IPv6 mit Linux
(Einführung)

Tutorial

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

IPv6-Kongress
Frankfurt/Main, Deutschland
22. - 23. Mai 2014
http://www.ipv6-kongress.de/

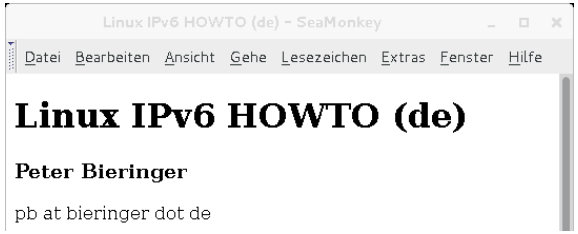
Über mich

- **Wohnhaft in München (Deutschland)**
- **Beschäftigt als *Senior IT Architect* bei *Giesecke & Devrient 3S GmbH***

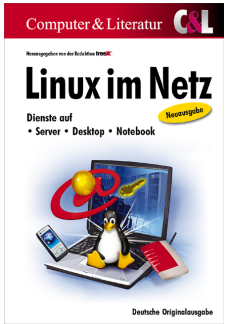
Giesecke & Devrient
Creating Confidence.

- **Mitbegründer und Kernmitglied von *Deep Space 6***

- **Autor des "Linux IPv6 HowTo"**



- **Mitautor des Buches "Linux im Netz" (2006)**
  - 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 18 Jahre IPv6-Erfahrung!

Inhalt

IPv6 mit Linux (Einführung)

- IPv6-Konfiguration
- Fehlersuche
- IPv6-Anbindung
- IPv6-aktive Services
- IPv6-Firewalling (ip6tables, nftables) 

- IPv6 Network Address Translation (NAT)
- IPv6 in virtualisierten Umgebungen

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
- **Adress-Auswahl** 

IPv6 in Linux Voraussetzungen

- **IPv6 im Linux-Kernel aktiviert?**
  - Datei `/proc/net/if_inet6` existiert?

```
# grep -w lo /proc/net/if_inet6
00000000000000000000000000000001 01 80 10 80       lo
```

- **IPv6 im Linux-Kernel aktivieren, falls notwendig**
  - 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 (für Router, 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 1000
    inet6 fe80::224:21ff:fe01:2345/64 scope link
       valid_lft forever preferred_lft forever


# ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::224:21ff:fe01:2345  prefixlen 64  scopeid 0x20<link>
        ether 00:24:21:01:23:45  txqueuelen 1000  (Ethernet)
        ...
```

IPv6-Adresskonfiguration Statische Zuweisung

- **Werkzeug: `ip`**

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

- **Beispiele**

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

- **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
```

IPv6-Adresskonfiguration Router Advertisements (1)

- **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 Router Advertisements (2)

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

```
interface eth0
{
   ...
   DNSSL demo.bieringer.de {
       AdvDNSSLLifetime 600;
   };
   RDNSS fec0::1 {
       AdvRDNSSLifetime 600;
   };
};
```

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

```
$ cat /etc/resolv.conf

# Generated by NetworkManager
domain demo.bieringer.de
search demo.bieringer.de
nameserver fec0::1
```

IPv6-Adresskonfiguration DHCPv6

- **Server**
  - Software: ISC-DHCP (4.2.6), dibbler (0.8.4)
  - Verteilung von
    - IPv6-DNS-Server
    - DNS Search List
- **Client**
  - Software
    - dhclient (4.2.6), z.B. in Fedora 20
    - NetworkManager
  - Unterstützt
    - IPv6-DNS-Server
    - DNS Search List

dhcpv6 (1.2.0) – Weiterentwicklung gestoppt seit 09/2009 

IPv6-Adresskonfiguration Privacy (manuell)

- **Konfiguration (manuell)**
  - auf CLI

```
# 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-Adresskonfiguration Privacy (permanent)

- **Konfiguration (permanent)**
  - Änpassung der Datei /etc/sysctl.conf
  - pro Schnittstelle (muß zum Zeitpunkt bereits existieren!)

```
net.ipv6.conf.eth0.use_tempaddr=2
```

  - für alle Schnittstellen

```
net.ipv6.conf.all.use_tempaddr=2
net.ipv6.conf.default.use_tempaddr=2
```

  - Aktivieren auf CLI

```
# sysctl -p
```

  - Prüfung auf CLI

```
# sysctl -a |grep tempaddr
net.ipv6.conf.all.use_tempaddr = 2
net.ipv6.conf.default.use_tempaddr = 2
net.ipv6.conf.eth0.use_tempaddr = 2
net.ipv6.conf.eth1.use_tempaddr = 2
net.ipv6.conf.lo.use_tempaddr = 2
```

  - Restart der Schnittstelle oder Reboot notwendig!

IPv6-Adresskonfiguration Privacy (NetworkManager)

- **Konfiguration via NetworkManager** (0.9.9.1-5.git20140319.fc21)
  - Vorhandene Interfaces

```
# nmcli connection
NAME    UUID                                  TYP             GERÄT
ens4v1  d0fc2b2e-5fa0-4675-96b5-b723ca5c46db  802-3-ethernet  ens4v1
```

  - Aktuelle IPv6-Privacy-Konfiguration

```
# ip -o addr show dev ens4v1 | grep temporary | wc -l
0

# nmcli connection show ens4v1 |grep ip6-privacy
ipv6.ip6-privacy:                       -1 (unbekannt)
```

  - Anpassung & Restart des Interfaces

```
# nmcli connection modify ens4v1 ipv6.ip6-privacy 2

# nmcli connection down ens4v1; nmcli connection up ens4v1
```

  - Neue IPv6-Privacy-Konfiguration

```
# nmcli connection show ens4v1 |grep ip6-privacy
ipv6.ip6-privacy:                       2 (aktiviert, temporäre IP bevorzugen)

