14 Firewalling und Masquerading

In der bisherigen Konfiguration bildet der Linux-Server eine recht brauchbare Firewall (Brandmauer): Er erlaubt keinerlei direkte Verbindung zwischen einem Rechner im Intranet und einem Rechner im Internet. Das ist die extremste Form einer Firewall.

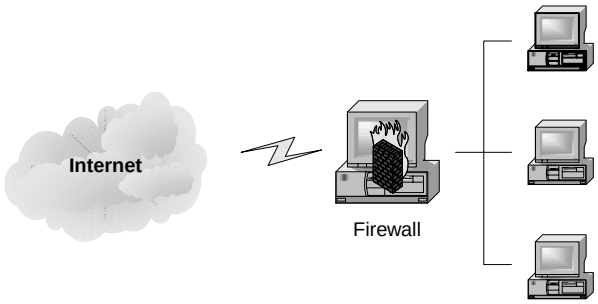

Abbildung 14.1: Firewall

Da Sie immer den Server als Stellvertreter benutzt haben, können Sie trotzdem Internet-Dienste nutzen. Mails z.B. haben Sie nur zwischen Client und Server ausgetauscht. Der Server wiederum hat dann die Mail mit dem Internet-Rechner des Providers ausgetauscht. Webseiten haben Sie über den Proxy-Cache Squid, also wieder vom Server bezogen. Squid hat die Seiten aus dem Internet geholt. In den bisherigen Beispielen tauchte nie eine direkte Internet-Verbindung auf.

In diesem Kapitel lernen Sie, wie man, unter Berücksichtigung von Sicherheitsaspekten, einen direkten Zugriff von lokalen Rechnern aus auf das Internet ermöglicht.

Dazu erfahren Sie zuerst etwas über die Grundlagen des Routing, Forwarding und Masquerading und dann über das Konfigurieren eines einigermaßen internettauglichen Routers.

14.1 Grundlagen

14.1.1 TCP/IP Das Internet-Protokol

TCP/IP ist eine Sammlung von Internet-Protokollen mit unterschiedlichen Aufgaben.

- Grundlage ist das *Internet-Protokol* (IP), ein verbindungsloses Protokoll ohne Komponenten zur Sicherung der Datenübertragung. Zu den Aufgaben von IP gehört es, Datenpakete zu adressieren. Dazu dient die IP-Adresse aus einer 4Byte-Zahl, die man üblicherweise in der Form 192.168.1.1 darstellt. Eine weitere Aufgabe von IP ist das Aufteilen der Daten in Pakete, die die darunter liegende Schicht (z.B. Ethernet) übertragen kann, sowie das korrekte Zusammensetzen der übertragenen Pakete beim Empfänger. Auf diesem Internet-Protokoll setzen weitere Protokolle auf.
- Das verbindungsorientierte *Transmission Control Protocol* (TCP), das bekannteste Protokoll auf dieser Ebene, setzt auf IP auf. Bevor es mit der eigentlichen Datenübertragung beginnt, baut es zunächst eine Verbindung zum Empfänger auf. Dann erst schickt es die Datenpakete ab, die der Empfänger quittieren muss. Bleibt diese Empfangsbestätigung aus, so schickt es das entsprechende Paket erneut. Hierdurch ist sichergestellt, dass die Datenpakete vollständig beim Empfänger ankommen und TCP sie dort wieder in die richtige Reihenfolge bringt. Die Reihenfolge kann sich beim Versand verändern, da IP sich für jedes Paket andere Wege durchs Netz mit unterschiedlichen Laufzeiten suchen kann. Nach erfolgreicher Datenübertragung baut TCP die Verbindung zwischen den Rechnern wieder ab. Die Verwaltung der Verbindung kostet Zeit und Übertragungskapazität. Daher gibt es für weniger sensible Verbindungen weitere Protokolle.
- UDP, das *User Datagramm Protocol*, ist ein verbindungsloses Protokoll. Es dient zum Übertragen kurzer Nachrichten. Nameserver-Anfragen werden über UDP abgewickelt. Wenn keine Antwort kommt, dann wird einfach eine neue Anfrage gestellt, eventuell an einen anderen Nameserver. Auch Streaming-Video und Netzwerkspiele arbeiten oft mit UDP, hier geht es vor allem um die höhere Performance. Außerdem ist es hier nicht weiter tragisch bzw. sowieso nicht reparabel, wenn ein Datenpaket verloren ist.
- 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 schickt der Zielrechner die Fehlermeldung `Port unreachable` per ICMP zurück. Auch Routing-Informationen und Ping werden über ICMP weitergeleitet.

