auf ein Datenbyte. TCP signalisiert damit, dass sich an dieser Stelle im Datenstrom wichtige Daten befinden, die sofort gelesen werden sollten.
- *Optionen* (variabel 0 bis 44 Byte)
  Das Options-Feld bietet die Möglichkeit Funktionen bereitzustellen, die im TCP-Protokollkopf nicht vorgesehen sind. In TCP sind drei Optionen definiert:
  - 0 End of Option List (1 Byte),
  - 1 No-Operation (1 Byte)
  - 2 Maximum Segment Size (4 Byte)
  - 8 Timestamp Value (10 Byte)

Mit der Option Maximale Segmentgröße kann ein Host die maximale Anzahl Nutzdaten übermitteln, die er annehmen will bzw. annehmen kann. Während eines Verbindungsaufbaus kann jede Seite ihr Maximum an Nutzdaten übermitteln, die kleinere der beiden Zahlen wird als maximale Nutzdatengröße für die Übertragung übernommen. Wird diese Option von einem Host nicht unterstützt wird als Standardwert die Vorgabe von 536 Byte verwendet.

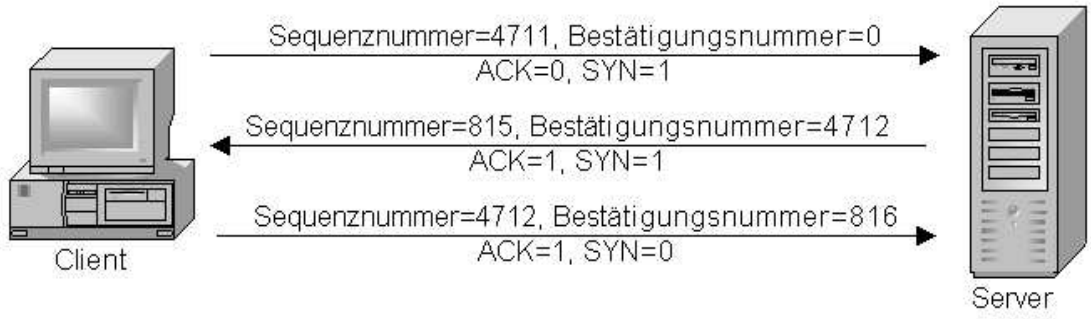
- *Füllzeichen* (variable)
  Dient dazu die Headerlänge trotz des variablen Optionen-Feldes auf volle 32 Bit Worte zu bringen.

Bevor man eine TCP-Verbindung benutzen kann, muß man sie zuerst einmal aufbauen. Dazu dient ein Dreier-Handshake. Nach dem eigentlichen Datenaustausch muss die Verbindung auch wieder abgebaut werden.

Im folgenden Beispiel will ein Rechner (Client) eine Verbindung zu einem anderen Rechner (Server) aufbauen und von dort z.B. eine HTML-Seite beziehen.

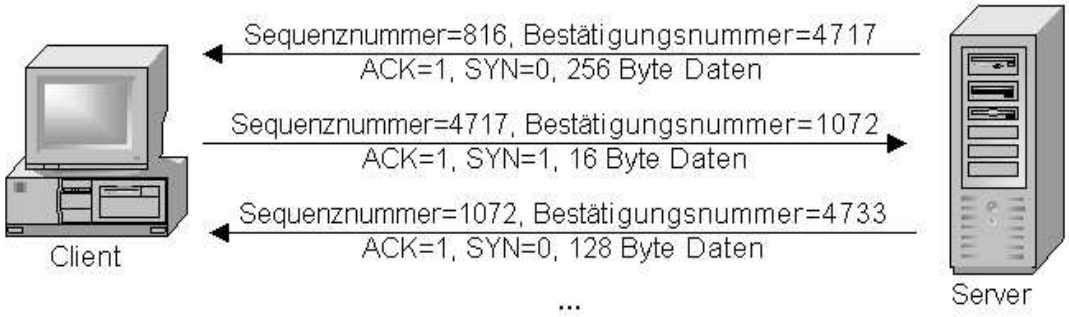
2.5.1 TCP-Verbindungsaufbau

Der Client schickt bei diesem Dreier-Handshake das erste Datenpaket, mit dem er den Verbindungsaufbau einleitet. Der Server antwortet mit einem Bestätigungspaket, worauf der Client die Verbindung als aufgebaut (established) erklärt.



2.5.2 TCP-Datenaustausch

Nach dem Verbindungsaufbau schickt der Server das erste Datenpaket, bei vielen Dienste eine Begrüßungsmeldung (in der Zwischenzeit wurden 5 Byte übertragen).



2.5.3 TCP-Verbindungsabbau

Zum Beenden der Verbindung sendet der Client ein Paket mit gesetztem FIN-Bit. Wenn nun der Server ebenfalls ein Paket mit gesetztem FIN-Bit schickt, dann ist die Verbindung beendet.

2.6. Ethernetframe

Zum Versand werden alle Datenpake in Ethernet-Frames verpackt, um dann über die Leitung versandt zu werden.

Die Felder haben fogende Bedeutung:

- Zieladresse (6 Byte)
  MAC-Adresse des Empfängers
- Zieladresse (6 Byte)
  MAC-Adresse des Empfängers
- Type (2 Byte)
  - 0800 IP-Protokoll
  - 0806 ARP-Protokoll
- Daten (variabel)
  Die eigentlichen Daten mit mindestens 46 Byte und höchstens 1500 Byte.
- CRC (4Byte)
  Eine Prüfsumme, Cyclic Redundancy Check, über das gesamte Datenpaket. Ein Paket mit fehlerhafter Prüfsumme wird immer verworfen.

3. Verbindung II: Programme zur Kontrolle und Manipulation von Datenpaketen

Für eine eingehende Beschäftigung mit der TCP/IP Protokollfamilie benötigt man Tools bzw. Programme zur Erzeugung von Datenpaketen und Programme zu Anzeige bzw. Filterung von Datenpaketen (Sniffer).

3.1. TCPDUMP

Dieses Programm ist eines der bekanntesten Snifferprogramme und wird bei den meisten Linux-Distributionen mit ausgeliefert.
Die Möglichkeiten von tcpdump sind sehr umfangreich, deshalb hier nur ein paar mögliche Anwendungen:

```
tcpdump -i eth0 'ether proto \arp'
```

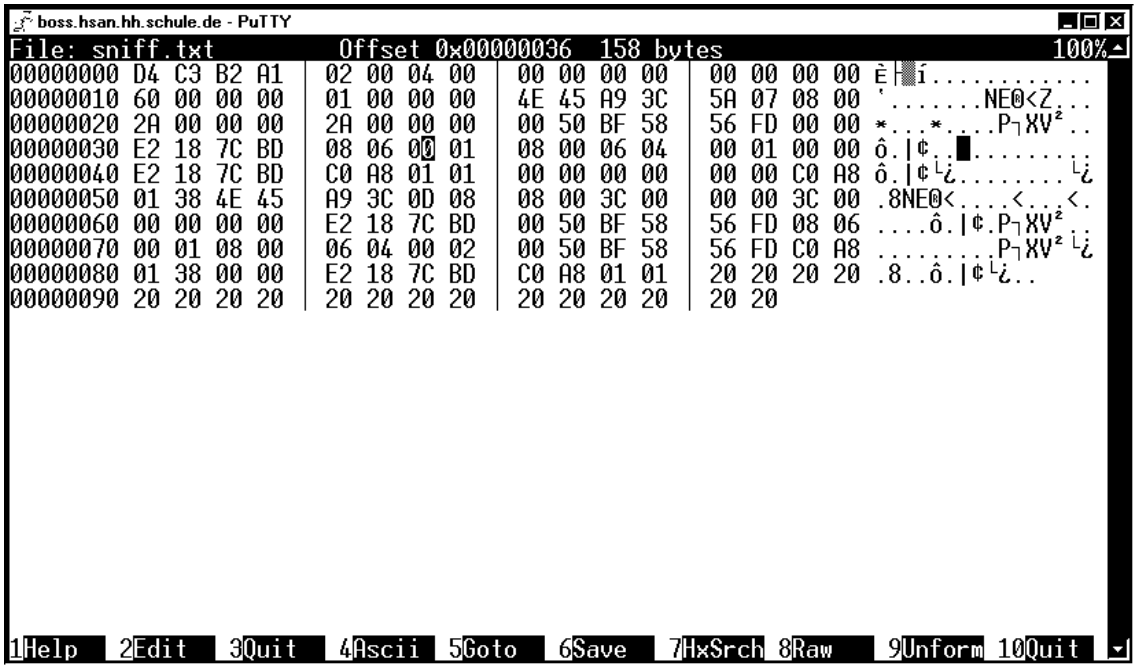
protokolliert alle Datenpakete auf dem Interface eth0, mit dem Protokoll arp.
Tcpdump gibt die Pakete nicht einfach so aus, wie sie übers Netz kommen, sondern interpretiert sie.

```
12:41:07.640057 arp who-has boss.hsan.hh.schule.de (28:ca:4:8:1d:0) tell hh1-2.hsan.hh.schule.de
12:41:07.773440 arp reply boss.hsan.hh.schule.de is-at 0:0:e2:18:7c:bd
```

