IPv6 mit Linux
(Einführung)

Tutorial

Dr. Peter Bieringer
Deep Space 6
peter@deepspace6.net
http://www.deepspace6.net/

IPv6-Kongress
Frankfurt/Main, Deutschland
20. - 21. Mai 2010
http://www.ipv6-kongress.de/

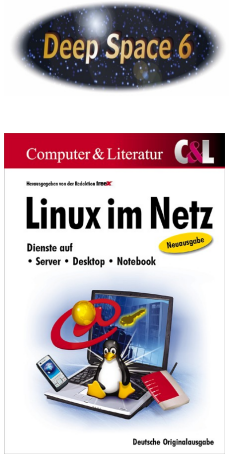
Inhalt

IPv6 mit Linux (Einführung)

- IPv6-Konfiguration
- Fehlersuche
- IPv6-Anbindung
- IPv6-aktive Services
- IPv6-Firewalling
- IPv6 QoS
- IPv6 in virtualisierten Umgebungen (libvirt & kvm)

Über mich

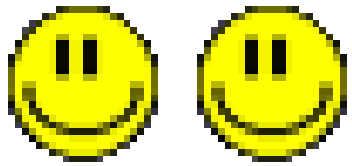
- **Wohnhaft in München (Deutschland)**
- **Beschäftigt als Netzwerkspezialist bei *Venyon GmbH***
  - Aufbau kleiner Rechenzentren für OverTheAir (OTA)-Personal-isierung von NFC-fähigen Mobiltelefonen
- **Mitbegründer und Kernmitglied von *Deep Space 6***
- **Autor des "Linux IPv6 HowTo"**
- **Mitautor des Buches "Linux im Netz"**
  - Grundlagen von TCP/IP incl. IPv6, DNS, DHCP



Meine Internet- & IPv6-Historie

- **1993: Erster Kontakt mit dem Internet (Univ., SunOS)**
- **1996: Erste Erfahrungen mit IPv6 und Linux**
- **1997: *IPv6 & Linux - HowTo*, *initscripts-ipv6***
- **1999: *IPv6 & Linux - Current Status***
- **2001: *Linux IPv6 HOWTO*, *ipv6calc***
- **2002: Mitbegründer von *Deep Space 6***

inzwischen 14 Jahre IPv6-Erfahrung!



IPv6-Konfiguration von Linux

IPv6-Konfiguration

- **Adressen**
  - Automatisch
    - Link-Local
  - Statisch
    - Site-Local, Unique-Local, Global
  - Dynamisch
    - Router Advertisement Daemon
    - DHCPv6
- **Routing**
  - Statisch
  - via Autokonfiguration
  - Dynamisch

IPv6 in Linux Voraussetzungen

- **IPv6 im Linux-Kernel bereits aktiviert**
  - Test: Datei existiert: `/proc/net/if_inet6`
- **IPv6 im Linux-Kernel aktivieren**
  - Modul laden

```
# modprobe ipv6
```

  - Prüfen

```
# lsmod |grep -w 'ipv6'
ipv6                  223810  22 ip6t_REJECT,nf_conntrack_ipv6

# cat /proc/net/if_inet6
00000000000000000000000000000001 01 80 10 80       lo
fe8000000000000022421fffe012345 02 40 20 80     eth0
```

- **IPv6-Forwarding aktivieren (bei Bedarf)**

```
# sysctl -w net.ipv6.conf.all.forwarding=1
```

IPv6-Adresskonfiguration Ansicht

- **Werkzeuge `ip` oder `ifconfig`**

```
# ip -6 addr show dev INTERFACE
# ifconfig INTERFACE
```

- **Beispiele**

```
# ip -6 addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_ fast qlen 100
    inet6 fe80::224:21ff:fe01:2345/64 scope link
       valid_lft forever preferred_lft forever


# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:24:21:01:23:45
          inet6 addr: fe80::224:21ff:fe01:2345/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          ...
```

IPv6-Adresskonfiguration Statische Zuweisung

- **Werkzeuge `ip` oder `ifconfig`**

```
# ip -6 addr add IPV6ADDRESS/PREFIXLENGTH dev INTERFACE
# ifconfig INTERFACE inet6 add IPV6ADDRESS/PREFIXLENGTH
```

- **Beispiele**

```
# ip -6 addr add 2001:db8:0:1::1/64 dev eth0
# ifconfig eth0 inet6 add 2001:db8:0:1::1/64
```

- **Ergebnis**

```
# ip -6 addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qlen 1000
    inet6 2001:db8:0:1::1/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::224:21ff:fe01:2345/64 scope link
       valid_lft forever preferred_lft forever
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:24:21:01:23:45
          inet6 addr: 2001:db8:0:1::1/64 Scope:Global
          inet6 addr: fe80::224:21ff:fe01:2345/64 Scope:Link
```

IPv6-Adresskonfiguration Router Advertisements

- **Konfigurationsdatei auf Router: `/etc/radvd.conf`**

```
interface eth0
{
    AdvSendAdvert on;
     ...
    prefix 2001:db8:0:1::/64
    {
        ...
        AdvPreferredLifetime 86400;
        AdvValidLifetime 604800;
    };
```

- **Ergebnis auf Client nach Eintreffen eines RAs**

```
# ip -6 addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qlen 1000
    inet6 2001:db8:0:1::224:21ff:fe01:2345/64 scope global
       valid_lft 604711sec preferred_lft 86311sec
    inet6 fe80::224:21ff:fe01:2345/64 scope link
       valid_lft forever preferred_lft forever
```

IPv6-Adresskonfiguration Privacy

- **Aktivieren von IPv6-Privacy gemäß RFC 4941 (3041)**
  - Konfiguration

```
# sysctl -w net.ipv6.conf.eth0.use_tempaddr=2
```

  - Restart der Schnittstelle notwendig

```
# ip link set dev eth0 down
# ip link set dev eth0 up
```

- **Ergebnis auf Client nach Eintreffen eines RAs**

```
# ip link show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 ...
    link/ether 00:24:21:01:23:45 brd ff:ff:ff:ff:ff:ff

# ip -6 addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qlen 1000
    inet6 2001:db8:0:1:8992:3c03:d6e2:ed72/64 scope global secondary dynamic
       valid_lft 604711sec preferred_lft 86311sec
    inet6 2001:db8:0:1::224:21ff:fe01:2345/64 scope global
       valid_lft 604711sec preferred_lft 86311sec
    ...
```

IPv6-Routing Ansicht

- **Werkzeuge `ip` oder `route`**

```
# ip -6 route show [dev INTERFACE]
# route -n -A inet6
```

- **Beispiele**

```
# ip -6 route show dev eth0
fe80::/64 dev eth0  metric 256
      expires 21296936sec  mtu 1500 advmss 1440 hoplimit 4294967295
2001:db8:0:1::/64 dev eth0  proto kernel  metric 256
      expires 604545sec  mtu 1500 advmss 1440 hoplimit 4294967295
default via fe80::224:21ff:fe00:1 dev eth0  proto kernel  metric 1024
      expires 1637sec mtu 1500 advmss 1440 hoplimit 64


# route -n -A inet6 | grep -w eth0
2001:db8:0:1::/64      ::                      UA    256    94    0 eth0
fe80::/64              ::                      U     256    0     0 eth0
::/0                   fe80::224:21ff:fe00:0001 UGDA  1024   36    0 eth0
ff00::/8               ::                      U     256    0     0 eth0
```

IPv6-Routing Konfiguration

- **Werkzeuge `ip` oder `route`**

```
# ip -6 route add IPV6NETWORK/PREFIXLENGTH via IPV6ADDRESS
# route -A inet6 add IPV6NETWORK/PREFIXLENGTH gw IPV6ADDRESS
```

- **Beispiele**

```
# ip -6 route add 2001:db8:0:2::/64 via 2001:db8:0:1::1
# route -A inet6 add 2001:db8:0:2::/64 gw 2001:db8:0:1::1
```

- **Ergebnis**

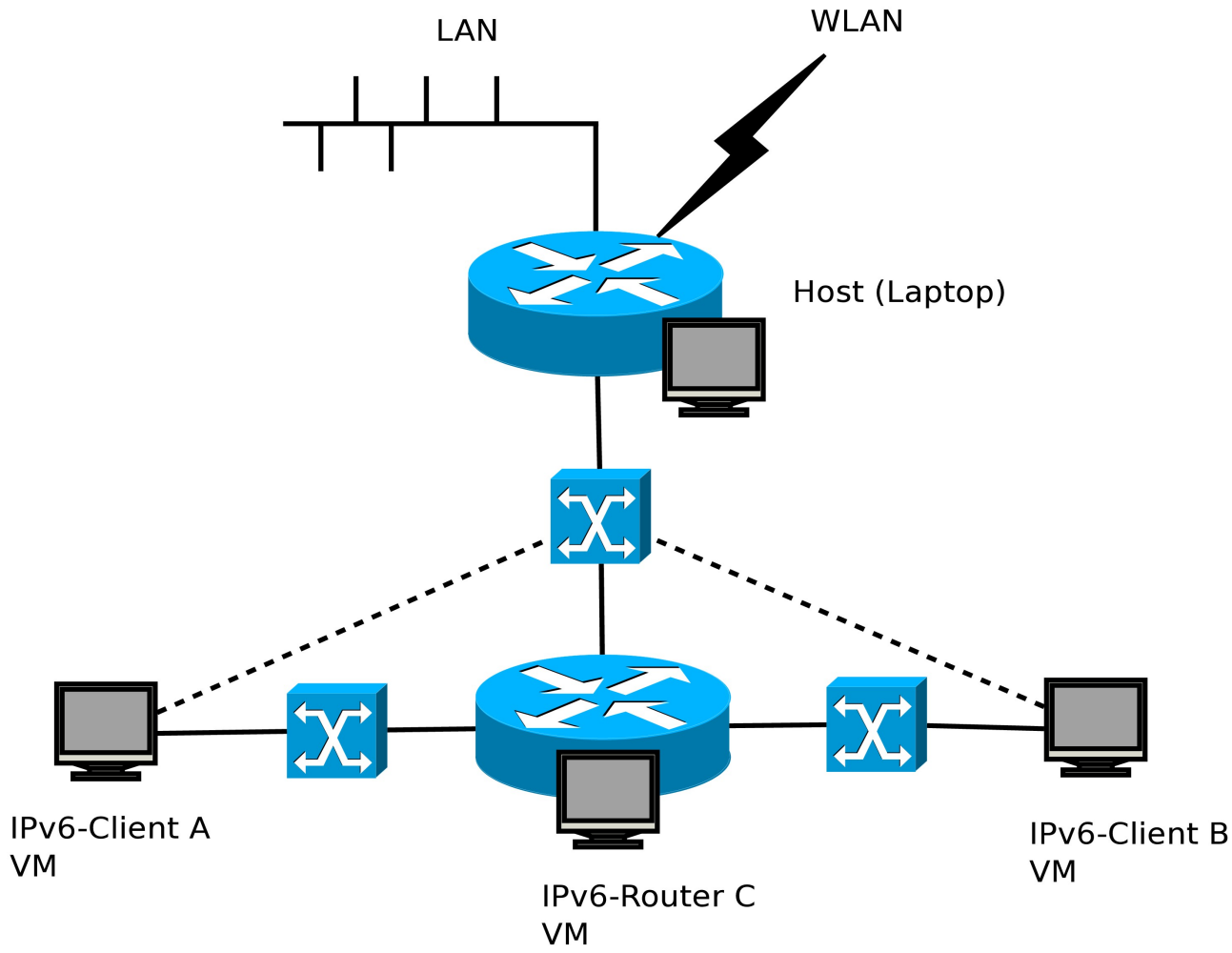
```
# ip -6 route show dev eth0 | grep "via 2001"
2001:db8:0:2::/64 via 2001:db8:0:1::1 dev eth0  proto kernel
      metric 1024 expires 1637sec mtu 1500 advmss 1440 hoplimit 64

# route -n -A inet6 | grep -w eth0 | grep "2001" | grep "G"
2001:db8:0:2::/64   2001:db8:0:1::1      UG    1024   0        0 eth0
```