# ip -o addr show dev ens4v1 | grep temporary | wc -l
2
```

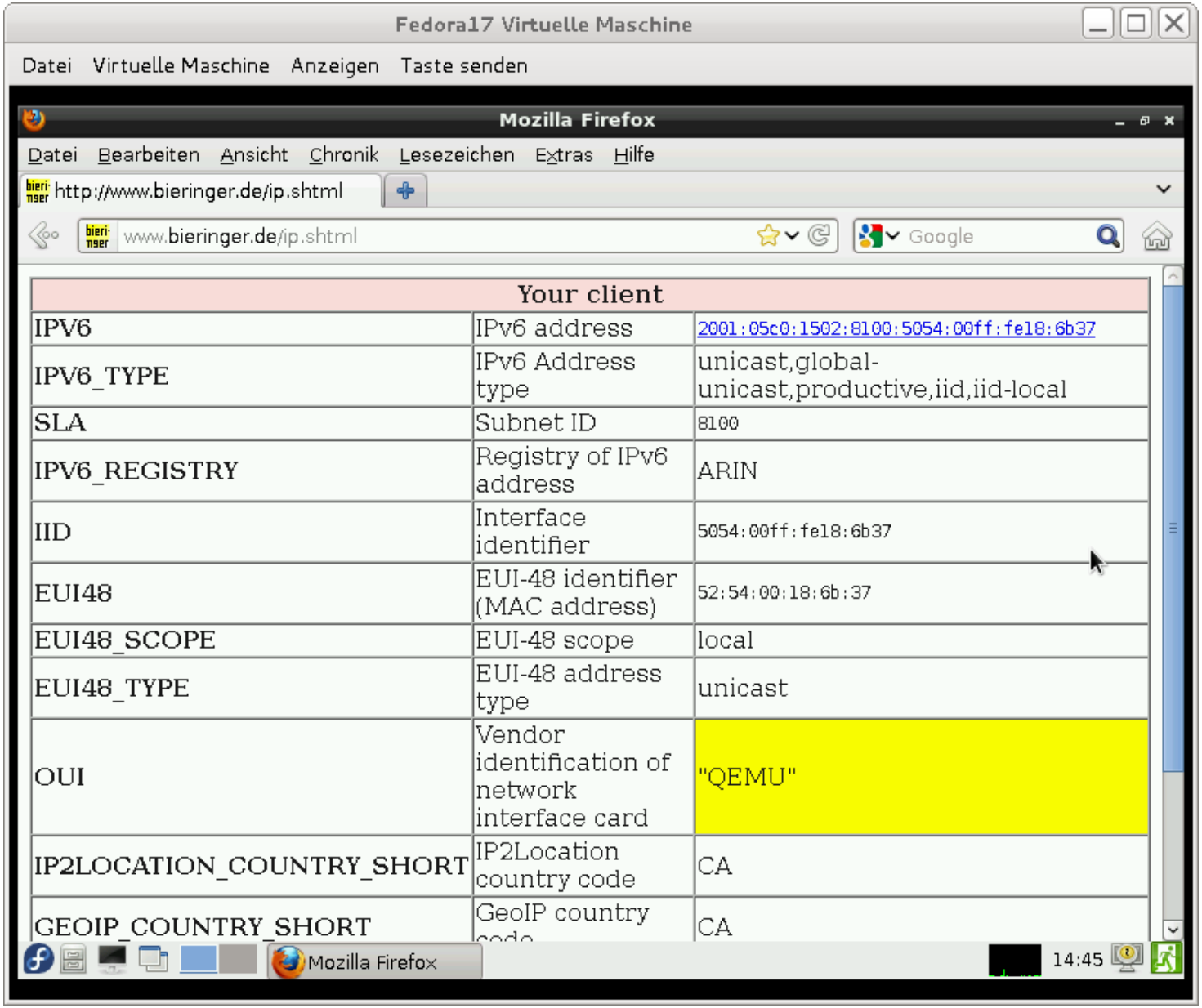
IPv6-Adresskonfiguration Privacy (Test & Status)

- **IPv6 Privacy Test**
  - Test: http://ip.bieringer.de/
    - EUI64_SCOPE: iid-privacy

- **IPv6 Privacy Status nach Installation**

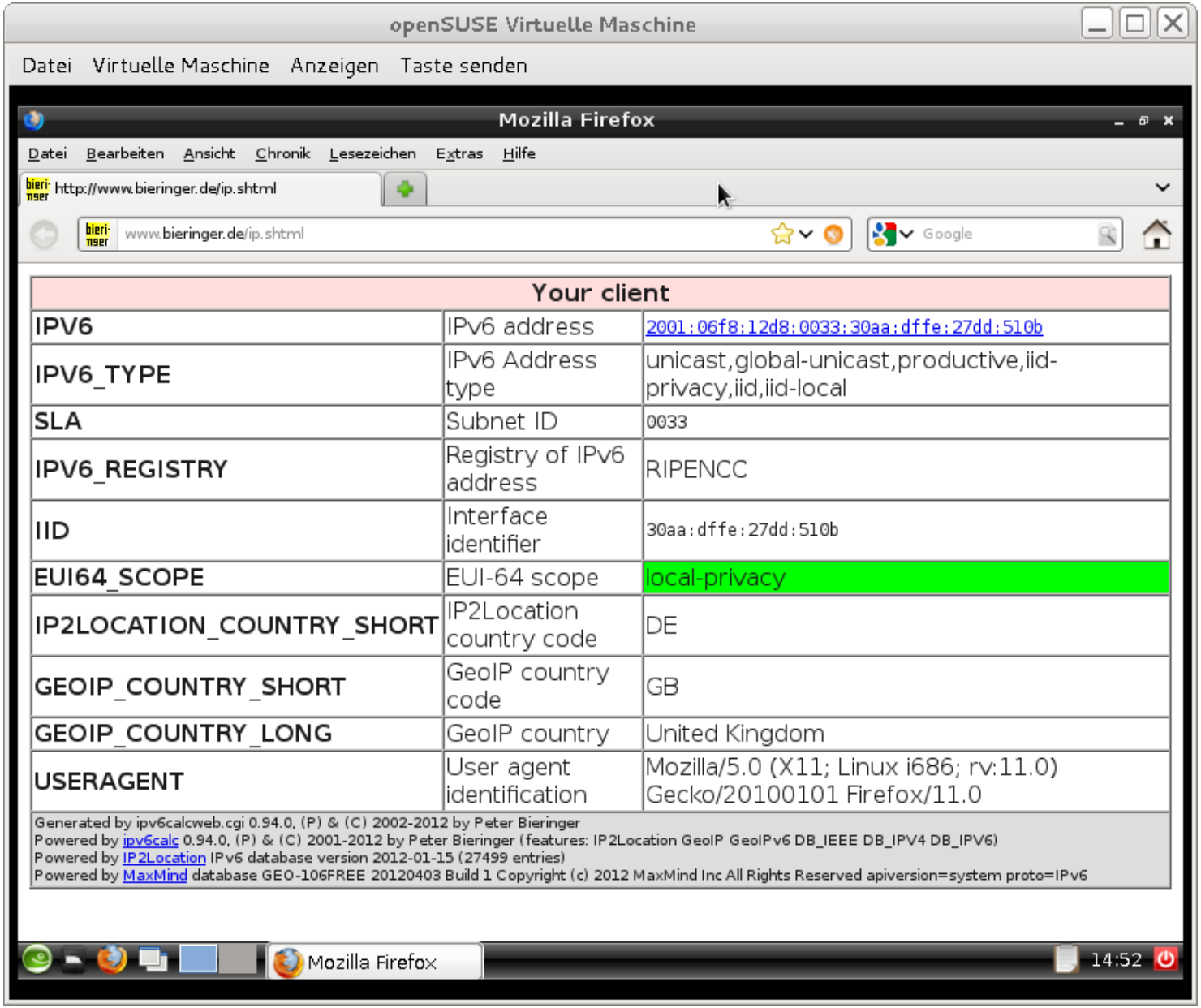

CentOS/RHEL:	5, 6, 7	
Fedora:	16, 17, 18, 19, 20	
Ubuntu:	11.10, 12.04, 13.04	14.04
Debian	6.0.4	
openSUSE:		12.1, 12.3, 13.1

IPv6-Adresskonfiguration Privacy – Test (1)



Your client		
IPV6	IPv6 address	2001:05c0:1502:8100:5054:00ff:fe18:6b37
IPV6_TYPE	IPv6 Address type	unicast,global-unicast,productive,iid,iid-local
SLA	Subnet ID	8100
IPV6_REGISTRY	Registry of IPv6 address	ARIN
IID	Interface identifier	5054:00ff:fe18:6b37
EUI48	EUI-48 identifier (MAC address)	52:54:00:18:6b:37
EUI48_SCOPE	EUI-48 scope	local
EUI48_TYPE	EUI-48 address type	unicast
OUI	Vendor identification of network interface card	"QEMU"
IP2LOCATION_COUNTRY_SHORT	IP2Location country code	CA
GEOIP_COUNTRY_SHORT	GeoIP country code	CA

IPv6-Adresskonfiguration Privacy – Test (2)



Your client		
IPV6	IPv6 address	2001:06f8:12d8:0033:30aa:dffe:27dd:510b
IPV6_TYPE	IPv6 Address type	unicast,global-unicast,productive,iid-privacy,iid,iid-local
SLA	Subnet ID	0033
IPV6_REGISTRY	Registry of IPv6 address	RIPENCC
IID	Interface identifier	30aa:dffe:27dd:510b
EUI64_SCOPE	EUI-64 scope	local-privacy
IP2LOCATION_COUNTRY_SHORT	IP2Location country code	DE
GEOIP_COUNTRY_SHORT	GeoIP country code	GB
GEOIP_COUNTRY_LONG	GeoIP country	United Kingdom
USERAGENT	User agent identification	Mozilla/5.0 (X11; Linux i686; rv:11.0) Gecko/20100101 Firefox/11.0

Generated by ipv6calcweb.cgi 0.94.0, (P) & (C) 2002-2012 by Peter Bieringer
Powered by ipv6calc 0.94.0, (P) & (C) 2001-2012 by Peter Bieringer (features: IP2Location GeoIP GeoIPv6 DB_IEEE DB_IPV4 DB_IPV6)
Powered by IP2Location IPv6 database version 2012-01-15 (27499 entries)
Powered by MaxMind database GEO-106FREE 20120403 Build 1 Copyright (c) 2012 MaxMind Inc All Rights Reserved apiversion=system proto=IPv6

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

- **Werkzeug: `ip`**

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

- **Beispiele**