14.1.2 Kontaktformen

Haben Sie für die Rechner im lokalen Netz offizielle IP-Adressen (s.u.), so müssen Sie nur erreichen, dass der Linux-Server Datenpakete vom Device für das lokale Netzwerk (`eth0`) auf das Device für die Internetanbindung (`ippp0` oder `ppp0`) weiterleitet. Diese Weiterleitung bezeichnet man als *IP-Forwarding*.

Besitzen die Rechner im lokalen Netz private Adressen, so hilft IP-Forwarding nicht viel, weil der erste Router im Internet die Anfragen Ihrer lokalen Rechner sofort verwerfen würde. Router im Internet sind so konfiguriert, dass sie Anfragen von oder an private Netz-Adressen nicht weiterleiten. In diesem Fall müsste der Server in der Anfrage die lokale IP des Clients durch seine legale, bei der Anwahl übermittelte IP ersetzen. Trifft die Antwort ein, so muss er die Server-IP wieder durch die Client-IP ersetzen, damit er die Daten lokal zustellen kann. Diese IP-Ersetzung bezeichnet man als *Masquerading*.

Direkter Kontakt mit dem Internet ist für einen Rechner immer gefährlich. Im Internet sind Millionen von Benutzern unterwegs, von denen einige sich ihre Zeit damit vertreiben, fremde Rechner anzugreifen. Will man seine Rechner schützen, so muss man alle Datenpakete filtern und verdächtige Pakete entfernen, also wieder eine Firewall aufbauen.

Wer wirklich auf Sicherheit bedacht ist, wird Forwarding und Masquerading vermeiden, da ohne diese beiden Funktionen die Rechner im lokalen Netz von außen nicht erreichbar und damit auch nicht angreifbar sind. Dafür sind umgekehrt die Rechner im Internet auch nicht direkt erreichbar.

Will man erlauben, dass Nutzer direkt mit fremden Rechnern kommunizieren, sich also mit einem fremden Mailserver (z.B. `gmx.de`) oder mit einem fremden News-Server verbinden, so muss man in der einen oder anderen Form Datenpakete weiterleiten.

14.1.3 Forwarding

Will man ein Netz mit offiziellen IP-Adressen über eine Leitung an das Internet anbinden, so muss der Gateway-Rechner sowohl eine Verbindung mit dem lokalen Netz (`eth0`) als auch mit dem Internet (`eth1`, `ippp0` oder `ppp0`) besitzen und Pakete zwischen diesen beiden Geräten weiterleiten.

Tipp: Offizielle IP-Adressen sind solche, die der Provider fest vergibt. Zwei der Adressen (die niedrigste und die höchste) gehen für die Netzwerkadresse und die Broadcastadresse ab. Hat man z.B. acht solcher Adressen, so kann man damit 6 Rechner versorgen.

Um diese Weiterleitung zu aktivieren, müssen Sie auf ihrem Linux-Server das nach Voreinstellung abgeschaltete IP-Forwarding aktivieren. Dazu geben Sie folgenden Befehl an der Konsole ein:

```
echo  "1"  >  /proc/sys/net/ipv4/ip_forward
```

Damit schreiben Sie eine "1" an die Speicherstelle, die den Kernel anweist, das Forwarding zu aktivieren. Deaktivieren können Sie das Forwarding dann mit:

```
echo  "0"  >  /proc/sys/net/ipv4/ip_forward
```

Abfragen können Sie den Schalterzustand mittels:

```
cat  /proc/sys/net/ipv4/ip_forward
```

Sollten Sie hierbei Fehlermeldungen erhalten, so unterstützt Ihr Kernel kein Forwarding. In diesem Fall müssen Sie Ihren Kernel neu kompilieren. Zum Glück sind die Standardkernel der üblichen Distributionen passend eingerichtet, so dass wir auf das Erzeugen eines neuen Kernels hier nicht weiter eingehen.