IPv6-Adresskonfiguration DHCPv6

- **Server**
  - Software: ISC-DHCP (4.1.1), dibbler (0.7.3)
  - Verteilung von
    - IPv6-DNS-Server
    - DNS Search List
- **Client**
  - Software: dhclient (4.1.1), z.B. in Fedora 12
  - Unterstützt
    - IPv6-DNS-Server
    - DNS Search List

dhcpv6 (1.2.0) – Weiterentwicklung gestoppt seit 09/2009 

Demo Aufbau


LAN
WLAN
Host (Laptop)
IPv6-Client A
VM
IPv6-Router C
VM
IPv6-Client B
VM

Demo IPv6 DHCP

- **IPv6-only Client-Konfiguration**
  - Fedora 12: /etc/sysconfig/network-scripts/ifcfg-INTERFACE
    - Statische IPv6-Adresse vom DHCP-Server

```
DHCLIENTARGS="-6"
```

    - Temporäre IPv6-Adresse vom DHCP-Server

```
DHCLIENTARGS="-6 -T"
```

  - Test mit Browser: http://www.ipv6.bieringer.de/ (IPv6 only)

Fehlersuche

Werkzeuge für Fehlersuche
Verbindungstest

- **Verbindungstest mit `ping6` (analog zu IPv4)**

```
$ ping6 -I INTERFACE IPV6LINKLOCALADDRESS
$ ping6 HOSTNAME|IPV6ADDRESS
```

- **Beispiele**
  - Link-Local (Angabe der Schnittstelle notwendig)

```
$ ping6 -I eth0 -c 1 fe80::224:21ff:fe01:2345
PING fe80::224:21ff:fe01:2345 from fe80::224:21ff:fe00:1 eth0: 56 data bytes
64 bytes from fe80::224:21ff:fe01:2345: icmp_seq=0 hops=64 time=445 usec
```

  - Global

```
$ ping6 -c 1 2001:db8:0:1::224:21ff:fe01:2345
PING 2001:db8:0:1::224:21ff:fe01:2345 from 2001:db8:0:1::1 eth0: 56 data bytes
64 bytes from 2001:db8:0:1::224:21ff:fe01:2345: icmp_seq=0 hops=64 time=445
usec
```

Werkzeuge für Fehlersuche
Maximum Transfer Unit

- **MTU durch Tunnels bzw. Transport verringert**

IPv4	MTU	IPv4 (-20)	TCP-MSS (-20)	UDP ICMP-ECHO (-8)
Ethernet	1500	1480	1460	1472
PPPoE (-8)	1492	1472	1452	1464
IPv6	**MTU**	**IPv6 (-40)**	**TCP-MSS (-20)**	**UDP ICMP-ECHO (-8)**
Ethernet	1500	1460	1440	1452
PPPoE (-8)	1492	1452	1432	1444
IPv6 in IPv4 über Ethernet (-20)	1480	1440	1420	1432
IPv6 in IPv4 über PPPoE (-28)	1472	1432	1412	1424
IPv6 in UDP in IPv4 über Ethernet (-28)	1472	1432	1412	1424
IPv6 in UDP in IPv4 über PPPoE (-36)	1464	1424	1404	1416
Minimum gemäß RFC	1280	1240	1220	1232

Werkzeuge für Fehlersuche
Maximum Transfer Unit

- **Störung der PTMU-Discovery bei Blockade von ICMP**
- **Test**
  - ICMP ECHO Payload 1452 = MTU 1500

```
$ ping6 -c 1 -M do -s 1452 www.six.heise.de
PING www.six.heise.de(www.six.heise.de) 1452 data bytes

1452 bytes from www.six.heise.de: icmp_seq=1 ttl=57 time=76.8 ms
```

  - ICMP ECHO Payload 1453 = MTU 1501

```
$ ping6 -c 1 -M do -s 1453 www.six.heise.de
PING www.six.heise.de(www.six.heise.de) 1453 data bytes

From 2001:db8:0:1:224:21ff:fe00:1 icmp_seq=1 Packet too big: mtu=1500
```

- **TCP-MSS “Clamping” (Klammerung) mit ip6tables**

```
# ip6tables -t mangle -I FORWARD  -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --set-mss 1404
# ip6tables -t mangle -I OUTPUT   -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --set-mss 1404
# ip6tables -t mangle -I INPUT    -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --set-mss 1404
```

Werkzeuge für Fehlersuche Routenverfolgung

- **Routenverfolgung mit `traceroute` (analog zu IPv4)**
  - Neue Versionen von traceroute unterstützen zusätzlich IPv6

```
$ traceroute -6 HOSTNAME|IPV6ADDRESS
```

- **Beispiel**

```
$ traceroute -6 -q 1 www.bieringer.de
traceroute to www.bieringer.de (2001:a60:9002:1::186:3), 30 hops max, 80 byte
packets
 1  2001:db8:0:1::1 (2001:db8:0:f101::1)  0.249 ms
 2  gw-24.muc-02.de.sixxs.net (2001:a60:f000:17::1)  43.743 ms
 3  2001:a60:0:30::1 (2001:a60:0:30::1)  48.427 ms
 4  www.bieringer.de (2001:a60:9002:1::186:3)  48.427 ms
```

Werkzeuge für Fehlersuche Router/Neighbor Discovery

- **Werkzeug `ip`**

```
# ip -6 neigh show
```

- **Beispiele**

```
# ip -6 neigh show
fe80::224:21ff:fe00:1 dev eth0 lladdr 00:01:23:45:67:89 router REACHABLE
fe80::224:21ff:fe67:89ab dev eth0  FAILED
```

  - Wichtige Flags
    - REACHABLE erreichbar, vor kurzem benutzt
    - STALE war erreichbar, länger nicht genutzt
    - INCOMPLETE Neighbor Discovery aktiv
    - FAILED Neighbor Discovery erfolglos (Host nicht am Link)

Werkzeuge für Fehlersuche Router/Neighbor Advertisements

- **Werkzeug `tcpdump`**

```
# tcpdump -n icmp6
```

- **Beispiele**
  - Router Advertisement

```
fe80::224:21ff:fe00:1 > ff02::1: icmp6: router advertisement
 (chlim=64, router_ltime=30, reachable_time=0, retrans_time=0)
 (prefix info: LAR valid_ltime=2592000, preffered_ltime=604800,
  prefix=2001:db8:0:1::/64)
 (src lladdr:0:24:21:0:0:1)
 (len 88, hlim 255)
```

  - Neighbor Solictation

```
2001:db8:0:1:224:21ff:fe01:2345 > ff02::1:ff00:10: icmp6: neighbor
sol: who has 2001:db8:0:1::10
 (src lladdr:0:24:21:1:23:45)
 (len 32, hlim 255)
```

IPv6-Anbindung

IPv6-Anbindung

Typ	Tunneltyp	IPv4-Adresse	IPv6-Adresse	IPv6-Netz	Anbieter bzw. Methode
Nativ			Statisch	ja	ISP
Tunnel	IPv6 in IPv4	Statisch	Statisch	ja	SixXS, gogo6, Hurricane Electric und andere
Tunnel	IPv6 in IPv4	Dynamisch	Statisch	ja	SixXS (Heartbeat) gogo6 (v6anyv4)
Tunnel	IPv6 in IPv4	Statisch	Statisch	ja	6to4
Tunnel	IPv6 in IPv4	Dynamisch	Dynamisch	ja	6to4
Tunnel	IPv6 in UDP in IPv4	Stat./Dyn.	Statisch	ja	SixXS (AYIYA) gogo6 (v6udpv4)
Tunnel	IPv6 in UDP in IPv4	Statisch	Statisch	nein	Teredo (Miredo)
Tunnel	IPv6 in UDP in IPv4	Dynamisch	Dynamisch	nein	Teredo (Miredo)

IPv6-Anbindung
Nativ

- **ISP routet direkt Präfixe über die Anbindung**
- **Weiterleitung in das Intranet durch Aufsplittung**
  - direkt: diverse /64-Präfixe
  - über Router: hierarchisch oder Routing-Protokoll
- **Verteilung der Präfixe an Clients und Server**
  - durch Router Advertisment Daemon
  - statische Konfiguration (auch zusätzlich möglich)

IPv6-Anbindung
Tunnel

- **IPv6-ISP stellt einen Präfix zur Verfügung**
  - /64-Präfix für einzelne Clients
  - /48 bzw. /56-Präfix für Standorte
- **Automatischer Präfix durch IPv4-Adresse**
  - /48 durch 6to4
- **Lokaler Tunnelendpunkt terminiert Tunnel**
  - Getunnelte IPv6-Pakete werden ausgepackt
  - Lokales Netz transportiert IPv6 nativ
    - Weitere Tunnels (z.B. Inter-Standort) sind möglich

