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.

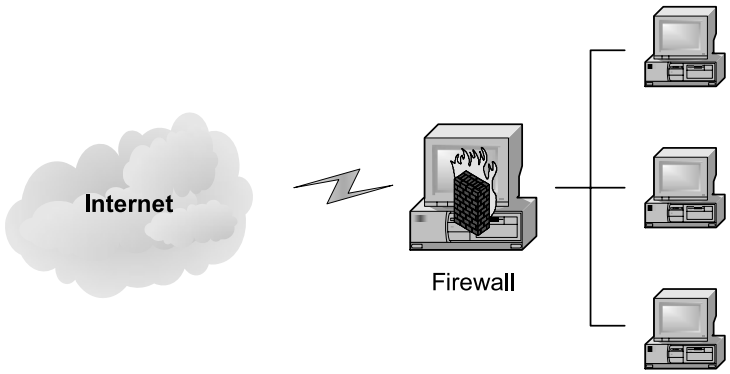

Abbildung 14.1: Firewall

Da Sie den Linux-Server immer als Stellvertreter benutzt haben, können Sie an den Arbeitsplatzrechnern trotzdem Internetdienste 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 Internetverbindung der Clients auf.

Lesen Sie in diesem Kapitel, wie man unter Berücksichtigung von Sicherheitsaspekten, lokalen Clients einen direkten Zugriff auf das Internet ermöglichen kann.

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 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*.

Nutzen Sie für 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 Netzadressen nicht weiterleiten. In diesem Fall müsste der Server in der Anfrage die lokale IP des Clients durch seine offizielle, 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 Client-Rechner im lokalen Netz von außen nicht erreichbar und damit auch nicht angreifbar sind. Dafür können die lokalen Rechner auch nicht selbst direkt auf Rechner im Internet zugreifen.

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

14.1.2 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: Provider können offizielle IP-Adressen fest vergeben. Zwei der Adressen (die niedrigste und die höchste) des Adressraums gehen für die Netzwerk-Adresse und die Broadcast-Adresse ab. Hat man z. B. acht solcher Adressen, so kann man damit 6 Rechner versorgen und 256 Adressen reichen für 254 Geräte.

Um diese Weiterleitung zu aktivieren, müssen Sie auf ihrem Linux-Server das nach Voreinstellung abgeschaltete IP-Forwarding aktivieren. Dazu geben Sie in einem IPv4-Netz 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, unterstützt der Kernel Ihres Linux-Servers kein Forwarding. In diesem Fall müssen Sie dessen Kernel neu kompilieren. Zum Glück richten die verbreiteten Distributionen die Standardkernel passend ein.

Um das Forwarding dauerhaft zu aktivieren, müssen Sie den angegebenen Befehl in Ihr Boot-Skript aufnehmen. Bei SuSE-Linux gehen Sie dazu im YaST-Kontrollzentrum auf *System* • *Editor für /etc/sysconfig-Daten* • *Network* • *General* und suchen in der Parameterliste den Schalter *IP_FORWARD* und setzen ihn auf *yes*.

