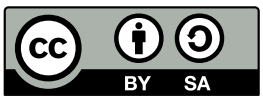
# Netzwerk Teil 2

Zinching Dang

11. Januar 2016



## Unterschied Host – Router

- Standardverhalten eines Linux-Rechners: Host
- nur IP-Pakete mit Zieladressen, die dem Rechner zugeordnet sind, werden angenommen
- IP-Adresse muss auf einem Interface konfiguriert sein
- Pakete für andere IP-Adressen werden verworfen
- über Sysctl kann Forwarding aktiviert werden
- Rechner verhält sich dann wie ein Router
- leitet Pakete für fremde IP-Adressen anhand der Routing-Tabelle weiter



Aus Netzwerksicht lassen sich Rechner in Hosts und Router unterteilen. Ein Host nimmt nur Pakete an, die für eine seiner IP-Adressen bestimmt sind, wohingegen ein Router auch Pakete für fremde IP-Adressen annimmt und diese weiterleitet. Standardmäßig verhält sich ein Linux-Rechner als Host. Dies ist sinnvoll, da ein unkontrolliertes Weiterleiten von IP-Paketen aus Sicherheitssicht unerwünscht ist.

## Forwarding per Sysctl aktivieren

- IPv4:
  `echo 1 > /proc/sys/net/ipv4/conf/all/forwarding`
- IPv6:
  `echo 1 > /proc/sys/net/ipv6/conf/all/forwarding`
- oder dauerhaft in `/etc/sysctl.conf`:

```
1 net.ipv4.conf.all.forwarding=1
2 net.ipv6.conf.all.forwarding=1
```

- Nebenwirkungen: deaktiviert u. a. stateless autoconfig für IPv6



Aus einem Host wird ein Router, indem Forwarding aktiviert wird. Dies ist unter Linux über Sysctl möglich. Forwarding kann für IPv4 und IPv6 getrennt aktiviert werden.

Wie bei allen Sysctl-Einstellungen kann dies über Dateien in `/proc/sys/` geschehen. Die dort getätigten Einstellungen gehen aber bei einem Neustart des Rechners verloren. Daher ist es sinnvoll, die Datei `/etc/sysctl.conf` zu verwenden.

Da das Einschalten von Forwarding aus dem Host einen Router macht, werden dabei automatisch weitere Einstellungen angepasst. Am wichtigsten ist hier, dass stateless autoconfiguration für IPv6 abgeschaltet wird.

## iproute2

- Ersatz für u. a. `ifconfig`, `route`, `arp`
- kann den Status von Netzwerk-Interfaces anzeigen und ändern
- IP-Adressen hinzufügen und löschen
- Routen anzeigen und ändern
- ARP-Tabelle anzeigen und ändern
- erweiterte Funktionalität im Vergleich zu den Vorgänger-Werkzeugen
- z. B. mehrere IP-Adressen auf einem Interface



Iproute2 wurde als Ersatz für die veralteten Netzwerkkonfigurations-Werkzeuge wie `ifconfig`, `route` und `arp` entwickelt. Neben einem einheitlichen Interface stellt es Funktionen zur Verfügung, die sonst nur mit Zusatzprogrammen oder gar nicht realisiert werden konnten.

Der größte Teil der Funktionalität wird über die Unterbefehle von `ip` bereitgestellt. Die Unterbefehle können auch abgekürzt werden um das Eingeben der Befehle zu erleichten. Beispielsweise lässt sich `ip adddress show` zu `ip a s` abkürzen.

## ip link

- Link-Status anzeigen: `ip link show dev eth0`
- von allen Interfaces: `ip link show`
- Interface einschalten: `ip link set eth0 up`
- ausschalten: `ip link set eth0 down`



Mit `ip link` kann der Status von Netzwerk-Interfaces angezeigt und manipuliert werden. Es können, wie schon in Teil 1 gesehen, auch virtuelle Netzwerkinterfaces hinzugefügt werden.