Um das Forwarding dauerhaft zu aktivieren, müssen Sie den angegebenen Befehl in Ihr Boot-Script aufnehmen. Bei SuSE-Linux gehen Sie dazu in YaST auf *Administration des Systems • Konfigurationsdatei verändern* und suchen in der Parameterliste den Schalter *IP_FORWARD* und setzen ihn auf *yes*.

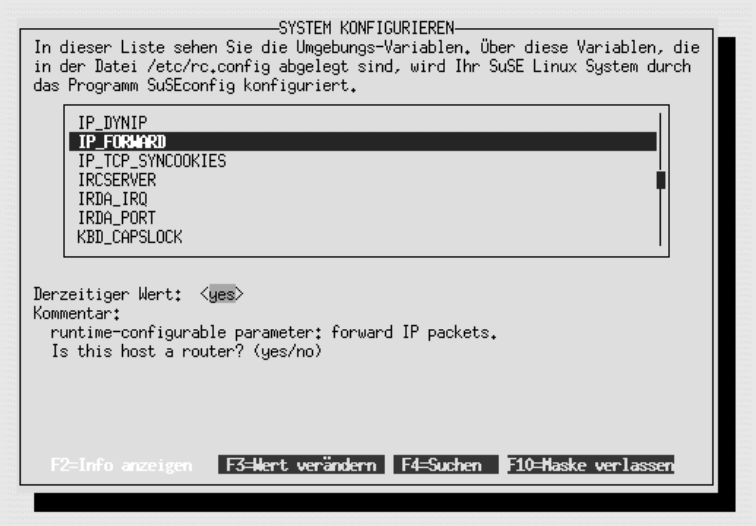

Abbildung 14.2: YaST –IP_FORWARD

Damit das Script `/sbin/init.d/boot` diesen Schalter auswerten kann, sollte man den Linux-Server rebooten, um die Einstellung zu aktivieren.

Wenn die Routen richtig gesetzt sind, dann besteht jetzt eine direkte Verbindung zwischen allen Rechnern im lokalen Netz und dem Internet.

Man kann das z.B. mit einem `ping` von einem Client-Rechner im Netz testen:

```
ping www.linuxbu.ch
```

Hier erfolgt jetzt eine Rückmeldung:

```
PING wird ausgeführt für www.linuxbu.ch [195.37.188.187] mit
32 Bytes Daten:
Antwort von 195.37.188.187: Bytes=32 Zeit<10ms TTL=255
Antwort von 195.37.188.187: Bytes=32 Zeit<10ms TTL=255
Antwort von 195.37.188.187: Bytes=32 Zeit<10ms TTL=255
Antwort von 195.37.188.187: Bytes=32 Zeit<10ms TTL=255
Ping-Statistik für 195.37.188.187:    Pakete: Gesendet = 4,
Empfangen = 4, Verloren = 0 (0% Verlust),Ca. Zeitangaben in
Millisek.:    Minimum = 0ms, Maximum =  0ms, Mittelwert =  0ms
```

14.1.4 Grundlagen zum Routing

Router ermöglichen den Datenaustausch zwischen zwei Netzwerken (Subnetzen). Dabei dürfen die Subnetze eine unterschiedliche Hardwarebasis besitzen, wie das z.B. bei Ethernet und Telefonleitungen der Fall ist. Wichtig ist nur, dass beide Netze mit dem gleichen Protokoll, TCP/IP arbeiten.

Für einen Datentransport zwischen Subnetzen benötigt das System Information über die IP-Adressen und die zugehörigen Net-Devices. Die statischen Informationen stehen in der Datei `/etc/route.conf` (dargestellt sind hier nur Auszüge).

```
# Destination     Dummy/Gateway  Netmask          Device
#
192.168.1.0       0.0.0.0        255.255.255.0    eth0
194.95.238.253    0.0.0.0        255.255.255.255  ippp0
```

Die erste Zeile legt fest, das alle IP-Adressen von `192.168.1.0` bis `192.168.1.255` über das Device `eth0` erreichbar sind (255 Adressen, da die letzte Stelle der Netmask `0` ist). Ein Gateway muss nicht angegeben werden, denn das wäre der Rechner selbst, also steht hier nur der Dummy `(0.0.0.0)`.

