Netzfiltertabellen mit *nftables* verwalten

Filtriert

Udo Seidel

Linux-Administratoren steht der Umstieg auf *nftables* bevor.

Linux besaß bereits mehrere Paketfilter-Implementierungen: von *ipfwadm* zu *ipchains* und *iptables* zu *nftables*. Letzteres ist seit Version 3.13 Bestandteil des Linux-Kernels und löst das bis dahin bevorzugte *iptables* ab. Es enthält die meisten *iptables*-Funktionen sowie einige zusätzliche Eigenschaften und ersetzt die Kommandos des Vorgängers durch das Werkzeug *nft*. Dank der Kompatibilitätsschicht von *nftables* lassen sich die alten Kommandos und Einstellungen zunächst weiterverwenden.

Das neue Paketfiltertool arbeitet mit Tabellen, Ketten (Chains) und Regeln, die zugleich Hierarchiestufen bilden. Tabellen sind ein Container für Ketten, bei denen man reguläre und Basisketten unterscheidet. Diese sind wiederum Behälter für Regeln, die das Filtern übernehmen. Letztere benutzen die von *ipchains* bekannten Einstiegspunkte in den Netzwerkstack: Prerouting, Input, Forward, Output, Postrouting und das neue Ingress. Zuerst erzeugt man eine Tabelle, legt darin eine Kette an und definiert schließlich Regeln in dieser Kette. Anders als bei *ipchains* gibt es keine vordefinierten Konstrukte. Regeln bei *nftables* sind mit einer Nummer (Handle) und einer ketteninternen Position versehen, die der eindeutigen Identifizierung und exakten Positionierung im Gesamtkonstrukt dienen.

Mit Regel-Matches identifiziert man Pakete im Netzverkehr für besondere Behandlung und gelangt an Informationen wie TCP-Sequenznummern oder UDP-Portnummern. Derzeit gibt es mehr als 20 Matches-Kategorien wie TCP oder IPv6. Regel-Statements hingegen definieren die Art der Behandlung, die entweder in Urteilen (Verdicts) oder in Aktionen besteht. Urteile akzeptieren oder verwerfen ein Paket oder leiten es um. Aktionen sind etwa Protokollierung, Kapazitätsbeschränkung, Adressübersetzung (NAT) oder schlichtes Paketezählen.

Eine wesentliche Neuerung bei *nftables* ist das Konzept der Familien (Families): *IP*, *IP6*, *ARP*, *BRIDGE* sowie *INET* und *NETDEV*. Alle zueinandergehörigen Tabellen müssen sich in derselben Familie befinden, was vor allem für das Generieren von Tabellen und Ketten relevant ist. Die Hauptarbeit steckt in der letzten Hierarchiestufe, den Regeln. Egal ob Tabelle, Kette oder Regel – die grundlegenden Kommandos sehen immer gleich aus:

```
nft <add|list|flush|delete> <table|chain|rule> ↵
                                <WeitereOptionen>
```

Für einen typischen Regelsatz erzeugt man zuerst mit *nft add table inet t1* eine Tabelle der Familie *INET* mit dem Namen *t1*. Für das Filtern von Paketen ist eine sogenannte Basiskette nötig. Beim Anlegen muss man den Einstiegspunkt (Hook) in den Netzwerkstack sowie den Typ als Filter angeben und eine Basispriorität definieren:

```
nft create chain inet t1 k1 { type filter ↵
                        hook input priority 0\; }
```

In der Kette *k1* in Tabelle *t1* lassen sich anschließend Regeln anlegen. Die erste behandelt eingehenden Verkehr über das Loopback-Gerät als unverdächtig:

```
nft add rule inet t1 k1 iif lo accept
```

Die Option *inet* ist der erste Einstiegspunkt in *nftables*. Gleichnamige Tabellen und Ketten lassen sich in verschiedenen Familien anlegen. Deswegen sind *t1* und *k1* nicht eindeutig – erst durch *inet* weiß *nftables*, wohin die Regel gehört.

```
nft add rule inet t1 k1 ct state ↵
                    established,related accept
```

stellt sicher, dass die Kommunikation nach „draußen" funktioniert. Eingehende Verbindungen sind nur auf dem TCP-Port 22 erlaubt:

```
nft add rule inet t1 k1 tcp dport 22 ct ↵
                              state new accept
nft add rule inet t1 k1 drop
```

Das Ergebnis ist in Listing 1 dargestellt. Die Option *–a* zeigt die interne Nummerierung der Regeln, die fürs weitere Verarbeiten nützlich ist. Zusätzliche Regeln für eingehende Verbindungen muss man an der richtigen Stelle einfügen – im vorliegenden Fall wäre das bei Regel/Handle 5:

```
nft add rule inet t1 k1 position 4 tcp dport ↵
                        80 ct state new accept
```

Das erlaubt eingehende Verbindungen auf Port 80 und schiebt diese Regel nach Handle 4 ein. Dies lässt sich durch eine Regel für beide Ports vereinfachen, die die anderen ersetzt (Listing 2).

Fazit

Die ersten Schritte mit *nftables* sind nicht unbedingt leicht, aber das Verwalten der Netzwerktabellen mit nur einem Werkzeug ist eine Verbesserung. Neben der Projektdokumentation (siehe ix.de/ix1801125) sollte man sich *iptables-translate* ansehen, das beim Weitergeben des *iptables*-Erbe hilft. (tiw)

Listing 1: Beispielregelsatz für einen normalen Arbeitsplatz-PC

```
nft list ruleset -a
table inet t1 {
        chain k1 {
                type filter hook input priority 0; policy accept;
                iif "lo" accept # handle 2
                ct state established,related accept # handle 3
                tcp dport ssh ct state new accept # handle 4
                drop # handle 5
        }
}
```

Listing 2: Eine Regel an Position 4 einfügen

```
nft add rule inet t1 k1 position 4 tcp dport { 22, 80 } ct state new accept
 && nft delete rule inet t1 k1 handle 4
 && nft delete rule inet t1 k1 handle 6
```

Dr. Udo Seidel

ist studierter Mathe-Physik-Lehrer.

Alle Links: www.ix.de/ix1801125

© Copyright by Heise Medien
Persönliches PDF für alle Leser der iX aus 30625 Hannover