IPv6-Anbindung
Tunnel ISP SixXS

- **Hauptstandort: Niederlande**
  - PoPs in etlichen Ländern verfügbar
    - Deutschland: aktuell 5 (seit 20.05.2009)
- **URL: http://www.sixxs.net/**
- **Möglicher Subnetz-Präfix: /48**
- **Mögliche Tunnelmethoden**
  - Statisch
    - stabile globale IPv4-Adresse bei SixXS hinterlegt (Proto 41)
  - Dynamisch (bei temporärer IPv4-Adresse)
    - Client-Daemon "aiccu" im Heartbeat-Modus (Proto 41 & UDP/3740)
  - Aus dem Intranet heraus (IPv6-in-UDP-in-IPv4)
    - Client-Daemon "aiccu" im AYIYA-Modus (UDP/5072)

IPv6-Anbindung
Tunnel ISP Gogo6 (Hexago)

- **Standort: Kanada**
- **URL:**
  - http://www.gogo6.com/
- **Möglicher Subnetz-Präfix: /56**
- **Mögliche Tunnelmethoden**
  - Statisch
    - Client-Daemon "gogoc" im v6v4-Modus (Proto 41)
  - Dynamisch (bei temporärer IPv4-Adresse)
    - Client-Daemon "gogoc" im v6anyv4-Modus (Proto 41 & UDP/3653)
  - Aus dem Intranet heraus (IPv6-in-UDP-in-IPv4)
    - Client-Daemon "gogoc" im v6udpv4-Modus (UDP/3653)

IPv6-Anbindung
Tunnelbroker Hurricane Electric

- **Standort: USA**
- **URLs:**
  - http://www.he.net/
  - http://tunnelbroker.net/
- **Möglicher Subnetz-Präfix: /48**
- **Mögliche Tunnelmethoden**
  - Statisch
    - stabile globale IPv4-Adresse hinterlegt

IPv6-Anbindung
Teredo

- **URLs:**
  - http://de.wikipedia.org/wiki/Teredo
- **Kein Subnetz möglich**
- **Mögliche Tunnelmethoden**
  - IPv6-in-UDP-in-IPv4 (sog. Teredo-Bubbles, 3544/udp)
- **Linux-Client: miredo-client**

Demo IPv6-Anbindung

- **Nativ lokaler ISP**
  - incl. Fehlersuche mit "ip neigh", "tcpdump"
- **Tunnel zu SixXS (AYIYA)**
- **Tunnel zu gogo6 (v6udpv4)**
- **Tunnel via Teredo**

IPv6-aktive Services

IPv6-aktive Services
Secure Shell (SSH)

- Aktuelle Linux-Distributionen: SSH bereits IPv6-fähig
- Konfiguration: `/etc/ssh/sshd_config`

```
AddressFamily   any
ListenAddress   ::
Port            22
```

- Prüfung

```
# netstat -nlpt | grep ssh
tcp        0      0 0.0.0.0:22      0.0.0.0:*         LISTEN      2425/sshd
tcp        0      0 :::22           :::*              LISTEN      2425/sshd
```

- Einschränkung bei tcp_wrapper: `/etc/hosts.allow`

```
sshd:           [2001:db8:0:1::]/64
```

IPv6-aktive Services
Apache Webserver (httpd)

- Apache Webserver ist nativ IPv6-fähig ab 2.x
- Konfiguration: `/etc/httpd/conf/httpd.conf`

```
Listen [2001:a60:9002:1::186:2]:80
<VirtualHost [2001:a60:9002:1::186:2]:80 212.18.21.186:80>
```

- Prüfung

```
# netstat -nlpt | grep httpd
tcp        0      0 2001:a60:9002:1::186:2::80  :::*  LISTEN   2426/httpd
```

- Test:
  - http://[2001:a60:9002:1::186:2]/
- Eintrag in der DNS-Zone

```
mirrors.bieringer.de    IN AAAA    2001:a60:9002:1::186:2
```

Demo IPv6-Services

- **Zugriff auf SSH von extern über IPv6**
- **Zugriff auf Apache von extern über IPv6**

Absicherung & Firewalling

Angriffsrisiko

- **Angriffsrisiko für verschiedene Anbindungen**

Protokoll:	**IPv4**			**IPv6**
Verbindung:	direkte Anbindung	Lokales Netzwerk kein Port-Forwarding	Lokales Netzwerk Port-Forwarding	
Angriff auf:				
Netzwerk-Stack	mittel	niedrig	mittel	hoch
Server-Applikationen	hoch	n/a	hoch	sehr hoch

- Netzwerk Stack von IPv6 noch nicht so ausgereift
- Firewalling bei IPv6 auf Client und Server sehr wichtig!
  - Kein impliziter Schutz durch NAT vorhanden!
  - System ist vom Internet aus direkt ansprechbar!

Unnötige offene Ports

- **Oft unnötige offene Ports durch Standardinstallation**
- **Prüfung (als Benutzer „root“)**

```
# netstat -nlptu
# lsof -n -i | grep -v '\->'
# rpcinfo -p
```

- **Unnötige Dienste abschalten**
  - Fedora / Red Hat Enterprise Linux:

```
# chkconfig DIENST off; service DIENST stop
```

Gefährdete offene Ports

- **Beispiel: Fedora 10 Standardinstallation**

```
Aktive Internetverbindungen (Nur Server)
Pro  RQ SQ Local Address     Foreign A  State   PID/Program name
tcp  0  0  0.0.0.0:111       0.0.0.0:*  LISTEN  2084/rpcbind
tcp  0  0  0.0.0.0:22        0.0.0.0:*  LISTEN  2425/sshd
tcp  0  0  127.0.0.1:631     0.0.0.0:*  LISTEN  2561/cupsd
tcp  0  0  0.0.0.0:34138     0.0.0.0:*  LISTEN  2097/rpc.statd
tcp  0  0  :::111            :::*       LISTEN  2084/rpcbind
tcp  0  0  :::22             :::*       LISTEN  2425/sshd
tcp  0  0  ::1:631           :::*       LISTEN  2561/cupsd
udp  0  0  0.0.0.0:44082     0.0.0.0:*          2474/avahi-daemon
udp  0  0  0.0.0.0:68        0.0.0.0:*          4057/dhclient
udp  0  0  0.0.0.0:987       0.0.0.0:*          2084/rpcbind
udp  0  0  0.0.0.0:53084     0.0.0.0:*          2097/rpc.statd
udp  0  0  0.0.0.0:5353      0.0.0.0:*          2474/avahi-daemon
udp  0  0  0.0.0.0:1001      0.0.0.0:*          2097/rpc.statd
udp  0  0  0.0.0.0:111       0.0.0.0:*          2084/rpcbind
udp  0  0  0.0.0.0:631       0.0.0.0:*          2561/cupsd
udp  0  0  :::987            :::*               2084/rpcbind
udp  0  0  :::111            :::*               2084/rpcbind
```

  - Absicherung notwendig bzw. prüfen!
    - IPv6: SSH (22) und rpc-bind (111, 987)

Absicherungsmöglichkeiten

- **Bindung der Dienste an lokale (private) Adressen**
- **Diensteigene ACLs aktivieren**
- **tcp_wrapper (wenn durch Dienst unterstützt)**
- **Lokales Firewalling**
- **Firewalling am Gateway**

Alle Möglichkeiten nutzen – sicher ist sicher!

Absicherungsmöglichkeiten ohne Firewalling

- **Bindung der Dienste an lokale Adressen**
  - Dienste (Beispiele): samba, squid, httpd, cups
  - IPv4: z.B. 192.168.1.x
  - IPv6: Site-Local oder Unique-Local
  - Prüfung mit: `netstat -nlptu`
- **Diensteigene ACLs aktivieren**
  - Dienste (Beispiele): samba, squid, httpd, cups
  - Prüfung: Verbindungsaufbau von nicht erlaubten Adressen
- **tcp_wrapper (wenn durch Dienst unterstützt)**
  - Dienste (Beispiele): rpc-Dienste, openssh
  - Datei `/etc/hosts.allow` anpassen
  - Prüfung: Verbindungsaufbau von nicht erlaubten Adressen

Linux-Firewalling allgemein

- **Paketfilter im aktuellen Linux-Kernel: „netfilter“**
  - URL: http://www.netfilter.org/
  - ersetzte 2001 ipchains ab Kernel 2.3.x
  - Stateless IPv6-Unterstützung seit Kernel 2.4.x (Januar 2001)
    - Nur Einschränkungen auf TCP-Flag- & Portebene möglich
  - Stateful IPv6-Firewalling ab Kernel 2.6.20 (seit Februar 2007)
  - Benutzer-Werkzeug: *iptables* (IPv4) bzw. *ip6tables* (IPv6)
- **Filter-Möglichkeiten**
  - Layer 2 bis 4
    - seit 2.6.20 *connection tracking* bei IPv6 vorhanden
      - Red Hat Enterprise Linux 4.x und 5.x: nur stateless!

