Kapitel 3

iptables: Das Administrationsprogramm für die Linux-Firewall

3.1	Unterschiede in den Firewall-Mechanismen von IPFW und Netfilter	104
3.2	Features von iptables	107
3.3	Syntax von iptables	114
3.4	Zusammenfassung	134

In Kapitel 2 haben wir die grundlegenden Ideen und Konzepte hinter einer Paketfilter-Firewall besprochen. Jede eingebaute Kette hat ihre eigene Policy. Eine einzelne Regel kann sich nicht nur auf eine Kette beziehen, sondern auch auf ein bestimmtes Netzwerkinterface, einen Protokolltyp (z.B. TCP, UDP oder ICMP), einen Serviceport oder einen ICMP-Nachrichtentyp. Für die Ketten `INPUT`, `OUTPUT` und `FORWARD` werden jeweils eigene Regeln für das Annehmen, Ablehnen und kommentarlose Verwerfen von Paketen definiert. Mehr dazu erfahren Sie am Ende dieses Kapitels sowie in Kapitel 6. Das folgende Kapitel wird mithilfe all dieser Konzepte eine einfache, selbstgebastelte Firewall für ein einzelnes System vorstellen.

In diesem Kapitel geht es um `iptables`, das Firewall-Administrationsprogramm, mit dem man eine Netfilter-Firewall erstellt. Wenn Sie schon eines der älteren Programme aus der IPFW-Technik kennen, `ipfwadm` oder `ipchains`, dann wird `iptables` Ihnen ziemlich bekannt vorkommen. Jedoch verfügt es über viel mehr Features, ist flexibler und unterscheidet sich in Einzelheiten ganz wesentlich von der älteren Software.

3.1 Unterschiede in den Firewall-Mechanismen von IPFW und Netfilter

Dieses Buch basiert auf Red Hat Linux 7.x. Linux beinhaltet in seinen neueren Versionen eine neue Implementierung der Firewall, genannt *Netfilter*. Weil sich `iptables` so grundlegend von dem älteren `ipchains` unterscheidet, versuche ich erst gar nicht, noch auf die ältere Implementation einzugehen.

Der folgende Abschnitt richtet sich an einen Leser, der `ipchains` kennt oder es sogar momentan benutzt. Wenn Sie mit `iptables` erstmals eine Linux-Firewall kennen lernen, überspringen Sie am besten alles bis zur Überschrift »Der Weg eines Paketes im Netfilter«.

Bei der Konversion von `ipchains` werden Sie einige geringfügige Unterschiede in der Syntax von `iptables` feststellen. In erster Linie geht es dabei darum, dass die Netzwerkinterfaces für ankommende und abgehende Pakete getrennt identifiziert werden. `iptables` verfügt über einen stark modularen Aufbau, und die einzelnen Module müssen teilweise explizit geladen werden. Logging von Paketen ist jetzt ein eigenes Target für eine Regel, nicht mehr nur eine Option. Der Verbindungszustand lässt sich auf Wunsch überwachen. Adress- und Portübersetzung sind jetzt logisch vom reinen Paketfiltern getrennt. Adressübersetzung auf Absender- und Empfängerseite ist vollständig implementiert. »Masquerading« als Begriff bezieht sich nun auf eine spezialisierte Form der Absenderadressübersetzung. Port-Forwarding und Adressübersetzung auf der Empfängerseite sind direkt unterstützt, man benötigt dafür keine Drittsoftware wie `ipmasqadm` mehr.

MASQUERADING IN ÄLTEREN LINUX-VERSIONEN
Falls Linux noch neu für Sie ist: `iptables` unterstützen die so genannte Network Address Translation, kurz NAT oder auf deutsch Netzadressübersetzung genannt, vollständig. Früher hieß NAT unter Linux »Masquerading«. Dabei handelte es sich um eine einfache und unvollständige Implementierung der Adressübersetzung für Absenderadressen. Sie war geeignet für Leute, die nur eine einzige öffentliche IP-Adresse besaßen oder zugeteilt erhielten, aber für mehrere Rechner auf einem privaten Netzwerk den Zugang zum Internet wollten. Bei abgehenden Paketen von internen Hosts wurde die Absender-IP in die öffentliche IP-Adresse übersetzt.

Der wichtigste Unterschied zwischen den IPFW-basierten Systemen und dem neuen Netfilter liegt in dem Weg, den ein Paket durch das Betriebssystem hindurch geht. Dadurch kommt es zu kleinen Unterschieden im Aufbau der Firewall.

Für Benutzer von `ipchains` ist ein genaues Verständnis dieses Unterschiedes, der in den nächsten beiden Abschnitten ausführlich diskutiert wird, sehr wichtig. Oberflächlich sehen sich `iptables` und `ipchains` ganz ähnlich, aber in der Praxis handelt es sich um sehr unterschiedliche Systeme. Syntaktisch völlig korrekte `iptables`-Regeln können dennoch eine völlig andere Auswirkung haben, als es unter `ipchains` der Fall war. Das kann recht verwirrend sein! Wenn Sie sich mit den `ipchains` schon auskennen, müssen Sie die Unterschiede ständig im Hinterkopf behalten.

3.1.1 Der Weg eines Paketes in IPFW

Bei IPFW (und damit meine ich sowohl `ipfwadm` als auch `ipchains`) gab es drei eingebaute Ketten für den Paketfilter. Alle von außen ankommenden Pakete durchliefen die Input-Chain. Wenn diese das Paket annahm, wurde es an das Routing-Modul übergeben. Die Funktionen daraus entschieden, ob das Paket lokal ausgeliefert oder an einen anderen Rechner weitergeleitet werden sollte. (Siehe Abbildung 3.1.)

Ein weiterzuleitendes Paket wurde ein zweites Mal gefiltert, diesmal in der Forward-Chain. Falls diese es annahm, ging das Paket an die Output-Chain.

Sowohl lokal erzeugte als auch von anderen Rechnern weitergeleitete Pakete wurden an die Output-Chain übergeben. Nur dort akzeptierte Pakete wurden tatsächlich gesendet.

Empfangene und gesendete lokale Pakete (Loopback) mussten insgesamt also zwei Filter durchlaufen, weitergeleitete Pakete drei.

Der Pfad für das Loopback-Interface bestand aus zwei Ketten. Abbildung 3.2 zeigt, dass jedes Loopback-Paket zunächst die Output-Chain passieren musste, bevor es über das Loopback-Interface »verschickt« wurde. Anschließend musste es durch die Input-Chain hindurchwandern.

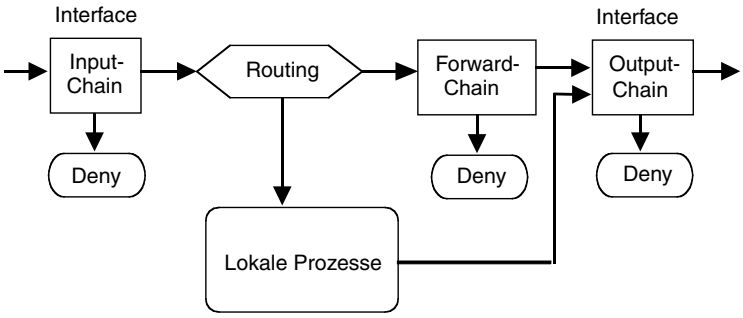

Abbildung 3.1: Der Weg eines Paketes in IPFW (Abbildung basiert auf dem Linux IPCHAINS-HOWTO von Rusty Russel, v1.0.8.)

Dieser Weg von Loopback-Paketen demonstriert, warum z.B. X-Windows hängenblieb, wenn man ein Firewall-Skript benutzte, das Loopback-Pakete nicht berücksichtigte, bzw. wenn man einmal kein Firewall-Skript benutzte, obwohl man DENY als Voreinstellung aktiviert hatte.

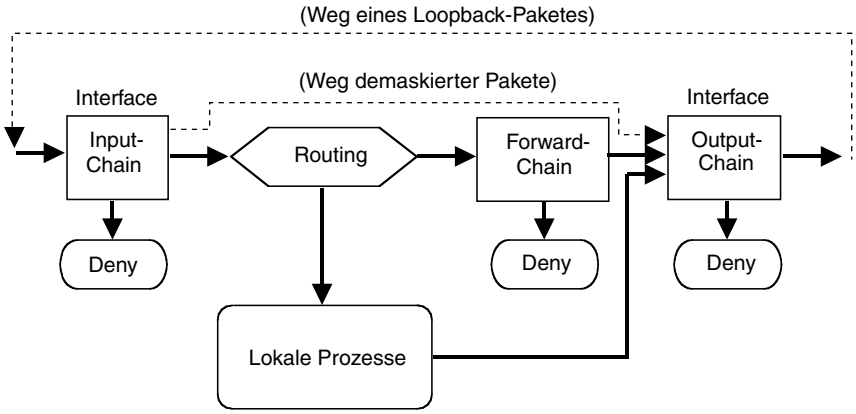

Abbildung 3.2: Der Weg von Loopback- und Masquerading-Paketen in IPFW (Abbildung basiert auf dem Linux IPCHAINS-HOWTO von Rusty Russel, v1.0.8.)

Abbildung 3.2 zeigt auch den Weg von Antworten auf Masquerading-Pakete, die ins LAN weitergeleitet werden sollten. Diese durchliefen zunächst die Input-Chain, wurden dann aber *nicht* an das Routingmodul weitergeleitet. Stattdessen wurden die Pakete direkt an die Output-Chain übergeben. Ankommende Pakete als Teil einer Masquerading-Verbindung wurden also nur zweimal gefiltert, abgehende Pakete hingegen dreimal.

3.1.2 Der Weg eines Paketes im Netfilter

Netfilter (`iptables`) kennt die drei eingebauten Filterketten bzw. Chains INPUT, OUTPUT und FORWARD. Ankommende Pakete werden zunächst von der Routingfunktion bearbeitet. Diese entscheidet, ob das Paket in die Input- oder die Forward-Chain des Rechners gelangt. Siehe Abbildung 3.3.