```
# ip -6 route add 2001:db8:0:2::/64 via 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
```

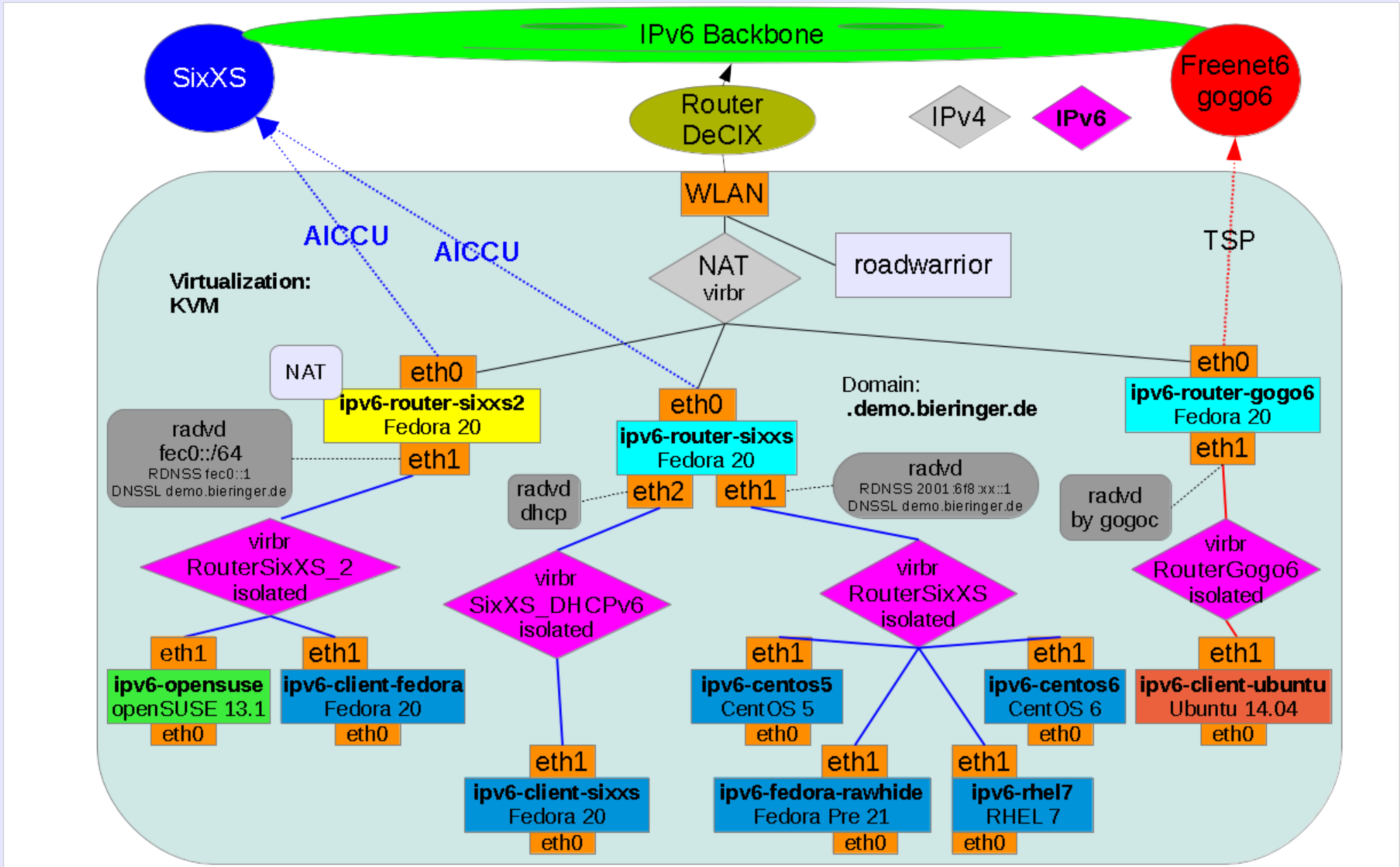
IPv6
Address Selection

- **Destination Address Selection**
  - Mehr als eine IPv6-Adresse in DNS-Antwort via *getaddrinfo*
    - Ja: RFC 3484 *Default Address Selection for Internet Protocol version 6*
      - Beeinflussung der Sortierung mit Hilfe von /etc/gai.conf
- **Source Address Selection**
  - Mehr als ein "globaler" Prefix pro Interface
    - Ja: Möglichkeit der Bindung zu Ziel-Präfix mit: **`ip addrlabel`**
  - Beispiel von 2 Bindungen

```
# ip addrlabel
...
prefix 2a01:238:423d:8800::/64 label 300   # dst-B
prefix 2001:4dd0:ff00:834::/64 label 200   # dst-A
prefix 2001:6f8:900:8cbc::/64  label 300   # src-B
prefix 2001:6f8:12d8:2::/64    label 200   # src-A
...
```

Siehe auch: http://mirrors.deepspace6.net/Linux+IPv6-HOWTO/resolver.html

Demo Aufbau


IPv6 Backbone
SixXS
Router
DeCIX
IPv4
IPv6
Freenet6
gogo6
WLAN
AICCU
AICCU
TSP
Virtualization:
KVM
NAT
virbr
roadwarrior
NAT
eth0
ipv6-router-sixxs2
Fedora 20
eth1
radvd
fec0::/64
RDNSS fec0::1
DNSSL demo.bieringer.de
eth0
ipv6-router-sixxs
Fedora 20
eth2
eth1
radvd
dhcp
radvd
RDNSS 2001:6f8:xx::1
DNSSL demo.bieringer.de
Domain:
.demo.bieringer.de
eth0
ipv6-router-gogo6
Fedora 20
eth1
radvd
by gogoc
virbr
RouterSixXS_2
isolated
virbr
SixXS_DHCPv6
isolated
virbr
RouterSixXS
isolated
virbr
RouterGogo6
isolated
eth1
ipv6-opensuse
openSUSE 13.1
eth0
eth1
ipv6-client-fedora
Fedora 20
eth0
eth1
ipv6-client-sixxs
Fedora 20
eth0
eth1
ipv6-centos5
CentOS 5
eth0
eth1
ipv6-centos6
CentOS 6
eth0
eth1
ipv6-client-ubuntu
Ubuntu 14.04
eth0
eth1
ipv6-fedora-rawhide
Fedora Pre 21
eth0
eth1
ipv6-rhel7
RHEL 7
eth0

Demo IPv6 DHCP

- **IPv6-only Client-Konfiguration**
  - Fedora >= 16: /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)

Demo IPv6 DHCP (Probleme)

- **DHCPv6-Server**
  - IPv6-Firewall blockt DHCPv6-Anfragen
    - udp/547 erlauben von fe80::/64

```
-A INPUT -s fe80::/64 -p udp -m udp --dport 547 -j ACCEPT
```

- **DHCPv6-Client**
  - DHCPv6 wird von ip6tables nicht als stateful erkannt
    - Firewall anpassen für eingehende Pakete vom DHCPv6-Server
      - Fedora < 17
      - OpenSUSE 12
    - udp/546 erlauben von fe80::/64

```
-A INPUT -d fe80::/64 -p udp -m udp --dport 546 -j ACCEPT
```

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-Präfix	IPv6-Netz	Anbieter bzw. Methode
Nativ		Statisch	Statisch	ja	ISP (Firmenanschluß)
Nativ		Dynamisch	Dynamisch	ja	ISP via DHCPv6
Nativ		Dynamisch	Statisch	ja	ISP via DHCPv6
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

- **Firmenkunden (entsprechender Anschluß):**
  - ISP routet direkt Präfixe über die Anbindung
- **Privatkunden**
  - IPv6 über DHCPv6 (z.B. Mnet: /56)
- **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 (2011)
- **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 hinterlegen

IPv6-Anbindung
Teredo

- **URLs:**
  - http://de.wikipedia.org/wiki/Teredo
- **Kein Subnetz möglich (eigentlich)**
  - Workaround: IPv6-NAT
- **Mögliche Tunnelmethoden**
  - IPv6-in-UDP-in-IPv4 (sog. Teredo-Bubbles)
    - Port: 3544/udp
      - Zusätzlich hinter NAT “high-ports” ausgehend notwendig
- **Linux-Client: miredo-client**

Demo IPv6-Anbindung

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

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:           [fd00::]/7
```