Will man die Pakete als solche auswerten, so macht es Sinn diese in eine Datei zu leiten, da nicht alle Zeichen darstellbar sein dürften.

```
tcpdump -i eth0 -w sniff.txt 'ether proto \arp'
```

Leitet die Pakete in die Datei sniff.txt um. Diese Datei kann man dann mit einem beliebigen Hex-Editor betrachten. Ein dafür vollkommen ausreichendes Werkzeug dafür ist der mc, der Midnight Commander. Dort führt man den Leuchtbalken auf die Datei sniff.txt und drückt F3 (Anzeige). Innerhalb des Anzeige-Programmes kann man dann mit F4 in den Hex-Modus umschalten, was folgende Darstellung ergibt.



Tcpdump kann auch eine Aufzeichnung interpretieren, zur Angabe des Dateinamens dient dann der Schalter -r.

```
tcpdump  -X  -n  -r sniff.txt
```

Der Schalter -X bewirkt, dass sowohl die interpretierten Daten, als auch die rohen Daten angezeigt werden. Mit -n wird die Namensauflösung unterdrückt, so dass Tcpdump nur IP-Adressen anzeigt und nicht die Namen.

```
07:44:46.526170 arp who-has 192.168.1.56 tell 192.168.1.1
0x0000   0001 0800 0604 0001 0000 e218 7cbd c0a8        ............|...
0x0010   0101 0000 0000 0000 c0a8 0138                  ...........8
07:44:46.526349 arp reply 192.168.1.56 is-at 0:50:bf:58:56:fd
0x0000   0001 0800 0604 0002 0050 bf58 56fd c0a8        .........P.XV...
0x0010   0138 0000 e218 7cbd c0a8 0101 2020 2020        .8....|.........
0x0020   2020 2020 2020 2020 2020 2020 2020            ..............
```

Das Programm erlaubt eine sehr einfache Analyse der Daten auf dem Netzwerk

3.2. ngrep

Es gibt viele mit tcpdump ähnliche Programme unter Linux, wie z.B. ngrep ein Sniffer-Programm mit Suchmustern wie bei grep. Dieses Programm ist bei der SuSE-Distribution dabei. Ein typischer Aufruf sieht folgendermaßen aus:

```
ngrep -d eth0 -i "rv_uname | rv_passwd"
```

Mit diesem Aufruf sucht ngrep in den Datenpaketen auf eth0 nach den Schlüsselwörtern rv_uname bzw rv_passwd, die bei der Webmail-Anmeldung von Web.de auftauchen. Ngrep zeigt dann nur Datenpakete mit Treffern an.

3.3. Spak

Das Programmpaket Spak dient dazu Datenpakete *nach Wunsch* zu erzeugen.
Die Programm-Quellen von http://www.xenos.net/software/spak/ ließen sich nicht compilieren, es gibt aber ein rpm-Paket http://rpmfind.net//linux/RPM/contrib/libc6/i386/spak-0.6b-1.i386.html, das auch auf aktuellen Systemen zu installieren ist. Das Paket besteht aus den einzelnen Programmen (jeweils im Verzeichnis /usr/bin):

- sendpacket
- sendeth
- makeudp
- maketcp
- makeip
- makeeth
- makearp

Für jedes der Programme bekommt man eine kurze Anleitung, wenn man es mit dem Schalter -h aufruft. Eine weitere Informationsquelle findet sich unter /usr/doc/spak-0.6b/README.
Die Programme arbeiten jeweils nur auf einer der Protokollebenen. Zum gezielten Versenden eines Datenpaketes muss man immer mehrere der Programme nacheinander aufrufen, wie in dem folgenden Beispiel.

```
#!/bin/sh

#File   : arp
#Purpose: Create and send an ARP request.

#Location of programs used
MAKEETH=/usr/bin/makeeth
MAKEARP=/usr/bin/makearp
SENDETH=/usr/bin/sendeth

SRC=192.168.1.2
DST=192.168.1.1
SRC_MAC=00:20:AF:F7:56:42

$MAKEARP -op 1 -di $DST -si $SRC -sm $SRC_MAC |
      $MAKEETH -s $SRC_MAC -i - -t 0x806 | $SENDETH -i -
```

Hier wird zuerst ein ARP-Paket erzeugt. Der Schalter -op 1 legt fest, dass es sich um ein ARP-Request handelt, -sm die MAC-Adresse der Quelle, -di die gesuchte IP-Adresse und -si die IP-Adresse des Absenders.
Das ARP-Paket wird jetzt noch in einen Ethernetframe verpackt, hier gibt -s die MAC-Adresse des Absenders an, -t den Ethernet-Typ und -i legt fest, aus welcher Datei die Eingabedaten genommen werden. Steht hier statt eines Dateinamens nur „-„ dann wird die Standardeingabe benutzt.
Zuletzt geht das Datenpaket an das Programm sendeth, was für den eigentlichen Versand zuständig ist. Auch hier legt -i die Eingabedatei fest. Für sendeth sind noch die Schalter -v und -vv interessant, die Informationen über das verschickte Datenpaket an der Konsole ausgeben.

Für ein normales Datenpaket ist der Ablauf:

Datei mit Daten -> maketcp -> makeip -> sendpacket

Ein Bespiel für das Senden eines beliebigen Textes zeigt das folgende Listing.

```
#!/bin/sh

# Location of programs used
MAKETCP=maketcp
MAKEIP=makeip
SENDPACKET=sendpacket

SRC=80.137.148.10
SRC_PORT=313
DST=195.37.209.130
DST_PORT=25

echo "Hallo" | $MAKETCP $SRC $SRC_PORT $DST $DST_PORT -ss -i - |
      $MAKEIP $SRC $DST -i - -sd | $SENDPACKET $DST -v
```

Hier wird der Text Hallo an das Programm *maketcp* übergeben mit den zugehörigen Adressangaben. Das erstelle Datenpaket geht dann weiter an *makeip* und letztendlich an *sendpacket*.
Den Aufbau der Datenpakete kann man sehr genau einstellen, dazu dienen Parameter für *maketcp* und *makeip*.

Maketcp kennt die folgenden Schalter:

```
  -a        Set the acknowledge number to <ack_num>.  Default: random
  -do       Set the data offset to <offset>.  Default: 0
  -h        Print this help message.
  -c        Set the header checksum to <hdr_cksum>.  Default: correct value
  -i        Read the packet data from the file <input_file>.  If <input_file>
            is -, the data will be read from stdin.
  -o        Send the output to <output_file>.  If this argument is not given,
            the output will be sent to stdout.
  -r1 - -r6 Set reserved bits, (-r5 sets bit 5).  Default: not set
  -s        Set the sequence number to <seq_num>.  Default: random
  -sa       Set the ACK bit.  Default: not set
  -sf       Set the FIN bit.  Default: not set
  -sp       Set the PSH bit.  Default: not set
  -sr       Set the RST bit.  Default: not set
  -ss       Set the SYN bit.  Default: not set
  -su       Set the URG bit.  Default: not set
  -uo       Set the urgent offset to <offset>.  Default: 0
  -v        Turn on verbose mode, (print packet data).
  -w        Set the window size to <win_size>.  Default: 512
```