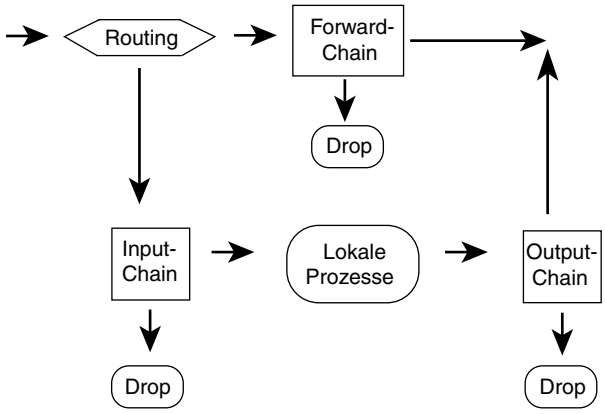

Abbildung 3.3: Der Weg eines Paketes im Netfilter

Wenn ein Paket für den eigenen Rechner von den Regeln der INPUT-Chain akzeptiert wird, dann wird es lokal ausgeliefert. Wenn ein Paket für einen fremden Rechner die FORWARD-Chain passiert, schickt das Betriebssystem es an das passende Netzwerkinterface.

Von lokalen Programmen erzeugte, abgehende Pakete durchlaufen die OUTPUT-Chain. Tun sie das erfolgreich, gelangen sie an das Netzwerkinterface. Damit wird bei Netfilter jedes Paket genau einmal gefiltert. (Ausnahme sind Loopback-Pakete, die zweimal gefiltert werden.)

3.2 Features von iptables

`iptables` benutzt für verschiedene Arten der Paketbearbeitung jeweils eigene Tabellen für die Regeln. Diese Regeltabellen sind als funktionell getrennte Module implementiert. Die drei wichtigsten Module sind: die normale Regeltabelle `filter`; die Tabelle `nat` für die Network Adress Translation; und eine spezialisierte Tabelle zur Paketbearbeitung namens `mangle`. Jedes dieser drei Module verfügt wiederum über eigene Modulerweiterungen, die dynamisch nachgeladen werden, sobald man sie das erste Mal benutzt.

Voreingestellt ist immer die `filter`-Tabelle. Die anderen Tabellen lassen sich durch Kommandozeilenoptionen auswählen.

Die `filter`-Tabelle verfügt im Wesentlichen über folgende Eigenschaften:

- Operationen mit den drei eingebauten Chains (INPUT, OUTPUT und FORWARD) sowie mit zusätzlichen, frei definierbaren Chains
- Hilfe
- Entscheidung darüber, ob ein Paket angenommen (ACCEPT) oder verworfen werden soll (DROP)
- Zugriff auf die Felder des IP-Headers, d.h. Protokoll, Absender- und Empfängeradresse, Netzwerkinterfaces, Fragmentverarbeitung
- Zugriff auf die Felder der TCP-, UDP- und ICMP-Header

Für die `filter`-Tabelle existieren zwei Sorten von Erweiterungsmodulen: Target-Module (was soll mit dem Paket geschehen?) und Match-Module (zur genaueren und/oder flexibleren Auswahl von Paketen). Die Target-Erweiterungen bieten die Funktionalitäten REJECT (zum Zurückweisen von Paketen mit Fehlermeldung an den Absender) und LOG (zum Protokollieren bestimmter Pakete). Mit den Match-Erweiterungen lassen sich Pakete gezielt auswählen, u.A. anhand folgender Kriterien:

- TCP-Verbindungszustand
- Portlisten (dafür ist das Multiport-Modul zuständig)
- Hardwareadresse (Ethernet-MAC-Adresse) des Absenders
- Benutzerkennung, Gruppe, Prozessnummer oder Prozessgruppe des sendenden Programms
- Type-of-Service-Feld (TOS) aus dem IP-Headers – dieses Feld kann z.B. durch Einträge in der `mangle`-Tabelle manipuliert werden
- Ein `iptables`-spezifisches Mark-Feld – wird ebenfalls durch die `mangle`-Tabelle gesetzt
- Auswahl von Paketen zur Begrenzung des Datendurchsatzes

Die `mangle`-Tabelle verfügt über zwei Erweiterungen: Das `MARK`-Modul kann dem `iptables`-spezifischen Mark-Feld eines Paketes einen Wert zuweisen; das `TOS`-Modul bearbeitet das gleichnamige Feld aus dem IP-Header.

> **ZUKÜNFTIGE ERWEITERUNGEN VON `IPTABLES`**
> Der gesamte Netfilter befindet sich in ständiger Weiterentwicklung. Wenn man sich den Quellcode ansieht, erkennt man eine Reihe bereits vorbereiteter Module, die vermutlich schon verfügbar sein werden, wenn Sie dieses Buch lesen. Andere Module hingegen sollen wohl nicht veröffentlicht werden – in mindestens einem Fall handelt es sich um ein reines Demonstrations- bzw. Beispiel-Modul.

> Beispielsweise existiert bereits jetzt eine Erweiterung für die `filter`-Tabelle namens TCPMSS sowie eine gleichnamige Erweiterung für die `mangle`-Tabelle. Diese Module erlauben die Übergabe eines Zahlenwertes im Optionsfeld des TCP-Headers, und zwar in den SYN- und SYN/ACK-Paketen, die während des Verbindungsaufbaus ausgetauscht werden. Über diese TCP-MSS-Option kann man den jeweiligen Verbindungspartner über die größtmögliche Segmentgröße informieren, die man akzeptiert – unabhängig vom MTU-Wert der Netzwerkverbindung, der möglicherweise größer ist.

Die `nat`-Tabelle verfügt über Erweiterungsmodule für Adress- und Port-Übersetzung auf Absender- und Empfängerseite. Mit diesen Modulen sind die folgenden Formen der Network Address Translation möglich:

- SNAT: Source NAT, also Adressübersetzung auf Absenderseite
- DNET: Destination NAT, Adressübersetzung auf Empfängerseite
- MASQUERADE: Eine spezialisierte Form der Absenderadressübersetzung für Verbindungen, denen eine vorübergehende, veränderliche, dynamisch ausgewählte IP-Adresse zugewiesen wird, also typischerweise für Einwählverbindungen ins Internet.
- REDIRECT: Eine spezialisierte Form der Empfängeradressübersetzung, die ein Paket an ein Programm auf dem eigenen Computer weiterleitet, völlig unabhängig von den Empfängerangaben im IP-Header.

3.2.1 Features der `filter`-Tabelle

`iptables` beinhaltet u.A. die folgenden neuen Möglichkeiten beim Paketfiltern:

- Portlisten für Absender und Empfänger
- Zugriff auf die Verbindungsflags von TCP-Verbindungen
- Zugriff auf das Optionsfeld des TCP-Headers
- Überwachung des Verbindungszustandes für den Datenaustausch mit TCP, UDP und ICMP
- Zugriff auf das TOS-Feld des IP-Headers
- Zugriff auf die MAC-Adresse des Absenders
- Erweiterte Protokollierungsmöglichkeiten
- Paketauswahl zur Begrenzung des Datendurchsatzes
- Erweiterte Möglichkeiten zum Zurückweisen unerwünschter Pakete
- Übergabe von Paketen an Userspace-Programme über Warteschlangen
- Filtern abgehender Pakete nach Benutzerkennung, Gruppe oder Prozessnummer des sendenden Programms

Für Absender- und Empfängerport sind kommagetrennte Listen von bis zu 15 Ports möglich. (Allerdings kann man diese Portlisten nicht mit den bereits früher möglichen Portbereichen kombinieren.)

Der Zugriff auf alle Verbindungsflags des TCP-Headers ist möglich, und man kann die Entscheidung über den Verbleib eines Paketes darauf basieren. So ist z.B. die Abwehr von Stealth-Scans möglich.

Eine TCP-Option bietet die Übergabe der größten erlaubten Segmentgröße, die ein Rechner akzeptiert. Das Filtern nach dieser Option ist allerdings schon ein sehr spezieller Fall. (In der Box »Zukünftige Erweiterungen von `iptables`« haben wir das schon kurz angesprochen.)

Man kann den Zustand einer TCP-Verbindung sowie auch einen UDP-Datenaustausch überwachen. So lassen sich Pakete einer Verbindung zuordnen und muss nicht immer bei jedem Paket neu entscheiden. Ein als Teil einer bestehenden Verbindung erkanntes Paket muss nicht die ganze lange Liste von Regeln für Einzelpakete durchlaufen und wird damit schneller verarbeitet. Sobald die Verbindung als solche akzeptiert worden ist, werden nachfolgende Pakete sofort zugelassen.

Das TOS-Feld ist eigentlich von eher historischem Interesse. Normalerweise wird es einfach ignoriert oder aber von Routern im Zusammenhang mit den neuen Definitionen für »differenzierten Service« DS (wer mehr bezahlt, erhält schnelleren Datendurchsatz) interpretiert. Indem man nach dem Inhalt des TOS-Feldes filtert, weist man Paketen lokal unterschiedliche Prioritäten zu.

Ankommende Pakete lassen sich nach der MAC-Absenderadresse filtern. Das ist eine relativ begrenzte Möglichkeit der lokalen Zugriffskontrolle, denn MAC-Adressen werden nicht über Router hinweg weitergegeben.

Protokollaufzeichnungen können durch benutzerdefinierte Texte ergänzt werden. Den so entstehenden Meldungen kann man Loglevel zuweisen und sie damit in `/etc/syslog.conf` unterscheiden. So kann man in `syslog.conf` das Logging an- und ausschalten bzw. entscheiden, welche Meldungen in welchen Dateien abgespeichert werden.

Zur Begrenzung des Datendurchsatzes definiert man eine *Burst-Rate* für den Beginn der Verbindung; anschließend greift eine Begrenzung in Form von Paketen pro Sekunde. Bei aktivierter Durchsatzbegrenzung ist die Voreinstellung, dass nach einem Burst von fünf Paketen nur noch drei Pakete pro Stunde erlaubt sind. Mit anderen Worten, wenn jemand z.B. eine Ping-Flut versucht, dann dürften die ersten fünf Pings frei passieren. Das nächste Ping-Paket wäre erst nach zwanzig Minuten erlaubt, egal wie viele Echo-Requests ankommen, das folgende erst nach weiteren 20 Minuten. Ob die ankommenden Pakete protokolliert werden oder nicht, hängt von den nachfolgenden Regeln für die Pakete ab.

Für REJECT kann man angeben, welche ICMP-Fehlermeldung (bzw. RST für TCP-Pakete) zurückgeschickt werden soll. Der IPv4-Standard schreibt für TCP vor, dass sowohl RST als auch ICMP-Fehlermeldungen angenommen werden

müssen, wobei RST das normale TCP-Verhalten ist. In der Voreinstellung schickt `iptables` dem Absender entweder überhaupt keine Nachricht (bei DROP) bzw. eine ICMP-Fehlermeldung (bei REJECT).

Neben REJECT ist QUEUE ein weiteres Target mit besonderen Eigenschaften. QUEUE übergibt das Paket mittels des Netlink-Gerätes an ein Programm im Userspace. Wenn dort kein Programm zuhört, wird das Paket stillschweigend verworfen.

