Linux Kommandoreferenz

Shell-Befehle von A bis Z

Um das Backup später wieder einzuspielen, führen Sie diese Kommandos aus:

```
user$  mysqladmin  -u user -p create dbname
user$  mysql  -u user -p --default-character-set=utf8 dbname < backup.sql
```

namei [optionen] datei

`namei` zeigt die Zugriffsrechte aller Verzeichnisse, die den Pfad der angegebenen Datei bilden:

```
user$  namei -l ~/.ssh
f: /home/kofler/.ssh
drwxr-xr-x root   root   /
drwxr-xr-x root   root   home
drwxr-xr-x kofler kofler kofler
drwx------ kofler kofler .ssh
```

nc [optionen] [hostname/ip-adresse] [port]

Das Kommando `nc` ist unter der Bezeichnung *Netcat* bekannter. Es leitet TCP/UDP-Ports auf die Standardeingabe/-ausgabe um und bietet eine Menge weiterer Funktionen. Sie können mit dem Kommando ähnlich wie mit `telnet` interaktiv Netzwerkprotokolle wie HTTP oder SMTP ausprobieren. Mit `nc` können Sie aber auch Dateien über einen beliebigen Netzwerk-Port kopieren, einen Chat realisieren oder eine simple Backdoor einrichten, die über einen Port Kommandos entgegennimmt und ausführt. Da verwundert es nicht, dass `nc` unter Hackern und Penetration-Testern außerordentlich beliebt ist.

Bei einigen Linux-Distributionen ist das Kommando `nc` im gleichnamigen Paket enthalten, bei anderen Distributionen müssen Sie `netcat` installieren. Beachten Sie, dass es unterschiedliche Implementierungen von Netcat gibt. So kommt unter Debian und Ubuntu `netcat-traditional` zum Einsatz, während RHEL eine Variante der `nmap`-Entwickler anbieten (*https://nmap.org/ncat*). In der Praxis ergeben sich daraus keine großen Unterschiede, allerdings sind möglicherweise einzelne Optionen je nach Version anders (oder gar nicht) implementiert.

- `-4` oder `-6`

  verwendet ausschließlich IPv4 oder IPv6.

- `-l`

  wartet am angegebenen Port auf einen Verbindungsaufbau (*listen*).

- `-p portnr`

  legt den lokalen Port (Source-Port) fest. Der üblicherweise am Ende des `nc`-Kommandos angegebene Port ist hingegen der Ziel-Port (Destination-Port).

- `-x proxyadr:portnr`

  verwendet die angegebene Proxy-Adresse und den dazugehörenden Port.

Eine Menge weiterer Optionen beschreibt `man nc`. Eine mögliche Alternative zu `nc` ist das Kommando `socat`, das in diesem Buch aber nicht behandelt wird. Es unterstützt auch das Protokoll SCTP, kann über Proxyserver arbeiten, auch serielle Schnittstellen bedienen und die Daten für die Übertragung verschlüsseln:

http://www.dest-unreach.org/socat

Beispiele

Um eine Datei über einen beliebigen Port (hier 1234) von Host 1 nach Host 2 zu kopieren, starten Sie zuerst auf Host 2 (rechte Spalte) den Empfänger und initiieren die Übertragung der Datei dann auf Host 1 (linke Spalte).

```
                                  host2$  nc -l 1234 > datei
host1$  nc host2 1234 < datei
```

Ebenso einfach führen Sie einen Chat durch. Sie müssen sich mit Ihrem Gesprächspartner lediglich auf einen Port einigen. Der Chat wird auf einem Rechner mit `nc -l` initiiert (linke Spalte). Damit überwacht `nc` den angegebenen Port 1234 und wartet auf einen Verbindungsaufbau.

