Linux-Paketfilter

Michael Dienert

13. Januar 2022

Inhaltsverzeichnis

1 Paketfilterung mit dem Linux-Kernel

1.1 Tabellen und Filterketten

- Die Firewall besteht aus **Tabellen**.
- Eine Tabelle enthält mehrere Filter-**Ketten**.
- Eine Kette besteht aus **Regeln**, die Regeln sind also die Kettenglieder. Die Regeln einer Kette werden nacheinander durchlaufen. Trifft eine Regel zu, wird die Kette verlassen.
- Eine Regel endet mit der Angabe eines Sprung-**Ziels**. Das Ziel bestimmt, was mit dem Paket gemacht wird: DROP, ACCEPT, DNAT, ... oder ob man zu einer anderen Kette springt.
- Die Sprungziele am *Ende* der fest definierten, eingebauten Ketten nennt man *Policies*. Diese können nur die Werte **ACCEPT** oder **DROP** haben.

1.1.1 Die Tabellen FILTER, NAT und MANGLE

Es gibt standardmässig die fünf Tabellen:

filter ist die Standardtabelle. Ist keine Tabelle angegeben (Option `-t`), wird *filter* verwendet.

nat Die Tabelle für NAT wird mit `-t nat` aufgerufen.

mangle Mit Hilfe der Tabelle *mangle* können die Header von Paketen manipuliert werden.

raw Wird hier ebenfalls nicht betrachtet.

security Auch auf diese Tabelle wird hier nicht eingegangen.

1.1.2 Sprungziele

Sprungziele (targets) bestimmen, wie mit dem Paket verfahren wird. Die Ziele werden mit **-j** oder **–jump** aufgerufen. Es gibt (vordefiniert, Liste nicht vollst.) :

- DROP
- ACCEPT
- MASQUERADE: gibt es nur in der nat-Tabelle
- DNAT: für Port-Forwarding; gibt es nur in der nat-Tabelle

1.1.3 Vordefinierte Filterketten

Es gibt 5 vordefinierte *Ketten* (in Blocksatz):

PREROUTING erste Kette, da muss der gesamte Verkehr durch (gut für z.B. *port forwarding* = *Destination NAT*)

INPUT Kette für Pakete, die *für* den Host selbst bestimmt sind

FORWARD Kette für Pakete, die geroutet werden

OUTPUT Kette für Pakete, die *vom* Host selbst stammen

POSTROUTING letzte Kette, da muss der gesamte Verkehr durch (für *Source NAT*)