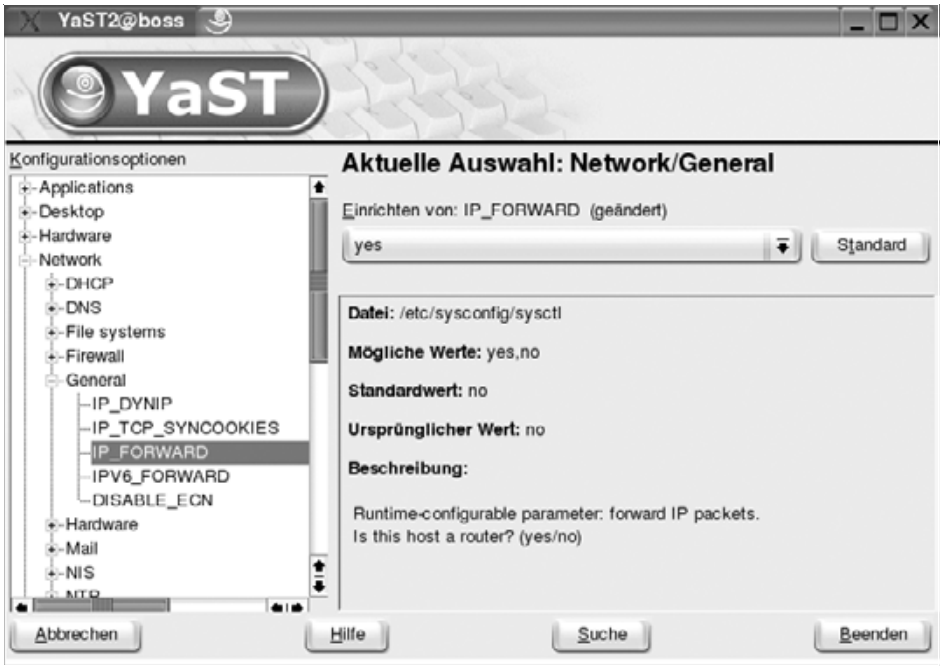

Abbildung 14.2: YaST : IP_FORWARD

Damit das Skript `/etc/init.d/boot` diesen Schalter auswerten kann, sollte man den Linux-Server rebooten.

Wenn die Routen richtig gesetzt sind, besteht danach 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 meldet der angesprochene Rechner www.linuxbu.ch (213.70.186.2) den korrekten Empfang der Datenpakete zurück:

```
PING www.linuxbu.ch (213.70.186.2) from 192.168.1.2 : 56(84)
↵ bytes of data.
64 bytes from 213.70.186.2: icmp_seq=1 ttl=245 time=79.997
↵ msec
64 bytes from 213.70.186.2: icmp_seq=2 ttl=245 time=75.554
↵ msec
64 bytes from 213.70.186.2: icmp_seq=3 ttl=245 time=77.367
↵ msec
64 bytes from 213.70.186.2: icmp_seq=4 ttl=245 time=77.301
↵ msec

--- www.linuxbu.ch ping statistics ---
4 packets transmitted, 4 received, 0% loss, time 3034ms
rtt min/avg/max/mdev = 75.554/77.554/79.997/1.622 ms
```

14.1.3 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.

Hinweis: Den folgende Abschnitt sollten Sie als Hintergrundinformation betrachten. In der Regel werden Sie die Routing-Informationen nicht direkt bearbeiten, sondern dies YaST und den Startscripten der Netzwerkgeräte überlassen. Bei der Suche nach eventuellen Fehlern benötigen Sie Informationen über das Routing.

Für einen Datentransport zwischen Subnetzen benötigt der Linux-Kernel Information über die IP-Adressen und die zugehörigen Net-Devices. Die statischen Informationen stehen in der Datei `/etc/sysconfig/network/routes`. Diese Datei können Sie direkt mit einem Texteditor oder besser über da YaST-Kontrollzent-

rum unter *Netzwerkgeräte* • *Netzwerkkarte* • *Konfiguration ändern* • *eth0 Bearbeiten* • *Routing* bearbeiten.

`/etc/sysconfig/network/routes`

```
# 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, dass 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, dann ist das 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 vom Provider 255 IP-Adressen 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          Device
195.37.188.187   194.95.238.253         255.255.255.255  ippp0
```

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

```
# default                  Provider IP
default        194.95.238.253     0.0.0.0    ippp0
```

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

Diese Datei konfiguriert die immer vorhandenen statischen Routen. Das Startscript `/etc/init.d/network` 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, normalerweise die Netzwerkkarte für das lokale Netz (meist eth0), das Gerät für den Internetzugriff (hier ippp0) und das Loopback-Device (lo):

Ausgabe von `/sbin/ifconfig`:

```
eth0      Protokoll:Ethernet  Hardware Adresse 00:50:BF:55:8D:46
          inet Adresse:192.168.1.2  Bcast:192.168.1.255
        ↵ Maske:255.255.255.0
          inet6 Adresse: fe80::250:bfff:fe55:8d46/64
        ↵ Gültigkeitsbereich:Verbindung
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:488 errors:0 dropped:0 overruns:0 frame:0
          TX packets:285 errors:0 dropped:0 overruns:0 carrier:0
          Kollisionen:0 Sendewarteschlangenlänge:100
          RX bytes:42368 (41.3 Kb)  TX bytes:29573 (28.8 Kb)
          Interrupt:11 Basisadresse:0xe000

ippp0     Protokoll:Punkt-zu-Punkt Verbindung
          UP PUNKTZUPUNKT 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
          Kollisionen:0 Sendewarteschlangenlänge:30
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

lo        Protokoll:Lokale Schleife
          inet Adresse:127.0.0.1  Maske:255.0.0.0
          inet6 Adresse: ::1/128 Gültigkeitsbereich:Maschine
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:790 errors:0 dropped:0 overruns:0 frame:0
          TX packets:790 errors:0 dropped:0 overruns:0 carrier:0
          Kollisionen:0 Sendewarteschlangenlänge:0
          RX bytes:55860 (54.5 Kb)  TX bytes:55860 (54.5 Kb)
```