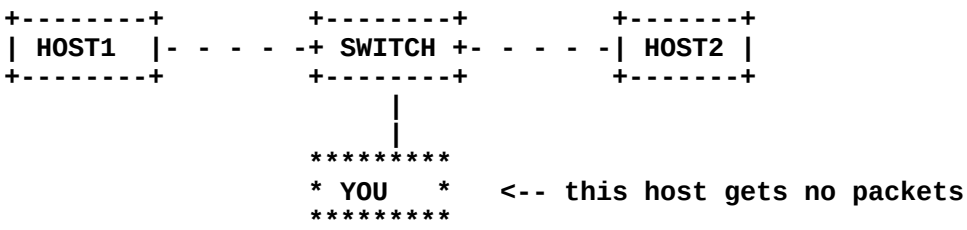
Makeip verfügt über diese Schalter:

```
  -c    Set the header checksum to <hdr_cksum>.
  -fo   Set the fragment offset to <fragment_offset>.
  -h    Print a help message.
  -hl   Set the header length to <hdr_len>.
  -i    Read the packet data from the file <input_file>.  If <input_file> is
        -, the data will be read from stdin.
  -id   Set the packet ID to <id>.
  -o    Send the output to the file <output_file>.  If this argument is not
        given, the packet will be written to stdout.
  -of   Set the options listed in <opt_file>, (under construction).
  -p    Set the protocol to <protocol>, (number).
  -pr   Set the priority to <priority>.
  -tl   Set the total packet length.
  -ttl  Set time to live to <time>.
  -s6   Set the 6th bit in the type of service field.
  -s7   Set the 7th bit in the type of service field.
  -sd   Set the "low delay" bit.
  -sf   Set the first bit in the flag field.
  -sdf  Set the "don't fragment" bit.
  -smf  Set the "more fragments" bit.
  -sr   Set the "high reliability" bit.
  -st   Set the "high throughput" bit.
  -vr   Set the IP version to <version>.
  -v    Turn on verbose mode, (print packet data).
```

Damit steht der Konstruktion (nahezu) beliebiger Datenpakete nichts mehr im Weg.

3.4. ARP0c

ARP0c http://www.phenoelit.de/fr/tools.html ist ein Programm mit dem man Verbindungen auch in geswitchten Netzen überwachen kann, indem es die Arp-Tabellen der beteiligten Rechner manipuliert.
In einem geswitchten Netzt sieht es für einen potentiellen Sniffer folgendermaßen aus:



Der Angreifer sieht hier nur ARP-Anfragen oder andere Arten von Broadcast-Verkehr, der wenig interessant ist. Genau deshalb werden in vielen Netzen Switches auch eingesetzt.
Beim Einsatz von ARP0c antworten der richtige Rechner und der ARP0c-Rechner auf ARP-Anfragen. Während der richtige Rechner nur einmalig antwortet fährt der ARP0c mit Antworten fort, um den Zielrechner informiert zu halten.
Die meisten Rechner verwerfen daraufhin die richtige Antwort und glauben den Aussagen des ARP0c-Rechners. Alle Datenpakete an den richtigen Rechner laufen damit über den ARP0c-Rechner, der sich darum kümmert die Datenpakete auch beim eigentlichen Empfänger abzuliefern, damit die Verbindung nicht unterbrochen wird.

ARP0c wird über zwei Textdateien konfiguriert. Die erste Datei (hier routes.txt) beschreibt das Netzwerk. Hier steht in einer Zeile:

```
Netzwerkadresse Netzwerkmaske Gateway
```

Beispiel

```
192.168.1.0     255.255.255.0   192.168.1.1
```

Die zweite Datei (hier server.txt) beschreibt die Verbindung, die umgeleitet werden soll. In dieser Datei können mehrere Verbindungen beschrieben werden. In jeder Zeile steht:

```
host1   host2
```

Beispiel:

```
192.168.1.1             192.168.1.92
```

Dann muss nur noch das Programm aufgerufen werden:

```
ARP0c -i < interface > -r < routingtable.file > -a < agressive_intercept.file >
```

Beispiel:

```
ARP0c -i eth0 -r routes.txt -a server.txt -v
```

3.5 conflictd

Eine interessante Anwendung von ARP stellt das Programm conflictd http://packetstormsecurity.org/DoS/conflictd.tar.gz zur Verfügung. Es versendet verfälschte ARP-Pakete, wodurch am Zielrechner ein popup-Fenster mit einer Fehlermeldung erzeugt wird.
Der Aufruf

```
conflict-DoS eth0 192.168.1.56
```

erzeugt dann auf dem angegriffenen Rechner etwa 100 popup-Fenster mit wenig aussagekräftigem Inhalt:

Gerade Windows-Anwender, die sehr viel mit der Maus arbeiten sind eine Weile beschäftigt, bis all diese Fenster beseitigt sind. Die werden dann auch kaum die nette MAC-Adresse würdigen wollen.

3.6 datapool

Die TCP/IP Implementierungen eigentlich aller Systeme weisen irgendwelche kleinen Fehler auf, die ein Angreifer ausnutzen kann. Für jeden dieser Fehler gibt es dann auch ein Programm, das ihn für einen Angriff ausnutzt. Die Wahl des richtigen Programmes hängt aber immer vom Betriebssytem und dessen Aktualität ab.
Zur Vereinfachung derartiger Angriffe finden sich im Internet Programmpakete, die möglichst viele Angriffsprogramme sammeln und nacheinander auf einen Zielrechner loslassen, solange bis dieser offline ist. Ein entsprechendes Programmpaket ist datapool von http://packetstorm.mirror.widexs.nl/DoS/datapool3.3.tar.gz.

Die Nutzung des Programmpaketes ist einfach, der Aufruf von datapool.sh liefert folgenden Hilfetext.

```
Usage: ./datapool.sh [-p] [portlow-porthigh] [-x] [-v] [logfile] [-k]
 [-d] [destination ip] [-s] [-l] [T1|T3|OC3|Modem|Slowass]
 [-i] [source ip] [-c] [-t] [#of attacks] [-r] [attackname]
Options:
[-d]: Specifies destination IP or hostname: REQUIRED
[-p]: Specifies port range to scan.  ex: -p 1-1024
[-x]: "Don't stop till they drop"
[-v]: Logs results of scan to file.  ex: -v logfile.log
[-s]: Scan ports only.
[-l]: Specifies line speed. Choose from T1,T3,OC3,Modem, and Slowass.
[-i]: Specifies source IP.  ex: -i 127.0.0.1
[-k]: Wait till host is online, then attack.
[-c]: Never stop attacking.
[-t]: Number of simultaneous attacks to launch. ex: -t 4
[-r]: Run this attack only.  ex: -r onetwothreefour
      Note: attacknames can be found in datapool.fc
```

Der einfachste Aufruf wäre also:

```
datapool.sh  -d 192.168.1.56
```

Es ist schon erschreckend, wie schnell viele Rechner vor diesem Programm in die Knie gehen. Das hängt auch damit zusammen, das Windows-Nutzer ihre Systeme nur selten aktualisieren bzw. die vorhandenen Updates einspielen.

4. Rechner

In diesem Abschnitt soll es darum gehen, was man über einen Rechner im Netz erfahren kann, wenn man etwas Wissen über TCP/IP besitzt oder die richtigen Tools kennt.
Zu den wichtigsten Informationen gehört das Betriebssytem und die offenen Ports. Beide Information lassen sich oft sogar für Systeme ermitteln, die durch eine Firewall geschützt sind.

4.1. OS Fingerprinting

Viele Größen im TCP/IP Protokoll sind nicht zwingend vorgeschrieben bzw. manche Betriebssystem-Versionen halten sich nicht an die Vorschriften. Aus diesen Größen lassen sich Rückschlüsse auf das Betriebssystem eines Rechners ziehen.
Die interessantesten TCP/IP Eigenschaften hierfür sind:

- TTL — Die Lebesndauer für ein ausgehendes Paket wird von den Betriebssystemen unterschiedlich gesetzt.
- Window Size — Auch die Window-Größe ist sehr unterschiedlich.
- DF — Viele, aber nicht alle Betriebssysteme setzen dieses Bit.
- TOS — Auch beim Type of Service Feld existieren Unterschiede.

Die folgende (nicht ganz aktuelle) Tabelle von http://project.honeynet.org/papers/finger/traces.txt gibt einen Überblick über die Bandbreite an unterschiedlichen Werte.