1.2 Zusammenfassung

filter
FORWARD
INPUT
OUTPUT

Tabelle 1: Tabelle filter

nat
PREROUTING
OUTPUT
POSTROUTING

Tabelle 2: Tabelle nat

mangle
PREROUTING
POSTROUTING
OUTPUT
INPUT
FORWARD

Tabelle 3: Tabelle mangle

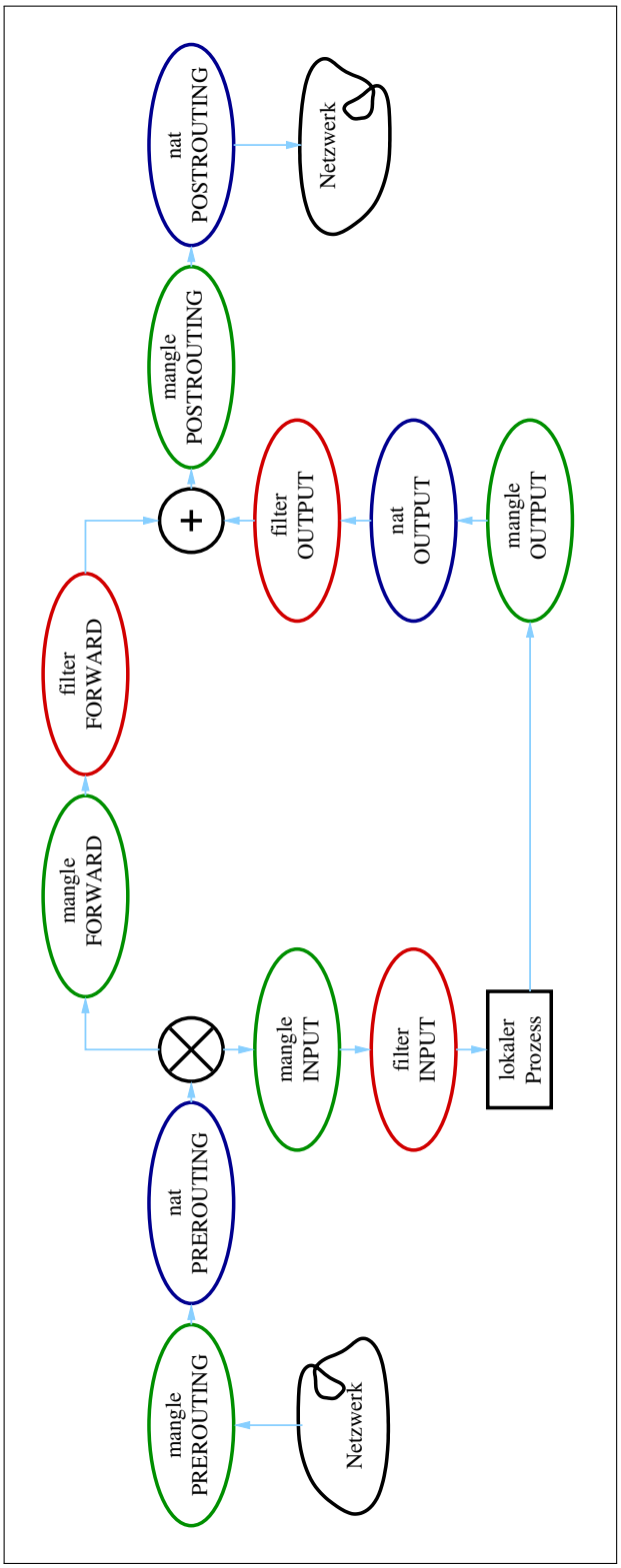

Abbildung 1: Weg der Pakete durch iptables

2 Stateful Inspection

Bei einer Kommunikation zwischen zwei Rechnern über TCP sind Quell- und Ziel IP-Adressen und Quell- und Ziel Ports beteiligt. Möchte man diese Kommunikation mit einer Firewall filtern, gibt es ein Problem wegen der Vielzahl an möglichen IP-Adressen und der dynamischen Ports.

Im Beispiel soll eine Firewall für einen Host so konfiguriert werden, dass keinerlei Zugriffe von Aussen nach Innen erlaubt sind. Wir setzen alle Policies auf `DROP`.

Gleichzeitig sollen von diesem Rechner aus aber http/https-Requests möglich sein.

Wir erlauben also Pakte mit dem Zielport 80 oder 443 von innen nach aussen (http-Request).

Nun müssen wir irgendwie die http-Response-Pakete von aussen nach innen durchlassen. Wir können aber nicht nach der Quell-IP filtern, es gibt ja beliebig viele Webserver.

Bliebe evtl. der Quellport, der ist beim http-Response Port 80 oder 443: wir müssten Pakete mit dem Quellport 80 und 443 erlauben.

Diese Regel stellt nun ein Problem dar: jeder Angreifer kann leicht seine Pakte mit dem *Quell-Port* 80 oder 443 versehen und sie würden somit durchgelassen.

Viel besser wäre es, die zweite Regel für den *Zielport* aufzustellen. Da der Quell-Port (der für die Antwort des Servers zum Zielport wird) beim Anfragen der Webseite aber *dynamisch* vom jeweiligen Host vergeben wird, ist die Portnummer vorab nicht bekannt.

Das Problem wird nun so gelöst, dass die Firewall *selbstständig temporäre* Regeln einführt, die dann für die dynamischen Ports und IP-Adressen einer Verbindung gelten.

Hierzu wird der Zustand der Verbindung von der FW mitprotokolliert (tracking). Bei *netfilter* existiert hierfür das Modul `conntrack`.

Die Handbuchseite von `conntrack` erreicht man über

```
man iptables-extensions
```

3 Ein paar einfache, praktische Übungen

Zuerst ein Rettungs-Skript, mit dem man die Firwall wieder deaktivieren kann:

```
#!/bin/sh

echo "Testskript fuer iptables"

# iptables wird im folgenden immer ohne option -t aufgerufen
# daher beziehen sich alle kommandos auf die tablelle filter (default)

echo "Alle vordefinierten Ketten (INPUT, OUTPUT, FORWARD) in der Tabelle filter loeschen"

iptables -F

echo "Alle Benutzereigenen Ketten loeschen"

iptables -X

echo "Setze alle Policies (Sprungziele der Ketten INPUT, OUTPUT, FORWARD) auf ACCEPT"

iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
```

Ein einfaches Skript, das den Zugriff auf einen Webserver erlaubt. Die Policy der Ketten ist hier auf `ACCEPT` gesetzt.

```
echo "Alle Regeln in den vordefinierten Ketten (INPUT, OUTPUT, FORWARD)"
echo "in der Tabelle filter loeschen"

iptables -F # F = flush=klospuelung

echo "Alle Benutzereigenen Ketten loeschen"
iptables -X

echo "Setze alle Policies (Sprungziele der Ketten INPUT, OUTPUT, FORWARD)"

iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT

#einfach, ohne conntrack
#iptables -A INPUT -p tcp --dport 80 -j ACCEPT
#iptables -A INPUT -p tcp --dport 443 -j ACCEPT
#iptables -A INPUT -j DROP

#etwas genauer, mit hilfe von conntrack:
iptables -A INPUT -p tcp -m conntrack --ctstate NEW --dport 80 -j ACCEPT
iptables -A INPUT -p tcp -m conntrack --ctstate NEW --dport 443 -j ACCEPT
iptables -A INPUT -p tcp -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -j DROP
```