Absicherungsmöglichkeiten mit Firewalling

- **Aktivieren von lokalem Firewalling auf jedem Client**
  - INPUT-Chain
    - Vorsicht bei Filterung von ICMPv6
      - verhindert unter Umständen Neighbor/Router/PMTU-Discovery
- **Aktivieren von Gateway-Firewalling auf jedem Router**
  - FORWARD-Chain

Firewalling Regelsatzgeneratoren

- **Distributionseigene Werkzeuge**
  - Red Hat / CentOS: system-config-securitylevel-tui
  - Fedora: system-config-firewall (ab FC6 auch für IPv6)
- **Unabhängige Regelsatzgeneratoren für IPv4 und IPv6**
  - *fwbuilder* seit 3.0, GUI
    - URL: http://www.fwbuilder.org/
  - *Shorewall Firewall* seit 4.2.4, CLI
    - URL: http://www.shorewall.net/
  - Aufbau eigenes Regelwerks, CLI
  - Hilfsskripte: *ip-firewalling*, CLI
    - URLs:
      - ftp://ftp.aerasec.de/pub/linux/repository/public/redhat/enterprise/4/i386/
      - ftp://ftp.bieringer.de/pub/linux/IPv6/ip-firewalling/ (Updates)

Beispiel für IPv6 netfilter Regelwerk

- **Minimales Regelwerk (für *ip6tables-restore*):**

```
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p ipv6-icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -j REJECT --reject-with icmp6-adm-prohibited
-A FORWARD -j REJECT --reject-with icmp6-adm-prohibited
COMMIT
```

- **Erlauben eingehend SSH von überall her:**

```
-A INPUT -m state --state NEW -m tcp -p tcp --syn --dport 22 -j ACCEPT
```

- **Mehr Tipps stehen zur Verfügung unter:**
  http://www.tldp.org/HOWTO/Linux+IPv6-HOWTO/chapter-firewalling-security.html

Demo IPv6-Firewalling

- **Fedora / Red Hat / CentOS: /etc/sysconfig/ip6tables**

IPv6 QoS unter Linux

IPv6 QoS unter Linux / Filter

- **Schnittstelle, Quelle, Ziel, IP-Version, Protokoll, Port**
  - Beispiel: eth1, IPv6, TCP (6), ssh (22)
    - Filterdefinition direkt in “tc”

```
# tc filter add dev eth1 parent 1: protocol ipv6 u32 match ip6 protocol 6 0xff
match ip6 dport 22 0xffff flowid 1:2
```

    - Markierung im Linux-Firewalling

```
# ip6tables -A POSTROUTING -t mangle -p tcp --dport 22 -j MARK --set-mark 32
# tc filter add dev eth1 parent 1: protocol ipv6 handle 32 fw flowid 1:4
```

- **Flow Label (Beispiel: 0x21)**

```
# tc filter add dev eth1 parent 1: protocol ipv6 u32 match ip6 flowlabel 0x21
0x3ffff flowid 1:3
```

- **Traffic Class (Beispiel: 0x11)**

```
# tc filter add dev eth1 parent 1: protocol ipv6 u32 match u32 0x01100000
0x0ff00000 at 0 flowid 1:3
```

Achtung: tc filter ... match ip6 priority ...“ erzeugt falschen „match“ (Red Hat Bugzilla 584913)

Demo IPv6 QoS (1)

- **Konfigurieren von Filter**
  - Filter für *Flow Label* und *Traffic Class*
- **Trigger eines “Match” mit ping6**
  - Flow Label 0x21 (33)

```
# ping6 -c 2 -F 21 DESTINATION
```

  - Traffic Class 0x11 (17)

```
# ping6 -c 2 -Q 11 DESTINATION
```

Achtung: aktuell interpretiert ping6 die Eingabewerte für -F und -Q als Hexadezimal

- **Testen eines “Match”**

```
# tc -s filter show dev DEVICE | grep -B 1 match

filter parent 1: protocol ipv6 pref 49149 u32 fh 803::800 order 2048 key ht
803 bkt 0 flowid 1:3  (rule hit 5 success 2)
  match 01100000/0ff00000 at 0 (success 2 )

filter parent 1: protocol ipv6 pref 49150 u32 fh 802::800 order 2048 key ht
802 bkt 0 flowid 1:3  (rule hit 3 success 2)
  match 00000021/0003ffff at 0 (success 2 )
```

Demo IPv6 QoS (2)

- **Konfigurieren von Filter**
  - TCP-Port 5001 (0x1389) auf 50 MBit/s limitiert
- **Test auf Limitierung mit iperf abhängig vom TCP-Port**
  - Server

```
# iperf -V -s -p 5001
# iperf -V -s -p 5002
```

  - Client

```
# iperf -V -c DESTINATION -p 5001
# iperf -V -c DESTINATION -p 5002
```

- **Testen auf Wirksamkeit**

```
# tc -s filter show dev DEVICE | grep -B 1 match

filter parent 1: protocol ipv6 pref 49151 u32 fh 801::800 order 2048 key ht
801 bkt 0 flowid 1:2  (rule hit 18873 success 1846)
  match 00000600/0000ff00 at 4 (success 18870 )
  match 00001389/0000ffff at 40 (success 1846 )
```

IPv6 in virtualisierten Umgebungen (libvirt)

Status von libvirt / virt-manager

- **Fedora 12 mit libvirt 0.7.1 und virt-manager 0.8.2**
  - IPv6 ist immer noch ein Stiefkind
  - virt-manager verlangt IPv4-Setup für VM 
  - libvirt erzeugt KEINE passenden ip6tables-Einträge für Bridges (Red Hat Bugzilla 567124) 
    - Handarbeit oder Skript notwendig, welches *iptables* Regeln zu *ip6tables* Regeln kopiert
  - Kernel hat unerwartet Geschwindigkeitsprobleme bei Benutzung von “virtio” als Treiber und einer VM als Router (RH-BZ 571234)
    - Client (VM) <--> Router (VM) <--> Server (VM) 

Weitere Informationen

IPv6 & Linux bezogene Information

- ***Linux IPv6 HOWTO***
  - Schwerpunkt: ausgiebige Information über IPv6 in Linux
    http://www.tldp.org/HOWTO/Linux+IPv6-HOWTO/ (nur English)
    http://mirrors.bieringer.de/ (en, de, fr, it) 
  - URLs zu allen Übersetzungen und weiterführende Informationen
    http://www.bieringer.de/linux/IPv6/
- ***Current Status of IPv6 Support for Networking Applications***
  - IPv6-Status von netzwerkfähigen Applikationen
    http://www.deepspace6.net/docs/ipv6_status_page_apps.html

Kontakt-Information

peter@deepspace6.net

http://www.deepspace6.net/

pb@bieringer.de

http://www.bieringer.de/pb/

http://www.bieringer.de/linux/IPv6/

http://mirrors.bieringer.de/

Vielen Dank für die Teilnahme!

Fragen & Antworten

Tutorial mit Notizen ist als PDF per E-Mail erhältlich!

Dankeschön an

Johannes Endres, c't (Einladung)

AERAsec Network Services and Security GmbH (IPv6 Web- & E-Mail Hosting)

IPv6 mit Linux
(Einführung)

Tutorial

Dr. Peter Bieringer
Deep Space 6
peter@deepspace6.net
http://www.deepspace6.net/

IPv6-Kongress
Frankfurt/Main, Deutschland
20. - 21. Mai 2010
http://www.ipv6-kongress.de/

Inhalt

IPv6 mit Linux (Einführung)

- IPv6-Konfiguration
- Fehlersuche
- IPv6-Anbindung
- IPv6-aktive Services
- IPv6-Firewalling
- IPv6 QoS
- IPv6 in virtualisierten Umgebungen (libvirt & kvm)

Über mich

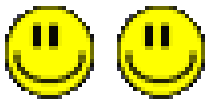
- **Wohnhaft in München (Deutschland)**
- **Beschäftigt als Netzwerkspezialist bei *Venyon GmbH***
  - Aufbau kleiner Rechenzentren für OverTheAir (OTA)-Personalisierung von NFC-fähigen Mobiltelefonen
- **Mitbegründer und Kernmitglied von *Deep Space 6***
- **Autor des "Linux IPv6 HowTo"**
- **Mitautor des Buches "Linux im Netz"**
  - Grundlagen von TCP/IP incl. IPv6, DNS, DHCP

Dr. Peter Bieringer – IPv6 mit Linux (Tutorial) – IPv6-Kongress – 20. - 21. Mai 2010, Frankfurt/Main, Deutschland 21.05.10 19:06:35 3

Abkürzungen:
OTA: Over The Air
NFC: Near Field Communication

Meine Internet- & IPv6-Historie

- **1993: Erster Kontakt mit dem Internet (Univ., SunOS)**
- **1996: Erste Erfahrungen mit IPv6 und Linux**
- **1997: *IPv6 & Linux - HowTo*, *initscripts-ipv6***
- **1999: *IPv6 & Linux - Current Status***
- **2001: *Linux IPv6 HOWTO*, *ipv6calc***
- **2002: Mitbegründer von *Deep Space 6***

inzwischen 14 Jahre IPv6-Erfahrung! 

Titel durch Klicken hinzufügen

IPv6-Konfiguration von Linux

IPv6-Konfiguration

- **Adressen**
  - Automatisch
    - Link-Local
  - Statisch
    - Site-Local, Unique-Local, Global
  - Dynamisch
    - Router Advertisement Daemon
    - DHCPv6
- **Routing**
  - Statisch
  - via Autokonfiguration
  - Dynamisch

Aktuelle Routing-Daemons für Linux: quagga, bird

IPv6 in Linux Voraussetzungen

- **IPv6 im Linux-Kernel bereits aktiviert**
  - Test: Datei existiert: `/proc/net/if_inet6`
- **IPv6 im Linux-Kernel aktivieren**
  - Modul laden

```
# modprobe ipv6
```

  - Prüfen

```
# lsmod |grep -w 'ipv6'
ipv6                  223810  22 ip6t_REJECT,nf_conntrack_ipv6
# cat /proc/net/if_inet6
00000000000000000000000000000001 01 80 10 80       lo
fe8000000000000022421fffe012345 02 40 20 80     eth0
```

- **IPv6-Forwarding aktivieren (bei Bedarf)**

```
# sysctl -w net.ipv6.conf.all.forwarding=1
```


Vollständiges Deaktivieren von IPv6 bei Bedarf möglich durch folgende Moduloption:

```
option ipv6 disable=1
```

IPv6-Adresskonfiguration Ansicht

- **Werkzeuge `ip` oder `ifconfig`**

```
# ip -6 addr show dev INTERFACE
# ifconfig INTERFACE
```

- **Beispiele**

```
# ip -6 addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_ fast qlen 100
    inet6 fe80::224:21ff:fe01:2345/64 scope link
       valid_lft forever preferred_lft forever


# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:24:21:01:23:45
          inet6 addr: fe80::224:21ff:fe01:2345/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          ...
```


IPv6-Adresskonfiguration Statische Zuweisung

Werkzeuge `ip` oder `ifconfig`

```
# ip -6 addr add IPV6ADDRESS/PREFIXLENGTH dev INTERFACE
# ifconfig INTERFACE inet6 add IPV6ADDRESS/PREFIXLENGTH
```

Beispiele

```
# ip -6 addr add 2001:db8:0:1::1/64 dev eth0
# ifconfig eth0 inet6 add 2001:db8:0:1::1/64
```