# OS	VERSION	PLATFORM	TTL	WINDOW	DF	TOS
DC-OSx	1.1-95	Pyramid/NILE	30	8192	n	0
Windows	9x/NT	Intel	32	5000-9000	y	0
NetApp	OnTap	5.1.2-5.2.2	54	8760	y	0
HPJetDirect	?	HP_Printer	59	2100-2150	n	0
AIX	4.3.x	IBM/RS6000	60	16000-16100	y	0
AIX	4.2.x	IBM/RS6000	60	16000-16100	n	0
Cisco	11.2	7507	60	65535	y	0
DigitalUnix	4.0	Alpha	60	33580	y	16
IRIX	6.x	SGI	60	61320	y	16
OS390	2.6	IBM/S390	60	32756	n	0
Reliant	5.43	Pyramid/RM1000	60	65534	n	0
FreeBSD	3.x	Intel	64	17520	y	16
JetDirect	G.07.x	J3113A	64	5804-5840	n	0
Linux	2.2.x	Intel	64	32120	y	0
OpenBSD	2.x	Intel	64	17520	n	16
OS/400	R4.4	AS/400	64	8192	y	0
SCO	R5	Compaq	64	24820	n	0
Solaris	8	Intel/Sparc	64	24820	y	0
FTX(UNIX)	3.3	STRATUS	64	32768	n	0
Unisys	x	Mainframe	64	32768	n	0
Netware	4.11	Intel	128	32000-32768	y	0
Windows	9x/NT	Intel	128	5000-9000	y	0
Windows	2000	Intel	128	17000-18000	y	0
Cisco	12.0	2514	255	3800-5000	n	192
Solaris	2.x	Intel/Sparc	255	8760	y	0

4.2. Weitere Informationen

Wenn der Serverbetreiber nicht aufpasst, dann kann man über das Netz eine Vielzahl weiterer Informationen über seinen Rechner abfragen.
Ein sehr nützliches Tool in diesem Zusammenhang ist das Programm hping, das inzwischen in der Version 2 zur Verfügung steht: http://www.hping.org/.
Im einfachsten Fall ist *hping* ein Ersatz für das übliche *ping*.

```
boss:~ # ping -c 2 www.lohbruegge.de
PING www.lohbruegge.de (212.227.109.22) from 80.137.182.236 : 56(84) bytes of data.
64 bytes from kundenserver.de (212.227.109.22): icmp_seq=1 ttl=248 time=70.473 msec
64 bytes from kundenserver.de (212.227.109.22): icmp_seq=2 ttl=248 time=68.319 msec

--- www.lohbruegge.de ping statistics ---
2 packets transmitted, 2 received, 0% loss, time 1003ms
rtt min/avg/max/mdev = 68.319/69.396/70.473/1.077 ms
```

Über den Schalter -c 2 legt man in beiden Programmen fest, dass nur zwei Pakete verschickt werden sollen (mit -1 bekäme man sogar ICMP-Pakete).

```
boss:~ # hping -c 2 www.lohbruegge.de
HPING www.lohbruegge.de (ppp0 212.227.109.22): NO FLAGS are set, 40 headers + 0 data
bytes
len=40 ip=212.227.109.22 flags=RA seq=0 ttl=248 id=9156 win=0 rtt=64.7 ms
len=40 ip=212.227.109.22 flags=RA seq=1 ttl=248 id=9892 win=0 rtt=65.1 ms

--- www.lohbruegge.de hping statistic ---
2 packets tramitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 64.7/64.9/65.1 ms
```

Spannend wird die Nutzung von hping dann, wenn man mit Rechnern zu tun hat, die auf ICMP-Meldungen nicht reagieren. Hping benutzt nämlich TCP und da kann man dann die Möglichkeiten des Protokolles voll ausnutzen.
Eines der Rechnersysteme, die per Ping nicht erreichbar sind ist www.microsoft.com

```
boss:~ # ping www.microsoft.com
PING www.microsoft.akadns.net (207.46.197.100) from 80.137.182.236 : 56(84) bytes of
data.

From 207.46.129.179: icmp_seq=14 Packet filtered
From 207.46.129.179 icmp_seq=14 Packet filtered
```

Ganz gelegentlich erhält man vom Router eine Meldung host unreachable, normalerweise verschwinden die Pakete einfach. Die IP-Adresse für www.microsoft.com variiert übrigens sehr stark, da hier viele Rechner hinter dem Namen stecken und über einen Lastverteiler zugeordnet werden.
Ein einfaches hping www.microsoft.com funktioniert hier aber auch nicht, da hping dann den TCP Port 0 anspricht, der hier auch vom Router gefiltert wird. Da aber auf dem Rechner sicherlich der Port 80 zur Verfügung steht könnte man diesen Port ansprechen, was interesanterweise auch einigen Paketen gelingt.

```
boss:~ # hping -p 80 www.microsoft.com
HPING www.microsoft.com (ppp0 207.46.230.220): NO FLAGS are set, 40 headers + 0 data
bytes
len=40 ip=207.46.230.220 flags=RA seq=1 ttl=50 id=43840 win=0 rtt=264.1 ms

--- www.microsoft.com hping statistic ---
18 packets tramitted, 1 packets received, 95% packet loss
round-trip min/avg/max = 264.1/264.1/264.1 ms
```

Um alle Pakete ans Ziel zu bekommen muss man noch das Syn-Bit setzen, dann klappt der Ping auch auf diesen Rechner.

```
boss:~ # hping -c 2 -p 80 -S www.microsoft.com
HPING www.microsoft.com (ppp0 207.46.230.219): S set, 40 headers + 0 data bytes
len=44 ip=207.46.230.219 flags=SA DF seq=0 ttl=49 id=26298 win=16616 rtt=269.5 ms
len=44 ip=207.46.230.219 flags=SA DF seq=1 ttl=49 id=47504 win=16616 rtt=264.2 ms

--- www.microsoft.com hping statistic ---
2 packets tramitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 264.2/266.8/269.5 ms
```

Selbstverständlich kann hping auch als Ersatz für Traceroute dienen:

```
boss:~ # hping  -p 80 -S -T www.microsoft.com
HPING www.microsoft.com (ppp0 207.46.230.220): S set, 40 headers + 0 data bytes
hop=1 TTL 0 during transit from ip=217.5.98.161 name=UNKNOWN
hop=1 hoprtt=48.5 ms
hop=2 TTL 0 during transit from ip=217.237.156.250 name=UNKNOWN
hop=2 hoprtt=48.2 ms
hop=3 TTL 0 during transit from ip=62.156.131.138 name=NYC-gw14.USA.net.DTAG.DE
hop=3 hoprtt=168.6 ms
hop=4 TTL 0 during transit from ip=4.25.133.5 name=so-2-1-0.nycmny1-hcr3.bbnplanet.net
hop=4 hoprtt=168.7 ms
hop=5 TTL 0 during transit from ip=4.0.7.10 name=p12-0.nycmny1-nbr1.bbnplanet.net
hop=5 hoprtt=169.0 ms
hop=6 TTL 0 during transit from ip=4.24.4.17 name=so-6-0-0.chcgil2-br2.bbnplanet.net
hop=6 hoprtt=186.4 ms
hop=7 TTL 0 during transit from ip=4.24.9.62 name=so-1-0-0.dnvtco1-br2.bbnplanet.net
hop=7 hoprtt=215.7 ms
hop=8 TTL 0 during transit from ip=4.24.11.37 name=so-7-0-0.dnvtco1-br1.bbnplanet.net
hop=8 hoprtt=213.8 ms
hop=9 TTL 0 during transit from ip=4.24.11.233 name=so-1-0-0.sttlwa2-br2.bbnplanet.net
hop=9 hoprtt=239.6 ms
hop=10 TTL 0 during transit from ip=4.24.11.189 name=so-7-0-0.sttlwa2-br1.bbnplanet.net
hop=10 hoprtt=240.8 ms
hop=11 TTL 0 during transit from ip=4.24.11.202 name=so-0-0-0.sttlwa1-hcr1.bbnplanet.net
hop=11 hoprtt=239.5 ms
hop=12 TTL 0 during transit from ip=4.24.10.234 name=so-7-0-0.sttlwa1-hcr2.bbnplanet.net
hop=12 hoprtt=241.2 ms
hop=13 TTL 0 during transit from ip=4.24.10.241 name=p1-0.sttlwa1-cr2.bbnplanet.net
hop=13 hoprtt=239.6 ms
hop=14 TTL 0 during transit from ip=4.25.89.6 name=p2-0.msseattle.bbnplanet.net
hop=14 hoprtt=262.8 ms
hop=15 TTL 0 during transit from ip=207.46.154.9 name=UNKNOWN
hop=15 hoprtt=268.9 ms
hop=16 TTL 0 during transit from ip=207.46.155.10 name=UNKNOWN
hop=16 hoprtt=263.9 ms
hop=17 TTL 0 during transit from ip=207.46.224.197 name=iustmscomc1201-ge-6-0.msft.net
hop=17 hoprtt=262.9 ms
len=44 ip=207.46.230.220 flags=SA DF seq=19 ttl=50 id=23864 win=16616 rtt=265.2 ms
```

Nach dem Erreichen des Zielrechners läuft alles wie ein normaler hping weiter.