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**
  - “ipv6loganon” (im Paket “ipv6calc”) ab 0.90.0 kann Logfiles on-the-fly anonymisieren
    - Standard in httpd.conf

```
CustomLog logs/access_log combined
```

    - Mit Anonymisierung

```
CustomLog "|/usr/bin/ipv6loganon -f -a /var/log/httpd/access_log"  ¬
  combined
```

    - Mit Anonymisierung und “cronolog”

```
CustomLog "|/usr/bin/ipv6loganon -f |/usr/sbin/cronolog          ¬
  /var/log/httpd/access.log-%Y%m%d" combined
```

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!
  - (Noch) Kein impliziter Schutz durch NAT vorhanden!
  - System ist vom Internet aus (noch) 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 20 Standardinstallation**

```
Aktive Internetverbindungen (Nur Server)
Proto Recv-Q Send-Q Local Address   Foreign Address  State  PID/Program name
tcp        0      0 0.0.0.0:58755   0.0.0.0:*        LISTEN 710/rpc.statd
tcp        0      0 0.0.0.0:111     0.0.0.0:*        LISTEN 695/rpcbind
tcp        0      0 0.0.0.0:22      0.0.0.0:*        LISTEN 764/sshd
tcp        0      0 127.0.0.1:631   0.0.0.0:*        LISTEN 1692/cupsd
tcp6       0      0 :::111          :::*             LISTEN 695/rpcbind
tcp6       0      0 :::22           :::*             LISTEN 764/sshd
tcp6       0      0 ::1:631         :::*             LISTEN 1692/cupsd
tcp6       0      0 :::33689        :::*             LISTEN 710/rpc.statd
udp        0      0 0.0.0.0:57652   0.0.0.0:*               710/rpc.statd
udp        0      0 127.0.0.1:323   0.0.0.0:*               544/chronyd
udp        0      0 0.0.0.0:68      0.0.0.0:*               901/dhclient
udp        0      0 0.0.0.0:52806   0.0.0.0:*               531/avahi-daemon: r
udp        0      0 0.0.0.0:61534   0.0.0.0:*               901/dhclient
udp        0      0 0.0.0.0:867     0.0.0.0:*               695/rpcbind
udp        0      0 0.0.0.0:111     0.0.0.0:*               695/rpcbind
udp        0      0 127.0.0.1:888   0.0.0.0:*               710/rpc.statd
udp        0      0 0.0.0.0:123     0.0.0.0:*               544/chronyd
udp        0      0 0.0.0.0:5353    0.0.0.0:*               531/avahi-daemon: r
udp6       0      0 ::1:323         :::*                    544/chronyd
udp6       0      0 :::37975        :::*                    710/rpc.statd
udp6       0      0 :::867          :::*                    695/rpcbind
udp6       0      0 :::111          :::*                    695/rpcbind
udp6       0      0 :::123          :::*                    544/chronyd
udp6       0      0 :::31692        :::*                    901/dhclient
```

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 (fec0::/10, fd00::/7)
  - 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 (Februar 2007)
    - Red Hat EL 4 (2.6.9) und 5 (2.6.18): nur stateless!
  - Network Address Translation (NAT) ab Kernel 3.9.0 (April 2013)
    - in Verbindung mit iptables 1.4.18
  - Neues Framework "nftables" seit Kernel 3.13 (April 2014) 
- **Benutzer-Werkzeuge**
  - *iptables* (IPv4) bzw. *ip6tables* (IPv6)
  - *nft* (IPv4 & IPv6) 

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 (1)

- **Distributionseigene Werkzeuge**
  - Red Hat & Clones:
    - bis 5.x: system-config-securitylevel-tui
    - 6.x: system-config-firewall-tui
    - 7.x: firewallD (firewall-cmd)
  - Fedora:
    - bis 16: system-config-firewall-tui
    - ab 17: firewallD (firewall-cmd)
  - openSUSE: SuSEfirewall2

Firewalling Regelsatzgeneratoren (2)

- **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

Beispiel für IPv6 Regelwerk (ip6tables)

- **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 ohne Source-Filter:**

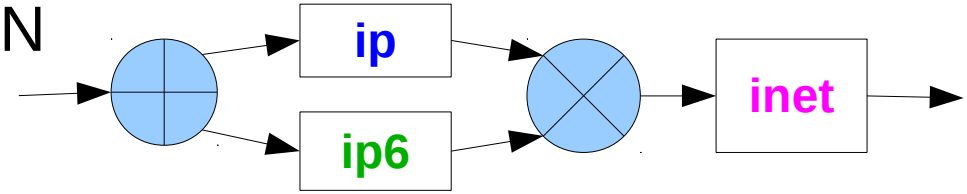
```
-A INPUT -m state --state NEW -m tcp -p tcp --syn --dport 22 -j ACCEPT
```

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

Besonderheiten von nftables

- **Dedizierte Tabellen für IPv4 und IPv6**
  - `table ip`
  - `table ip6`
- **Gemeinsame Tabelle für IPv4 und IP6** 

  - `table inet`
    - Protokol-Auswahl möglich mit
      - `meta nfproto ipv4`
      - `meta nfproto ipv6`
- **Paket durchläuft BEIDE relevanten Tabellen** 
  - Achtung: "accept" in BEIDEN Tabellen notwendig
  - Tipp
    - Verwendung von "marks" in den dedizierten Tabellen
    - "accept" von markierten Paketen in Tabelle "inet"



Beispiel für IPv6 Regelwerk (nftables)

- **Minimales Regelwerk (nur Nutzung von Tabelle "inet"):**

  - Connection Tracking für IPv4 und IPv6

```
# nft add rule inet filter input ct state established,related counter accept
```

  - Wichtige ICMPv6-Typen

```
# nft add rule inet filter input meta nfproto ipv6 icmpv6 type echo-request accept
# nft add rule inet filter input meta nfproto ipv6 icmpv6 type { nd-neighbor-advert, nd-neighbor-
  solicit, nd-router-advert} accept
```

  - IPv4-Ping

```
# nft add rule inet filter input meta nfproto ipv4 icmp type { echo-request } accept
```

  - Erlauben eingehend SSH ohne Source-Filter:

```
# nft add rule inet filter input tcp dport 22 ct state new tcp flags \& \(syn \| ack\) == syn ac-
  cept
```

  - Finale Reject-Regel

```
# nft add rule inet filter input counter reject
```

Eine Regel für IPv4 und IPv6

- **Widrigkeiten**
  - Überlagerung von existierendes ip(6)tables-Regelwerk
  - Kein Loglevel in Log-Option => default: kern.emerg

Beispiel für IPv6 netfilter NAT

- **Funktion gleichwertig zu IPv4**
  - IPv6 Masquerading

```
# ip6tables -t nat -I POSTROUTING -o sixxs -s fec0::/64 -j MASQUERADE
```

    - Maskierung interner IPv6-Clients durch Router-Adresse

  - IPv6 Ziel Weiterleitung

```
# ip6tables -t nat -A PREROUTING
  -d 2001:db8:0:1:5054:ff:fe01:2345 -i sixxs
  -j DNAT --to-destination fec0::5054:ff:fe01:2345
```

    - Weiterleitung externer IPv6-Adresse an interne Adresse

  - IPv6 Ziel/Port Weiterleitung

```
# ip6tables -t nat -A PREROUTING -i sixxs
  -p tcp --dport 8080
  -j DNAT --to-destination [fec0::1234]:80
```