Die zweite Zeile beschreibt eine ISDN-Verbindung mit fester IP. Die vom Provider angegebene Adresse (remote IP) ist `194.95.238.253`. Die Netzmaske `255.255.255.255` gibt an, dass zu diesem Device nur eine einzige IP gehört. Hätte man 255 IP-Adressen vom Provider bekommen, so müsste die zweite Zeile lauten

```
194.95.238.0      0.0.0.0         255.255.255.0    ippp0
```

Als Gateway dient hier wieder der Linux-Server selber.

Damit ist definiert, wie Datenpakete die IP-Adressen `192.168.1.1` bis `192.168.1.254` (`eth0`) und `194.95.238.253` (`ippp0`) erreichen können.

Nirgends ist aber festgelegt, wohin Anfragen an z.B. `195.37.188.187` (`www.linuxbu.ch`) gehen sollen.

Eine Möglichkeit wäre, dies konkret festzulegen:

```
#Host              Gateway (Provider IP)  Netmask
195.37.188.187     194.95.238.253         255.255.255.255
```

Statt für alle, die man erreichen möchte, Adressen einzutragen, definiert man einfacher ein Default-Gateway:

```
# default                 Provider IP
0.0.0.0                   194.95.238.253
```

Nun leitet der Linux-Router alle Anfragen, für die kein Routing festgelegt ist, an die angegebene IP weiter.

Diese Datei soll die statischen Routen, die immer vorhanden sind, konfigurieren. Der Router wertet die Eintragungen beim Start des Systems aus und übergibt sie an das Programm `/sbin/route`. Routen lassen sich auch im laufenden Betrieb setzen und löschen.

Beim Einrichten von Routen muss man unterscheiden zwischen

- solchen, die ein Device definieren, diese kann man mit `/sbin/ifconfig` (Interface konfigurieren) bearbeiten und
- solchen, die nur einen Weg definieren, diese kann man mit `/sbin/route` (Weg) verändern.

Ohne Parameter aufgerufen, zeigen beide Befehle die aktuellen Definitionen an:

Ausgabe von `/sbin/ifconfig`:

```
eth0      Link encap:Ethernet  HWaddr 00:80:5F:84:5E:A6
          inet addr:192.168.1.2  Bcast:192.168.1.255
Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:212 errors:0 dropped:0 overruns:0 frame:0
          TX packets:180 errors:0 dropped:0 overruns:0
carrier:0
          collisions:0 txqueuelen:100
          Interrupt:5 Base address:0x7000
```

```
ippp0     Link encap:Point-to-Point Protocol
          inet addr:194.95.53.7 P-t-P: 194.95.238.253
Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP DYNAMIC  MTU:1500
          ➥ Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:30

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:3924  Metric:1
          RX packets:123 errors:0 dropped:0 overruns:0 frame:0
          TX packets:123 errors:0 dropped:0 overruns:0 q
          ➥ carrier:0
          collisions:0 txqueuelen:0
```

Neben den Devices `eth0` und `ipp0` ist noch ein Device `lo` vorhanden, ohne dass man es irgendwo definiert hätte. Dieses Loopback-Device ist ein Pseudo-Device, das auf den aktuellen Rechner zeigt und sicherstellt, dass die Netzfunktionen lokal funktionieren.

Mit `ifconfig` müssen Sie in der Regel nicht selbst arbeiten. Die umfangreiche Liste der Parameter finden Sie in der Manpage zu `ifconfig`.

Ausgabe von `/sbin/route`:

```
Kernel IP routing table
Destination    Gateway  Genmask          Flags Metric Ref  Use Iface
194.95.238.253   *      255.255.255.255   UH     0     0    0  ippp0
192.168.1.0      *      255.255.255.0     U      0     0    0  eth0
loopback         *      255.0.0.0         U      0     0    0  lo
```

Die IP-Adressen für ippp0 werden bei Ihnen abweichen, da man diese meist beim Verbindungsaufbau dynamisch bezieht.