Auf dem zweiten Rechner wird `nc` ohne Optionen gestartet, um die Verbindung zum ersten Host herzustellen. Eine sichtbare Bestätigung des Verbindungsaufbaus gibt es zwar nicht, aber sobald nun einer der beiden Gesprächspartner Text eingibt (Standardeingabe) und mit ↵ bestätigt, erscheint der Text im Terminal des anderen Gesprächspartners (Standardausgabe).

```
host1$  nc -l 1234
                                  host2$  nc host1 1234
 wie geht's?
                                  wie geht's?
                                  gut
gut
<Strg>+<C>
```

Das Gefahrenpotential von Netcat zeigt sich im dritten Beispiel: Hier wird `nc` auf Host 1 so eingerichtet, dass es alle auf Port 1234 empfangenen Eingaben an die Shell `bash` weitergibt. Deren Ausgaben werden wieder zurück an den Sender übertragen. Von einem zweiten Host können nun Shell-Kommandos auf Host 1 ausgeführt werden. `ls` zeigt also Dateien, die sich auf Host 1 befinden!

```
host1$  nc -l 1234 -e /bin/bash
```

```
host2$  nc host1 1234
ls
datei1
datei2
datei3
```

Die Option `-e` zur Ausführung eines Kommandos steht allerdings nicht bei allen Netcat-Versionen zur Verfügung. Sie fehlt insbesondere bei der unter Debian und Ubuntu üblichen `netcat-traditional`-Implementierung. Abhilfe: Installieren Sie das Paket `nmap`, und führen Sie das dort enthaltene Kommando `ncat` aus.

ncdu [optionen] [verzeichnis]

`ncdu` ist eine interaktive Variante des Kommandos `du`. Normalerweise wird es einfach ohne Optionen und Parameter aufgerufen. Es zeigt dann die Unterverzeichnisse im aktuellen Verzeichnis an, die am meisten Platz beanspruchen.

Mit den Cursortasten und ↵ können Sie nun durch die Unterverzeichnisse navigieren. Diverse weitere Funktionen sind über Tastenkürzel zugänglich. Einen Überblick über die wichtigsten Kürzel gibt ?. Vorsicht, D löscht nach einer Rückfrage die aktuelle Datei bzw. das aktuelle Verzeichnis.

- `-r`

  führt das Kommando im Read-only-Modus aus. Ein versehentliches Löschen von Dateien oder Verzeichnissen ist damit ausgeschlossen.

- `-x`

  bleibt im aktuellen Dateisystem, berücksichtigt also keine `mount`-Verzeichnisse.

Beispiel

Der Platzbedarf der Verzeichnisse im Heimatverzeichnis wird auch durch Balken symbolisiert.

```
user$  ncdu
--- /home/kofler ---
    1,5 GiB [##########] /Nextcloud
    1,0 GiB [######    ] /Downloads
  528,5 MiB [###       ] /Bilder
  209,4 MiB [#         ] /Dokumente
  ...
```

needs-restarting [optionen]

Das unter Fedora, Red Hat und verwandten Distributionen verfügbare Kommando needs-restarting gibt an, ob der Rechner oder einzelne Programme aufgrund eines Updates des Kernels, einer grundlegenden Bibliothek oder einer Firmware-Datei neu gestartet werden müssen.

Bei Distributionen auf der Basis von Debian oder Ubuntu steht das Kommando nicht zur Verfügung. Dort weist die Datei /var/run/reboot-required auf einen erforderlichen Neustart hin.

- -r

  zeigt an, ob und warum ein Rechnerneustart erforderlich ist.

- -u

  listet Prozesse des aktuellen Benutzers auf, die neu gestartet werden müssen. Diese Option gilt standardmäßig.

Beispiel

Die folgenden Kommandos zeigen den Zustand eines Rechners, der neu gestartet werden sollte:

```
root#  needs-restarting -r
Core libraries or services have been updated since boot-up:
  * glibc
Reboot is required to fully utilize these updates.
More information: https://access.redhat.com/solutions/27943

root#  needs-restarting | sort -n
782 : /usr/lib/polkit-1/polkitd --no-debug
783 : /usr/libexec/power-profiles-daemon
784 : /usr/bin/qemu-ga --method=virtio-serial ...
787 : /usr/libexec/accounts-daemon
...
```

netplan [kommando]

Netplan (*https://netplan.io*) ist ein von Ubuntu entwickeltes Framework, das andere Netzwerk-Backends wie den NetworkManager sowie networkd (eine systemd-Komponente, siehe networkctl) konfiguriert und einbindet. Es kommt in Ubuntu zum Einsatz und wertet die Konfigurationsdateien aus, die sich in /etc/netplan befinden (siehe netplan.yaml).