Das Target RETURN dient dazu, eine benutzerdefinierte Regelkette abzubrechen, bevor sie vollständig abgearbeitet ist.

Lokal generierte Pakete können nach Benutzer, Gruppe, Prozessnummer oder Prozessgruppe des sie erzeugenden Programms gefiltert werden. So kann man den Zugriff auf Netzwerkdienste nicht nur über einfache Paketfilter regulieren, sondern auch den einzelnen Benutzern eines Systems unterschiedliche Rechte einräumen. Diese spezielle Option ist allerdings Allzweckcomputern mit mehreren Benutzeraccounts vorbehalten – Sie erinnern sich, eine Firewall sollte nicht über Benutzeraccounts verfügen.

3.2.2 Features der nat-Tabelle

Es gibt im Wesentlichen drei Formen der Adressübersetzung (NAT):

- *Traditionelles, unidirektionelles NAT für abgehende Pakete*: Für Netzwerke mit IP-Adressen aus einem privaten Adressbereich.
  - *Einfaches NAT*: Dabei werden nur IP-Adressen übersetzt. Typischerweise bildet man damit eine Reihe lokaler, privater Absenderadressen auf eine IP-Adresse aus einem öffentlichen Pool ab.
  - NAPT *(Network Address Port Translation*): Abbildung mehrerer lokaler Absenderadressen auf eine einzige öffentliche Adresse, z.B. das Masquerading von Linux.
- *Bidirektionelles NAT*: Bei der Adressübersetzung in beide Richtungen sind abgehende und ankommende Verbindungen möglich. Diese Form des NAT benutzt man z.B. zur Abbildung zwischen Adressbereichen von IPv4 und IPv6.
- *Doppeltes NAT*: Übersetzung von Absender- und Empfängeradressen in beide Richtungen ermöglicht wieder sowohl abgehende als auch ankommende Verbindungen. Doppeltes NAT benutzt man z.B., wenn sich die Adressbereiche von Absender- und Empfängernetzwerk überschneiden, z.B. wenn jemand fälschlicherweise öffentliche IP-Adressen benutzt, die eigentlich einer anderen Site gehören. Doppeltes NAT wird manchmal auch aus purer Bequemlichkeit eingesetzt, wenn die Rechner einer Site neu nummeriert worden sind, oder wenn ein neuer IP-Adressblock zugewiesen wurde und der Admin die Umstellung noch nicht auf den Rechnern vornehmen möchte.

Die NAT-Implementierung von `iptables` unterstützt NAT für Absender- (SNAT) und Empfängeradressen (DNAT). D.h. die Tabelle erlaubt die Modifikation sowohl der Absender- als auch der Empfängeradresse im Paketheader. Dabei stehen folgende drei Ketten zur Verfügung:

- Die PREROUTING-Kette modifiziert ein ankommendes Paket, bevor es die Routing-Funktion des Kernels erreicht (DNAT). Die Empfängeradresse im Header kann entweder durch den Localhost ersetzt werden (z.B. für transparente Proxies oder Port-Umleitung) oder durch eine andere IP-Adresse – für Host-Forwarding (entsprechend der Funktionalität des alten `ipmasqadm`, unter Linux spricht man von Port-Forwarding) oder Lastverteilung.
- Die OUTPUT-Kette korrigiert die Empfängeradresse in lokal erzeugten Paketen, bevor das Routing-Modul aktiv wird (DNAT oder REDIRECT). Meistens will man damit ein abgehendes Paket transparent an einen Proxy umleiten, aber man kann auch Port-Forwarding an einen anderen Rechner realisieren.
- Die POSTROUTING-Kette enthält Änderungen der Absenderadresse an Paketen, die über den eigenen Computer weitergeleitet werden (SNAT bzw. MASQUERADE). Diese Änderungen werden erst umgesetzt, wenn über das Routing des Paketes schon entschieden worden ist.

MASQUERADING UND IPTABLES

Bei `iptables` ist Masquerading eine Sonderform des NAT für Absenderadressen. Wenn die Verbindung verloren geht, vergisst der Rechner den Verbindungszustand sofort. Zielgruppe sind Verbindungen mit vorübergehend zugewiesener IP-Adresse, z.B. Einwahlverbindungen ins Internet. Selbst wenn der Benutzer die Verbindung sofort wieder aufbaut, erhält er trotzdem eine andere IP-Adresse als zuvor. (Wenn die Internetverbindung über ein Kabelmodem oder über DSL hergestellt wird, ist das oft anders: Bei dieser Technik erhält man oft immer die gleiche IP-Adresse.)

Bei regulärem SNAT hingegen merkt sich der Computer die bestehenden Verbindungen bis zu einem bestimmten Timeout. Wenn die Verbindung schnell genug wiederhergestellt wird, können Netzwerkprogramme ununterbrochen weiterarbeiten, weil sich die IP-Adresse nicht geändert hat und etwa verlorene TCP-Pakete einfach nochmals übertragen werden.

Diese Unterscheidung zwischen MASQUERADE einerseits und SNAT andererseits soll ein Problem früherer NAT-/Masquerading-Implementierungen von Linux vermeiden: Wenn eine Einwahlverbindung unterbrochen wurde und der Benutzer sie sofort wieder aufbaute, erhielt er eine neue IP-Adresse. Diese konnte er aber noch nicht gleich benutzen, weil die alten Informationen über IP-Adresse und bestehende NAT-Verbindungen noch im Speicher waren und erst nach einem Timeout freigegeben wurden.

Abbildung 3.4 zeigt die NAT-Ketten und ihren Bezug zum Routing-Modul des Kernels und zu den INPUT-, OUTPUT- und FORWARD-Ketten.

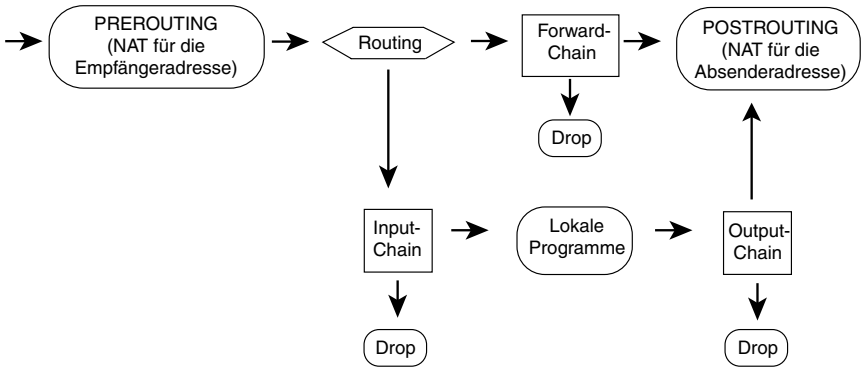

Abbildung 3.4: Der Weg von Paketen bei NAT (Abbildung basiert auf dem Linux-2.4-Paketfilter-HOWTO v1.0.1 und dem Linux-2.4-NAT-HOWTO v1.0.1.)

Achtung: Wenn ein Paket auf dem eigenen Computer erzeugt wird, sitzt die Routing-Funktion logisch zwischen dem lokalen Programm und der Output-Chain. Zunächst entscheidet das statische Routing über das Netzwerkinterface, über das die Pakete verschickt werden; erst anschließend werden die Regeln der Output-Chain angewandt.

3.2.3 Features der mangle-Tabelle

Mit der `mangle`-Tabelle »markieren« Sie ein Paket, d.h. Sie weisen ihm einen Netfilter-interne MARK-Wert zu. Gleichzeitig können Sie das TOS-Feld verändern, um eine lokale Entscheidung über Routing bzw. Weiterleitung zu beeinflussen. Die `mangle`-Tabelle hat zwei eingebaute Ketten:

- Die PREROUTING-Kette ändert ankommende Pakete unmittelbar beim Eingang auf einem Interface, noch bevor irgendwelche Entscheidungen über Routing oder lokale Auslieferung getroffen werden.
- Die OUTPUT-Kette ändert abgehende Pakete, die auf dem eigenen Rechner erzeugt worden sind.

Ein Linux-Router beachtet, je nach Konfiguration, entweder die in der `mangle`-Tabelle gesetzten TOS-Flags oder die Flags, die von anderen Rechnern im LAN gesetzt worden sind.

Die `iptables`-Dokumentation enthält nur wenig Information über das Markieren von Paketen. Es heißt dort lediglich, dass dieser interne Wert für die Quality-of-Service-Implementierung von Linux gebraucht wird und dass er für die Kommunikation zwischen den einzelnen Modulen von `iptables` benutzt werden kann.

Die vorangehenden Abschnitte bieten einen Überblick über die Eigenschaften von `iptables` und über die allgemeine Struktur und Funktionalität der einzelnen Module. Die folgenden Abschnitte zeigen die Syntax, mit der Sie auf die geschilderten Features zugreifen.

3.3 Syntax von iptables

Wie bereits geschildert, trennt `iptables` die Regeln für unterschiedliche Funktionen in jeweils eigene Tabellen auf. Wenn man auf eine Tabelle zugreifen möchte, die gerade nicht voreingestellt ist, muss man dies über eine Kommandozeilenoption angeben. Es existieren drei Tabellen:

- `filter`: Die `filter`-Tabelle ist voreingestellt. Sie enthält die eigentlichen Filterregeln für die Firewall. Ihre eingebauten Ketten sind:
  - INPUT
  - OUTPUT
  - FORWARD
- `nat`: Die `nat`-Tabelle enthält die Regeln für Adress- und Portübersetzung auf Absender- und Empfängerseite. Diese Regeln unterscheiden sich in ihrer Funktionalität von den Filterregeln der Firewall. Die eingebauten Ketten sind:
  - PREROUTING (DNAT/REDIRECT)
  - OUTPUT (DNAT/REDIRECT)
  - POSTROUTING (SNAT/MASQUERADE)
- `mangle`: Die `mangle`-Tabelle enthält Regeln zum Setzen besonderer Routingflags. Diese Flags werden später durch Regeln der `filter`-Tabelle abgefragt. Eingebaute Ketten sind
  - PREROUTING (für geroutete Pakete)
  - OUTPUT (für lokal erzeugte Pakete)

KONVENTIONEN BEI DER SYNTAXDARSTELLUNG
In der Computerwelt gibt es einige Standard-Konventionen für die Darstellung von Kommandozeilensyntax. Falls Unix oder Computer-Dokumentationen im Allgemeinen noch Neuland für Sie darstellen, finden Sie die im Folgenden gebrauchten Konventionen in Tabelle 3.1 noch einmal zusammengefasst.