## ip address

- IP-Adresse hinzufügen:
  `ip address add 192.0.2.4/28 dev eth0`
- löschen: `ip address del 192.0.2.4/29 dev eth0`
- anzeigen: `ip address show [dev eth0]`
- auch mit IPv6:
  `ip addr add 2001:db8:f00d::3/64 dev eth0`



`ip address` kann verwendet werden um IP-Adressen zu Interfaces hinzuzufügen und sie wieder zu löschen. Es können auch die momentan auf dem Interface konfigurierten Adressen angezeigt werden.

Das letzte Beispiel demonstriert auch das Abkürzen des Unterbefehls `address`.

Nachträglich zu einem Interface hinzugefügte Adressen gehen beim Herunterfahren des Rechners verloren. Es ist daher sinnvoll, diese beim Hochfahren automatisch neu hinzufügen zu lassen. Dies kann am besten erreicht werden, indem in `/etc/network/interfaces` bei der entsprechenden Interface-Definition die Zeile `up ip addr add ...` hinzugefügt wird. Durch das Schlüsselwort `up` können beim Aktivieren eines Interfaces beliebige Befehle ausgeführt werden.

## ip route

- Route hinzufügen:
  `ip route add 198.51.100.0/24 via 192.0.2.1 dev eth0`
- löschen analog mit `del`
- Routing-Tabelle anzeigen: `ip route show`
- für IPv6: `ip -6 route show`



Mit `ip route` kann die Routing-Tabelle manipuliert werden. `ip route` ist der Nachfolger für den alten Befehl `route`.

Auch Routen gehen beim Herunterfahren des Rechners verloren. Auch hier sollte ein Eintrag in `/etc/network/interfaces` hinzugefügt werden.

## ip neighbour

- Neighbour-Tabelle anzeigen: `ip neighbour show`
- für IPv4 auch als ARP-Tabelle bekannt



Mit `ip neighbour` (üblicherweise zu `ip neigh` abgekürzt), lässt sich die Neighbourhood-Tabelle, also die MAC-Adressen zu den lokalen IP-Adressen, mit denen der Rechner kommuniziert hat, anzeigen. Diese enthält zumindest die MAC-Adresse des Default-Gateways.

Unter IPv4 war diese Tabelle auch als ARP-Tabelle, nach dem Address Resolution Protocol, bekannt. Mit IPv6 wird statt ARP das Neighbourhood Discovery Protocol (NDP) verwendet, der Begriff ARP-Tabelle ist also nicht mehr angemessen.

## Lab 8.1: Routing

- Forwarding aktivieren
- dem Rechner eine weitere IP-Adresse hinzufügen
- Routen zu den zusätzlichen IP-Adressen auf den anderen Rechnern hinzufügen
- VLAN 7 deaktivieren und über Partner-Rechner erreichen

## iptables

- Paketfilter für IP-Pakete unter Linux
- `ip6tables` für IPv6
- kann eingehende, ausgehende und weitergeleitete Pakete filtern
- einsetzbar als Firewall auf Rechnern und auf Routern
- kann auch als NAT-Gateway verwendet werden
- in Linux 3.13 durch nftables ersetzt
- nftables in vielen aktuellen Distributionen noch nicht enthalten
- iptables über Kompatibilitätslayer weiter unterstützt



`iptables` wird unter Linux verwendet, um IP-Pakete zu filtern. Zudem können damit Pakete auch umgeschrieben werden, und dadurch z. B. Network Address Translation realisiert werden. Für IPv6 steht das Werkzeug `ip6tables` bereit, das sich sehr ähnlich bedienen lässt.

`iptables` wird üblicherweise als Firewall auf Rechner oder Routern eingesetzt. Auf Routern kann es auch zum Einrichten eines NAT-Gateways verwendet werden.