- `apply`

  führt `netplan generate` aus und aktiviert dann die geänderte Konfiguration.

- `generate`

  liest die Netplan-Konfiguration aus den YAML-Datein in `/etc/netplan`, `/lib/netplan` und `/var/netplan` und generiert die entsprechenden Konfigurationsdateien für die Netzwerk-Backends. Die neuen Dateien landen in `/run/NetworkManager` bzw. in `/run/systemd/network`.

  Die neue Konfiguration wird nicht aktiviert! Die geänderten Einstellungen werden erst wirksam, wenn das jeweilige Backend aufgefordert wird, die Dateien auszuwerten (bzw. beim nächsten Neustart).

- `ip leases` *`schnittstelle`*

  gibt die aktuellen DHCP-Lease-Daten für die betreffende Schnittstelle aus. Das funktioniert nur, wenn das Backend networkd und eine DHCP-Konfiguration im Einsatz sind. Andernfalls liefert das Kommando in die Irre führende Fehlermeldungen.

```
netstat [optionen]
```

`netstat` liefert Informationen über die Netzwerkaktivität auf dem lokalen Rechner. Wird das Kommando ohne Optionen aufgerufen, liefert es eine Liste aller offenen Internetverbindungen und Sockets.

- `-a`

  berücksichtigt auch nichtaktive Sockets, also Dienste, die auf einem Netzwerk-Port auf eine Verbindung warten (Zustand `LISTEN` bzw. bei deutscher Lokalisierung `HÖREN`).

- `-e`

  gibt bei Internetverbindungen zusätzlich den Benutzer an.

- `-n`

  liefert numerische Adressen und Port-Nummern anstelle von Host- und Netzwerkdienstnamen.

- `-p`

  gibt zusätzlich die Prozessnummer (PID) und den Prozessnamen des Programms an, das für die Verbindung verantwortlich ist. `netstat` benötigt `root`-Rechte, damit es Informationen über nichteigene Prozesse liefern kann.

- `-t / -u / -w / -X`

  schränkt die Ausgabe auf Verbindungen ein, die das Protokoll TCP, UDP, Raw oder Unix nutzen.

Beispiel

Das folgende Kommando listet alle aktiven Verbindungen (`established`) bzw. überwachten Ports (`LISTEN`) auf. Die Ausgabe wurde aus Platzgründen stark gekürzt.

```
root#  netstat -atupe
Active Internet connections (servers and established)
Proto Local Address           Foreign Addr   State    User     PID/Prog name
tcp   *:nfs                   *:*            LISTEN   root     -
tcp   *:ldap                  *:*            LISTEN   root     5842/slapd
tcp   localhost:mysql         *:*            LISTEN   mysql    5785/mysqld
tcp6  [::]:ssh                [::]:*         LISTEN   root     5559/sshd
udp   *:nfs                   *:*                     root     -
```

`networkctl` [kommando]

`networkctl` gehört zu den Administrationskommandos aus der systemd-Familie. Sofern networkd als Netzwerk-Backend eingesetzt wird, hilft `networkctl` bei der Analyse des Netzwerkstatus.

Standardmäßig ist networkd zwar bei vielen Distributionen installiert, aber nur bei wenigen auch aktiv. Zu den Ausnahmen zählt Ubuntu Server, wo networkd als Backend für das Ubuntu-eigene Netplan-System dient (siehe `netplan`). networkd wertet Konfigurationsdateien aus den Verzeichnissen `/etc/systemd/network`, `/lib/systemd/network` sowie in `/run/systemd/network` aus (siehe `/etc/systemd/networkd/networkd.network`).

- `list`

  listet alle Netzwerkschnittstellen auf. Die `SETUP`-Spalte des Ergebnisses zeigt, ob die Schnittstelle durch networkd gesteuert wird (`configured`) oder nicht (`unmanaged`).

- `lldp`

  zeigt andere Geräte im Netzwerk, die über das *Link Layer Discovery Protocol* (LLDP) entdeckt wurden.

- `status` [*schnittstelle*]

  liefert detaillierte Informationen über den Status einer Netzwerkschnittstelle. Wenn keine Schnittstelle angegeben wird, versucht das Kommando, den gesamten Netzwerkstatus zusammenzufassen.