    - Weiterleitung von Port 8080 via Router nach intern auf Port 80

Demo IPv6-NAT

- **Fedora 20 als Router**
- **Fedora 20 (2. Installation)**
  - als Host: 1:1 NAT
  - als Client: Masquerading
- **OpenSUSE**
  - als Host: 1:1 NAT
  - als Client: Masquerading
  - als Server: Portweiterleitung von 8080 auf 80

IPv6 in virtualisierten Umgebungen (libvirt, VirtualBox)

Status von libvirt / virt-manager

- **Allgemein**
  - IPv6 ist immer noch ein Stiefkind
- **libvirt 1.1.3.4 (Fedora 20)**
  - IPv6-Konfiguration via virtsh (bzw. XML-Datei)
  - libvirt erzeugt passenden ip6tables-Einträge für Bridges NUR wenn IPv6 explizit im XML konfiguriert
    - inkonsistent zu IPv4 
- **virt-manager 1.0.1 (Fedora 20)**
  - IPv6-Unterstützung fehlt (noch immer) 

Status von VirtualBox

- **Aktuell verfügbar: 4.3.10**
  - Host-Only-Netzwerk “vboxnetX”
    - Zuweisung einer IPv6-Adresse möglich 
  - Interne Netzwerke sind **nicht** als Linux-Bridges realisiert
    - Keine Probleme mit Host-Firewall 

Weitere Informationen

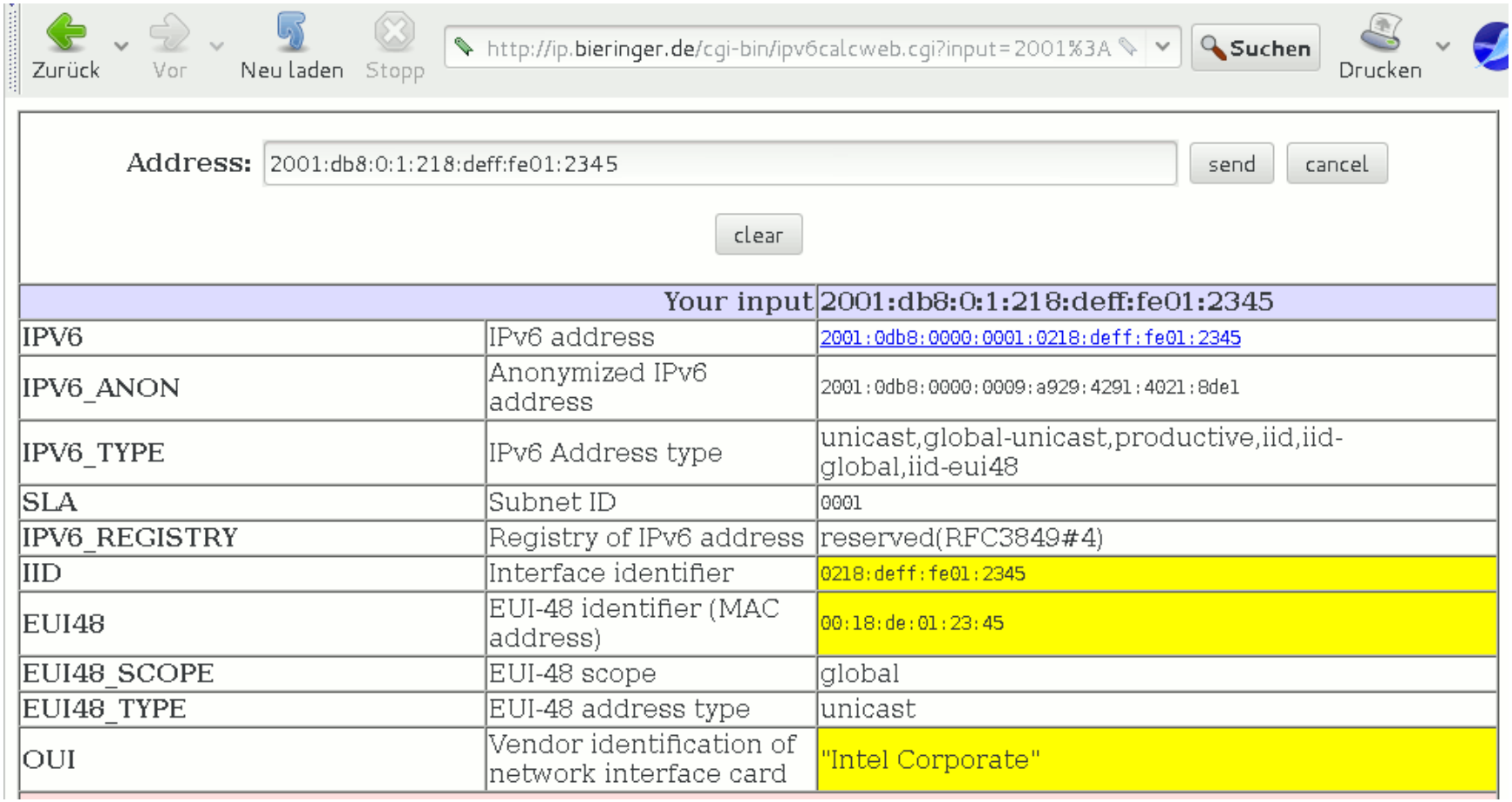
IPv6 & Linux bezogene Information

- ***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)
    http://mirrors.deepspace6.net/howtos/ (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

ipv6calc Online Tool

- **URL: http://ip.bieringer.de/cgi-bin/ipv6calcweb.cgi**
  - Betrieb von ipv6calcweb.cgi im “Form”-Modus
  - Information über MAC, IPv4 & IPv6-Adressen



Your input		2001:db8:0:1:218:deff:fe01:2345
IPV6	IPv6 address	2001:0db8:0000:0001:0218:deff:fe01:2345
IPV6_ANON	Anonymized IPv6 address	2001:0db8:0000:0009:a929:4291:4021:8de1
IPV6_TYPE	IPv6 Address type	unicast,global-unicast,productive,iid,iid-global,iid-eui48
SLA	Subnet ID	0001
IPV6_REGISTRY	Registry of IPv6 address	reserved(RFC3849#4)
IID	Interface identifier	0218:deff:fe01:2345
EUI48	EUI-48 identifier (MAC address)	00:18:de:01:23:45
EUI48_SCOPE	EUI-48 scope	global
EUI48_TYPE	EUI-48 address type	unicast
OUI	Vendor identification of network interface card	"Intel Corporate"

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 bzw. über Veranstalter erhältlich!

Dankeschön an

Jürgen Seeger, iX (Einladung)

IPv6 mit Linux
(Einführung)

Tutorial

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

IPv6-Kongress
Frankfurt/Main, Deutschland
22. - 23. Mai 2014
http://www.ipv6-kongress.de/

Über mich

- **Wohnhaft in München (Deutschland)**
- **Beschäftigt als *Senior IT Architect* bei *Giesecke & Devrient 3S GmbH***

Giesecke & Devrient
Creating Confidence.

- **Mitbegründer und Kernmitglied von *Deep Space 6***

- **Autor des "Linux IPv6 HowTo"**

- **Mitautor des Buches "Linux im Netz" (2006)**
  - 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 18 Jahre IPv6-Erfahrung!