Element	Beschreibung
`|`	Ein senkrechter Strich trennt alternative Optionen voneinander. Beispielsweise existieren die meisten Befehle von `iptables` in einer Kurz- und einer Langform, z.B. `-L` und `--list`.
`<Wert>`	Spitze Klammern stehen für einen frei wählbaren Wert, z.B. einen Textstring oder eine Zahl.
`[...]`	Optionale Kommandos, Optionen oder Werte stehen in eckigen Klammern. Beispielsweise ist bei den meisten Vergleichsoperatoren eine Negation des Vergleichs durch Voranstellen eines Ausrufezeichens möglich. Dieser optionale Negationsoperator steht meistens zwischen dem Vergleichsoperator und dem Wert, mit dem verglichen werden soll.
`<Wert>:<Wert>`	Ein Doppelpunkt steht für einen Wertebereich. Die beiden angegebenen Werte stehen für den kleinsten und den größten erlaubten Wert. Weil Bereiche an sich optional sind, findet man diese Form am häufigsten als `<Wert>[:<Wert>]`.
`<Wert>,<Wert>`	Ein Komma steht für eine Werteliste. Das überprüfte Feld muss mit einem der angegebenen Werte übereinstimmen. Weil Listen an sich optional sind, findet man diese Form am Häufigsten als `<Wert>[,<Wert>[,<Wert>]...]`.

Tabelle 3.1: Konventionen für die Darstellung der Syntax an der Kommandozeile

3.3.1 Befehle für die filter-Tabelle

Für die Befehle der `filter`-Tabelle ist das Modul `ip_tables` zuständig. Die entsprechende Funktionalität wird durch Laden des Moduls aktiviert. Dies geschieht in der Regel automatisch, sobald Sie den `iptables`-Befehl das erste Mal verwenden.

filter-Befehle, die sich auf komplette Chains beziehen

Tabelle 3.2 zeigt die `iptables`-Befehle, die sich auf komplette Chains beziehen.

Befehl	Beschreibung
`-N | --new-chain <Chain>`	Erzeugt eine benutzerdefinierte Chain
`-F | --flush [<Chain>]`	Löscht alle Regeln einer Chain, oder alle Regeln aller Chains, sofern keine Chain explizit angegeben wird
`-X | --delete-chain [<Chain>]`	Löscht die angegebene benutzerdefinierte Chain, oder alle benutzerdefinierten Ketten, sofern keine Chain explizit angegeben wird

Tabelle 3.2: `iptables`-Befehle für komplette Chains

Befehl	Beschreibung
`-P | --policy <Kette> <Policy>`	Legt die voreingestellte Policy für eine der eingebauten Chains fest, also für INPUT, OUTPUT oder FORWARD. Erlaubte Policies sind `ACCEPT` und `DROP`.
`-L | --list [<Chain>]`	Zeigt die Regeln einer Kette an, oder alle Regeln aller Ketten, sofern keine Chain explizit angegeben wird
`-Z | --zero`	Setzt die Paket- und Byte-Zähler für alle Chains auf null zurück
`-h | <irgendein Befehl> -h`	Zeigt alle erlaubten Befehle und Optionen von `iptables` an, oder zeigt Syntax und Optionen für den angegebenen Befehl an

Tabelle 3.2: `iptables`-Befehle für komplette Chains (Forts.)

Der Hilfebefehl `-h` ist natürlich kein Befehl, der sich auf eine komplette Chain bezieht, aber ich wusste auch nicht, wo ich ihn sonst hätte einordnen sollen.

Für den Befehl zum Anzeigen einer Kette gibt es noch ein paar zusätzliche Optionen, siehe Tabelle 3.3.

Option	Beschreibung
`-L -n | --numeric`	Die Anzeige von IP-Adressen und Portnummern erfolgt numerisch, die Namen werden also nicht aufgelöst
`-L -v | --verbose`	Ausführliche Anzeige mit zusätzlichen Informationen zu jeder Regel, z.B. mit Byte- und Paketzählern, Regeloptionen, Netzwerkinterfaces usw.
`-L -x | --exact`	Zähler werden als genaue Werte angezeigt und nicht abgerundet
`-L --line-numbers`	Zeigt zusätzliche Zeilennummern mit der Position jeder Regel innerhalb der Chain an

Tabelle 3.3: Optionen für den List-Befehl

filter-Befehle für eine einzelne Regel

Die meistbenutzten Kommandos zum Anlegen und Löschen von Regeln innerhalb einer Chain finden Sie in Tabelle 3.4.

Befehl	Beschreibung
`-A | --append <Chain>`	Hängt eine neue Regel an das Ende der angegebenen Chain an
`-I | --insert <Chain>`	Fügt eine neue Regel am Anfang der angegebenen Chain ein

Tabelle 3.4: Befehle für einzelne Regeln

Befehl	Beschreibung
`-D | --delete <Chain> <Regelnummer>`	Löscht die Regel mit der angegebenen Nummer aus der Chain

Tabelle 3.4: Befehle für einzelne Regeln (Forts.)

Einfache Vergleiche in der filter-Tabelle

Die grundlegenden Vergleichsoperationen für die normale `filter`-Tabelle von `iptables` finden Sie in Tabelle 3.5.

Befehl	Beschreibung
`-i | --in-interface [!] [<Interface>]`	Gibt das Netzwerkinterface an, auf das sich eine Regel für ein ankommendes Paket der INPUT- oder FORWARD-Chain oder einer benutzerdefinierten Unter-Chain bezieht. Wenn kein Interface angegeben wird, gilt die Regel für alle Netzwerkinterfaces.
`-o | --out-interface [!] [<Interface>]`	Gibt das Netzwerkinterface an, auf das sich eine Regel für ein abgehendes Paket der OUTPUT- oder FORWARD-Chain oder einer benutzerdefinierten Unter-Chain bezieht. Wenn kein Interface angegeben wird, gilt die Regel für alle Netzwerkinterfaces.
`-p | --protocol [!] [<Protokoll>]`	Gibt das IP-Protokoll an, auf das sich die Regel bezieht. Erlaubte Protokollnamen sind `tcp`, `udp`, `icmp` sowie `all`. Als Protokoll sind dabei anstelle der Namen auch die numerischen Werte aus `/etc/protocols` erlaubt.
`-s | --source | --src [!] <Adresse>[/ <Maske>]`	Gibt eine IP-Adresse oder ein Netzwerk als Absenderadresse im IP-Header an.
`-d | --destination | --dst [!] <Adresse>[/ <Maske>]`	Gibt eine IP-Adresse oder ein Netzwerk als Empfängeradresse im IP-Header an.
`-j | --jump <Target>`	Gibt an, was mit Paketen geschehen soll, die zu der gerade formulierten Regel passen. Voreingestellte Targets sind ACCEPT und DROP, aber Sie können z.B. auch eine selbst definierte Chain angeben.

Tabelle 3.5: Einfache Vergleiche in der `filter`-Tabelle

TARGETS SIND OPTIONAL!
Wenn eine Regel auf ein Paket zutrifft, ohne dass ein Target angegeben wäre, wird das Paket zwar unter der betreffenden Regel gezählt, aber es durchläuft die Regeltabelle weiter.

Vergleiche in der filter-Tabelle für TCP-Pakete

Tabelle 3.6 zeigt die Optionen für TCP-Pakete.

-p tcp	Beschreibung
`--source-port | --sport [!] <Port>[:<Port>]`	Angabe des Absender-Ports
`--destination-port | --dport [!] <Port>[:<Port>]`	Angabe des Empfänger-Ports
`--tcp-flags [!] <Flag>[,<Flag>] <Gesetzt>[,<Gesetzt>]`	Angabe der TCP-Flags. Dabei werden zwei Gruppen von Flags angegeben: Nur die in der ersten Gruppe genannten Flags werden überprüft, und von Ihnen müssen genau die in der zweiten Gruppe enthaltenen Flags gesetzt sein. Z.B. fordert `--tcp-flags SYN,ACK,FIN,RST SYN`, dass SYN im Paket gesetzt und ACK, FIN und RST nicht gesetzt sind.
`[!] --syn`	Das SYN-Flag muss gesetzt sein, d.h. es muss sich um eine Verbindungsanfrage handeln.
`--tcp-option [!] <Zahl>`	Die einzige erlaubte TCP-Option ist die Angabe der maximalen Paketgröße, die der sendende Host akzeptieren will.

Tabelle 3.6: Vergleiche in der `filter`-Tabelle für TCP-Pakete

Vergleiche in der filter-Tabelle für UDP-Pakete

Tabelle 3.7 zeigt die Optionen für UDP-Header.

-p udp	Beschreibung
`--source-port | --sport [!] <Port>[:<Port>]`	Angabe des Absender-Ports
`--destination-port | --dport [!] <Port>[:<Port>]`	Angabe des Empfänger-Ports

Tabelle 3.7: Vergleiche in der `filter`-Tabelle für UDP-Pakete

Vergleiche in der filter-Tabelle für ICMP-Pakete

Tabelle 3.8 zeigt die Optionen für ICMP-Pakete.

-p icmp	Beschreibung
`--icmp-type [!] <Typ>`	Angabe des ICMP-Typs als Zahl oder Name. Der ICMP-Typ wird anstelle des Absender-Ports benutzt.

Tabelle 3.8: Vergleiche in der `filter`-Tabelle für ICMP-Pakete

Die wichtigsten unterstützten ICMP-Typen in numerischer und alphanumerischer Form lauten:

- `echo-reply` (0)
- `destination-unreachable` (3)
  - network-unreachable
  - host-unreachable
  - protocol-unreachable
  - port-unreachable
  - fragmentation-needed
  - network-unknown
  - host-unknown
  - network-prohibited
  - host-prohibited
- `source-quench` (4)
- `redirect` (5)
- `echo-request` (8)
- `time-exceeded` (10)
- `parameter-problem` (11)

WEITERE UNTERSTÜTZTE TYPEN

`iptables` unterstützt noch eine Reihe zusätzlicher, weniger häufiger oder nur von Routern eingesetzter Nachrichtentypen und -subtypen. Die vollständige Liste erhalten Sie durch Eingabe des folgenden Befehls:

```
iptables -p icmp -h
```

3.3.2 Target-Erweiterungen für die filter-Tabelle

Durch Erweiterungsmodule stehen zusätzliche Targets für die `filter`-Tabelle zur Verfügung. Diese beinhalten das Protokollieren von Paketen sowie die Option, ein Paket nicht einfach stillschweigend zu verwerfen, sondern eine konfigurierbare Fehlermeldung zurückzuschicken.