Beispiele

Die folgenden Ausgaben von `networkctl` sind auf einem Ubuntu-Server in einer virtuellen Maschine entstanden:

```
root#  networkctl list
IDX  LINK      TYPE       OPERATIONAL   SETUP
  1  lo        loopback   carrier       unmanaged
  2  ens3      ether      routable      configured
root#  networkctl status ens3
       Link File: /lib/systemd/network/99-default.link
    Network File: /run/systemd/network/10-netplan-ens3.network
            Type: ether
           State: routable (configured)
             ...
      HW Address: 52:54:00:0a:8d:fc
         Address: 138.201.20.182
                  fe80::5054:ff:fe0a:8dfc
         Gateway: 138.201.20.176 (Fujitsu Technology Solutions GmbH)
             DNS: 213.133.100.100
                  213.133.98.98
  Search Domains: ubuntu-buch.info
```

Das zweite Listing zeigt die Statuszusammenfassung einer anderen Server-Installation mit IPv6-Konfiguration:

```
root#  networkctl status
         State: routable
  Online state: online
       Address: 168.119.33.110 on eth0
                172.17.0.1 on docker0
                2a01:4f8:242:1f88::4 on eth0
                fe80::5054:ff:fe4a:7321 on eth0
       Gateway: 168.119.33.119 on eth0
                2a01:4f8:242:1f88::2 on eth0
           DNS: 213.133.100.100
                213.133.99.99
                2a01:4f8:0:1::add:1010
                2a01:4f8:0:1::add:9999
```

newaliases

Die Datei `/etc/aliases` enthält eine Alias-Liste für den E-Mail-Server, die z. B. dafür sorgt, dass alle E-Mails, die an `postmaster` adressiert sind, an `root` weitergeleitet werden. Damit Änderungen an dieser Datei vom Mail-Server berücksichtigt werden, müssen Sie `newaliases` ausführen.

newgrp [gruppenname]

Das Kommando newgrp bestimmt die gerade aktive Gruppe eines Benutzers, der mehreren Gruppen angehört. Die aktive Gruppe bestimmt, welcher Gruppe neu erzeugte Dateien angehören. Die zur Auswahl stehenden Gruppen können mit groups ermittelt werden. Wenn kein Gruppenname angegeben wird, wird die primäre Gruppe verwendet. Diese Gruppe gilt auch nach einem Login automatisch als aktive Gruppe.

Beispiel

Im folgenden Beispiel macht das newgroup-Kommando docuteam zur aktiven Gruppe des Benutzers kofler. Die neu erzeugte Datei newfile wird daher der Gruppe docuteam zugeordnet und kann von anderen Mitgliedern des Dokumentationsteams bearbeitet werden.

```
user$  groups
kofler docuteam wheel
user$  newgroups docuteam
user$  touch newfile
```

newusers datei

newusers liest eine Textdatei und erzeugt für jede Zeile einen neuen Benutzer. Die Textdatei weist prinzipiell dasselbe Format auf wie /etc/passwd. Allerdings müssen die Passwörter unverschlüsselt in der zweiten Spalte angegeben werden. Die meisten weiteren Parameter (UID, GID etc.) sind optional. newusers erzeugt für jeden angegebenen Benutzer einen neuen Account, wobei bei Bedarf auch die dazugehörenden Gruppen angelegt werden. Für fehlende Parameter wählt newusers geeignete Defaultwerte, wobei die Einstellungen in /etc/login.defs berücksichtigt werden.

Beachten Sie, dass newusers zwar neue Heimatverzeichnisse erzeugt, wenn deren Ort in der sechsten Spalte angegeben ist. Das Kommando kümmert sich aber nicht darum, den Inhalt von /etc/skel dorthin zu kopieren.