Inhalt

IPv6 mit Linux (Einführung)

- IPv6-Konfiguration
- Fehlersuche
- IPv6-Anbindung
- IPv6-aktive Services
- IPv6-Firewalling (ip6tables, nftables) 

- IPv6 Network Address Translation (NAT)
- IPv6 in virtualisierten Umgebungen

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
- **Adress-Auswahl** 

Aktuelle Routing-Daemons für Linux: quagga, bird

IPv6 in Linux Voraussetzungen

- **IPv6 im Linux-Kernel aktiviert?**
  - Datei `/proc/net/if_inet6` existiert?

```
# grep -w lo /proc/net/if_inet6
00000000000000000000000000000001 01 80 10 80       lo
```

- **IPv6 im Linux-Kernel aktivieren, falls notwendig**
  - 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 (für Router, bei Bedarf)**

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


Vollständiges Deaktivieren von IPv6 bei Bedarf möglich durch folgende Moduloption (z.B. in /etc/modprobe.d/local.conf):

```
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 1000
    inet6 fe80::224:21ff:fe01:2345/64 scope link
       valid_lft forever preferred_lft forever


# ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::224:21ff:fe01:2345  prefixlen 64  scopeid 0x20<link>
        ether 00:24:21:01:23:45  txqueuelen 1000  (Ethernet)
        ...
```


Ältere Ausgabe von "ifconfig":

```
# 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

Werkzeug: ip

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

Beispiele

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

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
```


•Werkzeug: **ifconfig**

```
# ifconfig INTERFACE inet6 add IPV6ADDRESS/PREFIXLENGTH
```

•Beispiele

```
# ifconfig eth0 inet6 add 2001:db8:0:1::1/64
```

•Ergebnis

```
# 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 (1)

- **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 Router Advertisements (2)

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

```
interface eth0
{
   ...
   DNSSL demo.bieringer.de {
       AdvDNSSLLifetime 600;
   };
   RDNSS fec0::1 {
       AdvRDNSSLifetime 600;
   };
};
```

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

```
$ cat /etc/resolv.conf
# Generated by NetworkManager
domain demo.bieringer.de
search demo.bieringer.de
nameserver fec0::1
```


Nur neuere Kernel in Verbindung mit neueren NetworkManager sind in der Lage, DNSSL und RDNSS Optionen im RouterAdvertisement auszuwerten.

Siehe auch https://fedoraproject.org/wiki/Networking/Addressing

„rdnssd“ aus dem Paket ndisc6 kann alternativ benutzt werden, um /etc/resolv.conf ändern, falls auf einem System kein NetworkManager läuft oder vorhanden ist – unterstützt aber nur RDNSS!

Wireshark-Dump:

```
ICMPv6 Option (Recursive DNS Server fec0::1)
    Type: Recursive DNS Server (25)
    Length: 3 (24 bytes)
    Reserved
    Lifetime: 600
    Recursive DNS Servers: fec0::1 (fec0::1)
ICMPv6 Option (DNS Search List Option demo.bieringer.de)
    Type: DNS Search List Option (31)
    Length: 4 (32 bytes)
    Reserved
    Lifetime: 600
    Domain Names: demo.bieringer.de
    Padding
```

IPv6-Adresskonfiguration DHCPv6

- **Server**
  - Software: ISC-DHCP (4.2.6), dibbler (0.8.4)
  - Verteilung von
    - IPv6-DNS-Server
    - DNS Search List
- **Client**
  - Software
    - dhclient (4.2.6), z.B. in Fedora 20
    - NetworkManager
  - 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;
     range6 2001:db8:0:1::/64 temporary;
     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;
}
```

IPv6-Adresskonfiguration Privacy (manuell)

- **Konfiguration (manuell)**
  - auf CLI

```
# 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-Adresskonfiguration Privacy (permanent)

- **Konfiguration (permanent)**
  - Änpassung der Datei /etc/sysctl.conf
  - pro Schnittstelle (muß zum Zeitpunkt bereits existieren!)

```
net.ipv6.conf.eth0.use_tempaddr=2
```

  - für alle Schnittstellen

```
net.ipv6.conf.all.use_tempaddr=2
net.ipv6.conf.default.use_tempaddr=2
```

  - Aktivieren auf CLI

```
# sysctl -p
```

  - Prüfung auf CLI

```
# sysctl -a |grep tempaddr
net.ipv6.conf.all.use_tempaddr = 2
net.ipv6.conf.default.use_tempaddr = 2
net.ipv6.conf.eth0.use_tempaddr = 2
net.ipv6.conf.eth1.use_tempaddr = 2
net.ipv6.conf.lo.use_tempaddr = 2
```

  - Restart der Schnittstelle oder Reboot notwendig!

Dr. Peter Bieringer – IPv6 mit Linux (Tutorial) – IPv6-Kongress – 22. - 23. Mai 2014, Frankfurt/Main, Deutschland 23.05.14 20:16:23 14

Den sysctl-Aufruf von udev ausführen zu lassen, funktioniert nur, wenn das IPv6-Modul bereits geladen ist.

- Bei Fedora 14 funktioniert das nicht, weil udev schon von der initialen RAM-Disk gestartet wird, jedoch das IPv6-Modul zuvor nicht geladen wird.
- IPV6_PRIVACY="rfc3041" in ifcfg-DEVICE funktioniert eigentlich gar nicht, da bisher nicht vernünftig implementiert. Dass es dennoch manchmal klappt, scheint von Verzögerungen im Kernel beim Hochfahren der Schnittstelle zu kommen.

Siehe auch
http://www.heise.de/ct/hotline/IPv6-Privacy-bei-Fedora-1219825.html
http://www.heise.de/ct/hotline/IPv6-anonym-1100728.html
https://bugzilla.redhat.com/show_bug.cgi?id=250919

IPv6-Adresskonfiguration Privacy (NetworkManager)

- **Konfiguration via NetworkManager** (0.9.9.1-5.git20140319.fc21)
  - Vorhandene Interfaces

```
# nmcli connection
NAME    UUID                                  TYP             GERÄT
ens4v1  d0fc2b2e-5fa0-4675-96b5-b723ca5c46db  802-3-ethernet  ens4v1
```

  - Aktuelle IPv6-Privacy-Konfiguration

```
# ip -o addr show dev ens4v1 | grep temporary | wc -l
0
# nmcli connection show ens4v1 |grep ip6-privacy
ipv6.ip6-privacy:                       -1 (unbekannt)
```

  - Anpassung & Restart des Interfaces

```
# nmcli connection modify ens4v1 ipv6.ip6-privacy 2
# nmcli connection down ens4v1; nmcli connection up ens4v1
```

  - Neue IPv6-Privacy-Konfiguration