Routen, die nicht an ein Gerät gebunden sind, sondern einen Weg beschreiben, müssen Sie schon eher einmal setzen. Am häufigsten wird es darum gehen, eine Default-Route auf das `ippp0`- oder `ppp0`- Device zu setzen bzw. diese zu löschen. Nur wenn Sie eine Default-Route für die Wählverbindung eingerichtet haben, kann Ihr Linux-Server die Internet-Verbindung nutzen und auch den Clients im Netz zur Verfügung stellen.

Eine Default-Route auf `ippp0` wird mittels

```
/sbin/route add default dev ippp0
```

gesetzt und gelöscht mit

```
/sbin/route del default
```

Mit dem Setzen der Default-Route muss man sehr gezielt arbeiten, da alle Anfragen, für die kein Weg definiert ist, über den Default-Pfad gehen. Das löst dann eventuell eine Anwahl beim Provider aus. Sie müssen daher sicherstellen, dass alle lokalen Ziele auch über konkrete Routen erreichbar sind.

14.1.5 Internet-tauglichen Router konfigurieren

Ein internet-tauglicher Router muss also auf alle Fälle

- Routing-Informationen für das lokale Netz (meist `eth0`) und
- das Internet (`ppp0` oder `ippp0`) eingetragen und
- eine Default-Route auf das Internet-Device kennen.

Da die Dämonen bzw. die Start-Scripten die Routen für `ppp0` bzw. `ippp0` setzen, muss man nur darauf achten, dass diese eine Default-Route setzen, damit man die Verbindung auch vernünftig nutzen kann.

14.2 Masquerading

Beim Masquerading verstecken Sie ein ganzes lokales Netzwerk hinter einer einzigen IP-Adresse. Der Server fängt alle Datenpakete ab, die ins Internet weitergeleitet werden sollen, ersetzt die ungültige IP des Absenders durch seine gültige IP und schickt das Paket weiter ins Internet.

Eingehende Pakete sind immer an die gültige IP des Servers gerichtet. Da er alle weitergegebenen Anfragen in einer Tabelle vermerkt, kann er feststellen, welcher Rechner im Netz die Daten erwartet. Nun ersetzt er die Server-IP durch die lokale IP des Empfängers und stellt das Paket zu.

Der Client merkt von dieser doppelten Umsetzung nichts. Alle Internetdienste sind vom Client aus vollkommen transparent nutzbar, wenn man generell maskiert.

Für die konkrete Umsetzung des Masquerading benötigen Sie eine Unterstützung im Kernel, Module für spezielle Funktion und zum Steuern des Masquerading das Programm IPChains. Die Standardkernel von SuSE enthalten diese Unterstützung.

Das Programm IPChains gehört gewissermaßen zum Kernel 2.2.x. Bei den Kernelversionen davor war für den gleichen Zweck das Programm `ipfwadm` zuständig und bei dem Kernel 2.4.x wird es das Programm `iptables` sein. Zum Glück wird IPChains auch mit dem 2.4.x Kernel zusammenarbeiten, so dass die in diesem Text beschriebenen Grundlagen weiterhin gültig bleiben.

14.2.1 Masquerading mit IPChains

Die Standardinstallation von SuSE installiert normalerweise das Programm IPChains aus dem Paket `ipchains` der Serie `sec` nicht. Sie finden dieses Paket auf der CD mit der Evaluationsversion oder im Verzeichnis `sec` in der Datei `ipchains.rpm`.

IPChains steuert bzw. kontrolliert die Paketfilterung im Kernel. Der Kernel kennt drei Arten von Firewall-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, dann 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;
- DENY, er verwirft das Paket ohne Rückmeldung;
- REJECT, er verwirft das Paket, gibt aber per ICMP eine Rückmeldung an den Absender.

In einer Forward-Chain ist zusätzlich auch MASQ zulässig, womit man das Maskieren veranlasst.

Im Paket-Header sind u.a. folgende Informationen vorhanden, die mit Regeln überprüft werden können:

- Absender-IP und -Port (`-s Source`),
- Ziel-IP und -Port (`-d Destination`),
- Protokoll (`-p Protocol`).

Fragt man die eingestellten Regeln mit `ipchains -L` ab, so erhält man:

Ausgabe von IPChains `-L`

```
Chain input (policy ACCEPT):
Chain forward (policy ACCEPT):
Chain output (policy ACCEPT):
```

Für alle drei Chains liegt die 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. Folgendermaßen stellen Sie auf Masquerading um:

```
ipchains  -P forward MASQ
```

Danach haben alle Rechner im Netz vollen Internet-Zugriff.