Seit Linux 3.13 steht das Nachfolge-Projekt nftables bereit, das `iptables` langfristig ersetzen soll. Allerdings ist nftables in den meisten aktuellen Distributionen noch nicht verfügbar, sodass `iptables` weiterhin benötigt wird. Zudem wird auch wenn sich nftables verbreitet hat, `iptables` weiterhin über einen Kompatibilitätslayer unterstützt werden.

## Tabellen und Chains

- jedes Paket durchläuft verschiedene Tabellen, die wiederum Chains enthalten
- Chains enthalten Filter-Regeln, die nacheinander ausgewertet werden
- die erste zutreffende Regel wird angewandt
- wichtigste Tabellen:
  - filter: für „normales“ Filtern
  - nat: für NAT



Bei der Verarbeitung durch den Linux-Kernel durchläuft ein Netzwerk-Paket verschiedene Tabellen. Diese Tabellen haben festgelegte Aufgaben. Die in diesem Kurs behandelten Tabellen sind filter und nat.

Die Tabelle filter dient dem Filtern von IP-Pakten auf Basis von festzulegenden Kriterien. Dabei können Pakete, die für das System bestimmt sind, Pakete, die vom System versendet wurden und Pakete, die das System für andere Rechner weiterleitet, gefiltert werden.

Die Tabelle nat wird verwendet, um Pakete umzuschreiben. Dadurch kann Network Address Translation realisiert werden.

Jede Tabelle enthält sogenannte Chains, in denen die eigentlichen Filterregeln enthalten sind. Die Regeln in den Chains werden nacheinander ausgewertet.

## Tabellen

- Tabelle filter aufgeteilt in die Chains
  - INPUT: Pakete für den lokalen Rechner
  - FORWARD: Pakete, die vom Rechner weitergeleitet werden (Router)
  - OUTPUT: Pakete, die vom lokalen Rechner erzeugt wurden
- Tabelle nat aufgeteilt in die Chains
  - PREROUTING: Pakete manipulieren, bevor entschieden wird, wie sie weitergeleitet werden
  - OUTPUT: Pakete, die vom lokalen Rechner erzeugt wurden, manipulieren
  - POSTROUTING: Pakete manipulieren, nachdem entschieden wurde, wie sie weitergeleitet werden



Die Tabelle filter enthält die Chains INPUT, FORWARD und OUTPUT. Der Chain INPUT verarbeitet Pakete, die für den Rechner an sich bestimmt sind. Der Chains FORWARD verarbeitet Pakete, die vom Rechner als Router weitergeleitet werden. Der Chain OUTPUT verarbeitet Pakete, die vom Rechner versendet werden. Ein Netzwerk-Paket durchläuft nur einen dieser Chains.

Die Tabelle nat enthält die Chains PREROUTING, OUTPUT und POSTROUTING. Der Chain PREROUTING verarbeitet Pakete, bevor entschieden wird, wie diese weitergeleitet werden. Üblicherweise wird hier die Ziel-Adresse von Paketen umgeschrieben (Destination NAT). Der Chain OUTPUT verarbeitet Pakete, die vom Rechner selbst erzeugt wurden. Der Chain POSTROUTING verarbeitet Pakete, nachdem entschieden wurde, wie diese weitergeleitet werden sollen. Üblicherweise wird hier die Quell-Adresse von Paketen umgeschrieben (Source NAT, Masquerading).