Beispiel

Die folgenden Zeilen zeigen die minimalistische Textdatei users.txt, die den Anforderungen von newusers entspricht. Da users.txt Klartextpasswörter enthält, müssen Sie darauf achten, dass niemand außer root die Datei lesen kann, bzw. die Datei nach der Ausführung von newusers wieder löschen.

```
huber:geheim1:::Hermann Huber:/home/huber:/bin/bash
moser:geheim2:::Gabi Moser:/home/moser:/bin/bash
schmidt:geheim3:::Peter Schmidt:/home/schmidt:/bin/bash
```

`newusers` erzeugt nun die drei neuen Benutzer `huber`, `moser` und `schmidt` sowie gleichnamige primäre Gruppen. `newusers` entscheidet sich selbst für geeignete UIDs und GIDs.

```
root#  newusers users.txt
```

`nft` kommando [optionen]

Die meisten aktuellen Linux-Distributionen verwenden mittlerweile das neue Firewall-System *nftables*, das seinen Vorgänger *Netfilter* ablöst. Zur Konfiguration der Firewall kommt jedoch oft noch das alte Kommando `iptables` zum Einsatz, das dank einer Kompatibilitätsschicht weiterhin funktioniert.

Neue bzw. nftables-spezifische Funktionen können aber nur mit dem Kommando `nft` gesteuert werden. Die folgende Beschreibung gibt einen groben Überblick über die wichtigsten Unterkommandos und Optionen dieses Kommandos. Viel mehr Details zur Funktionsweise von nftables sowie zur Syntax von `nft` liefern `man nft` sowie das folgende Wiki:

https://wiki.nftables.org

Bevor Sie `nft` aufrufen, sollten Sie sich vergewissern, dass auf Ihrer Distribution nicht schon eine Firewall läuft. Unter Fedora, RHEL und SUSE ist z. B. *Firewalld* aktiv (siehe `firewall-cmd`). Selbst definierte Firewall-Regeln führen dann oft zu Konflikten mit der Firewall Ihrer Distribution.

nft-Nomenklatur

Für `nft` gilt eine eigene Nomenklatur: Firewall-Regeln (*rules*) sind immer Teil einer Kette (*chain*). Tabellen (*tables*) bestehen wiederum aus mehreren Ketten. Anders als bei `iptables` gibt es weder vordefinierte Tabellen noch vordefinierte Ketten. Sie können somit beliebig viele eigene Tabellen aus ebenfalls selbst definierten Ketten zusammenstellen.

Tabellen, Ketten und Regeln gelten für verschiedene Typen von Netzwerkpaketen, die den folgenden Familien (*families*) zugeordnet sind:

- `ip` (nur IPv4)
- `ip6` (nur IPv6)
- `inet` (für Regeln, die gleichermaßen für IPv4 und IPv6 gelten)
- `arp` (Level-2-Regeln, werden vor Level-3-Regeln ausgewertet)
- `bridge` (Switching-Regeln)
- `netdev` (Low-Level-Regeln, werden vor allen anderen Regeln ausgewertet und ermöglichen z. B. eine besonders effiziente Abwehr von DDOS-Angriffen)

nft-Optionen

- `-a`

  baut in die Ausgaben von `nft list` sogenannte *handles* ein. Das sind Nummern, die jedes Objekt innerhalb einer Gruppe (also z. B. eine Regel innerhalb einer Kette) eindeutig bezeichnen. Handles ermöglichen es, gezielt einzelne Objekte zu löschen bzw. zu ersetzen oder die Position anzugeben, wo neue Objekte eingefügt werden sollen.

- `-f` *fname*

  liest die auszuführenden Kommandos aus der angegebenen Datei. Dabei sind zwei Syntaxvarianten zulässig: Die Datei kann entweder zeilenweise gewöhnliche `nft`-Kommandos enthalten oder eine hierarchisch strukturierte Abfolge von Tabellen, Ketten und Regeln. Der Aufbau der Datei für die zweite Variante entspricht der Ausgabe von `nft list ruleset`.

- `-S`

  verwendet in Ausgaben die in `/etc/services` angegebenen Namen von Netzwerkdiensten anstelle von Port-Nummern (also z. B. `ssh` anstelle von 22).

- `-v`

  zeigt die Versionsnummer an.

nft-Kommandos

- `add chain` *family tname cname*

  erzeugt eine neue Kette mit dem Namen *cname* in der angegebenen Tabelle.

- `add rule` *family tname cname matches statements*

  fügt der angegebenen Kette *cname* eine neue Regel hinzu. Dabei gibt *matches* an, für welche Pakete die Regel gilt. *statements* beschreibt, was mit den betreffenden Paketen passieren soll.