Zusätzlich zu den Devices `eth0` und `ippp0` gibt es noch ein Device `lo`, ohne dass man es irgendwo definiert hätte. Dieses *Loopback-Device* ist ein Pseudo-Device, zeigt auf den aktuellen Rechner und stellt sicher, 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 Routentabelle
Ziel            Router          Genmask         Flags Metric
↵ Ref    Use Iface
192.168.1.0     *         255.255.255.0   U     0      0
↵ 0 eth0
194.95.238.25   *      255.255.255.255  UH     0       0     0
↵ ippp0
```

Die IP-Adressen für ippp0 werden bei Ihnen abweichen.

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 Internetverbindung nutzen und auch den Clients im Netz zur Verfügung stellen.

Eine Default-Route auf `ippp0` wird mit

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

gesetzt und gelöscht mit

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

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

14.1.4 Internettauglichen Router konfigurieren

Ein internettauglicher Router muss also auf alle Fälle

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

kennen.

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

14.2 Masquerading

Masquerading versteckt ein ganzes lokales Netzwerk hinter einer einzigen IP-Adresse. Der Server fängt alle Datenpakete ab, die vom lokalen Netz ins Internet weitergeleitet werden sollen, ersetzt die private IP des Absenders durch seine eigene offizielle 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 diesem 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 Funktionen und zum Steuern das Programm `iptables`. Die Standardkernel von SuSE enthalten diese Unterstützung.

Das Programm `iptables` gehört gewissermaßen zum Kernel 2.4.x. Bei den Kernel-Versionen davor war für den gleichen Zweck das Programm `ipfwadm` zuständig bzw. bei den Kerneln 2.2.x war es das Programm `ipchains`.

Auch wenn `ipchains` im Prinzip noch mit 2.4.x Kerneln zusammenarbeitet, sollten Sie mit `iptables` arbeiten, weil nur dieses alle aktuellen Möglichkeiten unterstützt.

14.2.1 Masquerading mit iptables

Die Standardinstallation von SuSE installiert normalerweise das Programm `iptables`. Ansonsten finden Sie dieses in der Paketgruppe *Productivity* • *Networking* • *Security* im Paket `iptables` oder auf der `CD1` in der entsprechenden rpm-Datei.

`Iptables` steuert bzw. kontrolliert die Paketfilter im Kernel. 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.

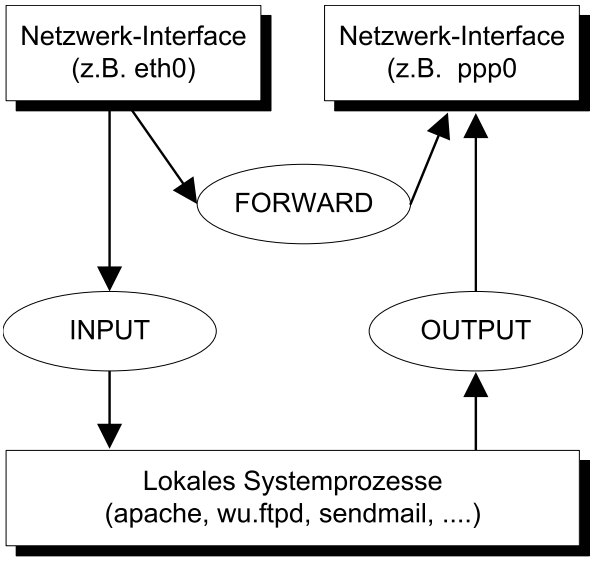

Abbildung 14.3: iptables: die Chains

Als Ergebnis dieser Überprüfung ergibt sich für jedes Datenpaket eine der folgenden Möglichkeiten:

- ACCEPT, der Kernel transportiert das Paket weiter;
- DROP, er verwirft das Paket ohne Rückmeldung;
- REJECT, er verwirft das Paket, informiert aber den Absender per ICMP.

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 gibt dieses die aktuellen Einstellungen für Input, Forward und Output aus.

Ausgabe von `iptables -L`

```
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 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 müssen Sie 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*.