Ergebnis

```
# ip -6 addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qlen 1000
    inet6 2001:db8:0:1::1/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::224:21ff:fe01:2345/64 scope link
       valid_lft forever preferred_lft forever
```

```
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:24:21:01:23:45
          inet6 addr: 2001:db8:0:1::1/64 Scope:Global
          inet6 addr: fe80::224:21ff:fe01:2345/64 Scope:Link
```


IPv6-Adresskonfiguration Router Advertisements

- **Konfigurationsdatei auf Router: `/etc/radvd.conf`**

```
interface eth0
{
   AdvSendAdvert on;
    ...
   prefix 2001:db8:0:1::/64
   {
      ...
      AdvPreferredLifetime 86400;
      AdvValidLifetime 604800;
   };
```

- **Ergebnis auf Client nach Eintreffen eines RAs**

```
# ip -6 addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qlen 1000
    inet6 2001:db8:0:1::224:21ff:fe01:2345/64 scope global
       valid_lft 604711sec preferred_lft 86311sec
    inet6 fe80::224:21ff:fe01:2345/64 scope link
       valid_lft forever preferred_lft forever
```


Mit "ifconfig" wird nur die Adresse, jedoch nicht die Lebenszeit angezeigt:

```
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:24:21:01:23:45
          inet6 addr: 2001:db8:0:1:224:21ff:fe01:2345/64 Scope:Global
          inet6 addr: fe80::224:21ff:fe01:2345/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

IPv6-Adresskonfiguration Privacy

- **Aktivieren von IPv6-Privacy gemäß RFC 4941 (3041)**
  - Konfiguration

```
# sysctl -w net.ipv6.conf.eth0.use_tempaddr=2
```

  - Restart der Schnittstelle notwendig

```
# ip link set dev eth0 down
# ip link set dev eth0 up
```

- **Ergebnis auf Client nach Eintreffen eines RAs**

```
# ip link show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 ...
    link/ether 00:24:21:01:23:45 brd ff:ff:ff:ff:ff:ff
# ip -6 addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qlen 1000
    inet6 2001:db8:0:1:8992:3c03:d6e2:ed72/64 scope global secondary dynamic
       valid_lft 604711sec preferred_lft 86311sec
    inet6 2001:db8:0:1::224:21ff:fe01:2345/64 scope global
       valid_lft 604711sec preferred_lft 86311sec
    ...
```


IPv6-Routing Ansicht

Werkzeuge `ip` oder `route`

```
# ip -6 route show [dev INTERFACE]
# route -n -A inet6
```

Beispiele

```
# ip -6 route show dev eth0
fe80::/64 dev eth0  metric 256
      expires 21296936sec  mtu 1500 advmss 1440 hoplimit 4294967295
2001:db8:0:1::/64 dev eth0  proto kernel  metric 256
      expires 604545sec  mtu 1500 advmss 1440 hoplimit 4294967295
default via fe80::224:21ff:fe00:1 dev eth0  proto kernel  metric 1024
      expires 1637sec mtu 1500 advmss 1440 hoplimit 64

# route -n -A inet6 | grep -w eth0
2001:db8:0:1::/64      ::                      UA    256    94    0 eth0
fe80::/64              ::                      U     256    0     0 eth0
::/0                   fe80::224:21ff:fe00:0001 UGDA  1024   36    0 eth0
ff00::/8               ::                      U     256    0     0 eth0
```


Wichtige Flags bei "route":

U Route ist aktiv (Up)
A Route ist durch ein Advertisement gesetzt worden
G Route benutzt ein Gateway
D Route ist von einem Daemon, ICMP redirect (IPv4) oder Router Advertisement (IPv6) generiert worden

IPv6-Routing Konfiguration

Werkzeuge `ip` oder `route`

```
# ip -6 route add IPV6NETWORK/PREFIXLENGTH via IPV6ADDRESS
# route -A inet6 add IPV6NETWORK/PREFIXLENGTH gw IPV6ADDRESS
```

Beispiele

```
# ip -6 route add 2001:db8:0:2::/64 via 2001:db8:0:1::1
# route -A inet6 add 2001:db8:0:2::/64 gw 2001:db8:0:1::1
```

Ergebnis

```
# ip -6 route show dev eth0 | grep "via 2001"
2001:db8:0:2::/64 via 2001:db8:0:1::1 dev eth0  proto kernel
      metric 1024 expires 1637sec mtu 1500 advmss 1440 hoplimit 64

# route -n -A inet6 | grep -w eth0 | grep "2001" | grep "G"
2001:db8:0:2::/64   2001:db8:0:1::1      UG    1024   0        0 eth0
```


IPv6-Adresskonfiguration DHCPv6

- **Server**
  - Software: ISC-DHCP (4.1.1), dibbler (0.7.3)
  - Verteilung von
    - IPv6-DNS-Server
    - DNS Search List
- **Client**
  - Software: dhclient (4.1.1), z.B. in Fedora 12
  - Unterstützt
    - IPv6-DNS-Server
    - DNS Search List

dhcpv6 (1.2.0) – Weiterentwicklung gestoppt seit 09/2009 

Konfigurationsbeispiele für DHCP-Server

ISC BIND:

```
subnet6 2001:db8:0:1::/64 {
     range6 2001:db8:0:1::129 2001:db8:0:1::254;
     option dhcp6.name-servers fec0:0:0:1::1;
     option dhcp6.domain-search "ipv6.local";
}
```

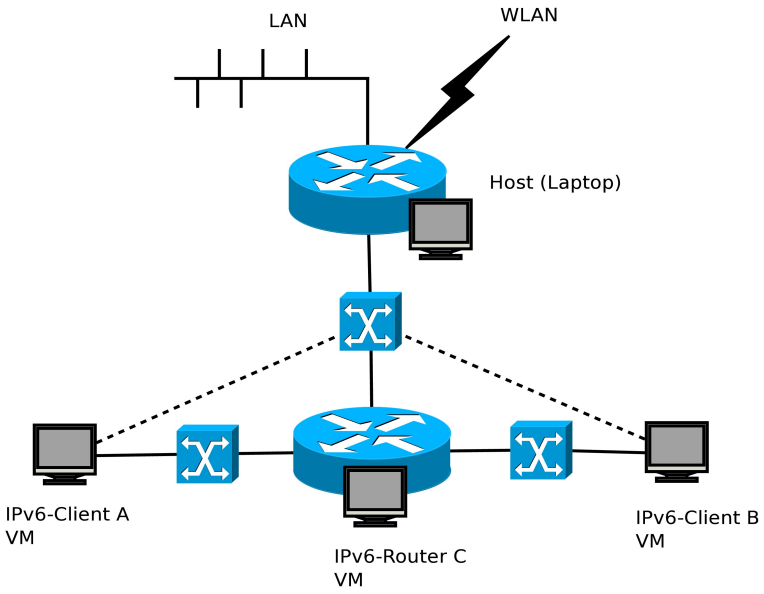
Dibbler:

```
iface "eth1" {
 class {
   pool 2001:db8:0:1::/64
 }
option dns-server fec0:0:0:1::1
option domain ipv6.local
```

Zur Vermeidung von 2 Adressen mit gleichem Prefix (1x Autokonfiguration, 1x DHCP) radvd umstellen, daß nur die Link-Local-Adresse vom Router verteilt wird:

```
interface INTERFACE
{
     AdvSendAdvert on;
     MinRtrAdvInterval 30;
     MaxRtrAdvInterval 100;
}
```

Demo Aufbau



IPv6-Client A & B: Prefixe/Adressen durch DHCPv6 bzw. Router-Advertisements vom IPv6-Router C

IPv6-Router C betreibt:
•dhcpv6
•Router Advertisment Daemon
•Caching Nameserver

Laptop betreibt:
•IPv4 Hiding NAT für IPv6-VMs

Demo IPv6 DHCP

- **IPv6-only Client-Konfiguration**
  - Fedora 12: /etc/sysconfig/network-scripts/ifcfg-INTERFACE
    - Statische IPv6-Adresse vom DHCP-Server

DHCLIENTARGS="-6"

    - Temporäre IPv6-Adresse vom DHCP-Server

DHCLIENTARGS="-6 -T"

  - Test mit Browser: http://www.ipv6.bieringer.de/ (IPv6 only)

Dr. Peter Bieringer – IPv6 mit Linux (Tutorial) – IPv6-Kongress – 20. - 21. Mai 2010, Frankfurt/Main, Deutschland 21.05.10 19:06:35 16

Konfigurationsbeispiele für ISC DHCP-Client

Einfacher Aufruf im Debugmodus:

```
# dhclient -d -6 INTERFACE
```

Aufruf erzeugt durch ifup-Skript (Fedora)

```
/sbin/dhclient -6 -1 -q -lf /var/lib/dhclient/dhclient-eth1.leases -pf
/var/run/dhclient-eth1.pid eth1
```

Achtung: für Experimente ist es sehr hilfreich, die Lease-Datei zwischendurch zu löschen.

Automatische Registierung des Hostnamens im DNS (AAAA & PTR) ist möglich durch:
`DHCLIENTARGS="$DHCLIENTARGS -F $HOSTNAME"`

Titel durch Klicken hinzufügen

Fehlersuche

Werkzeuge für Fehlersuche Verbindungstest

Verbindungstest mit `ping6` (analog zu IPv4)

```
$ ping6 -I INTERFACE IPV6LINKLOCALADDRESS
$ ping6 HOSTNAME|IPV6ADDRESS
```

Beispiele

- Link-Local (Angabe der Schnittstelle notwendig)

```
$ ping6 -I eth0 -c 1 fe80::224:21ff:fe01:2345
PING fe80::224:21ff:fe01:2345 from fe80::224:21ff:fe00:1 eth0: 56 data bytes
64 bytes from fe80::224:21ff:fe01:2345: icmp_seq=0 hops=64 time=445 usec
```

- Global

```
$ ping6 -c 1 2001:db8:0:1::224:21ff:fe01:2345
PING 2001:db8:0:1::224:21ff:fe01:2345 from 2001:db8:0:1::1 eth0: 56 data bytes
64 bytes from 2001:db8:0:1::224:21ff:fe01:2345: icmp_seq=0 hops=64 time=445
usec
```


"ping6" an link-local oder Multicast-Adressen ohne Angabe des Interfaces (-I INTERFACE) scheitert, da nicht eindeutig (es kann keine Routingtabelle zu Hilfe genommen werden):