  Anstelle von `add` sind auch die Schlüsselwörter `insert` und `replace` erlaubt, um eine Regel an einer bestimmten Stelle in die Regelliste einzubauen bzw. um eine vorhandene Regel durch eine neue zu ersetzen.

- `add table` *family tname*

  erzeugt eine neue Tabelle für die angegebene Familie.

- `delete/flush chain` *family tname cname*

  entfernt die angegebene Kette aus der Tabelle. `flush` löscht alle Regeln der Kette, nicht aber die Kette selbst.

- `delete rule` *`family tname cname`* `handle` *`handle`*

  löscht die durch *`handle`* spezifizierte Regel aus der Kette *`cname`*.

- `delete/flush table` *`family tname`*

  löscht die angegebene Tabelle. Bei `flush` werden zwar alle Ketten und Regeln der Tabelle gelöscht; die nun leere Tabelle bleibt aber erhalten.

- `flush ruleset`

  löscht sämtliche Tabellen inklusive aller darin enthaltenen Ketten und Regeln. Das entspricht einem kompletten Reset der Firewall. Die Firewall akzeptiert nun jedes Paket.

- `list tables`
  `list table` *`family name`*
  `list chain` *`family tname cname`*

  listet Tabellen, Ketten und Regeln auf.

- `list ruleset`

  listet sämtliche Tabellen, Ketten und Regeln auf. Die durch geschwungene Klammern strukturierte Ausgabe erfüllt die Syntaxregeln für `nft -f`, kann also als Grundlage für eine neue Regeldatei verwendet werden. Mit der zusätzlichen Option `-j` erfolgt die Ausgabe im JSON-Format.

- `monitor [filterkriterien]`

  zeigt nftables-Ereignisse an (z. B. als Debugging-Hilfe).

Beispiele

Die beiden ersten Kommandos zeigen, welche Firewall-Tabellen auf dem Testrechner (Fedora 32) definiert sind und welche Ketten und Regeln für die Tabelle `firewalld` innerhalb der Familie `ip` gelten. (nftables erlaubt die Verwendung übereinstimmender Tabellennamen für unterschiedliche Familien. Deswegen muss beim zweiten `list`-Kommando auch die Familie angegeben werden.)

```
root#  nft list tables
table bridge filter
table bridge nat
table inet firewalld
table ip firewalld
table ip6 firewalld

root#  nft list table ip firewalld
table ip firewalld {
   chain nat_PREROUTING {
```

```
      type nat hook prerouting priority dstnat + 10; policy accept;
      jump nat_PREROUTING_ZONES
   }

   chain nat_PREROUTING_ZONES {
      iifname "enp1s0" goto nat_PRE_FedoraWorkstation
      goto nat_PRE_FedoraWorkstation
   }
   ...
```

`systemctl` deaktiviert nun den vorgegebenen Firewall-Dämon von Fedora:

```
root#  systemctl disable --now firewalld
```

Jetzt kann mit `nft` eine eigene Firewall zusammengesetzt werden. Es ist nicht üblich, die Regeln durch den wiederholten Aufruf von `nft` zu definieren. Wesentlich eleganter ist es, die Regeln in einer Textdatei zu formulieren und diese Datei wie ein Script auszuführen. Dazu geben Sie in der Zeile `nft -f` wie einen Shell-Interpreter an.

```
#!/sbin/nft -f
# vorhandene Firewall löschen
flush ruleset

# neue Tabelle für IPv4 und IPv6
table inet myfilter {
   chain input {
      type filter hook input priority 0; policy drop;
      # fehlerhafte Pakete blockieren
      ct state invalid drop
      # Pakete von selbst erzeugten Verbindungen immer akzeptieren
      ct state {established, related} accept
      # internen Netzwerkverkehr akzeptieren
      iif lo accept
      # Pakete an Loopback aber von externen Adressen blockieren
      iif != lo ip daddr 127.0.0.1/8 drop
      iif != lo ip6 daddr ::1/128 drop
      # ICMP-Pakete akzeptieren
      ip protocol icmp accept
      ip6 nexthdr icmpv6 accept
      # SSH-Verbindung von außen akzeptieren
      tcp dport 22 accept
      # bei Server-Konfiguration: weitere Regeln für
      # HTTP + HTTPS (Port 80 + 443), Samba usw.
      # ...
   }
   # kein Forwarding
   chain forward {
      type filter hook forward priority 0; policy drop;
   }
}
```

Nun können Sie das Script ausführen:

```
root#  chmod +x myfirewall
root#  ./myfirewall
```

`nft -f` führt zuerst einen Syntax-Check durch. Wenn es dabei Fehler feststellt, bricht es den Vorgang mit einer Fehlermeldung ab. Die bisherige Firewall bleibt in diesem Fall erhalten.