Tabelle 3.9 zeigt die Optionen für das Target LOG; Tabelle 3.10 erklärt die Option für das Target REJECT.

-j LOG	Beschreibung
`--log-level <Syslog-Loglevel>`	Als Loglevel geben Sie eine Priorität für das Syslog an, und zwar entweder in numerischer oder symbolischer Form. Eine vollständige Liste finden Sie in der Datei `/usr/include/sys/syslog.h`. Dabei handelt es sich um die gleichen Loglevel, die auch in `/etc/syslog.conf` Verwendung finden: `emerg` (0), `alert` (1), `crit` (2), `err` (3), `warn` (4), `notice` (5), `info` (6) und `debug` (7).
`--log-prefix <"Beschreibung">`	Als Beschreibung können Sie – in Anführungszeichen – einen beliebigen String angeben, der mit ausgegeben wird, wenn diese Regel eine Protokollmeldung auslöst.
`--log-ip-options`	Dieser Befehl bewirkt, dass im IP-Header enthaltene Optionen mit protokolliert werden.
`--log-tcp-sequence`	Dieser Befehl bewirkt, dass die im TCP-Header enthaltene Sequenznummer mit protokolliert wird.
`--log-tcp-options`	Dieser Befehl bewirkt, dass im TCP-Header enthaltene Optionen mit protokolliert werden.

Tabelle 3.9: Target-Erweiterung LOG

-j REJECT	Beschreibung
`--reject-with <ICMP-3-Subtyp`	In der Voreinstellung wird eine ICMP-Typ-3-Meldung vom Subtyp `icmp-port-unreachable` zurückgegeben, wenn die Annahme eines Paketes mit Fehlermeldung verweigert wird. Sie können sich stattdessen aber einen beliebigen Subtyp des Typs 3 aussuchen, z.B. `icmp-net-unreachable`, `icmp-host-unreachable`, `icmp-proto-unreachable`, `icmp-net-prohibited` oder `icmp-host-prohibited`.
`--reject-with tcp-reset`	Eingehende TCP-Pakete werden auf Wunsch mit der eher Standard-konformen TCP-RST-Meldung verweigert, nicht mit einer ICMP-Fehlermeldung.
`--reject-with echo-reply`	Bei abgelehnten Ping-Echo-Requests kann als »Fehlermeldung« eine simulierte Antwort (ein `echo-reply`) zurückgegeben werden. Die Firewall erzeugt also eine Antwort auf das Ping, ohne die Anfrage an den eigentlich angesprochenen Host weiterzuleiten.

Tabelle 3.10: Target-Erweiterung REJECT

3.3.3 Erweiterungen für die Vergleiche in der filter-Tabelle

Die so genannten Match-Erweiterungen für die `filter`-Tabelle erlauben Zugriff auf die Felder der TCP-, UDP- und ICMP-Header. Sie unterstützen auch einige

neue Features von `iptables`, z.B. die Überwachung des Verbindungszustandes, Portlisten, den Zugriff auf die Hardware-MAC-Adresse sowie den Zugriff auf das TOS-Feld des IP-Headers.

> **SYNTAX FÜR DIE MATCH-ERWEITERUNGEN**
> Für diese Erweiterungen ist der Befehl `-m` oder `--match` nötig, der das Erweiterungsmodul lädt. Es folgen die jeweils gewünschten Optionen.

Match-Erweiterung für die filter-Tabelle: Multiport-Listen

Bei Multiport-Listen kann eine einzige Regel sich auf bis zu 15 Ports beziehen. Innerhalb der Liste sind Leerzeichen nicht erlaubt, d.h. zwischen den Kommas und den Ports darf kein Abstand bleiben. Das Mischen von Portbereichen und Portlisten ist nicht erlaubt. Der Befehl `-m multiport` muss unmittelbar nach der Protokollangabe `-p <Protokoll>` stehen.

Tabelle 3.11 zeigt die mit dieser Erweiterung verfügbaren Optionen.

-m | --match multiport	**Beschreibung**
`--source-port <Port>[,<Port>]`	Angabe von Absenderports
`--destination-port <Port>[,<Port>]`	Angabe von Empfängerports
`--port <Port>[,<Port>]`	Absender- und Empfängerport sind identisch und in der Liste enthalten

Tabelle 3.11: Die Multiport-Erweiterung

Die Multiport-Syntax ist manchmal ein bisschen gemein. Ich darf daher ein paar Beispiele und Tricks demonstrieren:

Die folgende Regel sperrt ankommende Pakete an die UDP-Ports für NFS und `lockd`:

```
iptables -A INPUT -i eth0 -p udp\
         -m multiport --destination-port 2049,4045 -j DROP
```

Die nächste Regel sperrt abgehende Verbindungsanfragen an hohe Ports, die zu den TCP-Diensten NFS, `socks` und `squid` gehören:

```
iptables -A OUTPUT -o eth0 -p tcp\
         -m multiport --destination-port 2049,1080,3128 --syn -j REJECT
```

Achten Sie darauf, dass der Multiport-Befehl unmittelbar auf die Protokollangabe folgen muss! Wenn Sie beispielsweise das `--syn` zwischen `-p tcp` und `-m multiport` platziert hätten, wäre das Ergebnis nur ein Syntaxfehler gewesen.

Ein ähnliches Beispiel – die folgenden Zeilen sind korrekt:

```
iptables -A INPUT -i <Interface> -p tcp \
        -m multiport --source-port 80,443 ! --syn -j ACCEPT
```

Hingegen resultiert dies in einem Syntaxfehler:

```
iptables -A INPUT -i <Interface> -p tcp ! --syn \
        -m multiport --source-port 80,443 -j ACCEPT
```

Zudem ist die korrekte Platzierung der Parameter für Absender und Empfänger nicht besonders intuitiv. Die folgenden beiden Varianten sind korrekt:

```
iptables -A INPUT -i <Interface> -p tcp -m multiport \
        --source-port 80,443 \
        ! --syn -d $IPADDR --dport 1024:65535 -j ACCEPT
```

und

```
iptables -A INPUT -i <Interface> -p tcp -m multiport \
        --source-port 80,443 \
        -d $IPADDR ! --syn --dport 1024:65535 -j ACCEPT
```

Dies hingegen ist syntaktisch falsch:

```
iptables -A INPUT -i <Interface> -p tcp -m multiport \
        --source-port 80,443 \
        -d $IPADDR --dport 1024:65535 ! --syn -j ACCEPT
```

Das Modul sorgt auch noch für ein paar überraschende Nebeneffekte. Die beiden gerade gezeigten korrekten Regeln werden plötzlich fehlerhaft, wenn man den Bezug auf das SYN-Flag entfernt:

```
iptables -A INPUT -i <Interface> -p tcp -m multiport \
        --source-port 80,443 \
        -d $IPADDR --dport 1024:65535 -j ACCEPT
```

Diese beiden Regeln hingegen sind erlaubt:

```
iptables -A OUTPUT -o <Interface> \
       -p tcp -m multiport --destination-port 80,443 \
       ! --syn -s $IPADDR --sport 1024:65535 -j ACCEPT

iptables -A OUTPUT -o <Interface> \
       -p tcp -m multiport --destination-port 80,443 \
      --syn -s $IPADDR --sport 1024:65535 -j ACCEPT
```

Beachten Sie, dass die Option `--destination-port` für das Multiport-Modul nicht identisch ist mit dem Argument `--destination-port` bzw. `--dport` im normalen Vergleichsmodul für `-p tcp`-Argumente.

Es bleibt zu hoffen, dass die Interpretation der Argumente für die einzelnen Module in Zukunft etwas flexibler wird.

Match-Erweiterung für die filter-Tabelle: Durchsatzbegrenzung

Die Begrenzung des Datendurchsatzes ist vor allem nützlich, um die Zahl der Protokollmeldungen zu reduzieren. Tabelle 3.12 zeigt die verfügbaren Optionen.

-m | --match limit	Beschreibung
`--limit <Rate>`	Höchstzahl von Paketen innerhalb einer gegebenen Zeit
`--limit-burst <Zahl>`	Höchstzahl von Paketen, die passieren dürfen, bevor die durch `--limit` angegebene Zeitbegrenzung wirksam wird

Tabelle 3.12: Die Limit-Erweiterung

Die Burst-Rate legt fest, dass zunächst einmal eine gewisse Anzahl von Paketen ohne zeitliche Begrenzung akzeptiert wird. Die Voreinstellung ist fünf. Wenn diese Zahl erreicht worden ist, werden weitere Pakete nur noch entsprechend der eingestellten Begrenzung akzeptiert. Die Voreinstellung hierfür liegt bei drei Paketen pro Stunde. Die Zeiteinheit, in der die Pakete gezählt werden, ist wählbar: `/second`, `/minute`, `/hour` oder `/day`.

Mit anderen Worten funktioniert die Voreinstellung so: Wenn innerhalb des Zeitlimits von einer Stunde die Burst-Rate von fünf Paketen erreicht wird, dann sind in der Folge nur noch drei Pakete pro Stunde erlaubt, eines in zwanzig Minuten, egal wie viele Pakete tatsächlich ankommen. Wenn innerhalb dieser Zeitgrenze kein Paket gesehen wird, wird der Burst-Zahl wieder eines gutgeschrieben.

Diese Durchsatzbegrenzung lässt sich viel besser an einem Beispiel demonstrieren, als sie mit Worten zu beschreiben. Die folgende Regel begrenzt die Protokollierung ankommender Ping-Pakete auf eines pro Sekunde, sobald einmal in einer beliebigen Sekunde mehr als fünf Echo-Requests angekommen sind:

```
iptables -A INPUT -i eth0 \
         -p icmp --icmp-type echo-request \
         -m limit --limit 1/second -j LOG
```

Natürlich können Sie damit auch die Annahme von Paketen begrenzen. Die folgenden beiden Regeln zusammen besagen, dass nur noch ein Ping-Paket pro Sekunde angenommen wird, sobald einmal innerhalb einer Sekunde fünf Echo-Requests angekommen sind. Alle weiteren Pings werden stillschweigend verworfen!

```
iptables -A INPUT -i eth0 \
         -p icmp --icmp-type echo-request \
         -m limit --limit 1/second -j ACCEPT

iptables -A INPUT -i eth0 \
         -p icmp --icmp-type echo-request -j DROP
```

Die nächste Regel begrenzt die Zahl der Protokollmeldungen, die bei verworfenen ICMP-Redirects erzeugt werden. Sobald einmal innerhalb von 20 Minuten

fünf Pakete protokolliert worden sind, werden in der folgenden Stunde maximal drei weitere Meldungen im Syslog generiert, eine alle zwanzig Minuten:

```
iptables -A INPUT -i eth0 \
         -p icmp --icmp-type redirect \
         -m limit -j LOG
```

In diesem letzten Beispiel gehen wir davon aus, dass das betreffende Paket und alle weiteren – nicht explizit von einer bestimmten Filterregel ausgewählten – Redirect-Pakete durch die DROP-Voreinstellung der INPUT-Chain gelöscht werden.