Leider nur fast. Ein paar Dienste machen Probleme. Dazu gehört FTP, da dieser Dienst mit zwei verschiedenen Ports arbeitet. Über den Datenkanal empfängt man Pakete, die man über den Kommandokanal angefordert hat. Darauf ist unser Firewall bisher nicht eingestellt.

Für die meisten problematischen Dienste gibt es inzwischen Module, die diese Probleme überwinden können. Diese Module müssen Sie noch laden. Eine Lösung besteht darin, das folgende Programm zu erstellen, welches die Default-Policy auf Masquerading stellt und die benötigten Module lädt.

```
/sbin/init.d/maske
```

```
#!/bin/sh
#ip-masquerading
#
#
MSQ_MODULES="ip_masq_autofw ip_masq_ftp ip_masq_cuseeme
➥ ip_masq_irc ip_masq_mfw ip_masq_portfw ip_masq_quake
➥ ip_masq_raudio ip_masq_user ip_masq_vdolive"

case "$1" in
  start)
    for I in $MSQ_MODULES;  do
      /sbin/modprobe  $I
    done
    /sbin/ipchains  -P  forward  MASQ
    # hier koennen eigene Regeln folgen
  ;;
  stop)
    for I in $MSQ_MODULES;  do
```

```
        /sbin/rmmod  $I
      done
      /sbin/ipchains  -P  forward  ACCEPT
    ;;
    *)
      echo "usage $0 start | stop"
    ;;
esac
exit
```

Wollen Sie das Script schon beim Starten aktivieren, so legen Sie es im Ordner `/sbin/init.d` an und erstellen in `/sbin/init.d/rc2.d` die Links zum Starten und Stoppen:

```
ln -s ../maske K20maske
ln -s ../maske S20maske
```

Damit ist das Masquerading vollständig funktionsfähig.

14.2.2 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 `IPCHAINS` 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 ist zu beachten, dass die Reihenfolge eine Rolle spielt. 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, was für aufwendigere Regelwerke die Strukturierung erleichtert. Auf Benutzer-Chains geht dieses Buch nicht ein, weitere Hinweise hierzu finden sie in der Manpage zu IPChains.

Der erste Parameter von IPChains 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:

```
ipchains -P forward DENY
```

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

```
 ipchain -A forward -s 192.168.1.0/24  -j MASQ
```

Hiermit fängt eine Regel an die Forward-Chains an. Der Kernel maskiert alle Pakete mit Quellen aus dem lokalen Netz (die IP beginnt mit `192.168.1`).

Für die Regel überprüft der Kernel hier den Absender (`-s`). Liegt der Absender im lokalen Netz, dann springt die Regel zu MASQ (`-j`), das Paket wird maskiert.

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. `192.168.1.0/24` bedeutet, wenn die ersten 24 Bit der IP diesem Muster entsprechen, dann fällt die IP unter unsere Regel. Haben Sie keine IP angegeben, so sind alle Adressen gemeint, was Sie konkret mit `0/0` angeben könnten.

Wenn Sie keine Angabe über Ports gemacht haben, bezieht sich das Muster auf alles Ports. Sie können jedoch wie oben einen Port einzeln angeben oder über `von:bis` einen Bereich von Ports. Mit `30:144` würden Sie 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 Sie nicht nur über ihre Nummern angeben, sondern auch über ihre Bezeichnung:

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

Bisher haben Sie kein Protokoll angegeben, also gilt die Regel für alle Protokolle. Im folgenden Beispiel (aus dem `IPCHAINS-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.

```
ipchains -A input -s 127.0.0.1 -p icmp -j DENY
```

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 Sie wissen, dass die Regel die einzige bzw. erste Regel in der Chain Input ist, können Sie sie mit

```
ipchains -D input 1
```

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

```
ipchains -D input -s 127.0.0.1 -p icmp -j DENY
```

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),
- `-y` lässt nur TCP-Antwortpakete durch `-p TCP -s 192.168.1.1 -y`.