Aus den Daten von hping kann man noch viel mehr entnehmen. Die Änderung der ID läßt Rückschlüsse auf den Datendurchsatz des Rechners zu. Damit man nicht jeweils erst die Differenz bilden muß kennt hping den Schalter -r, der genau das bewirkt.

```
boss:~ # hping  -p 80 -S -r www.hh.schule.de
HPING www.hh.schule.de (ppp0 134.100.212.78): S set, 40 headers + 0 data bytes
len=44 ip=134.100.212.78 flags=SA seq=0 ttl=55 id=43081 win=32736 rtt=68.5 ms
len=44 ip=134.100.212.78 flags=SA seq=1 ttl=55 id=+21 win=32736 rtt=67.4 ms
len=44 ip=134.100.212.78 flags=SA seq=2 ttl=55 id=+7 win=32736 rtt=66.8 ms
len=44 ip=134.100.212.78 flags=SA seq=3 ttl=55 id=+11 win=32736 rtt=67.9 ms
len=44 ip=134.100.212.78 flags=SA seq=4 ttl=55 id=+16 win=32736 rtt=67.7 ms
len=44 ip=134.100.212.78 flags=SA seq=5 ttl=55 id=+13 win=32736 rtt=66.4 ms

--- www.hh.schule.de hping statistic ---
6 packets tramitted, 6 packets received, 0% packet loss
round-trip min/avg/max = 66.4/67.4/68.5 ms
```

Die id muss sich zwischen zwei hpings mindestens um 1 erhöhen. Jede weitere Erhöhung ist ein Hinweis auf ein Datenpaket an einen anderen Rechner. Damit kann man den Traffic auf dem fernen Rechner abschätzen. Zum Vergleich ein Rechner mit großer Last:

```
boss:~ # hping  -p 80 -S -r www.microsoft.com
HPING www.microsoft.com (ppp0 207.46.230.219): S set, 40 headers + 0 data bytes
len=44 ip=207.46.230.219 flags=SA DF seq=0 ttl=49 id=5419 win=16616 rtt=268.5 ms
len=44 ip=207.46.230.219 flags=SA DF seq=1 ttl=49 id=+32872 win=16616 rtt=266.6 ms
len=44 ip=207.46.230.219 flags=SA DF seq=2 ttl=49 id=+721 win=16616 rtt=266.6 ms
len=44 ip=207.46.230.219 flags=SA DF seq=3 ttl=49 id=+29298 win=16616 rtt=267.6 ms
len=44 ip=207.46.230.219 flags=SA DF seq=4 ttl=49 id=+5536 win=16616 rtt=269.3 ms

--- www.microsoft.com hping statistic ---
5 packets tramitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 266.6/267.7/269.3 ms
```

Der Serverbetreiber kann die Informationsmöglichkeit unterbinden, dann wird als id immer 0 geliefert.

Aus den TCP-Daten kann man häufig sogar ermitteln, wie lange ein Rechner in Betrieb ist, dazu gibt es eine Timestamp-Feld. Aus der Differenz zwischen den Timestamps zweier Pakete kann hping die Uptime (Betriebszeit) errechnen:

```
boss:~ # hping  -c 2 -p 80 -S --tcp-timestamp www.hamburg.de
HPING www.hamburg.de (ppp0 62.181.130.64): S set, 40 headers + 0 data bytes
len=56 ip=62.181.130.64 flags=SA DF seq=0 ttl=249 id=39947 win=65500 rtt=60.4 ms
  TCP timestamp: tcpts=82284235

len=56 ip=62.181.130.64 flags=SA DF seq=1 ttl=249 id=39948 win=65500 rtt=59.0 ms
  TCP timestamp: tcpts=82284334
  HZ seems hz=100
  System uptime seems: 9 days, 12 hours, 34 minutes, 3 seconds


--- www.hamburg.de hping statistic ---
2 packets tramitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 59.0/59.7/60.4 ms
```

Diese Informationen liefern in der Regel nur Unix-Rechner und auch dort lassen sie sich relativ einfach unterbinden.

4.3. Nmap

Für das Sammeln von Informationen über einen Rechner im Internet setzt man in der Regel fertige Tools ein, die sog. Portscanner. Der Klassiker unter den Protscannern ist das Programm nmap von http://www.insecure.org/nmap/index.html. Viele der in diesem Kapitel beschriebenen Möglichkeiten zur Sammlung von Informationen stammen von den Entwicklern dieses Programmes.

Im einfachsten Fall ruft man *nmap* mit der Adresse des Zielrechners auf:

```
nmap www.zielrechner.de
```

nmap führt dann einen Portscan mit Verbindungsaufbau durch, das entspricht dem ausführlicheren Kommando

```
nmap -sT www.zielrechner.de
```

Der Schalter *-s* steht hierbei für Scan, der Parameter *T* für Connect.

Nmap kennt die folgenden Scanmethoden:

-sT Connect Scanning, führt mit jedem der interessierenden Ports auf dem Zielrechner einen Verbindungsaufbau durch. Diese Art des Scans wird auf den Zielrechnern aber sehr oft protokolliert bzw. abgeblockt.

-sS Syn Scanning (stealth), auch als halb offenes Scanning bezeichnet beruht auf der Nutzung von Datenpaketeten mit gesetztem Syn-Bit. Der Zielrechner antwortet mit gesetzten ACK+SYN Bits, wenn der Port offen ist bzw. mit RST wenn der Port nicht aktiv ist. Auch diese Scans finden sich oft in den Log-Dateien wieder.

-sF Fin stealth Scan, bei dem ein Paket mit gesetztem FIN-Bit genutzt wird. Offene Ports ignorieren dieses Bit meistens, geschlossene antworten mit einem RST-Paket. Man kann auch alle Schalter setzen (Xmas-Paket) -sX bzw. keineen Schalter (Null-Paket) -sN.

Weitere wichtige Schalter von nmap sind:

-O Aktiviert das OS-Fingerprinting, nmap versucht das Betriebssystem des Zielrechners zu ermittlen.

-v bzw. -vv erhöhen die Geschwätzigkeit von nmap.

Mit nmap kann man eine große Zahl von Rechnern auf einen Schlag scannen.

```
nmap -sS -O www.zielrechner.de/24
```

Scannt alle Rechner, deren IP in den ersten drei Oktetts mit der von www.zielrechner.de übereinstimmt, das können 254 Rechner sein.