Match-Erweiterung für die filter-Tabelle: Verbindungszustand

Statische Paketfilter überwachen immer nur einzelne Pakete. Ohne Kontextbezug werden bei jedem Paket nur seine eigenen Daten angesehen: Absender- und Empfängeradresse, Ports, Protokoll und die in diesem konkreten Paket gesetzten TCP-Flags. ICMP-Meldungen werden als isolierte Ereignisse angesehen.

Das Erweiterungsmodul für den Verbindungszustand enthält zusätzliche Überwachungs- und Aufzeichnungsfunktionen, mit denen man diese statische, kontextfreie Technik ergänzen kann. Immer wenn eine TCP-Verbindung oder ein UDP-Datenaustausch beginnt, werden Daten über den Verbindungszustand aufgezeichnet. Die folgenden Pakete können nun nicht mehr nur anhand der statischen Informationen überprüft werden, die in ihnen selbst enthalten sind, sondern man kann sie im Kontext eines Datenaustausches interpretieren. Mit anderen Worten: Ein Teil des kontextbezogenen Wissens, das normalerweise nur in der Transportschicht von TCP bzw. der Anwendungsschicht von UDP zur Verfügung steht, wird nun auf der Ebene des Paketfilters sichtbar.

Nachdem eine Verbindung aufgebaut und akzeptiert worden ist, werden folgende Pakete als Teil dieses Datenaustausches identifiziert. Etwa auftauchende ICMP-Meldungen werden einem bestimmten Austausch zugeordnet.

(In der Terminologie der Informatik bezeichnet man eine Sammlung von Werten oder Attributen als Tupel, wenn sie gemeinsam ein Ereignis oder Objekt eindeutig identifizieren. Ein UDP- oder TCP-Paket wird durch Protokoll (UDP oder TCP), Absender- und Empfängeradressen sowie Absender- und Empfängerports eindeutig identifiziert; diese Daten bilden ein Tupel.)

Überwachung bestehender Verbindungen: Bei TCP ist es nicht ganz so offensichtlich, welche Vorteile sich aus der Überwachung des Verbindungszustandes ergeben. Schließlich enthalten TCP-Pakete definitionsgemäß Informationen über den Verbindungszustand. Bei UDP ist der unmittelbare Vorteil ganz klar: Man kann Antworten von anderen Paketen unterscheiden. Wenn beispielsweise eine DNS-Anfrage unser System verlässt und damit ein neuer UDP-Datenaustausch beginnt, sind – innerhalb eines bestimmten Zeitfensters – ankommende UDP-Antworten von der IP-Adresse und dem Port erlaubt, an die das ursprüngliche Paket gerichtet war. Ankommende UDP-Pakete von anderen Hosts oder Ports werden nicht ange-

nommen. Sie sind nicht Teil dieser Verbindung. ICMP-Fehlermeldungen werden angenommen, wenn sie einer bestehenden TCP- oder UDP-Verbindung zugeordnet werden können.

Optimierung der Firewall-Geschwindigkeit und Reduktion ihrer Komplexität: Hier sind die Vorteile bei TCP eindeutiger. Das Konzept der *Datenflüsse* dient der Optimierung und Beschleunigung einer Firewall. Das Hauptziel eines solchen Datenflusses ist die Umgehung der normalen Firewall-Regeln für ein Paket. In einigen Fällen erreicht man dadurch ein wesentlich schnelleres Handling des TCP-Paketes: Wenn das Paket sofort als Teil einer bereits genehmigten Verbindung erkannt wird, können die übrigen Firewall-Filter komplett übersprungen werden. Die Kenntnis des Verbindungszustandes kann bei TCP-Verbindungen daher einen riesigen Vorteil in der Performance des Paketfilters darstellen. Gleichzeitig kann man die typischen Regeln für das TCP-Protokoll zu einer einzigen Regel zusammenfassen. Die Zahl der Filterregeln reduziert sich dadurch (zumindest theoretisch – dass es in der Praxis nicht immer so ist, sehen Sie später noch).

Der Hauptnachteil liegt darin, dass die Tabelle mit den ganzen Verbindungen viel mehr Speicher in Anspruch nimmt, als die normalen Firewall-Regeln allein. Ein Router mit 70.000 gleichzeitig erlaubten Verbindungen beispielsweise benötigt eine riesige Menge an Hauptspeicher, um für jede einzelne Verbindung den Zustand zu überwachen. Aus Geschwindigkeitsgründen führt man die Zustandsüberwachung oft in der Hardware durch – assoziative Zugriffe auf die Tabelle können dadurch evtl. gleichzeitig oder parallel durchgeführt werden. Aber egal, ob sie in Form eines Hardwaremoduls oder eines Softwareprogramms implementiert sind, kontextsensitive Firewalls müssen notfalls immer auf die traditionellen Methoden zurückgreifen können, wenn der Speicher einmal ausgeht und ein Eintrag in der Verbindungstabelle nicht mehr möglich ist.

Das Erstellen, Nachschlagen und Löschen der Einträge in der Zustandstabelle kostet auch Zeit. In vielen Fällen ist der dadurch entstehende Overhead unter dem Strich schlecht für das System. Bei einem fortlaufenden Datenaustausch wie z.B. einer FTP-Verbindung oder einer Multimedia-Übertragung über UDP gewinnt man Zeit. In beiden Fällen werden potenziell riesige Mengen an Paketen übertragen, und entsprechend viele Filterregeln müssten bei einem konventionellen System überprüft werden. Hingegen ist die Überwachung des Verbindungszustandes bei einfachen DNS- oder NTP-Abfragen kontraproduktiv. Das Aufzeichnen und Löschen der Verbindung kann in solchen Fällen ebensoviel Prozessorzeit (und ungleich mehr Hauptspeicher!) beanspruchen wie ein einfaches Abklopfen einzelner Filterregeln.

Bei Firewalls, die vor allem WWW-Daten abwickeln, sind die Vorteile zumindest fragwürdig. Der Client-Server-Austausch im Web ist meistens eher kurz.

Bei `telnet`- und `ssh`-Verbindungen sind die Effekte schwierig vorherzusagen. Wenn reichlich Datenverkehr vorliegt und viele Verbindungen aktiv sind, kann eine kontextsensitive Firewall schneller sein, weil sie die normalen Firewall-

Regeln überspringt. Hingegen wird eine ruhige Session mit wenig Aktivität durch den Benutzer wahrscheinlich zum Timeout kommen. Die aufgezeichneten Daten über die Verbindung werden aus der Zustandstabelle der Firewall gelöscht. Sobald das nächste Paket ankommt, passiert es die ganzen traditionellen Firewall-Regeln und führt dann wieder zu einem neuen Eintrag in der Zustandstabelle.

Tabelle 3.13 zeigt die Optionen für das Verbindungszustands-Modul.

-m | --match state	**Beschreibung**
`--state <Zustand>[,<Zustand>]`	Die Regel trifft zu, wenn sich die Verbindung in einem der angegebenen Zustände befindet. Erlaubte Werte für den Zustand sind `NEW`, `ESTABLISHED`, `RELATED` und `INVALID`.

Tabelle 3.13: Das Modul für Verbindungszustände

Nun werden Informationen über TCP-Verbindungszustände und bestehende UDP-Kommunikationen aufgezeichnet. Verbindungen lassen sich nun nach den Kriterien »neu«, »bestehend« und »verwandt« filtern:

- `NEW` bezieht sich auf eine Verbindungsanfrage mit TCP-SYN-Flag oder auf das erste UDP-Paket.
- `ESTABLISHED` meint die fortlaufenden TCP-ACK-Pakete nach erfolgtem Verbindungsaufbau, bzw. alle folgenden UDP-Pakete zwischen den gleichen IP-Adressen und Ports, sowie ICMP-Echo-Replies, die auf einen vorherigen Echo-Request zurückgeschickt werden.
- `RELATED` (»verwandt«) bezieht sich momentan nur auf ICMP-Fehlermeldungen. Sekundäre FTP-Verbindungen werden über ein zusätzliches Modul zur Überwachung von FTP-Verbindungen abgewickelt. Wenn dieses Modul aktiviert ist, bezieht `RELATED` die sekundären FTP-Verbindungen mit ein.
- Ein ungültiges (`INVALID`) Paket wäre beispielsweise eine ankommende ICMP-Fehlermeldung, die sich nicht auf eine bestehende Verbindung bezieht, oder ein Echo-Reply ohne vorangegangenen Echo-Request.

Im Idealfall erlaubt `ESTABLISHED`, dass die gesamte Firewall-Konfiguration für einen bestimmten Dienst in einer einzigen Regel ausgedrückt werden kann. Diese erlaubt einfach das allererste Paket der Anfrage. Z.B. wäre mit `ESTABLISHED` für einen Web-Client nur noch eine Regel nötig, die das initiale SYN genehmigt. Ein DNS-Client benötigt nur eine einzige Regel, die eine abgehende UDP-Anfrage an den DNS-Server zulässt.

Wenn unbekannte Pakete per Voreinstellung abgelehnt werden, kann man (theoretisch) alle protokollspezifischen Filterregeln durch zwei allgemeine Regeln ersetzen, die ankommende und abgehende Pakete erlauben, sofern sie selbst Teil einer etablierten Verbindung sind oder mit einer solchen Verbindung »verwandt« sind. Anwendungsspezifische Regeln benötigt man dann nur noch für das jeweils erste Paket des Datenaustausches.

Eine kleine Site oder ein Privatrechner kommt mit so einer Konfiguration vermutlich zurecht. Bei einer größeren Site oder einer Firewall mit vielen gleichzeitig bestehenden Verbindungen wäre das aber kaum noch angemessen. Dies liegt in den Timeouts der Zustandstabelle begründet: Der Eintrag für eine Verbindung mit geringem Datenaufkommen wird aus Platzmangel durch einen Eintrag für eine andere Verbindung ersetzt. Das nächste Paket, das eigentlich dem gerade gelöschten Eintrag zugeordnet wäre, benötigt jetzt eine eigene Regel, die das Paket erlaubt und den Eintrag in der Zustandstabelle regeneriert.