Das obige Script wurde leicht modifiziert vom Gentoo-Wiki übernommen. Dort sowie im ArchLinux-Wiki finden Sie noch mehr Beispiele:

https://wiki.gentoo.org/wiki/Nftables/Examples
https://wiki.archlinux.org/title/Nftables

ngrep [optionen] [grep-suchausdruck] [pcap-filterausdruck]

`ngrep` ist ein sogenannter *Packet Sniffer*. Das Kommando liest den Netzwerkverkehr eines Ports mit und filtert ihn. `ngrep` bietet ähnliche Funktionen wie das Kommando `tcpdump` und greift wie dieses auf die `pcap`-Bibliothek zurück, um die Netzwerkpakete auszulesen. Im Unterschied zu `tcpdump` berücksichtigt `ngrep` aber auch den Inhalt der Pakete. Das funktioniert naturgemäß nur bei nicht verschlüsselten Protokollen, also z. B. bei HTTP oder FTP.

Der Suchausdruck ist wie bei `grep` als reguläres Muster zu formulieren. Für den `pcap`-Filterausdruck gilt die bei `tcpdump` zusammengefasste Syntax.

- `-d` *schnittstelle*|`any`

  bestimmt die Netzwerkschnittstelle.

- `-i`

  ignoriert die Groß- und Kleinschreibung im `grep`-Suchausdruck.

- `-v`

  invertiert die Suche. `ngrep` liefert somit nur die Pakete, in denen das `grep`-Suchmuster *nicht* erkannt wurde.

- `-w`

  interpretiert den `grep`-Suchausdruck als Wort.

- `-W byline`

  berücksichtigt bei der Ausgabe Zeilenumbrüche, was zu besser lesbaren Ausgaben führt.

Beispiel

Das folgende Beispiel lauscht auf allen Schnittstellen nach HTTP-Paketen, in denen die Schlüsselwörter `user`, `pass` usw. vorkommen:

```
root#  ngrep -d any -i 'user|pass|pwd|mail|login' port 80
interface: any
filter: (ip or ip6) and ( port 80 )
match: user|pass|pwd|mail|login

T 10.0.0.87:58480 -> 91.229.57.14:80 [AP] POST /index.php
  HTTP/1.1..Host: ... user=name&pass=geheim&login=Login
...
```

`nice` [optionen] programm

`nice` startet das angegebene Programm mit einer verringerten oder erhöhten Priorität. Das Kommando kann dazu eingesetzt werden, nicht zeitkritische Programme mit kleiner Priorität zu starten, um das restliche System nicht zu stark zu beeinträchtigen.

- `-n +/-`*n*

  gibt den `nice`-Wert vor. Standardmäßig (also ohne `nice`) werden Programme mit dem `nice`-Wert `0` gestartet. Ein Wert von `-20` bedeutet »höchste Priorität«, ein Wert von `+19` bedeutet »niedrigste Priorität«. Werte kleiner als `0` dürfen nur von `root` angegeben werden, d. h., die meisten Anwender können mit `nice` nur Programme mit reduzierter Priorität starten. Wenn auf diese Option verzichtet wird, startet `nice` das Programm mit dem `nice`-Wert von `+10`.

Beachten Sie, dass `nice` nur die CPU-Belastung steuert. Wenn Sie die I/O-Belastung eines Kommandos reduzieren möchten, setzen Sie besser `ionice` ein.