```
# nmcli connection show ens4v1 |grep ip6-privacy
ipv6.ip6-privacy:                       2 (aktiviert, temporäre IP bevorzugen)
# ip -o addr show dev ens4v1 | grep temporary | wc -l
2
```

Dr. Peter Bieringer – IPv6 mit Linux (Tutorial) – IPv6-Kongress – 22. - 23. Mai 2014, Frankfurt/Main, Deutschland 23.05.14 20:16:23 15

Siehe auch
https://fedoraproject.org/wiki/Tools/NetworkManager/IPv6

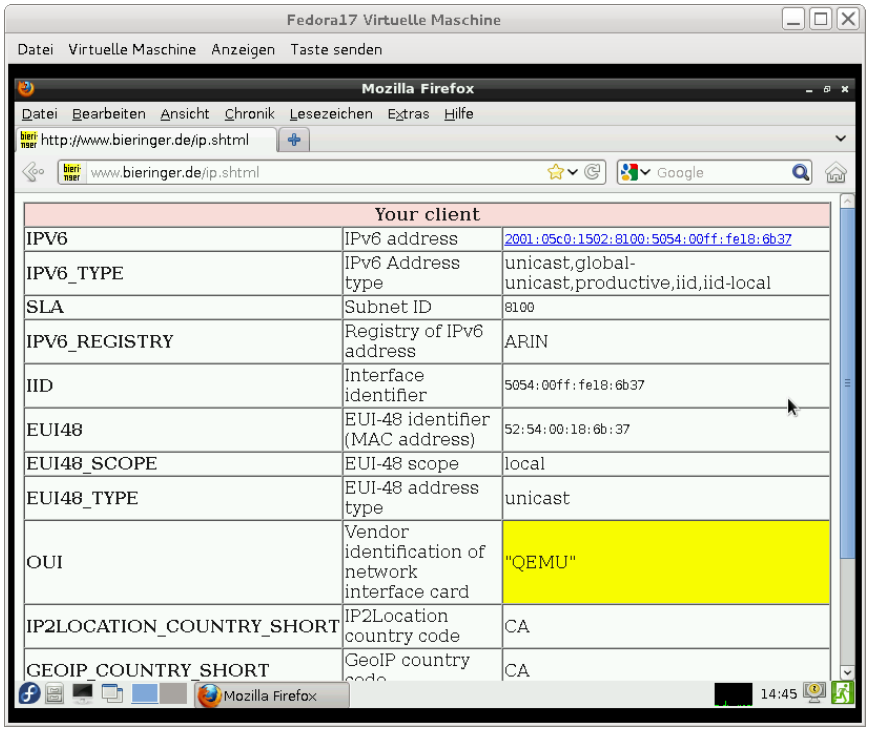
IPv6-Adresskonfiguration Privacy (Test & Status)

- **IPv6 Privacy Test**
  - Test: http://ip.bieringer.de/
    - EUI64_SCOPE: iid-privacy

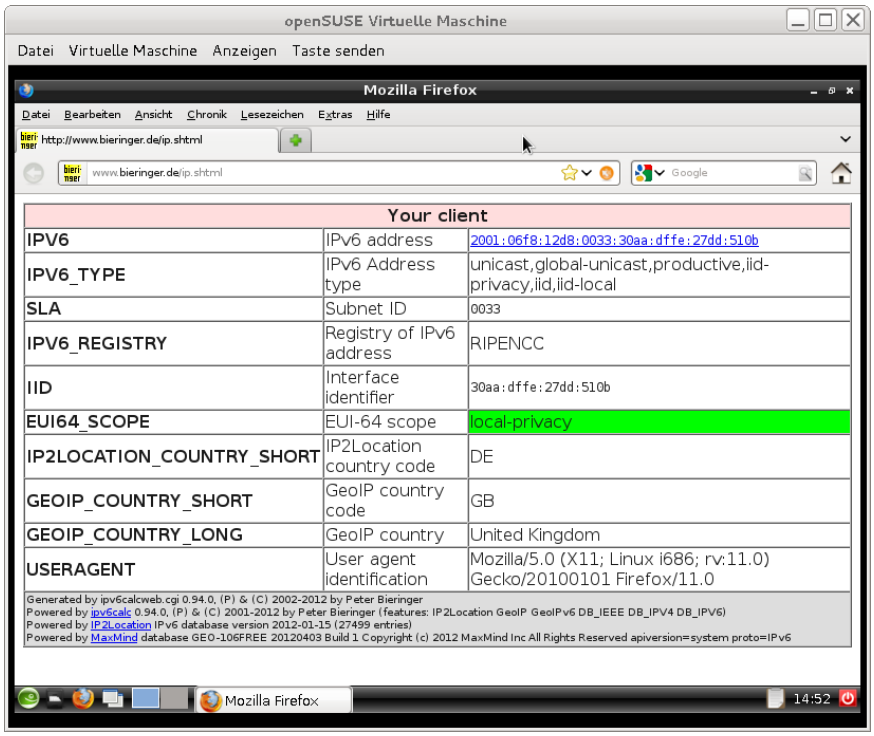
- **IPv6 Privacy Status nach Installation**


CentOS/RHEL:	5, 6, 7	
Fedora:	16, 17, 18, 19, 20	
Ubuntu:	11.10, 12.04, 13.04	14.04
Debian	6.0.4	
openSUSE:		12.1, 12.3, 13.1

IPv6-Adresskonfiguration Privacy – Test (1)



IPv6-Adresskonfiguration Privacy – Test (2)



Your client		
IPV6	IPv6 address	2001:06f8:12d8:0033:30aa:dffe:27dd:510b
IPV6_TYPE	IPv6 Address type	unicast,global-unicast,productive,iid-privacy,iid,iid-local
SLA	Subnet ID	0033
IPV6_REGISTRY	Registry of IPv6 address	RIPENCC
IID	Interface identifier	30aa:dffe:27dd:510b
EUI64_SCOPE	EUI-64 scope	local-privacy
IP2LOCATION_COUNTRY_SHORT	IP2Location country code	DE
GEOIP_COUNTRY_SHORT	GeoIP country code	GB
GEOIP_COUNTRY_LONG	GeoIP country	United Kingdom
USERAGENT	User agent identification	Mozilla/5.0 (X11; Linux i686; rv:11.0) Gecko/20100101 Firefox/11.0

Generated by ipv6calcweb.cgi 0.94.0, (P) & (C) 2002-2012 by Peter Bieringer
Powered by ipv6calc 0.94.0, (P) & (C) 2001-2012 by Peter Bieringer (features: IP2Location GeoIP GeoIPv6 DB_IEEE DB_IPV4 DB_IPV6)
Powered by IP2Location IPv6 database version 2012-01-15 (27499 entries)
Powered by MaxMind database GEO-106FREE 20120403 Build 1 Copyright (c) 2012 MaxMind Inc All Rights Reserved apiversion=system proto=IPv6

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

Werkzeug: ip

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

Beispiele

```
# ip -6 route add 2001:db8:0:2::/64 via 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
```


•Werkzeug: **route**

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

•Beispiele

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

•Ergebnis

```
# 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
```

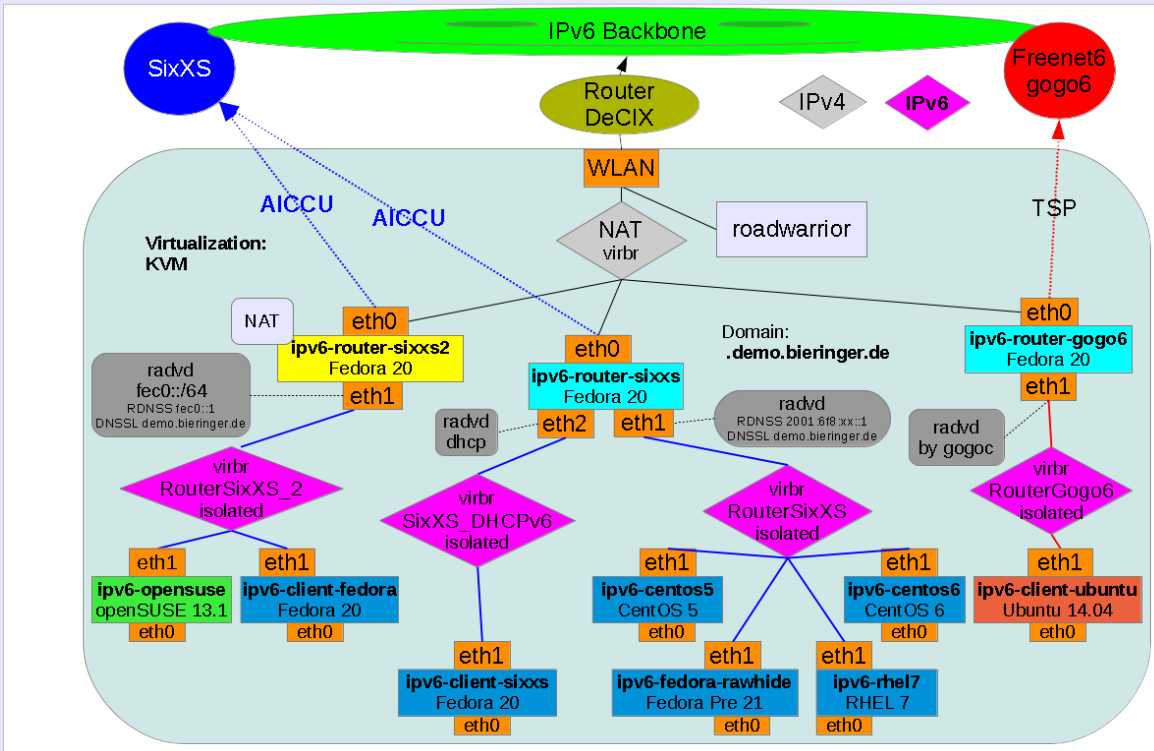
IPv6
Address Selection

- **Destination Address Selection**
  - Mehr als eine IPv6-Adresse in DNS-Antwort via *getaddrinfo*
    - Ja: RFC 3484 *Default Address Selection for Internet Protocol version 6*
      - Beeinflussung der Sortierung mit Hilfe von /etc/gai.conf
- **Source Address Selection**
  - Mehr als ein "globaler" Prefix pro Interface
    - Ja: Möglichkeit der Bindung zu Ziel-Präfix mit: **`ip addrlabel`**
  - Beispiel von 2 Bindungen