5. Paketfilter zum Absichern von Rechnern

Ein Rechner mit Kontakt zum Internet bzw. anderen Datennetzen ist immer gefährdet. Man kann die bestehenden Risiken nicht verhindern, nur vermindern. Eine übliche Möglichkeit dazu sind Paketfilter. Paketfilter analysieren Datenpakete nach bestimmten Kriterien und verwerfen Pakete, die diesen Kriterien nicht genügen. Filter gibt es auf zwei Ebenen:

- Paketebene
- Applikationsebene

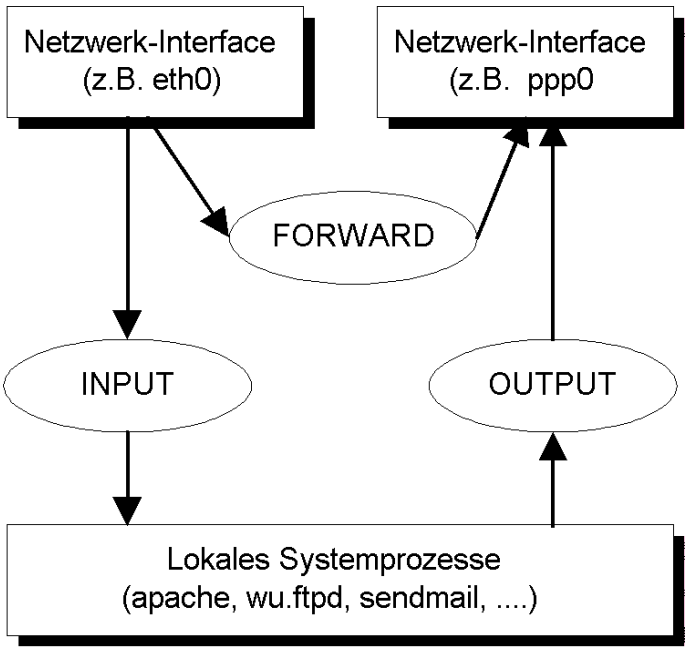
Sehr nützlich, aber auch speziell und meistens teuer sind Filter auf Applikationsebene. Ein derartige Filter wertet den Datenverkehr für einen speziellen Dienst aus und kann z.B. im Mailverkehr nach Viren suchen (läuft am GyLoh). Oder eine http-Filter könnte unerwünschte Daten bzw. Zugriffe von außen auf bestimmte Seiten blockieren. Zur Zeit gibt es immer mehr derartige Programme, relativ neu z.B. einen Filter der Samba-Zugriffe im Netz auf Viren untersucht.
Sehr weit verbreitet und sind Filter auf Paketebene. Hier wird jedes Datenpakte, das über ein Netzwerkgerät hereinkommt oder herausgeht, untersucht. Diese Untersuchungen sind mehr formaler Natur, es geht um Absender- oder Zieladressen, Protokolle bzw. die gesetzten Bits.
Paketfilter müssen zum Betriebssystem passen. Deshalb gibt es hier sehr unterschiedliche Systeme für Hardwarerouter, Windows- oder Linux-Systeme.
Auf Linux-Rechnern unterscheiden sich die verwendeten Paketfilter sogar je nach den Haupt-Kernelversionen. Für die Kernel 2.0.x war es ipfwadm, für 2.2.x ipchains und für die aktuellen 2.4.x Kernel ist es iptables.

5.1. Iptables

Iptables ist eine Erweiterung des älteren Systems ipchains. Genaugenommen sind diese Programme nicht wirklich Paketfilter, sondern sie steuern die Paketfilter, die in die jeweiligen Kernel bereits eingebaut sind. Der Kernel kennt drei Arten von Regeln (Chains):

- Input wendet er an, wenn ein Paket an einem Interface ankommt;
- Output wendet er an, bevor ein Datenpaket ein Interface verlässt;
- Forward benutzt er, wenn er ein Datenpaket von einem Interface zu einem anderen weiterleitet.

Jede Chain besteht aus einer Liste von Regeln, mit denen der Kernel jedes Datenpaket überprüft.



Die Regeln geben jeweils an, was zu tun ist, wenn der Header des Paketes einen bestimmten Aufbau besitzt. Wenn das Paket nicht den beschriebenen Aufbau hat, wendet der Kernel die nächste Regel an.

Als Ergebnis dieser Überprüfung ergibt sich für das Datenpaket eine der Möglichkeiten:
- ACCEPT, der Kernel transportiert das Paket weiter;
- DROP, er verwirft das Paket ohne Rückmeldung;
- REJECT, er verwirft das Paket, informiert aber per ICMP den Absender.

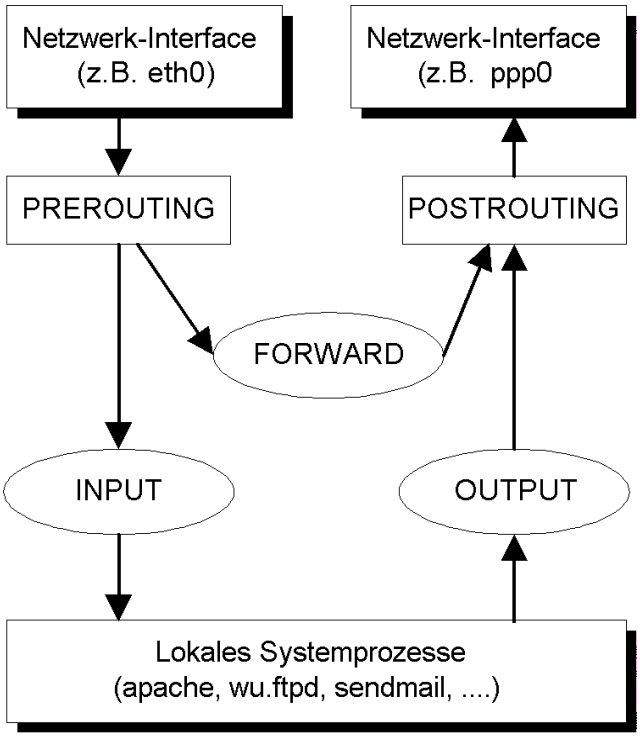
Eine wichtige Änderung gegenüber IPchains besteht darin, dass Forwarding-Pakete, also solche, die nicht für den Rechner selber bestimmt sind, bei iptables nur noch die Forward-Chain durchlaufen. Die Input- und Output-Regeln spielen für diese Pakete keine Rolle. Im Paket-Header kann iptables u.a. folgende Informationen mit Regeln überprüfen:
- Absender-IP und -Port (-s Source),
- Ziel-IP und -Port (-d Destination),
- Protokoll (-p Protocol).

Fragt man die eingestellten Regeln mit *iptables -L* ab, so erhält man:

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Für alle drei Chains liegt die Standard-Regel (Default-Policy) auf ACCEPT. Der Kernel wendet die Default-Policy an, wenn er ansonsten keine passende Regel findet.

Interessant ist vor allem die Forward-Chain. Hier leitet der Kernel momentan nur weiter, was für ein privates Netz unpraktisch ist, da der erste Router im Internet die Datenpakete aufgrund ihrer privaten IP-Adressen verwirft. Hier muss man noch erreichen, dass der Kernel bei Datenpaketen aus dem lokalen Netz die private IP-Adresse des Absenders durch seine gültige IP-Adresse ersetzt. Dazu gibt es bei IPTABLES neben den bisher angesprochenen Chains die zwei zusätzlichen Bereiche PREROUTING und POSTROUTING.



5.2. Masquerading

Wenn ein Datenpaket erfolgreich die FORWARD-Chain durchlaufen hat, danach also z.B. über ppp0 ins Internet gehen würde, muss die Absenderadresse geändert, also das Paket maskiert werden.

Die bisher angesprochenen Chains INPUT, OUTPUT und FORWARD gehören zur Default-Table *filter*, während PREROUTING und POSTROUTING zur Table *nat* gehören.
Die Table filter muss in den Regeln nicht explizit angeben, wohl aber die Table nat (network address translation), die für alle Veränderungen der Adressinformationen, also auch das Masquerading, zuständig ist.. Folgendermaßen fügt msn (-A) die Masquerading-Regel für das Output-Device (-o ppp0) an die POSTROUTING Chain der Table nat (-t nat) an:

```
iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
```

Sollte man bei der Eingabe dieser Regel eine Fehlermeldung erhalten, so ist vermutlich das Kernel-Modul für Nat nicht geladen; in diesem Fall gibt man

```
modprobe iptable_nat
```

ein und wiederholt danach die Nat-Regel.

Falls man sich beim Eintippen der Regeln verschreiben sollten, muss man die Regeln auch wieder loswerden können. Alle eingegebenen Regeln kann man auf einen Schlag löschen. Mit

```
iptables -F
```

löscht man alle Regeln der Default-Table filter und mit

```
iptables -t nat -F
```

alle Regeln der Table nat.