Ein einfaches Beispiel: Angenommen, ein lokaler DNS-Server arbeitet im »Cache-and-Forward-Modus«, d.h. er leitet Anfragen an einen Server im Netz weiter und hat einen Zwischenspeicher für die Ergebnisse. Für die Weiterleitung der Anfragen an den Server im Netz benimmt er sich aber wie ein echter Server, d.h. der DNS-Datenaustausch benutzt den Port 53 auf beiden beteiligten Rechnern. Die folgenden Regeln erlauben nun neue Anfragen (`NEW`), ankommende Antworten (`ESTABLISHED`) sowie ICMP-Fehlermeldungen, die im Zusammenhang mit der Anfrage stehen (`RELATED`):

```
iptables -A INPUT -m state \
         --state ESTABLISHED,RELATED -j ACCEPT

iptables -A OUTPUT --out-interface <Interface> -p udp \
         -s $IPADDR --source-port 53 -d $NAME_SERVER --destination-port 53 \
         -m state --state NEW,RELATED -j ACCEPT
```

Das DNS-Protokoll ist sehr einfach und besteht nur aus Frage und Antwort. Was ist mit einer Anwendung, die über längere Zeit eine kontinuierliche Verbindung ermöglicht, z.B. eine FTP-Kontrollverbindung oder eine `telnet`- oder `ssh`-Verbindung? Falls der Eintrag in der Zustandstabelle aus irgendeinem Grund vorzeitig gelöscht werden sollte, fehlt weiteren Paketen dieser Zustandseintrag und sie können keiner bestehenden Verbindung (`ESTABLISHED`) mehr zugeordnet werden.

Die folgenden Regeln berücksichtigen diese Eventualität:

```
iptables -A INPUT -m state \
         --state ESTABLISHED,RELATED -j ACCEPT

iptables -A OUTPUT -m state \
         --state ESTABLISHED,RELATED -j ACCEPT

iptables -A OUTPUT --out-interface <Interface> -p tcp \
         -s $IPADDR --source-port $UNPRIVPORTS \
         -d $FREMDER_TELNET_SERVER --destination-port 23 \
         -m state --state NEW, -j ACCEPT

iptables -A OUTPUT --out-interface <Interface> -p tcp ! --syn \
         -s $IPADDR --source-port $UNPRIVPORTS \
         -d $FREMDER_TELNET_SERVER --destination-port 23 \
```

```
        -j ACCEPT

iptables -A INPUT --in-interface <Interface> -p tcp ! --syn \
        -s $FREMDER_TELNET_SERVER --source-port 23 \
        -d $IPADDR --destination-port $UNPRIVPORTS \
        -j ACCEPT
```

Match-Erweiterung für die filter-Tabelle: MAC-Adressen

Tabelle 3.14 enthält die erlaubte Option für die Abfrage von MAC-Adressen.

-m | --match mac	Beschreibung
`--mac-source [!] <Adresse>`	Angabe einer Layer-2-Ethernet-Hardwareadresse im Absender eines Ethernet-Frames. Die Adresse wird in der Form `xx:xx:xx:xx:xx:xx` angegeben.

Tabelle 3.14: Zugriff auf MAC-Adressen

Denken Sie daran: MAC-Adressen werden über Router (oder an benachbarte Netzwerksegmente) nicht weitergegeben. Man kann auch nur Absenderadressen angeben! Die MAC-Erweiterung darf deshalb nur auf `in-interface`s eingesetzt werden, also z.B. in den Chains INPUT, PREROUTING oder FORWARD.

Die folgende Regel erlaubt ankommende `ssh`-Verbindungen nur von einem bestimmten Computer:

```
iptables -A INPUT -i <Lokales Netzwerkinterface> -p tcp \
        -m mac --mac-source xx:xx:xx:xx:xx:xx \
        --source-port 1024:65535 \
        -d $IPADDR --dport 22 -j ACCEPT
```

Match-Erweiterung für die filter-Tabelle: Erzeuger des Paketes

Die Optionen für die Match-Erweiterung `owner` sind in Tabelle 3.15 aufgeführt.

-m | --match owner	Beschreibung
`--uid-owner <Benutzerkennung>`	Überprüft die Benutzerkennung des erzeugenden Programms
`--gid-owner <Gruppenkennung>`	Überprüft die Gruppenkennung des erzeugenden Programms
`--pid-owner <Prozessnummer>`	Überprüft die Prozessnummer des erzeugenden Programms
`--sid-owner <Session-ID>`	Überprüft die Session-ID des erzeugenden Programms (SID oder PPID, Session-ID oder PID des Mutterprozesses)

Tabelle 3.15: Wer hat ein Paket generiert?

Diese Optionen beziehen sich auf das Programm, das ein Paket erzeugt hat. Diese Erweiterung kann nur in der OUTPUT-Chain verwendet werden.

Auf einer reinen Firewall ergibt die `owner`-Erweiterung keinen echten Sinn; nützlich ist sie vor allem auf Workstations.

Angenommen, Sie haben eine Firewall mit Monitor, aber ohne Tastatur. Die Administration erfolgt von einer lokalen Workstation aus. Nur ein einziger Benutzer dieser Workstation darf sich auf der Firewall einloggen. Auf der Workstation könnte man eine Paketfilter-Regel erstellen, die nur diesem User den Zugang zur Firewall erlaubt:

```
iptables -A OUTPUT -o eth0 -p tcp \
         -s $IPADDR --sport 1024:65535 \
         -d <IP-Adresse der Firewall> --dport 22 \
         -m owner --uid-owner <Benutzer-ID des Administrators> \
         --gid-owner <Gruppe des Admins> -j ACCEPT
```

Match-Erweiterung für die filter-Tabelle: Das Mark-Feld

Tabelle 3.16 erklärt den Gebrauch der Mark-Erweiterung.

-m | --match mark	Beschreibung
`--mark <Wert>[/<Maske>]`	Abfrage des Netfilter-internen Mark-Wertes

Tabelle 3.16: Das Mark-Feld

Wert und Maske sind vorzeichenlose `long`s. Wenn eine Maske angegeben ist, wird sie vor dem Vergleich durch logisches UND mit dem Wert verknüpft.

In folgendem Beispiel fragen wir ab, ob ein ankommendes `telnet`-Paket zwischen den angegebenen Absender- und Empfängeradressen zuvor markiert worden ist:

```
iptables -A FORWARD -i eth0 -o eth1 -p tcp \
         -s <irgendein Absender> --sport 1024:65535 \
         -d <irgendein Empfänger> --dport 23 \
         -m mark --mark 0x00010070 \
         -j ACCEPT
```

Der abgefragte Mark-Wert muss zu einem vorherigen Zeitpunkt der Paketverarbeitung gesetzt worden sein. Er kennzeichnet das Paket als etwas Besonderes, das in irgendeiner Form anders als »normale« Pakete behandelt werden soll.

Match-Erweiterung für die filter-Tabelle: Das TOS-Feld

Tabelle 3.17 zeigt den Gebrauch der Erweiterung für das TOS-Feld.

-m | --match tos	Beschreibung
`--tos <Wert>`	Abfrage des TOS-Feldes aus dem IP-Header

Tabelle 3.17: Abfrage des TOS-Feldes

Der angegebene Wert für das TOS-Feld kann einen der folgenden Werte annehmen, entweder in numerischer oder in symbolischer Form:

- `minimize-delay`, 16, `0x10` – kleinstmögliche Verzögerung
- `maximize-throughput`, 8, `0x08` – größtmöglicher Datendurchsatz
- `maximize-reliability`, 4, `0x04` – kleinstmöglicher Paketverlust
- `minimize-cost`, 2, `0x02` – kleinstmögliche Übertragungskosten
- `normal-service`, 0, `0x00` – unmarkiertes Paket, normale Behandlung

> **RELEVANZ DER TOS-BITS**
> Diese TOS-Bits sind eher von historischem Interesse. Linux unterstützt ihren lokalen Einsatz, und diverse Dokumente zum Setup von Linux-Firewalls beschreiben ihre Funktion. Trotzdem bleibt festzuhalten, dass die TOS-Bits im Allgemeinen nicht benutzt oder geprüft werden.

Für das *Differentiated Services Control Protocol* DSCP hat man das TOS-Feld als DS-Feld (Differenzierter Service) neu definiert.

Weitere Informationen zu diesem Konzept des Differentiated Service finden Sie in folgenden Dokumenten:

- RFC 2474 »Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers«
- RFC 2475 »An Architecture for Differentiated Services«
- RFC 2990 »Next Steps for the IP QoS Architecture«

Match-Erweiterung für die `filter`-Tabelle: »unsaubere« Pakete

Man findet keine Dokumentation darüber, in welcher Weise dieses Modul überprüft, ob ein Paket gültig ist. Das Modul zählt als »experimentell«, die Autoren von `iptables` raten von seiner Verwendung momentan noch ab.

Die folgende Zeile zeigt die Syntax für das `unclean`-Modul. Irgendwelche Optionen oder Argumente existieren nicht.

```
-m | --match unclean
```

Möglicherweise ist der Gebrauch des Moduls schon offiziell abgesegnet, wenn Sie dieses Buch lesen. Bis dahin können wir das Modul immerhin schon einmal für eine Demonstration der LOG-Optionen missbrauchen:

```
iptables -A INPUT -p ! tcp -m unclean \
         -j LOG --log-prefix "UNCLEAN packet: " \
         --log-ip-options

iptables -A INPUT -p tcp -m unclean \
         -j LOG --log-prefix "UNCLEAN TCP: " \
         --log-ip-options \
         --log-tcp-sequence --log-tcp-options

iptables -A INPUT -m unclean -j DROP
```

3.3.4 Target-Erweiterungen für die `nat`-Tabelle

Wie zuvor schon erwähnt, unterstützt `iptables` vier verschiedene Formen der Adressübersetzung: NAT der Absenderadresse (SNAT), NAT der Empfängeradresse (DNAT), Masquerading (MASQUERADE) als Sonderform der SNAT-Implementierung sowie eine lokale Portumleitung (REDIRECT). Jedes dieser in Kapitälchen gedruckten Targets ist in der `nat`-Tabelle verwendbar, wenn eine Regel sich über die Angabe `-t nat` auf diese Tabelle bezieht.