```
$ ping6 -c 1 fe80::224:21ff:fe01:2345
connect: Invalid argument
```

Werkzeuge für Fehlersuche
Maximum Transfer Unit

- **MTU durch Tunnels bzw. Transport verringert**

IPv4	MTU	IPv4 (-20)	TCP-MSS (-20)	UDP ICMP-ECHO (-8)
Ethernet	1500	1480	1460	1472
PPPoE (-8)	1492	1472	1452	1464
IPv6	**MTU**	**IPv6 (-40)**	**TCP-MSS (-20)**	**UDP ICMP-ECHO (-8)**
Ethernet	1500	1460	1440	1452
PPPoE (-8)	1492	1452	1432	1444
IPv6 in IPv4 über Ethernet (-20)	1480	1440	1420	1432
IPv6 in IPv4 über PPPoE (-28)	1472	1432	1412	1424
IPv6 in UDP in IPv4 über Ethernet (-28)	1472	1432	1412	1424
IPv6 in UDP in IPv4 über PPPoE (-36)	1464	1424	1404	1416
Minimum gemäß RFC	1280	1240	1220	1232

MTU: Maximum Transfer Unit
PPPoE: Point-to-Point-Protocol over Ethernet
MSS: Maximum Segment Size (maximale Nutzdaten) bei TCP

Werkzeuge für Fehlersuche Maximum Transfer Unit

- **Störung der PTMU-Discovery bei Blockade von ICMP**
- **Test**
  - ICMP ECHO Payload 1452 = MTU 1500

```
$ ping6 -c 1 -M do -s 1452 www.six.heise.de
PING www.six.heise.de(www.six.heise.de) 1452 data bytes
1452 bytes from www.six.heise.de: icmp_seq=1 ttl=57 time=76.8 ms
```

  - ICMP ECHO Payload 1453 = MTU 1501

```
$ ping6 -c 1 -M do -s 1453 www.six.heise.de
PING www.six.heise.de(www.six.heise.de) 1453 data bytes
From 2001:db8:0:1:224:21ff:fe00:1 icmp_seq=1 Packet too big: mtu=1500
```

- **TCP-MSS “Clamping” (Klammerung) mit ip6tables**

```
# ip6tables -t mangle -I FORWARD  -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --set-mss 1404
# ip6tables -t mangle -I OUTPUT   -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --set-mss 1404
# ip6tables -t mangle -I INPUT    -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --set-mss 1404
```


Falls Verbindungsprobleme beim Websurfen ins Internet auftreten, kann ein MTU (Maximum Transfer Unit) Problem – verursacht durch einen Tunnel zwischendrin – vorliegen, welcher "ICMP Packet Too Big" nicht zurücksendet (bzw. diese von einer Firewall gefiltert werden).

"-M do" ist die Option, um "Don't Fragment" zu aktivieren

Werkzeuge für Fehlersuche Routenverfolgung

- **Routenverfolgung mit `traceroute` (analog zu IPv4)**
  - Neue Versionen von traceroute unterstützen zusätzlich IPv6

```
$ traceroute -6 HOSTNAME|IPV6ADDRESS
```

- **Beispiel**

```
$ traceroute -6 -q 1 www.bieringer.de
traceroute to www.bieringer.de (2001:a60:9002:1::186:3), 30 hops max, 80 byte
packets
 1  2001:db8:0:1::1 (2001:db8:0:f101::1)  0.249 ms
 2  gw-24.muc-02.de.sixxs.net (2001:a60:f000:17::1)  43.743 ms
 3  2001:a60:0:30::1 (2001:a60:0:30::1)  48.427 ms
 4  www.bieringer.de (2001:a60:9002:1::186:3)  48.427 ms
```


Falls Verbindungsprobleme beim Websurfen ins Internet auftreten, kann ein MTU (Maximum Transfer Unit) Problem – verursacht durch einen Tunnel zwischendrin – vorliegen, welcher "ICMP Packet Too Big" nicht zurücksendet (bzw. diese von einer Firewall gefiltert werden).

Zur Feststellung, an welchem Router die MTU-Reduzierung stattfindet, kann "tracepath6" benutzt werden:

```
$ tracepath6 www.six.heise.de
 1?: [LOCALHOST]                      pmtu 1500
 1:  2001:db8:0:1::1                   1.047ms
 1:  2001:db8:0:1::1                   0.306ms
 2:  2001:db8:0:1::1                   0.279ms pmtu 1472
 2:  gw-24.muc-02.de.sixxs.net        67.895ms
 2:  gw-24.muc-02.de.sixxs.net        67.261ms
 3:  2001:a60:0:30::1                 70.776ms
 4:  ge3-4.c2.m.de.plusline.net       77.778ms asymm  8
 5:  2a02:2e0:1::51                   77.322ms asymm  7
 6:  2a02:2e0:1::55                   76.607ms
 7:  te6-1.c13.f.de.plusline.net      76.953ms
 8:  www.six.heise.de                 76.075ms reached
     Resume: pmtu 1472 hops 8 back 57
```

Achtung, "tracepath6" nutzt UDP-Pakete, die evtl. an Firewalls gefiltert werden können.

Werkzeuge für Fehlersuche Router/Neighbor Discovery

Werkzeug `ip`

```
# ip -6 neigh show
```

Beispiele

```
# ip -6 neigh show
fe80::224:21ff:fe00:1 dev eth0 lladdr 00:01:23:45:67:89 router REACHABLE
fe80::224:21ff:fe67:89ab dev eth0  FAILED
```

- Wichtige Flags
  - REACHABLE erreichbar, vor kurzem benutzt
  - STALE war erreichbar, länger nicht genutzt
  - INCOMPLETE Neighbor Discovery aktiv
  - FAILED Neighbor Discovery erfolglos (Host nicht am Link)

Werkzeuge für Fehlersuche Router/Neighbor Advertisements

- **Werkzeug `tcpdump`**

```
# tcpdump -n icmp6
```

- **Beispiele**
  - Router Advertisement

```
fe80::224:21ff:fe00:1 > ff02::1: icmp6: router advertisement
 (chlim=64, router_ltime=30, reachable_time=0, retrans_time=0)
 (prefix info: LAR valid_ltime=2592000, preffered_ltime=604800,
  prefix=2001:db8:0:1::/64)
 (src lladdr:0:24:21:0:0:1)
 (len 88, hlim 255)
```

  - Neighbor Solictation

```
2001:db8:0:1:224:21ff:fe01:2345 > ff02::1:ff00:10: icmp6: neighbor
sol: who has 2001:db8:0:1::10
 (src lladdr:0:24:21:1:23:45)
 (len 32, hlim 255)
```


Für Microsoft Windows gibt es "windump" (in Verbindung mit "winpcap"):
http://www.winpcap.org/windump/

Für bessere Darstellung empfiehlt sich die Nutzung von "wireshark", hat Text- & GUI-Modus: http://www.wireshark.org/

Text-Modus:

```
# tethereal -n -V icmp6
```

Titel durch Klicken hinzufügen

IPv6-Anbindung

IPv6-Anbindung

Typ	Tunneltyp	IPv4-Adresse	IPv6-Adresse	IPv6-Netz	Anbieter bzw. Methode
Nativ			Statisch	ja	ISP
Tunnel	IPv6 in IPv4	Statisch	Statisch	ja	SixXS, gogo6, Hurricane Electric und andere
Tunnel	IPv6 in IPv4	Dynamisch	Statisch	ja	SixXS (Heartbeat) gogo6 (v6anyv4)
Tunnel	IPv6 in IPv4	Statisch	Statisch	ja	6to4
Tunnel	IPv6 in IPv4	Dynamisch	Dynamisch	ja	6to4
Tunnel	IPv6 in UDP in IPv4	Stat./Dyn.	Statisch	ja	SixXS (AYIYA) gogo6 (v6udpv4)
Tunnel	IPv6 in UDP in IPv4	Statisch	Statisch	nein	Teredo (Miredo)
Tunnel	IPv6 in UDP in IPv4	Dynamisch	Dynamisch	nein	Teredo (Miredo)

Maximale Payload von Tunnelart abhängig
Test z.B. mit ping6 -M hint do -s PACKETSIZE DESTINATION

IPv6-Anbindung
Nativ

- **ISP routet direkt Präfixe über die Anbindung**
- **Weiterleitung in das Intranet durch Aufsplittung**
  - direkt: diverse /64-Präfixe
  - über Router: hierarchisch oder Routing-Protokoll
- **Verteilung der Präfixe an Clients und Server**
  - durch Router Advertisment Daemon
  - statische Konfiguration (auch zusätzlich möglich)

IPv6-Anbindung Tunnel

- **IPv6-ISP stellt einen Präfix zur Verfügung**
  - /64-Präfix für einzelne Clients
  - /48 bzw. /56-Präfix für Standorte
- **Automatischer Präfix durch IPv4-Adresse**
  - /48 durch 6to4
- **Lokaler Tunnelendpunkt terminiert Tunnel**
  - Getunnelte IPv6-Pakete werden ausgepackt
  - Lokales Netz transportiert IPv6 nativ
    - Weitere Tunnels (z.B. Inter-Standort) sind möglich

IPv6-Anbindung
Tunnel ISP SixXS

- **Hauptstandort: Niederlande**
  - PoPs in etlichen Ländern verfügbar
    - Deutschland: aktuell 5 (seit 20.05.2009)
- **URL: http://www.sixxs.net/**
- **Möglicher Subnetz-Präfix: /48**
- **Mögliche Tunnelmethoden**
  - Statisch
    - stabile globale IPv4-Adresse bei SixXS hinterlegt (Proto 41)
  - Dynamisch (bei temporärer IPv4-Adresse)
    - Client-Daemon "aiccu" im Heartbeat-Modus (Proto 41 & UDP/3740)
  - Aus dem Intranet heraus (IPv6-in-UDP-in-IPv4)
    - Client-Daemon "aiccu" im AYIYA-Modus (UDP/5072)

PoP: Point of Presence
AYIYA: Anything In Anything
http://www.sixxs.net/tools/ayiya/

Wichtige Parameter in **`/etc/aiccu.conf`**:

```
username USER/TUNNELID
password PASSFORTUNNEL
ipv6_interface sixxs
tunnel_id Txxxxx
```

Der Tunnelmodus ist per `tunnel_id` serverseitig definiert und kann nur bei SixXS über die Weboberfläche geändert werden.

IPv6-Anbindung
Tunnel ISP Gogo6 (Hexago)

- **Standort: Kanada**
- **URL:**
  - http://www.gogo6.com/
- **Möglicher Subnetz-Präfix: /56**
- **Mögliche Tunnelmethoden**
  - Statisch
    - Client-Daemon "gogoc" im v6v4-Modus (Proto 41)
  - Dynamisch (bei temporärer IPv4-Adresse)
    - Client-Daemon "gogoc" im v6anyv4-Modus (Proto 41 & UDP/3653)
  - Aus dem Intranet heraus (IPv6-in-UDP-in-IPv4)
    - Client-Daemon "gogoc" im v6udpv4-Modus (UDP/3653)

Wichtige Parameter in **`gogoc.conf`**

```
userid=USER
passwd=PASS
server=authenticated.freenet6.net
auth_method=any
tunnel_mode=v6udpv4
```

`tunnel_mode` ist abhängig von der gewählten Methode bzw. aktuellen Anbindung auf dem Client wählbar.

Aufruf:

```
# gogoc -n -f $HOME/gogoc.conf
```

IPv6-Anbindung
Tunnelbroker Hurricane Electric

- **Standort: USA**
- **URLs:**
  - http://www.he.net/
  - http://tunnelbroker.net/
- **Möglicher Subnetz-Präfix: /48**
- **Mögliche Tunnelmethoden**
  - Statisch
    - stabile globale IPv4-Adresse hinterlegt

IPv6-Anbindung Teredo

- **URLs:**
  - http://de.wikipedia.org/wiki/Teredo
- **Kein Subnetz möglich**
- **Mögliche Tunnelmethoden**
  - IPv6-in-UDP-in-IPv4 (sog. Teredo-Bubbles, 3544/udp)
- **Linux-Client: miredo-client**

Wichtige Parameter in: /etc/miredo/miredo.conf

```
InterfaceName    teredo
ServerAddress teredo.ipv6.microsoft.com
```

Aufruf:

```
# miredo -f
```

Demo IPv6-Anbindung

- **Nativ lokaler ISP**
  - incl. Fehlersuche mit "ip neigh", "tcpdump"
- **Tunnel zu SixXS (AYIYA)**
- **Tunnel zu gogo6 (v6udpv4)**
- **Tunnel via Teredo**

Titel durch Klicken hinzufügen

IPv6-aktive Services

IPv6-aktive Services
Secure Shell (SSH)

- **Aktuelle Linux-Distributionen: SSH bereits IPv6-fähig**
- **Konfiguration: `/etc/ssh/sshd_config`**

```
AddressFamily   any
ListenAddress   ::
Port            22
```

- **Prüfung**

```
# netstat -nlpt | grep ssh
tcp        0      0 0.0.0.0:22      0.0.0.0:*         LISTEN      2425/sshd
tcp        0      0 :::22           :::*              LISTEN      2425/sshd
```

- **Einschränkung bei tcp_wrapper: `/etc/hosts.allow`**

```
sshd:           [2001:db8:0:1::]/64
```


Wichtige "netstat"-Optionen

-l Reduzierung der Anzeige auf Listen-Ports (aktive Daemons)
-p Anzeige des Prozesses
-t nur TCP-Prozesse
-u nur UDP-Prozesse
-n numerisch, keine Rückwärtsauflösung von Adressen zu Namen
(unterdrückt Timeouts bei erfolglosen Auflöseversuchen)

IPv6-aktive Services
Apache Webserver (httpd)

- Apache Webserver ist nativ IPv6-fähig ab 2.x
- Konfiguration: `/etc/httpd/conf/httpd.conf`

```
Listen [2001:a60:9002:1::186:2]:80
<VirtualHost [2001:a60:9002:1::186:2]:80 212.18.21.186:80>
```

- Prüfung

```
# netstat -nlpt | grep httpd
tcp        0      0 2001:a60:9002:1::186:2::80  :::*  LISTEN   2426/httpd
```

- Test:
  - http://[2001:a60:9002:1::186:2]/
- Eintrag in der DNS-Zone

```
mirrors.bieringer.de    IN AAAA    2001:a60:9002:1::186:2
```


Demo IPv6-Services

- **Zugriff auf SSH von extern über IPv6**
- **Zugriff auf Apache von extern über IPv6**

Absicherung & Firewalling

Angriffsrisiko

Angriffsrisiko für verschiedene Anbindungen

Protokoll:	**IPv4**			**IPv6**
Verbindung:	direkte Anbindung	Lokales Netzwerk kein Port-Forwarding	Lokales Netzwerk Port-Forwarding	
Angriff auf:				
Netzwerk-Stack	mittel	niedrig	mittel	hoch
Server-Applikationen	hoch	n/a	hoch	sehr hoch

- Netzwerk Stack von IPv6 noch nicht so ausgereift
- Firewalling bei IPv6 auf Client und Server sehr wichtig!
  - Kein impliziter Schutz durch NAT vorhanden!
  - System ist vom Internet aus direkt ansprechbar!

Unnötige offene Ports

- **Oft unnötige offene Ports durch Standardinstallation**
- **Prüfung (als Benutzer „root“)**

```
# netstat -nlptu
# lsof -n -i | grep -v '\->'
# rpcinfo -p
```

- **Unnötige Dienste abschalten**
  - Fedora / Red Hat Enterprise Linux:

```
# chkconfig DIENST off; service DIENST stop
```


Gefährdete offene Ports

Beispiel: Fedora 10 Standardinstallation

```
Aktive Internetverbindungen (Nur Server)
Pro  RQ SQ Local Address     Foreign A  State   PID/Program name
tcp  0  0  0.0.0.0:111       0.0.0.0:*  LISTEN  2084/rpcbind
tcp  0  0  0.0.0.0:22        0.0.0.0:*  LISTEN  2425/sshd
tcp  0  0  127.0.0.1:631     0.0.0.0:*  LISTEN  2561/cupsd
tcp  0  0  0.0.0.0:34138     0.0.0.0:*  LISTEN  2097/rpc.statd
tcp  0  0  :::111            :::*       LISTEN  2084/rpcbind
tcp  0  0  :::22             :::*       LISTEN  2425/sshd
tcp  0  0  ::1:631           :::*       LISTEN  2561/cupsd
udp  0  0  0.0.0.0:44082     0.0.0.0:*          2474/avahi-daemon
udp  0  0  0.0.0.0:68        0.0.0.0:*          4057/dhclient
udp  0  0  0.0.0.0:987       0.0.0.0:*          2084/rpcbind
udp  0  0  0.0.0.0:53084     0.0.0.0:*          2097/rpc.statd
udp  0  0  0.0.0.0:5353      0.0.0.0:*          2474/avahi-daemon
udp  0  0  0.0.0.0:1001      0.0.0.0:*          2097/rpc.statd
udp  0  0  0.0.0.0:111       0.0.0.0:*          2084/rpcbind
udp  0  0  0.0.0.0:631       0.0.0.0:*          2561/cupsd
udp  0  0  :::987            :::*               2084/rpcbind
udp  0  0  :::111            :::*               2084/rpcbind
```

- Absicherung notwendig bzw. prüfen!
  - IPv6: SSH (22) und rpc-bind (111, 987)

"rpc-bind" ist der IPv6-fähige Nachfolger von "portmap"

Absicherungsmöglichkeiten

- **Bindung der Dienste an lokale (private) Adressen**
- **Diensteigene ACLs aktivieren**
- **tcp_wrapper (wenn durch Dienst unterstützt)**
- **Lokales Firewalling**
- **Firewalling am Gateway**

Alle Möglichkeiten nutzen – sicher ist sicher!

Absicherungsmöglichkeiten ohne Firewalling

- **Bindung der Dienste an lokale Adressen**
  - Dienste (Beispiele): samba, squid, httpd, cups
  - IPv4: z.B. 192.168.1.x
  - IPv6: Site-Local oder Unique-Local
  - Prüfung mit: `netstat -nlptu`
- **Diensteigene ACLs aktivieren**
  - Dienste (Beispiele): samba, squid, httpd, cups
  - Prüfung: Verbindungsaufbau von nicht erlaubten Adressen
- **tcp_wrapper (wenn durch Dienst unterstützt)**
  - Dienste (Beispiele): rpc-Dienste, openssh
  - Datei `/etc/hosts.allow` anpassen
  - Prüfung: Verbindungsaufbau von nicht erlaubten Adressen

Konfigurationsbeispiele:

cups [/etc/cups/cupsd.conf]:

```
Port 192.168.1.1:631
Port [2001:db8:0:1::1]:631
Allow From [2001:db8:0:1::]/64
Deny From All
```

samba [/etc/samba/smb.conf]:

```
interfaces = 192.168.1.1/24 2001:db8:0:1::1/64
bind interfaces only = Yes
host allow = 192.168.1. 2001:db8:0:1::/64
```

tcp_wrapper [/etc/hosts.allow]:

```
sshd:     [2001:db8:0:1::]/64
rpcbind:  [::1] [2001:db8:0:1::]/64
```

Linux-Firewalling allgemein

- **Paketfilter im aktuellen Linux-Kernel: „netfilter“**
  - URL: http://www.netfilter.org/
  - ersetzte 2001 ipchains ab Kernel 2.3.x
  - Stateless IPv6-Unterstützung seit Kernel 2.4.x (Januar 2001)
    - Nur Einschränkungen auf TCP-Flag- & Portebene möglich
  - Stateful IPv6-Firewalling ab Kernel 2.6.20 (seit Februar 2007)
  - Benutzer-Werkzeug: *iptables* (IPv4) bzw. *ip6tables* (IPv6)
- **Filter-Möglichkeiten**
  - Layer 2 bis 4
    - seit 2.6.20 *connection tracking* bei IPv6 vorhanden
      - Red Hat Enterprise Linux 4.x und 5.x: nur stateless!

Absicherungsmöglichkeiten mit Firewalling

- **Aktivieren von lokalem Firewalling auf jedem Client**
  - INPUT-Chain
    - Vorsicht bei Filterung von ICMPv6
      - verhindert unter Umständen Neighbor/Router/PMTU-Discovery
- **Aktivieren von Gateway-Firewalling auf jedem Router**
  - FORWARD-Chain

Folgende ICMPv6 Typen sollten tunlichst nicht blockiert werden:
- Packet too big
- Unreachable
- HOP limit exceeded
- Parameter problem

Firewalling Regelsatzgeneratoren

- **Distributionseigene Werkzeuge**
  - Red Hat / CentOS: system-config-securitylevel-tui
  - Fedora: system-config-firewall (ab FC6 auch für IPv6)
- **Unabhängige Regelsatzgeneratoren für IPv4 und IPv6**
  - *fwbuilder* seit 3.0, GUI
    - URL: http://www.fwbuilder.org/
  - *Shorewall Firewall* seit 4.2.4, CLI
    - URL: http://www.shorewall.net/
  - Aufbau eigenes Regelwerks, CLI
  - Hilfsskripte: *ip-firewalling*, CLI
    - URLs:
      - ftp://ftp.aerasec.de/pub/linux/repository/public/redhat/enterprise/4/i386/
      - ftp://ftp.bieringer.de/pub/linux/IPv6/ip-firewalling/ (Updates)

Beispiel für IPv6 netfilter Regelwerk

- **Minimales Regelwerk (für *ip6tables-restore*):**