## Filterregeln einfügen

- `iptables -t <tabelle> -A <chain> ... <Kriterien> ... -j <aktion>`
- fügt eine Regel ans Ende des angegbenen Chains an (`-A`: append)
- löschen mit `-D` statt `-A`
- auch möglich: `-I` (insert), `-R` (replace)
- `-t filter` ist Standard
- wichtigste Aktionen:
  - ACCEPT: das Paket akzeptieren
  - DROP: das Paket verwerfen
  - REJECT: das Paket verwerfen und eine Fehlermeldung an den Absender senden



Filterregeln werden mit dem Befehl `iptables` manipuliert. Filterregeln werden immer in eine Tabelle und dort in einen Chain eingefügt. Wird keine Tabelle angegeben, wird die Standard-Tabelle filter verwendet. Des weiteren enthält jede Regel ein oder mehrere Filter-Kriterien und eine Aktion, die ausgeführt wird, wenn das Paket den Kriterien entspricht. Mit der Option `-A` werden Regeln ans Ende eines Chains angehangen. Mit `-D` können sie wieder gelöscht werden. Da bei den Filterregeln die Reihenfolge von Bedeutung ist, können Regeln auch mit `-I` an eine bestimmte Stelle eingefügt werden (es kann eine Nummer angegeben werden, Standard ist der Anfang des Chains) oder mit `-R` ersetzt werden.

Die wichtigsten Aktionen in der Tabelle filter sind ACCEPT, wodurch das Paket akzeptiert wird, DROP, wodurch das Paket verworfen wird und REJECT, wodurch das Paket verworfen wird und der Absender eine Fehlermeldung erhält. Diese drei Aktionen werden terminating genannt, da die nachfolgenden Regeln im Chain nicht mehr ausgewertet werden. Ein Beispiel für eine non-terminating Aktion ist LOG, die eine Log-Meldung erzeugt, aber die Verarbeitung des Chains danach fortsetzt.

Bei der Entscheidung zwischen DROP und REJECT ist zu bedenken, dass eine Antwort an einen Angreifer eventuell nicht erwünscht ist, das einfache

Verwerfen von Paketen aber zu teilweise langen Timeouts und schwer zu untersuchenden Problemen führen kann. In der Regel ist REJECT daher zu bevorzugen.

## Filterkriterien

- `-s <adresse>`: nach Quell-IP filtern
- `-d <adresse>`: nach Ziel-IP filtern
- `-p tcp --sport <port>`: nach TCP-Quell-Port filtern
- `-p tcp --dport <port>`: nach TCP-Ziel-Port filtern
- `-p udp --sport <port>`: nach UDP-Quell-Port filtern
- `-p udp --dport <port>`: nach UDP-Ziel-Port filtern
- `-m state --state <state>`: nach Zustand der Verbindung filtern
- wichtigste Zustände: NEW, INVALID, ESTABLISHED, RELATED
- Kriterien können kombiniert werden
- `!` vor dem Kriterium kehrt die Bedeutung um



Die wichtigsten Filter-Kriterien sind auf der Folie aufgeführt. Neben der Quell- und Ziel-IP-Adresse wird häufig auch nach UDP- und TCP-Ports gefiltert um den Zugriff auf bestimmte Dienste einzuschränken.

`iptables` unterstützt auch auch das Filtern von TCP-Verbindungen nach ihrem Zustand. So können z. B. mit dem Zustand NEW alle Pakete erfasst werden, die eine neue Verbindung aufbauen. Die Zustände ESTABLISHED und RELATED treffen auf Pakete zu, die zu einer bestehenden Verbindung gehören. Mit INVALID können Pakete erfasst werden, die weder zu einer bestehenden Verbindung gehören, noch eine neue Verbindung aufbauen. Mit dem Filtern nach Zustand ist es z. B. möglich, alle eingehenden Pakete zu blockieren, außer solche, die zu von innen aufgebauten Verbindungen gehören. Ohne diese Einschränkung würde das Blockieren aller eingehenden Pakete dazu führen, dass auch keine Verbindungen mehr von innen aufgebaut werden können, da die Antworten auf diese Verbindungsversuche verworfen werden.

Strenggenommen lassen sich nur TCP-Verbindungen nach Zustand filtern, da UDP und ICMP zustandslose Protokolle sind. `iptables` versucht aber auch bei diesen Protokollen einen Zustand z. B. anhand von Port-Nummern zu ermitteln.