Target-Erweiterungen für SNAT

Die bekannteste Form der Adressübersetzung ist NAT für Absenderadressen und -ports (NAPT). Abbildung 3.5 zeigt, dass die Adressübersetzung nach der Entscheidung über das Routing erfolgt. SNAT ist nur in der POSTROUTING-Chain erlaubt und sinnvoll. Weil SNAT unmittelbar vor dem Absenden des Paketes erfolgt, darf nur ein Netzwerkinterface für abgehende Pakete angegeben werden.

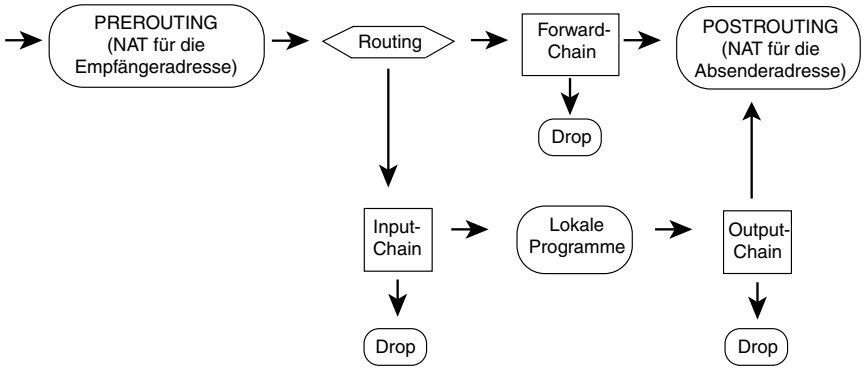

Abbildung 3.5: Der Weg eines Paketes bei NAT

Einige Leute nennen diese (häufigste) Form der Absender-NAT auch NAPT und bringen damit die Änderung der Portnummer zum Ausdruck. Die andere Form unidirektionalen NATs wird bei der Übersetzung zwischen einem privaten LAN

und einem ganzen Pool öffentlicher Adressen eingesetzt; die Portnummern werden dabei nicht angerührt.

NAPT verwendet man, wenn nur eine einzige öffentliche Adresse zur Verfügung steht. Weil sich eine ganze Reihe interner Maschinen den Absenderport teilen müssen, wird er geändert und durch einen freien Port auf der Firewall-Maschine ersetzt. Die von der internen Maschine eingesetzte Portnummer könnte ja schon von einem Prozess auf der Firewall oder einer anderen Maschine im LAN besetzt sein. Wenn ein Paket aus dem Internet zurückkommt, kann die NAT-Maschine nur anhand der Portnummer erkennen, dass das Paket nicht an sie selbst, sondern an eine interne Maschine gerichtet ist. Sie muss dann entscheiden, an welche LAN-Maschine das Paket zugestellt werden soll.

Eine Regel für SNAT sieht etwa so aus:

```
iptables -t nat -A POSTROUTING --out-interface <Interface>  ...  \
         -j SNAT --to-source <Adresse>[-<Adresse>][:<Port>-<Port>]
```

Wenn Sie mehr als eine öffentliche IP-Adresse besitzen, geben Sie einen Adressbereich an.

Auf Wunsch legen Sie explizit fest, auf welchen Portbereich der routenden Maschine der Absenderport abgebildet werden soll.

Target-Erweiterungen für Masquerading

Für die Absenderadressübersetzung bestehen in `iptables` zwei unterschiedliche Implementierungen: SNAT und MASQUERADE. Letzteres ist speziell für den Gebrauch auf Interfaces mit dynamisch zugewiesener IP-Adresse gedacht, insbesondere für vorübergehende Verbindungen, die vermutlich bei jedem neuen Verbindungsaufbau wieder eine andere IP-Adresse erhalten. Gerade für Einwählverbindungen per Modem oder ISDN ist MASQUERADE ideal geeignet.

Masquerading als Sonderform des SNAT erscheint nur in der POSTROUTING-Chain sinnvoll, und die Regel kann sich auch nur auf ein abgehendes Netzwerkinterface beziehen. Im Gegensatz zu dem unspezifischeren SNAT geben Sie bei MASQUERADE die neue Absenderadresse nicht explizit an. Stattdessen erfragt das System automatisch die jeweils aktuelle IP-Adresse des Netzwerkinterfaces.

Die Syntax für MASQUERADE lautet wie folgt:

```
iptables -t nat -A POSTROUTING --out-interface <Interface> ... \
         -j MASQUERADE [--to-ports <Port>[-<Port>]]
```

Sie können einen Wunschbereich angeben, aus dem die neue Portnummer für den Absenderport genommen wird.

Target-Erweiterungen für DNAT

Eine hochspezialisierte Form des NAT ist die Adressübersetzung für Empfängeradressen und -ports. Nützlich dürfte das vor allem für größere Privatsites oder kleinere Firmen sein: Wenn man nur über eine einzige IP-Adresse verfügt, kann man damit ankommende Verbindungen an interne Server weiterleiten, die selbst nicht öffentlich sichtbar sind. Mit anderen Worten: DNAT ersetzt früher notwendige Extra-Software wie z.B. `ipmasqadm`.

Wenn Sie sich noch einmal Abbildung 3.5 ansehen, erkennen Sie, dass die Übersetzung der Empfängeradresse und des Empfängerports vor einer Routingentscheidung stattfindet. DNAT ist in den Chains PREROUTING und OUTPUT erlaubt. In der PREROUTING-Chain ist DNAT in Verbindung mit einem Interface für ankommende Pakete erlaubt, in OUTPUT mit einem Interface für abgehende Pakete.

Die Syntax für DNAT lautet:

```
iptables -t nat -A PREROUTING --in-interface <Interface> ... \
         -j DNAT --to-destination <Adresse>[-<Adresse>][:<Port>-<Port>]

iptables -t nat -A OUTPUT --out-interface <Interface> ... \
         -j DNAT --to-destination <Adresse>[-<Adresse>][:<Port>-<Port>]
```

Wenn mehr als eine IP-Adresse benutzt wird, lässt sich die Empfängeradresse auf einen Adressbereich abbilden. Ähnlich kann der Empfängerport auf einen Bereich alternativer Ports auf der Empfängermaschine abgebildet werden.

Target-Erweiterungen für die lokale Portumleitung (REDIRECT)

Portumleitung ist ein Sonderfall des DNAT. Dabei wird ein Paket an einen bestimmten Port auf dem eigenen Computer umgelenkt. Ankommende Pakete, die sonst weitergeleitet worden wären, gelangen dadurch in die INPUT-Chain des Localhost. Von eigenen Programmen erzeugte, abgehende Pakete landen auf dem Loopback-Interface.

REDIRECT ist eine bequeme Kurzform für eine Umleitung eines Paketes an den eigenen Computer. Zusatznutzen entsteht dadurch nicht. Mit DNAT erreichen Sie denselben Effekt ebenso mühelos.

Damit ist auch REDIRECT nur in den PREROUTING- und OUTPUT-Chains erlaubt. In der PREROUTING-Chain ist es zusammen mit einem ankommenden Interface, in der OUTPUT-Chain mit einem abgehenden Interface erlaubt.

Die Syntax lautet:

```
iptables -t nat -A PREROUTING --in-interface <Interface> ... \
         -j REDIRECT [--to-ports <Port>[-<Port>]]
```

```
iptables -t nat -A OUTPUT --out-interface <Interface> ... \
        -j REDIRECT [--to-ports <Port>[-<Port>]]
```

Als Empfängerport darf wiederum auch ein Portbereich angegeben werden.

3.3.5 Befehle für die mangle-Tabelle

Die zusätzlichen Targets und Erweiterungen für die `mangle`-Tabelle sind in den Chains OUTPUT und PREROUTING erlaubt. Denken Sie daran, dass immer die `filter`-Tabelle voreingestellt ist. Wenn Sie die Features der `mangle`-Tabelle verwenden wollen, müssen Sie diese explizit durch Angabe von `-t mangle` auswählen.

Target-Erweiterungen für die mangle-Tabelle

Tabelle 3.18 zeigt die Target-Erweiterungen für die `mangle`-Tabelle:

-t mangle	Beschreibung
`-j MARK -set-mark <Wert>`	Legt einen Wert für das Netfilter-interne Mark-Feld fest
`-j TOS -set-tos <Wert>`	Setzt den TOS-Wert im IP-Header

Tabelle 3.18: Target-Erweiterungen für die `mangle`-Tabelle

In der `mangle`-Tabelle stehen Ihnen zwei Target-Erweiterungen zur Verfügung, MARK und TOS. Durch MARK manipulieren Sie einen vorzeichenlangen `long`-Wert für das Paket, der intern durch `iptables` gespeichert wird.

Ein Beispiel:

```
iptables -t mangle -A PREROUTING --in-interface eth0 -p tcp \
        -s <irgendeine Absenderadresse> --sport 1024:65535 \
        -d <irgendeine Empfängeradresse> --dport 23 \
        -j MARK --set-mark 0x00010070
```

Über TOS setzen Sie die TOS-Bits im IP-Header. Auch dafür ein Beispiel:

```
iptables -t mangle -A OUTPUT ... -j TOS --set-tos <TOS-Wert>
```

Die erlaubten TOS-Werte entsprechen denen aus der gleichnamigen Match-Erweiterung für die `filter`-Tabelle.

3.4 Zusammenfassung

In diesem Kapitel haben Sie die überwiegende Mehrzahl der `iptables`-Features kennen gelernt – zumindest waren die meistgenutzten Optionen dabei. Ich habe mich bemüht, Sie auf die Unterschiede zwischen Netfilter und IPFW hinzuweisen. Das dient als eine Vorbereitung für die Unterschiede in der Implementierung,

auf die Sie in den folgenden Kapiteln stoßen werden. Gesprochen haben wir auch über die modulare Aufteilung der Implementierung in drei getrennte Tabellen – `filter`, `mangle` und `nat`. Dabei war jeweils die Rede von Modulen für Target-Erweiterungen und Modulen für Match-Erweiterungen.

In Kapitel 4 besprechen wir ein einfaches Beispiel für eine Firewall. Ich stelle dabei Regeln gegen Adressfälschung, Denial-of-Service-Angriffe und andere fundamendale Regeln vor. Zweck des Kapitels ist weniger die Vorstellung einer allgemeingültigen Firewall, die Sie sich einfach herauskopieren können, als vielmehr die praktische Demonstration der in diesem Kapitel gezeigten Syntax.

Die später folgenden Kapitel werden immer spezieller. Separat gehen wir ebenso ein auf benutzerdefinierte Chains, Optimierung der Firewall, LANs und NAT wie auch auf Architekturmöglichkeiten für größere lokale Netze.