14.2.3 Sicherheitsphilosophien

Bei der Arbeit mit IPChains gibt es zwei grundsätzliche Philosophien, mit denen Sie vorgehen können:

- Vertrauen: Alles ist erlaubt, was Sie nicht explizit verbieten. Die Default-Policies stellen Sie bei diesem Ansatz auf *ACCEPT* bzw. *MASQ*.

- Misstrauen: Alles ist verboten, was Sie nicht explizit erlauben. Die Default-Policies stellen Sie dann auf *DENY* oder *REJECT*.

Die größere Sicherheit bietet Ihnen der misstrauische Ansatz. Er macht aber auch viel Arbeit, wenn Sie mehrere Dienst oder Protokolle freischalten müssen. Sie müssen hier sehr genau überlegen, welche sinnvollen Anforderungen Anwender in Ihrem Netz haben und welche Anwendungen wirklich eine Rolle spielen. Erst dann können Sie entscheiden, mit welcher Grundphilosophie Sie an Ihre Firewall herangehen.

14.2.4 Ein praktisches Beispiel

Für ein kleines lokales Netz sollten Sie das Forwarding nur auf die Dienste bzw. Rechner beschränken, für die ein direkter Zugriff vom Client aus notwendig ist. Die folgenden Regeln zeigen das exemplarisch:

```
sbin/ipchains -F forward
/sbin/ipchains -P forward REJECT
/sbin/ipchains -A forward -s 192.168.1.0/24  -d
➥ 185.37.188.0/24  -j MASQ
/sbin/ipchains -A forward -s 185.37.188.0/24  -d
➥ 192.168.1.0/24  -j MASQ
/sbin/ipchains -A forward  -p tcp -d 0/0 25 -j MASQ
/sbin/ipchains -A forward  -p tcp -d 0/0 110 -j MASQ
```

In der ersten Zeile löscht man erst einmal alle Regeln der Forward-Chain. Danach setzt man die Policy auf REJECT, was alle Transporte zwischen den Interfaces ablehnt. Dann ermöglicht man für alle Rechner `185.37.188.x` die Kommunikation durch das Maskieren, diese Rechner gelten als vertrauenswürdig bzw. sicher.

Zum Schluss öffnet man noch für alle Zielrechner die Ports `25` (SMTP) und `110` (POP). Dadurch können die Benutzer im lokalen Netz Post z.B. bei GMX abliefern und beziehen.

Weitere Rechner bzw. Ports sollte man nur auf konkreten Bedarf hin zulassen.

Für den Server selber haben Sie bisher keinerlei Schutz aktiviert, hier müssten Sie mit den Input- und Output-Chains arbeiten. Das lokale Netz ist so aber recht gut geschützt.

14.2.5 Accounting Rule

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

```
ipchains -L forward -v
```

die folgende Ausgabe (nach erfolgter Nutzung):

```
Chain forward (policy REJECT: 15 packets, 852 bytes):
pkts bytes target prot opt    tosa tosx ifname mark outsize
➥ source   destination ports
 311 12777 MASQ   tcp  ------ 0xFF 0x00 any
 ➥ anywhere anywhere    any ->   smtp
  47  2002 MASQ   tcp  ------ 0xFF 0x00 any
  ➥ anywhere anywhere    any ->   pop3
```

Der Kernel hat über die Policy REJECT 15 Pakete abgelehnt, 311 Pakete an SMTP-Ports maskiert und 47 Pakete an POP3-Ports versandt.

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:

```
ipchains -A input -s 192.168.1.1
```

Diese Regel zählt alle Pakete vom Host `192.168.1.1`. Der Befehl

```
ipchains -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 können Sie die Zähler über `ipchains -Z`, konkret für das letzte Beispiel mit

```
ipchains -Z input
```

SuSE liefert im Paket `firewall` der Serie `n` ein sehr umfangreiches Script für eine Firewall mit. Wer möchte, kann sich das Paket installieren und damit sein System abschotten. Das ist aber ein zweischneidiges Schwert, da man an einem derart komplexen System nur schwer etwas ändern kann. Für Sicherheitsfragen ist es notwendig, dass man genau weiß, was man tut. Von daher ist ein eigenes Script, so wie hier vorgestellt, weniger elegant, aber leicht zu warten.