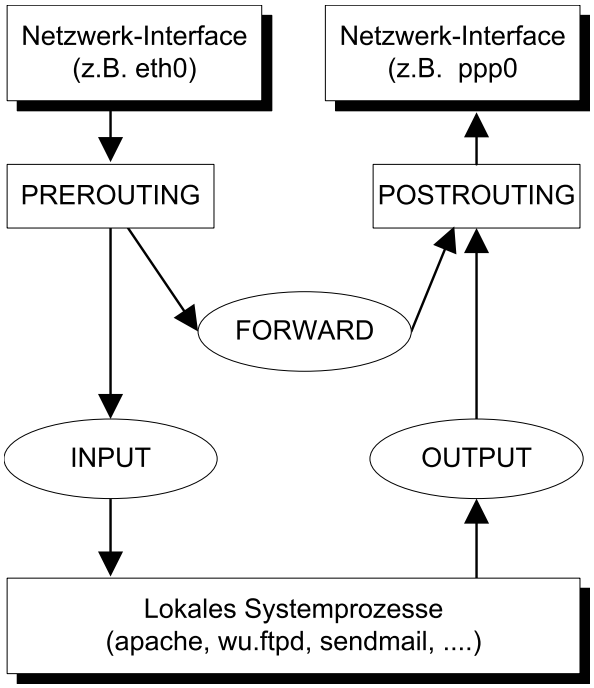

Abbildung 14.4: iptables: die Chains mit PREROUTING und POSTROUTING

Wenn ein Datenpaket erfolgreich die FORWARD-Chain durchlaufen hat, danach also z. B. über `ppp0` ins Internet gehen würde, muss der Kernel die Absenderadresse ändern, also Maskieren.

Die bisher angesprochenen Chains INPUT, OUTPUT und FORWARD gehören zur Default-Table *filter*, während PREROUTING und POSTROUTING zur Table *nat* (network address translation) gehören. Die Table *filter* müssen Sie in Ihren Regeln nicht explizit angeben, wohl aber die Table *nat*, die für alle Veränderungen der Adressinformationen, also auch das Masquerading, zuständig ist.

Fügen Sie die Masquerading-Regel (`-A`) für das Output-Device (`-o ppp0`) folgendermaßen an die POSTROUTING-Chain der Table *nat* (`-t nat`) an:

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

Sollten Sie beim Eingeben dieser Regel eine Fehlermeldung erhalten, ist vermutlich das Kernel-Modul für *nat* nicht geladen; geben Sie in diesem Fall

```
modprobe iptable_nat
```

ein und wiederholen Sie danach die *nat*-Regel.

Falls Sie sich beim Eintippen der Regeln verschreiben sollten, müssen Sie die Regeln auch wieder loswerden können. Alle von Ihnen eingegebenen Regeln können Sie auf einen Schlag löschen. Mit

```
iptables -F
```

löschen Sie 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.

Inzwischen können bestimmte Module für die meisten problematischen Dienste diese Probleme überwinden. 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. Das Skript beruht auf dem in Kapitel 4 beschriebenen Muster-Skript `skeleton`:

`/etc/init.d/maske`

```
#! /bin/sh
#
# /etc/init.d/maske
```

```
#
#   and its symbolic link
#
# /sbin/rcmaske
#
# System startup script for Masquerading
#
### BEGIN INIT INFO
# Provides: maske
# Required-Start: $network
# Required-Stop:
# Default-Start:  2 3 4 5
# Default-Stop:
# Description:    Start simple Firewall- Skript
### END INIT INFO

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
```

Eine ausführlichere Version dieses Skripts finden Sie auf `www.linuxbu.ch`.

Sie müssen das Skript nun noch mit

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

ausführbar machen

Sie können das Maske-Skript gut als Grundlage für eigene Experimente benutzen. Wenn Sie es beim Systemstart automatisch aktivieren wollen, müssen Sie mit

```
insserv  maske
```

die entsprechenden Links setzen lassen (Siehe Kapitel 4).

Damit ist das Masquerading vollständig funktionsfähig. Weitere Konfigurationsmöglichkeiten finden Sie in den folgenden Beispielen.

14.2.2 Firewalling

Die bisherigen Informationen über Paketfilterung reichen erst einmal aus, um das Masquerading zu aktivieren. Will man die Pakete genauer kontrollieren, 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 ist die Reihenfolge wichtig. Da der Kernel die erste zutreffende Regel abarbeitet, spielen spätere Regeln keine Rolle.

Neben den drei vorgegebenen Chains kann man auch eigene Chains (Benutzer-Chains) einrichten, um aufwändigere Regelwerke besser zu strukturieren.

Für eigene Chains gibt es folgende Regeln:

- Eine Benutzer-Chain definieren und benennen (-N);
- eine (leere) Benutzer-Chain löschen (-X).

Sie werden im weiteren Verlauf des Kapitels, im Abschnitt 14.2.4, ein Beispiel mit einer Benutzer-Chain kennenlernen.