Im nächsten Beispiel ist die Policy auf DROP gesetzt und man muss Pakete einer bestehenden Verbindung in beide Richtungen extra erlauben.

```
echo "Alle Regeln in den vordefinierten Ketten (INPUT, OUTPUT, FORWARD)"
echo "in der Tabelle filter loeschen"

iptables -F # F = flush=klospuelung

echo "Alle Benutzereigenen Ketten loeschen"
iptables -X

echo "Setze alle Policies (Sprungziele der Ketten INPUT, OUTPUT, FORWARD)"

iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP


iptables -A INPUT -p tcp -m conntrack --ctstate NEW --dport 80 -j ACCEPT
iptables -A INPUT -p tcp -m conntrack --ctstate NEW --dport 443 -j ACCEPT

#die folgenden beiden zeilen sind nur notwendig, wenn man von diesem
#host aus http-requests wegschicken moechte
iptables -A OUTPUT -p tcp -m conntrack --ctstate NEW --dport 80 -j ACCEPT
iptables -A OUTPUT -p tcp -m conntrack --ctstate NEW --dport 443 -j ACCEPT

iptables -A INPUT -p tcp -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -p tcp -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

4 De-Militarized Zone

Eine Anwendung des Netfilter-Moduls sind der Aufbau von De-Militarisierten Zonen.

Der Begriff *De-Militarisierte Zone* bezeichnet einen befriedeten Bereich, der zwei feindliche Lager voneinander trennt.

In der IT-Technik wird dieses Konzept auf zwei Arten realisiert. Es gibt eine zweistufige Anordnung, die zwei Router verwendet (Abb. 2) und eine etwas einfachere, einstufige Variante (Abb. 3), bei der ein Router mit drei Interfaces verwendet wird.

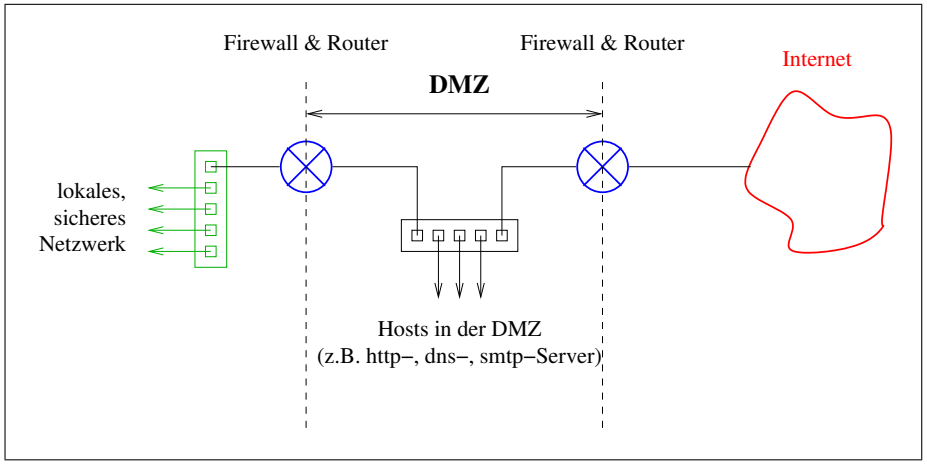

Abbildung 2: Zweistufige DMZ

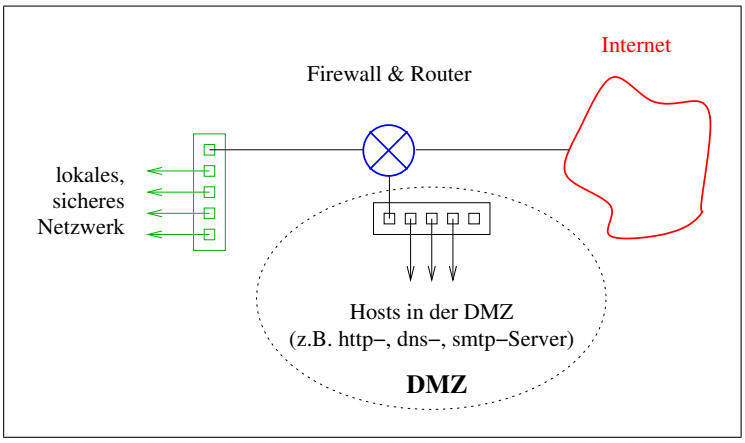

Abbildung 3: Einstufige DMZ

In den Abbildungen sind die Firewalls (Paketfilter) und die Router als ein Gerät dargestellt. Werden höchste Sicherheitsstandards verlangt, sollte man dafür zwei unabhängige Geräte einsetzen.