## Beispiel

```
1 iptables -A INPUT -s 192.0.2.0/27 -p tcp --dport 443
    -j ACCEPT
2 iptables -A INPUT -p tcp --dport 22 -j ACCEPT
3 iptables -A INPUT ! -s 203.0.113.0/24 -p tcp --dport
    25 -j ACCEPT
4 iptables -A INPUT -p tcp -m state ! --state
   ESTABLISHED,RELATED -j REJECT
```

- Zugriff von 192.0.2.0/27 auf Port 443 erlauben
- Zugriff von überall auf Port 22 erlauben
- Zugriff von überall außer 203.0.113.0/24 auf Port 25 erlauben
- Zugriff auf alle anderen TCP-Ports verbieten (außer bestehende Verbindungen)



In diesem Beispiel werden nacheinander vier Regeln an den INPUT-Chain der Tabelle filter angehangen. Die Reihenfolge ist in diesem Beispiel sehr wichtig, da die Pakete auf mehrere Regeln zutreffen können und nur die erste zutreffende Regel angewandt wird.

## NAT

- Network Address Translation
- allgemein: Verändern von Adressen in IP-Paketen
- häufigste Anwendung: mehrere private IPv4-Adressen auf eine öffentliche umschreiben
- auch Masquerading genannt
- bei vielen Internetzugängen nötig, da für viele Rechner nur eine öffentliche IP-Adresse bereit steht
- andere häufige Anwendung: Port-Forwarding eines Ports auf dem Gateway zu einer internen IP-Adresse



Network Address Translation wird allgemein verwendet um Adressen (und Port-Nummern) in IP-Paketen umzuschreiben. Aufgrund des begrenzten Vorrats an IPv4-Adressen ist es häufig nötig, viele Rechner hinter nur einer oder wenigen öffentlichen IP-Adressen zu betreiben. In diesem Fall kommt ein NAT-Gateway zum Einsatz, das die private Adresse des Rechners durch die öffentliche Adresse des Gateways ersetzt. Dieses Verfahren wird auch Masquerading genannt. Dies ist nicht die einzige Anwendung von NAT, aber die häufigste. Sollen Dienste auf den hinter einem NAT-Gateway befindlichen Rechnern von außen erreichbar sein, muss auf dem Gateway ein sogenanntes Port-Forwarding eingerichtet werden. Dabei werden auf bestimmten Ports ankommende Pakete an eine interne Adresse weitergeleitet.

## NAT mit iptables

- NAT-Regeln in der Tabelle nat, Chain POSTROUTING
- MASQUERADE-Aktion sorgt dafür, dass die Quell-IP durch die Adresse des externen Interfaces des Gateways ersetzt wird

```
1 iptables -t nat -A POSTROUTING -o eth0 -s
    192.168.254.0/24 -j MASQUERADE
```

- bei fester externer Adresse SNAT statt MASQUERADE
- Verbindungen werden nicht unterbrochen, wenn das Interface aus- und wieder eingeschaltet wird

```
1 iptables -t nat -A POSTROUTING -o bond0 -s
    192.168.2.0/24 -j SNAT --to-source 203.0.113.7
```



Im ersten Beispiel ist ein masquerading NAT mit `iptables` realisiert worden. Dazu wurde der POSTROUTING-Chain in der Tabelle nat verwendet. Alle Pakete, die den Rechner über das Interface `eth0` verlassen und eine Absender-Adresse aus dem Bereich `192.168.254.0/24` haben, werden umgeschrieben.

Im zweiten Beispiel wurde ebenfalls ein masquerading NAT realisiert, allerdings hat der Rechner in diesem Fall eine feste öffentliche IP-Adresse auf dem Interface `bond0` (203.0.113.7). Durch die Verwendung von SNAT statt MASQUERADE werden die von innen aufgebauten Verbindungen beim Aus- und Einschalten des Interfaces `bond0` nicht abgebrochen. Dies ist nur bei einer festen äußeren Adresse sinnvoll, da die Verbindungen bei einer Adress-Änderung auf jeden Fall abgebrochen werden.