Mit der oben angegebenen Nat-Regel haben alle Rechner im Netz fast vollen Internet-Zugriff. Nur ein paar Dienste machen noch Probleme. Dazu gehört FTP, da dieser Dienst mit zwei verschiedenen Ports arbeitet. Über den Datenkanal empfängt man per FTP Pakete, die man über den Kommandokanal angefordert hat. Darauf ist die hier beschriebene Firewall bisher nicht eingestellt. Für die meisten problematischen Dienste gibt es inzwischen Module, die diese Probleme überwinden können. Diese Module muss man noch laden. Eine Lösung besteht darin, das folgende Programm zu erstellen, welches die Default-Policy auf Masquerading stellt und die benötigten Module lädt. Das Script beruht auf dem Muster-Script sekeleton von SuSE:

```
#! /bin/sh
#
# /etc/init.d/maske
#
#   and symbolic its link
#
# /sbin/rcmaske
#
# System startup script for Masquerading
#
### BEGIN INIT INFO
# Provides: maske
# Required-Start: serial
# Required-Stop:
# Default-Start:  2 3 4 5
# Default-Stop:
# Description:    Start simple Firewall- Skript
```

```
### END INIT INFO

# Source SuSE config
. /etc/rc.config
. /etc/rc.config.local
# Determine the base and follow a runlevel link name.
base=${0##*/}
link=${base#*[SK][0-9][0-9]}

# Force execution if not called by a runlevel directory.
test $link = $base && START_MASKE=yes
test "$START_MASKE" = yes || exit 0

IPTABLES=/usr/sbin/iptables
MODPROBE=/sbin/modprobe
test -x $IPTABLES || exit 5
test -x $MODPROBE || exit 5

. /etc/rc.status

rc_reset

fw_dev="ppp0"

case "$1" in
    start)
        echo -n "Starting Maske Firewall Skript"
        $MODPROBE iptable_nat
        $MODPROBE ip_nat_ftp
        $MODPROBE ip_conntrack
        $MODPROBE ip_conntrack_ftp
        $IPTABLES -F
        $IPTABLES -t nat -F
        $IPTABLES -t nat -A POSTROUTING -o $fw_dev -j MASQUERADE
  # Remember status and be verbose
        rc_status -v
        ;;
    stop)
        echo -n "Shutting down Maske Skript"
        $IPTABLES -F
        $IPTABLES -t nat -F

        # Remember status and be verbose
        rc_status -v
        ;;
*)
        echo "Usage: $0 {start|stop}"
        exit 1
        ;;
esac
rc_exit
```

Das Script muss man noch mit

```
chmod  u+x  /etc/init.d/maske
```

ausführbar machen und eine Datei /etc/rc.config.local mit folgendem Inhalt anlegen:

```
#
# /etc/rc.config.local
#
START_MASKE="yes"
```

oder die Zeile mit einem "#" auskommentieren, welche die /etc/rc.config.local einbindet.

Man kann das Maske-Script gut als Grundlage für eigene Experimente benutzen. Wenn es beim Systemstart automatisch aktiviert werden soll, dann muss man mit

```
insserv  maske
```

die entsprechenden Links setzen lassen. Damit ist das Masquerading vollständig funktionsfähig.

5.3. Firewalling

Die bisherigen Informationen über Paketfilterung reichen erst einmal aus, um das Masquerading zu aktivieren. Will man eine genauere Kontrolle über die Pakete haben, so muss man tiefer in den Umgang mit IPTABLES einsteigen. Bezogen auf eine gesamte Chain kann man:

- Die Policy für eine eingebaute Chain ändern (-P);
- Alle Regeln in einer Chain listen (-L);
- Alle Regeln in einer Chain löschen (-F).

Bezogen auf einzelne Regeln kann man:

- Eine Regel an eine Chain anfügen (append) (-A);
- Eine Regel in eine Chain einfügen (insert) (-I);
- Eine Regel in einer Chain ersetzen (replace) (-R);
- Eine Regel in einer Chain löschen (delete) (-D);
- Die erste Regel in einer Chain, die zutrifft, löschen (-D).

Dabei spielt die Reihenfolge eine wichtige Rolle. Der Kernel arbeitet die erste Regel ab, die zutrifft. Spätere Regeln spielen dann keine Rolle mehr. Neben den drei vorgegebenen Chains kann man auch eigene Chains (Benutzer-Chains) einrichten, um aufwändigere Regelwerke besser zu strukturieren . Für eigenen Chains gibt es folgende Regeln:

- Eine Benutzerchain definieren und bennenen (-N);
- Eine (leere) Benutzerchain löschen (-X).

Wir werden im weiteren Verlauf des Kapitels ein Beispiel mit einer Benutzerchain kennenlernen. Der erste Parameter von IPTABLES gibt üblicherweise an, was man machen möchte (append, insert, ...). Danach gibt man an, auf welche Chains sich die Regel beziehen soll und zuletzt die eigentliche Regel. Ein paar kleine Beispiele:

```
iptables -P FORWARD DROP
```

Setzt die Policy für die Forward-Chain auf DROP. Alle Pakete zwischen den Interfaces würde der Kernel also abweisen, wenn er nicht noch eine passende positive Regel in der Chain findet.

```
iptables  -A FORWARD -s  192.168.1.51  -j DROP
```

Diese Regel unterbindet das Weiterleiten aller Datenpakete vom Rechner mit der IP-Adresse 192.168.1.51. Dieser Rechner kann aber noch auf lokale Serverdienste, wie z.B. Squid und den Apache zugreifen, aber nicht direkt aufs Internet. Für die Regel überprüft der Kernel hier den Absender (-s). Trägt das Datenpaket die angegebene IP-Adresse als Absender, springt die Regel zu DROP (-j), das Paket wird verworfen. Absender- und Zieladresse eines Paketes bestehen aus der Angabe von Adresse und Port: 192.168.1.2 80 (WWW-Port des Servers).

Statt der IP-Adresse kann man auch den Namen angeben. Gleichbedeutend wäre also

```
boss.lokales-netz.de 80
```

Da man oft mehrere ähnliche Adressen ansprechen will, kann man Gruppen angeben. Bei 192.168.1.0/24 fällt die IP unter unsere Regel, wenn die ersten 24 Bit der IP diesem Muster entsprechen. Hat man keine IP angegeben, so sind alle Adressen gemeint, was man auch konkret mit 0/0 angeben könnte. Wenn man keine Angabe über Ports gemacht hat, bezieht sich das Muster auf alles Ports. Man kann jedoch wie oben einen Port einzeln angeben oder mit von:bis einen Bereich von Ports. Mit 30:144 würde man also alle Ports von 30 bis 144 erreichen, mit :144 alle Ports von 0 bis 144, da die erste Angabe fehlt. Entsprechend wäre eine fehlende zweite Angabe mit der höchsten Portnummer identisch. Ports können nicht nur über ihre Nummern angeben werden, sondern auch über ihre Bezeichnung:

```
boss.lokales-netz.de www
```

Bisher haben wir kein Protokoll angegeben, also gilt die Regel für alle Protokolle. Im folgenden Beispiel (aus dem IPTABLES-HOWTO) unterbindet man einen ping auf 127.0.0.1 (Loopback-Device). Ping benutzt das Protokoll ICMP. Vor Anwendung der Regel sollte man sich mit

```
ping -c 1 127.0.0.1
```

überzeugen, dass man in der Grundeinstellung hier ein einzelnes (-c1) Paket erfolgreich übertragen kann.

```
iptables -I INPUT -s 127.0.0.1 -p icmp -j DROP
```

Damit wird der nächste ping von diese Adresse aus nicht mehr funktionieren, da das Antwortpaket nicht mehr durch die Firewall kommt. Ping wartet übrigens sehr lange, bevor er mit einer Fehlermeldung aufgibt. Ungeduldige brechen vorher mit (Strg)+(C) den Befehl ab. Bleibt zu klären, wie man diese Regel wieder löschen kann. Da wir wissen, dass die Regel die einzige bzw. erste Regel in der Chain Input ist, kann man sie mit

```
iptables -D INPUT 1
```

löschen. Das -D steht hier für Delete und erwartet die Angabe der Chain und die Nummer der Regel. Bei vielen Regeln ist dieser Weg unübersichtlich; dann ist es einfacher, die Regel mit dem Parameter -D noch einmal anzugeben:

```
iptables -D INPUT -s 127.0.0.1 -p icmp -j DROP
```

Bei den Regel-Parametern gibt es folgende Angaben:
-s Adresse(n) inklusive Port (Source),
-d Adresse(n) inklusive Port (Destination),
-i Device (Interface) + als Wildcard erlaubt,
-p Protokoll,
-j Aktion (Target),

5.3. Sicherheitsphilosophien

Bei der Arbeit mit IPTABLES gibt es zwei grundsätzliche Strategien:
Vertrauen: Alles ist erlaubt, was nicht explizit verboten ist. Die Default-Policies stellt man bei diesem Ansatz auf ACCEPT.
Misstrauen: Alles ist verboten, was nicht explizit erlaubt wurde. Die Default-Policies stellen man dann auf REJECT oder DROP.
Die größere Sicherheit bietet der misstrauische Ansatz. Er macht aber auch viel Arbeit, wenn man mehrere Dienste oder Protokolle freischalten muss. Man sollte hier sehr genau überlegen, welche sinnvollen Anforderungen Anwender im Netz haben und welche Anwendungen wirklich eine Rolle spielen. Erst dann kann man entscheiden, mit welcher Strategie man an die Firewall herangeht.