```
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p ipv6-icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -j REJECT --reject-with icmp6-adm-prohibited
-A FORWARD -j REJECT --reject-with icmp6-adm-prohibited
COMMIT
```

- **Erlauben eingehend SSH von überall her:**

```
-A INPUT -m state --state NEW -m tcp -p tcp --syn --dport 22 -j ACCEPT
```

- **Mehr Tipps stehen zur Verfügung unter:**
  http://www.tldp.org/HOWTO/Linux+IPv6-HOWTO/chapter-firewalling-security.html

Erlauben aller eingehender ICMPv6 -Pakete wichtig für Neighbor/Router-Discovery:

-A INPUT -p ipv6-icmp -j ACCEPT

Kann ggf. verfeinert werden durch Einschränkung des Hop-Limits (IPv6: TTL):

-A INPUT -p ipv6-icmp -j ACCEPT --match hl --hl-eq 1
-A INPUT -p ipv6-icmp -j ACCEPT --match hl --hl-eq 255

bzw. auf einzelne ICMPv6-Typen (Vorsicht: DROPs loggen und prüfen, gg.f. Regelsatz erweitern).

Demo IPv6-Firewalling

- **Fedora / Red Hat / CentOS: /etc/sysconfig/ip6tables**

Titel durch Klicken hinzufügen

IPv6 QoS unter Linux

IPv6 QoS unter Linux / Filter

- **Schnittstelle, Quelle, Ziel, IP-Version, Protokoll, Port**
  - Beispiel: eth1, IPv6, TCP (6), ssh (22)
    - Filterdefinition direkt in “tc”

```
# tc filter add dev eth1 parent 1: protocol ipv6 u32 match ip6 protocol 6 0xff
match ip6 dport 22 0xffff flowid 1:2
```

    - Markierung im Linux-Firewalling

```
# ip6tables -A POSTROUTING -t mangle -p tcp --dport 22 -j MARK --set-mark 32
# tc filter add dev eth1 parent 1: protocol ipv6 handle 32 fw flowid 1:4
```

- **Flow Label (Beispiel: 0x21)**

```
# tc filter add dev eth1 parent 1: protocol ipv6 u32 match ip6 flowlabel 0x21
0x3ffff flowid 1:3
```

- **Traffic Class (Beispiel: 0x11)**

```
# tc filter add dev eth1 parent 1: protocol ipv6 u32 match u32 0x01100000
0x0ff00000 at 0 flowid 1:3
```

Achtung: tc filter ... match ip6 priority ...“ erzeugt falschen „match“ (Red Hat Bugzilla 584913)

Anzeige der Filter incl. Trefferstatistik:

```
# tc -s filter show dev eth1
```

Demo IPv6 QoS (1)

- **Konfigurieren von Filter**
  - Filter für *Flow Label* und *Traffic Class*
- **Trigger eines "Match" mit ping6**
  - Flow Label 0x21 (33)

```
# ping6 -c 2 -F 21 DESTINATION
```

  - Traffic Class 0x11 (17)

```
# ping6 -c 2 -Q 11 DESTINATION
```

Achtung: aktuell interpretiert ping6 die Eingabewerte für -F und -Q als Hexadezimal

- **Testen eines "Match"**

```
# tc -s filter show dev DEVICE | grep -B 1 match
filter parent 1: protocol ipv6 pref 49149 u32 fh 803::800 order 2048 key ht
803 bkt 0 flowid 1:3  (rule hit 5 success 2)
  match 01100000/0ff00000 at 0 (success 2 )
filter parent 1: protocol ipv6 pref 49150 u32 fh 802::800 order 2048 key ht
802 bkt 0 flowid 1:3  (rule hit 3 success 2)
  match 00000021/0003ffff at 0 (success 2 )
```

Dr. Peter Bieringer – IPv6 mit Linux (Tutorial) – IPv6-Kongress – 20. - 21. Mai 2010, Frankfurt/Main, Deutschland 21.05.10 19:06:35 50

Ältere„ping6"-Versionen (vor Mai 2010) unterstützen „-F flowlabel" nicht.

Ältere Versionen von „traceroute" (< 2.0.14) können das Flowlabel nicht setzen.

Filter-Setup im Detail:

```
# tc qdisc add dev eth1 root handle 1: cbq avpkt 1000
bandwidth 1000Mbit
# tc class add dev eth1 parent 1: classid 1:1 cbq rate
1Mbit allot 1500 bounded
# tc class add dev eth1 parent 1: classid 1:2 cbq rate
50Mbit allot 1500 bounded
# tc class add dev eth1 parent 1: classid 1:3 cbq rate
10Mbit allot 1500 bounded
# tc class add dev eth1 parent 1: classid 1:4 cbq rate
200kbit allot 1500 bounded

# tc filter add dev eth1 parent 1: protocol ipv6 u32
match ip6 flowlabel 33 0x3ffff flowid 1:3
# tc filter add dev eth1 parent 1: protocol ipv6 u32
match u32 0x01100000 0x0ff00000 at 0 flowid 1:3
```

Demo IPv6 QoS (2)

- **Konfigurieren von Filter**
  - TCP-Port 5001 (0x1389) auf 50 MBit/s limitiert
- **Test auf Limitierung mit iperf abhängig vom TCP-Port**
  - Server

```
# iperf -V -s -p 5001
# iperf -V -s -p 5002
```

  - Client

```
# iperf -V -c DESTINATION -p 5001
# iperf -V -c DESTINATION -p 5002
```

- **Testen auf Wirksamkeit**

```
# tc -s filter show dev DEVICE | grep -B 1 match

filter parent 1: protocol ipv6 pref 49151 u32 fh 801::800 order 2048 key ht
801 bkt 0 flowid 1:2  (rule hit 18873 success 1846)
  match 00000600/0000ff00 at 4 (success 18870 )
  match 00001389/0000ffff at 40 (success 1846 )
```


Filter-Setup im Detail:

```
# tc qdisc add dev eth1 root handle 1: cbq avpkt 1000
bandwidth 1000Mbit
# tc class add dev eth1 parent 1: classid 1:1 cbq rate
1Mbit allot 1500 bounded
# tc class add dev eth1 parent 1: classid 1:2 cbq rate
50Mbit allot 1500 bounded
# tc class add dev eth1 parent 1: classid 1:3 cbq rate
10Mbit allot 1500 bounded
# tc class add dev eth1 parent 1: classid 1:4 cbq rate
200kbit allot 1500 bounded

# tc filter add dev eth1 parent 1: protocol ipv6 u32
match ip6 protocol 6 0xff match ip6 dport 5001 0xffff
flowid 1:2
```

Titel durch Klicken hinzufügen

IPv6 in virtualisierten Umgebungen
(libvirt)

Status von libvirt / virt-manager

- **Fedora 12 mit libvirt 0.7.1 und virt-manager 0.8.2**
  - IPv6 ist immer noch ein Stiefkind
  - virt-manager verlangt IPv4-Setup für VM 
  - libvirt erzeugt KEINE passenden ip6tables-Einträge für Bridges (Red Hat Bugzilla 567124) 
    - Handarbeit oder Skript notwendig, welches *iptables* Regeln zu *ip6tables* Regeln kopiert
  - Kernel hat unerwartet Geschwindigkeitsprobleme bei Benutzung von “virtio” als Treiber und einer VM als Router (RH-BZ 571234)
    - Client (VM) <--> Router (VM) <--> Server (VM) 

Skript für Übernahme von Regeln bzgl. virbr-Schnittstellen von IPv4 (iptables) nach IPv6 (ip6tables):

```
r=0; iptables-save -t filter | grep '^-A FORWARD' |
egrep '(-i virbr. -o virbr.|-i virbr. -j REJECT)' | sed
's/^-A FORWARD//' | sed 's/icmp/icmp6/' | while read
line; do r=$[ $r + 1 ]; ip6tables -I FORWARD $r $line;
done
```

Titel durch Klicken hinzufügen

Weitere Informationen

IPv6 & Linux bezogene Information

- ***Linux IPv6 HOWTO***
  - Schwerpunkt: ausgiebige Information über IPv6 in Linux
    http://www.tldp.org/HOWTO/Linux+IPv6-HOWTO/ (nur English)
    http://mirrors.bieringer.de/ (en, de, fr, it)
  - URLs zu allen Übersetzungen und weiterführende Informationen
    http://www.bieringer.de/linux/IPv6/
- ***Current Status of IPv6 Support for Networking Applications***
  - IPv6-Status von netzwerkfähigen Applikationen
    http://www.deepspace6.net/docs/ipv6_status_page_apps.html

Kontakt-Information

peter@deepspace6.net

http://www.deepspace6.net/

pb@bieringer.de

http://www.bieringer.de/pb/

http://www.bieringer.de/linux/IPv6/

http://mirrors.bieringer.de/

Titel durch Klicken hinzufügen

Vielen Dank für die Teilnahme!

Fragen & Antworten

Tutorial mit Notizen ist als PDF per E-Mail erhältlich!

Dankeschön an

Johannes Endres, c't (Einladung)

AERAsec Network Services and Security GmbH (IPv6 Web- & E-Mail Hosting)