## Port-Weiterleitungen

- werden verwendet um Dienste auf internen IP-Adressen von außen zugänglich zu machen
- Port auf der externen Adresse wird auf eine interne Adresse umgeleitet
- mit DNAT im Chain PREROUTING

```
1 iptables -t nat -A PREROUTING -p tcp --dport 22200
    -j DNAT --to-destination 192.168.2.100:22
```

- TCP-Port 22200 von außen wird auf Port 22 (SSH) der internen IP 192.168.2.100 umgeleitet



Port-Weiterleitungen werden verwendet, um Dienste auf internen Adressen von außen zugänglich zu machen. Dazu wird ein Port auf dem Gateway weitergeleitet. Mit `iptables` kann dies mit Regeln im PREROUTING-Chain realisiert werden.

In diesem Beispiel wird der TCP-Port 22200 auf der öffentlichen IP-Adresse zum TCP-Port 22 auf der internen Adresse 192.168.2.100 weitergeleitet. Dabei kommt Destination NAT (DNAT) zum Einsatz, da die Ziel-Adresse und der Ziel-Port umgeschrieben werden. Durch diese Weiterleitung kann von außen bei Verwendung des richtigen Ports eine Verbindung zum SSH-Server auf 192.168.2.100 aufgebaut werden.

## Verwaltung

- Anzeigen der Regeln mit `iptables -t <tabelle> -L`
- mehr Informationen mit `-v`
- Löschen aller Regeln mit `iptables -t <tabelle> -F`
- Zurücksetzen der Paketzähler: `iptables -t <tabelle> -Z`
- Regeln (und Zähler) werden beim Herunterfahren automatisch gelöscht
- müssen beim Hochfahren neu definiert werden



Neben dem Verwalten von Filterregeln kann `iptables` auch die vorhanden Regeln anzeigen. Dazu wird die Option `-L` verwendet. Da so nicht alle Informationen zu den Regeln angezeigt werden, kann zusätzlich die Option `-v` angegeben werden. So sind auch die Paket-Zähler für die einzelnen Regeln zu sehen.

Sollen alle Regeln gelöscht werden, kann die Option `-F` verwendet werden. Die Zähler lassen sich mit `-Z` zurücksetzen.

Beim Herunterfahren werden die Regeln automatisch gelöscht und die Zähler zurückgesetzt. Daher müssen die Regeln beim Hochfahren neu definiert werden.

## iptables-persistent

- iptables-persistent stellt einen Dienst bereit, der gespeicherte Regeln beim Hochfahren automatisch lädt
- Regeln werden in der Datei `/etc/iptables/rules.v4` gespeichert
- Speichern der Regeln mit `iptables-save > /etc/iptables/rules.v4`
- Manuelles Laden der Regeln: `iptables-restore < /etc/iptables/rules.v4`
- Bei Remote-Zugang besser `iptables-apply < /etc/iptables/rules.v4`
- IPv6-Regeln mit `ip6tables*` und `rules.v6` analog dazu



Mit dem Paket `iptables-persistent` können `iptables`-Regeln nach einem Neustart automatisch neu geladen werden. Das Paket enthält dazu ein Init-Skript, das beim Systemstart automatisch ausgeführt wird. Das Skript kann auch verwendet werden, um die vorhandenen Regeln zu speichern oder zu löschen.

Die dauerhaft vorhandenen Regeln werden in Text-Dateien in `/etc/iptables/` gespeichert und können dort mit Vorsicht auch von Hand editiert oder gelöscht werden.

## Lab 8.2: iptables

- Zugriff auf SSH einschränken
- Zugriff auf VLAN 7 von anderen Rechnern verbieten
- iptables-Regeln beim Hochfahren wiederherstellen