Der erste Parameter von `iptables` gibt üblicherweise an, was man machen möchte (`append`, `insert`,...). Danach nennt man die Chains, auf die sich die Regel beziehen soll und zuletzt die eigentliche Regel. Bitte lesen Sie die folgenden 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 verbietet das Weiterleiten aller Datenpakete vom Rechner mit der IP-Adresse 192.168.1.51. Dieser Rechner kann 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`) und verwirft das Paket.

Absender- und Zieladresse eines Pakets 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. Meinen Sie alle Adressen, können Sie auf die IP verzichten oder 0/0 schreiben.

Wenn Sie keine Angabe über Ports gemacht haben, bezieht sich das Muster auf alle Ports. Sie können jedoch wie oben einen Port einzeln angeben oder mit `von:bis` einen Bereich von Ports. Mit `30:144` würden Sie 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 wie:

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

Bisher haben Sie 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 dieser 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

```
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).

14.2.3 Sicherheitsphilosophien

Bei der Arbeit mit `iptables` gibt es zwei grundsätzliche Strategien:

- Vertrauen: Alles ist erlaubt, was Sie nicht explizit verbieten. Die Default-Policies stellen Sie bei diesem Ansatz auf *ACCEPT*.
- Misstrauen: Alles ist verboten, was Sie nicht explizit erlauben. Die Default-Policies stellen Sie dann auf *DENY* oder *DROP*.

Die größere Sicherheit bietet der misstrauische Ansatz. Er macht aber auch viel Arbeit, wenn Sie mehrere Dienste 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 Strategie Sie an Ihre Firewall herangehen.

14.2.4 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 legen Sie eine Benutzer-Chain *block* an, die einerseits Pakete durchlässt, die Antwortpakete sind (Status `ESTABLISHED`,…) oder neue Pakete, die nicht über `ppp0`, also das Internet, hereinkommen. Diese Regeln binden Sie dann in die INPUT und die FORWARD-Chain ein.

Zuletzt aktivieren Sie noch das Masquerading, das dann auf die Pakete wirkt, die die FORWARD-Chain erfolgreich passiert haben.

Ihr Rechner antwortet damit auf keinerlei Anforderungen aus dem Internet, nicht einmal einen Ping. Aus dem lokalen Netz heraus und vom Server selbst aus haben Sie aber vollen Zugriff auf das Internet.

Soll Ihr Server auf `ping` reagieren, dann müssen Sie am Anfang des Listings folgende Zeile ergänzen.

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

Wollen Sie auf Ihrem Server auch öffentlich Dienste anbieten, so müssen Sie diese explizit freischalten, beispielsweise den Port 80 für einen Web-Server:

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

Diese Regel muss aber vor der Definition der Benutzer-Chain *block* stehen, da sie sonst nicht mehr berücksichtigt wird.

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

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

```
iptables -Z INPUT
```

14.2.6 Logging-Rule

Mit `iptables kann man nicht nur` Pakete zählen, sondern auch Zugriffe gezielt protokollieren.

Wenn Sie Zugriffe auf den Telnet-Port 23 nicht nur sperren wollen, sondern auch protokollieren möchten, wer wann versucht, auf diesen Port zuzugreifen, fügen Sie 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, können Sie optional mit `--log-prefix` einen individuellen Text angeben.

Jeder versuchte Telnet-Zugriff auf Ihren Server hinterlässt folgende Einträge in Ihrer 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.

14.2.7 Limits

Zu den neuen Funktionen von `iptables` gehört die Möglichkeit, die Häufigkeit von Zugriffen 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, kann der Zielrechner unter der Last zusammenbrechen.

Mit der Limit-Option (`-m limit`) und deren Parameter `--limit 1/sec` können Sie einen derartigen Angriff abwehren:

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

Ihr Rechner beantwortet jetzt nur noch einen Ping pro Sekunde. Auch bei anderen Diensten sollten Sie die Anzahl der Zugriffe beschränken. 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, da dies Ihre Arbeitsgeschwindigkeit beschränken würde, 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 dem Verbindungsaufbau ist der Status der Pakete nicht mehr NEW, die Regel stört dann also nicht während der Verbindung.

14.2.8 SuSE firewall

SuSE liefert im Paket `SuSEfirewall2` ein sehr umfangreiches Skript für eine Firewall mit. Wer möchte, kann das Paket aktivieren und damit sein System abschotten. Das ist aber ein zweischneidiges Schwert, da man an einem derart komplexen System nur schwer etwas ändern kann. Bei Sicherheitsfragen ist es notwendig, dass man genau weiß, was man tut. Daher ist ein eigenes Skript, wie hier vorgestellt, leichter zu warten.