Beispiel

Das folgende Kommando startet das Script `mybackup.sh` mit niedrigerer Priorität:

```
user$  nice -n 10 mybackup.sh
```

`nl` [optionen] datei

`nl` nummeriert alle nichtleeren Zeilen der angegebenen Textdatei und schreibt das Ergebnis in die Standardausgabe. Durch die Einstellung der zahlreichen Optionen kann eine seitenweise Nummerierung, eine Nummerierung von Kopf- und Fußzeilen etc. erreicht werden.

```
nmap [optionen] hostname/ip-adresse/ip-adressbereich
```

Das Kommando `nmap` (*Network Mapper*) aus dem gleichnamigen Paket führt einen Port-Scan durch und versucht festzustellen, welche Netzwerkdienste auf dem angegebenen Rechner bzw. im angegebenen Netzwerk aktiv sind. `nmap` sollte ausschließlich zur Analyse eigener Rechner bzw. nach Rücksprache mit dem jeweiligen Administrator eingesetzt werden. Ein Port-Scan fremder Rechner kann als Einbruchsversuch gewertet werden!

- `-A`

  führt einen ausführlichen (»aggressiven«) Scan durch, entspricht `-sV -O -sC --traceroute`.

- `-F`

  berücksichtigt nur die 100 wichtigsten Ports aus `/usr/share/nmap/nmap-services` (schneller Scan).

- `-iL` *datei*

  scannt die in der Datei angegebenen IP-Adressen.

- `-oN` *datei* / `-oG` *datei* / `-oX` *datei.xml*

  schreibt die Ergebnisse in eine normale Textdatei, in eine Textdatei, die mit `grep` gut weiterverarbeitet werden kann, oder in eine XML-Datei. Ohne die Option verwendet `nmap` die Standardausgabe und das normale Textformat.

- `-O`

  versucht, das Betriebssystem zu erkennen; diese Option muss mit einer Scan-Option kombiniert werden, z. B. mit `-sS`, `-sT` oder `-sF`.

- `-p1-10,22,80`

  berücksichtigt nur die angegebenen Ports.

- `-Pn`

  verzichtet auf einen Ping-Test. Damit betrachtet `nmap` alle Hosts als online und führt auf jeden Fall einen Scan durch (langsam!).

- `-sL`

  listet alle Ports auf und gibt in der Vergangenheit zugeordnete Hostnamen an. Das gelingt besonders schnell, liefert aber veraltete Daten auch von Geräten, die aktuell gar nicht mehr online sind.

- `-sP`

  führt nur einen Ping-Test durch (schnell).

- `-sS`

  führt einen TCP-SYN-Scan durch (gilt per Default).

- `-sU`

  berücksichtigt auch UDP. Diese Option darf zusammen mit einer anderen `-s`-Option verwendet werden.

- `-sV`

  versucht, bei offenen Ports herauszufinden, welcher Service dort angeboten wird (*Service Version Detection*) bzw. welches Programm in welcher Version für den Dienst zuständig ist.

- `-T0` bis `-T5`

  wählt ein Timing-Schema. `-T5` ist am schnellsten. `-T3` gilt per Default. `-T0` und `-T1` sind extrem langsam, minimieren aber das Risiko, dass der Scan bemerkt wird.

- `-v`

  gibt detaillierte Informationen aus (*verbose*).

Sie müssen sich beim Aufruf für *eine* `-s`-Option entscheiden. Einzig `-sU` darf mit anderen `-s`-Optionen kombiniert werden. Generell ist die richtige Wahl der Optionen ein Kompromiss zwischen Gründlichkeit und Geschwindigkeit.

In vielen Fällen reicht `nmap -v -A name`, um einen ersten Überblick über die Netzwerkdienste des angegebenen Rechners zu bekommen. Fortgeschrittene `nmap`-Anwender finden weitere Details auf der `man`-Seite sowie auf den folgenden Webseiten:

https://insecure.org/nmap
https://nmap.org/book

Zu `nmap` existieren auch grafische Benutzeroberflächen, z. B. `nmap-frontend` oder `zenmap`.

Beispiele

Das folgende Kommando führt einen schnellen Netzwerk-Scan im lokalen Netzwerk durch (256 IP-Adressen). Dank der Konzentration auf die wichtigsten 100 Ports ist die Sache in gut zwei Sekunden erledigt. Die `nmap`-Ausgaben wurden aus Platzgründen stark gekürzt und zeigen nur die Ergebnisse von zwei der gefundenen Geräte:

```
root#  nmap -F  -T4 10.0.0.0/24
Nmap scan report for imac (10.0.0.2)
```