5.4. Ein praktisches Beispiel

Für ein kleines lokales Netz sollten Sie das Forwarding nur für die Rechner im lokalen Netz ermöglichen, neue Anfragen von außen sollten Sie normalerweise verwerfen. Die folgenden Regeln (frei nach dem Filtering HOWTO) zeigen das exemplarisch:

```
# definierten Zustand erstellen und alle Regeln löschen
iptables -F
iptables -t nat -F
# Ein Router sollte Pakete vom Typ destination-unreachable bearbeiten
iptables -A INPUT -i ppp0 -p icmp --icmp-type destination-unreachable
-j ACCEPT
# Kette erstellen, die neue Verbindung blockt, es sei denn, sie
# kommen von innen
iptables -N block
iptables -A block -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A block -m state --state NEW -i ! ppp0 -j ACCEPT  iptables
-A block -j DROP
# Von INPUT und FORWARD Ketten zu dieser Kette springen
iptables -A INPUT -j block
iptables -A FORWARD -j block
# Maskieren der lokalen Rechner
iptables -t nat -A POSTROUTING -o ppp0  -j MASQUERADE
```

In den ersten Zeile löscht man erst einmal alle Regeln der Table filter und der Table nat. Dann legt man eine Benutzerchain block an, die einerseits Pakete durchläßt, die Antwortpakete sind (Status ESTABLISHED, ...) oder neue Pakete, die nicht über ppp0, also das Internet hereinkommen. Diese Regeln bindet man dann in die INPUT und die FORWARD-Chain ein.
Zuletzt aktiviert man noch das Masquerading, das dann auf die Pakete wirkt, die die FORWARD-Chain erfolgreich passiert haben. Der Rechner antwortet damit auf keinerlei Anforderungen aus dem Internet, nicht einmal einen Ping beantwortet er. Aus dem lokalen Netz heraus, und von Server selber aus hat man aber vollen Zugriff auf das Internet.
Soll der Server auf ping reagieren, dann muss am Anfang des Listings folgende Zeile ergänzt werden.

```
iptables -A INPUT -i ppp0 -p icmp --icmp-type echo-request -j ACCEPT
```

Soll der Server auch öffentlich Dienste anbieten, so muss man diese explizit freischalten, beispielsweise den Port 80 für einen Webserver:

```
iptables -A INPUT -i ppp0 -p tcp --dport 80 -j ACCEPT
```

Diese Regel muss aber vor der Definition der Benutzerchain block stehen, da sie sonst nicht mehr berücksichtigt wird.

5.5 Accounting Rule

Der Kernel zählt für jede Regel mit, wie viele Datenpakete er der Regel unterworfen hat. Bezogen auf das vorangegangene Beispiel liefert

```
iptables  -L block  -v
```

die folgende Ausgabe (nach erfolgter Nutzung):

```
Chain block (2 references)
 pkts bytes target     prot opt in     out     source     destination
 184  9388 ACCEPT     all  --  any    any     anywhere   anywhere
     state RELATED,ESTABLISHED
  22  1824 DROP       all  --  any    any     anywhere   anywhere
```

Der Kernel hat über die erste Regel 184 Pakete akzeptiert und über die zweite Regel 22 Pakete abgelehnt. Will man generell zählen, wie viele Daten ein bestimmter Rechner

ins Internet übertragen hat, so ist die einfachste Regel eine ohne Ziel. Eine derartige Regel nennt man auch accounting rule, weil sie nur zum Zählen von Paketen, dem Accounting geeignet ist:

```
iptables -I INPUT  -s  192.168.1.1
```

Diese Regel zählt alle Pakete vom Host 192.168.1.1. Über den Schalter -I statt -A fügt man diese Regel am Anfang der Chain ein und hängen Sie nicht am Ende an, was keinen Effekt mehr hätte. Der Befehl

```
iptables -L INPUT -v
```

zeigt die Summe von Bytes und Paketen an, die das Interface passiert haben, nachdem die Regel zutreffend war, um das Datenaufkommen in einem Netz sehr differenziert auszuwerten. Zurücksetzen kann man die Zähler über iptables -Z, konkret für das letzte Beispiel mit

```
iptables -Z INPUT
```

5.6. Logging-Rule

Neben der bereits angesprochenen Möglichkeit die Pakete zu zählen hat man mit IPTABLES auch eine einfache Möglichkeit Zugriffe gezielt zu protokollieren. Angenommen, es sollen Zugriffe auf den Telnet-Port 23 nicht nur gesperrt, sondern auch protokolliert werden wer wann versucht auf diesen Port zuzugreifen. Dann fügen wir folgende Regel ein:

```
iptables -I INPUT -p tcp --dport 23 -j LOG --log-prefix
"Telnet-Zugriff: "
```

Das neue Sprungziel *LOG* protokolliert Zugriffe in den Dateien */var/log/messages* und */var/log/warn*. Im Gegensatz zu anderen Sprungzielen beendet LOG den Ablauf nicht, die weiteren Regeln arbeitet der Kernel also ganz normal ab. Um das Auswerten der Logdateien zu erleichtern, kann man optional mit --log-prefix einen individuellen Textes angeben. Ein Versuch eines Telnet-Zugriffs auf Ihren Server hinterläßt folgende Einträge in der Log-Datei.

```
Jan  4 14:58:22 boss kernel: Telnet-Zugriff: IN=eth0 OUT=
MAC=00:50:bf:55:8d:46:00:50:bf:58:56:fd:08:00 SRC=192.168.1.56
DST=192.168.1.2 LEN=48 TOS=0x00 PREC=0x00 TTL=128 ID=19228 DF
PROTO=TCP SPT=1092 DPT=23 WINDOW=8192 RES=0x00 SYN URGP=0
```

Diese hält Datum, Uhrzeit sowie die IP- und die MAC-Adresse des aufrufenden Rechners fest.

5.7. Limits

Zu den neuen Funktionen von iptables gehört die Möglichkeit Zugriffe in ihrer Häufigkeit zu beschränken. Eine Möglichkeit einen Rechner lahm zu legen besteht darin, ihn von vielen Rechnern im Netz aus mit einem Ping zu belasten. Im schlimmsten Fall mit dem Parameter -f (flood)

```
ping  -f  www.bei-mir-nicht.de
```

Damit schickt der Ping-Befehl seine Datenpakete so oft wie irgend möglich an den Zielrechner. Wenn das mehrere Hacker gleichzeitig machen, dann kann dies zu einem Zusammenbruch des Zielrechners führen. Mit der Limit-Option (-m limit) und deren Parameter --limit 1/sec kann man einen derartigen Angriff unterbinden:

```
iptables  -A INPUT -i ppp0 -p icmp --icmp-type echo-request -m limit
--limit 1/sec -j ACCEPT
```

Der Rechner beantwortet jetzt nur noch einen Ping pro Sekunde. Auch für manche Dienste macht Beschränkung der Anzahl der Zugriffe Sinn. Da es in der letzten Zeit viele Angriffe auf den SSH-Dämon gab sollten Sie diesen entsprechend schützen. Sie dürfen aber nicht alle Pakete zum SSH-Port limitieren, dass würde ja Ihre Arbeitsgeschwindigkeit beschränkem, sondern nur Pakete für den Verbindungsaufbau.

```
iptables -A INPUT -p tcp --dport 22 -m limit --limit 1/sec -m state
--state NEW  -j ACCEPT
```

Mit dieser Regel können Sie pro Sekunde nur eine Verbindung per SSH aufbauen, nach den Verbindungsaufbau ist der Status der Pakete nicht mehr NEW, die Regel stört dann also nicht während der Verbindung.