```
# ip addrlabel
...
prefix 2a01:238:423d:8800::/64 label 300   # dst-B
prefix 2001:4dd0:ff00:834::/64 label 200   # dst-A
prefix 2001:6f8:900:8cbc::/64  label 300   # src-B
prefix 2001:6f8:12d8:2::/64    label 200   # src-A
...
```

Siehe auch: http://mirrors.deepspace6.net/Linux+IPv6-HOWTO/resolver.html

Demo Aufbau


IPv6 Backbone
SixXS
Router DeCIX
IPv4
IPv6
Freenet6 gogo6
WLAN
AICCU
AICCU
NAT virbr
roadwarrior
TSP
Virtualization: KVM
NAT
eth0
ipv6-router-sixxs2 Fedora 20
eth1
radvd fec0::/64 RDNSS fec0::1 DNSSL demo.bieringer.de
eth0
ipv6-router-sixxs Fedora 20
Domain: .demo.bieringer.de
eth0
ipv6-router-gogo6 Fedora 20
eth1
radvd dhcp
eth2
eth1
radvd RDNSS 2001:6f8:xx::1 DNSSL demo.bieringer.de
radvd by gogoc
virbr RouterSixXS_2 isolated
virbr SixXS_DHCPv6 isolated
virbr RouterSixXS isolated
virbr RouterGogo6 isolated
eth1
ipv6-opensuse openSUSE 13.1
eth0
eth1
ipv6-client-fedora Fedora 20
eth0
eth1
ipv6-client-sixxs Fedora 20
eth0
eth1
ipv6-centos5 CentOS 5
eth0
eth1
ipv6-fedora-rawhide Fedora Pre 21
eth0
eth1
ipv6-centos6 CentOS 6
eth0
eth1
ipv6-rhel7 RHEL 7
eth0
eth1
ipv6-client-ubuntu Ubuntu 14.04
eth0
Dr. Peter Bieringer − IPv6 mit Linux (Tutorial) − IPv6-Kongress − 22. - 23. Mai 2014, Frankfurt/Main, Deutschland
23.05.14 20:16:23
22

Demo IPv6 DHCP

- **IPv6-only Client-Konfiguration**
  - Fedora >= 16: /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 – 22. - 23. Mai 2014, Frankfurt/Main, Deutschland 23.05.14 20:16:23 23

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.lease
-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"`

Andere Distributionen:

- OpenSUSE 12: BOOTPROTO='dhcp6'

Demo IPv6 DHCP (Probleme)

- **DHCPv6-Server**
  - IPv6-Firewall blockt DHCPv6-Anfragen
    - udp/547 erlauben von fe80::/64

```
-A INPUT -s fe80::/64 -p udp -m udp --dport 547 -j ACCEPT
```

- **DHCPv6-Client**
  - DHCPv6 wird von ip6tables nicht als stateful erkannt
    - Firewall anpassen für eingehende Pakete vom DHCPv6-Server
      - Fedora < 17
      - OpenSUSE 12
    - udp/546 erlauben von fe80::/64

```
-A INPUT -d fe80::/64 -p udp -m udp --dport 546 -j ACCEPT
```


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
```

Dr. Peter Bieringer – IPv6 mit Linux (Tutorial) – IPv6-Kongress – 22. - 23. Mai 2014, Frankfurt/Main, Deutschland 23.05.14 20:16:23 28

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
```

Dr. Peter Bieringer – IPv6 mit Linux (Tutorial) – IPv6-Kongress – 22. - 23. Mai 2014, Frankfurt/Main, Deutschland 23.05.14 20:16:23 29

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)
```

Dr. Peter Bieringer – IPv6 mit Linux (Tutorial) – IPv6-Kongress – 22. - 23. Mai 2014, Frankfurt/Main, Deutschland 23.05.14 20:16:23 31

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:

```
# tshark -n -V icmp6
```

IPv6-Anbindung

IPv6-Anbindung

Typ	Tunneltyp	IPv4-Adresse	IPv6-Präfix	IPv6-Netz	Anbieter bzw. Methode
Nativ		Statisch	Statisch	ja	ISP (Firmenanschluß)
Nativ		Dynamisch	Dynamisch	ja	ISP via DHCPv6
Nativ		Dynamisch	Statisch	ja	ISP via DHCPv6
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

- **Firmenkunden (entsprechender Anschluß):**
  - ISP routet direkt Präfixe über die Anbindung
- **Privatkunden**
  - IPv6 über DHCPv6 (z.B. Mnet: /56)
- **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 (2011)
- **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)

Installation unter Fedora
```
# yum install aiccu
```

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.

Start:
```
# service aiccu start
```

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)

Installation unter Fedora
```
# yum install gogoc
```

Wichtige Parameter in **`/etc/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.

Start:
```
# service gogoc start
```

Vordergrund-Modus:
```
# /usr/bin/gogoc -n -f /etc/gogoc/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 hinterlegen

IPv6-Anbindung Teredo

- **URLs:**
  - http://de.wikipedia.org/wiki/Teredo
- **Kein Subnetz möglich (eigentlich)**
  - Workaround: IPv6-NAT
- **Mögliche Tunnelmethoden**
  - IPv6-in-UDP-in-IPv4 (sog. Teredo-Bubbles)
    - Port: 3544/udp
      - Zusätzlich hinter NAT “high-ports” ausgehend notwendig
- **Linux-Client: miredo-client**

Installation unter Fedora
```
# yum install miredo
```

Wichtige Parameter in: /etc/miredo/miredo.conf
```
InterfaceName teredo
ServerAddress teredo.ipv6.microsoft.com
```

Aufruf:
```
# service miredo-client start
```

Demo IPv6-Anbindung

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

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:           [fd00::]/7
```

Dr. Peter Bieringer – IPv6 mit Linux (Tutorial) – IPv6-Kongress – 22. - 23. Mai 2014, Frankfurt/Main, Deutschland 23.05.14 20:16:24 42

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**
  - “ipv6loganon” (im Paket “ipv6calc”) ab 0.90.0 kann Logfiles on-the-fly anonymisieren
    - Standard in httpd.conf

```
CustomLog logs/access_log combined
```

    - Mit Anonymisierung

```
CustomLog "|/usr/bin/ipv6loganon -f -a /var/log/httpd/access_log"  ¬
  combined
```

    - Mit Anonymisierung und “cronolog”

```
CustomLog "|/usr/bin/ipv6loganon -f |/usr/sbin/cronolog            ¬
  /var/log/httpd/access.log-%Y%m%d" combined
```


Optionen „ipv6loganon“
-a : append to file
-f : flush after each line

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!
  - (Noch) Kein impliziter Schutz durch NAT vorhanden!
  - System ist vom Internet aus (noch) 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 20 Standardinstallation

```
Aktive Internetverbindungen (Nur Server)
Proto Recv-Q Send-Q Local Address   Foreign Address  State  PID/Program name
tcp        0      0 0.0.0.0:58755   0.0.0.0:*        LISTEN 710/rpc.statd
tcp        0      0 0.0.0.0:111     0.0.0.0:*        LISTEN 695/rpcbind
tcp        0      0 0.0.0.0:22      0.0.0.0:*        LISTEN 764/sshd
tcp        0      0 127.0.0.1:631   0.0.0.0:*        LISTEN 1692/cupsd
tcp6       0      0 :::111          :::*             LISTEN 695/rpcbind
tcp6       0      0 :::22           :::*             LISTEN 764/sshd
tcp6       0      0 ::1:631         :::*             LISTEN 1692/cupsd
tcp6       0      0 :::33689        :::*             LISTEN 710/rpc.statd
udp        0      0 0.0.0.0:57652   0.0.0.0:*               710/rpc.statd
udp        0      0 127.0.0.1:323   0.0.0.0:*               544/chronyd
udp        0      0 0.0.0.0:68      0.0.0.0:*               901/dhclient
udp        0      0 0.0.0.0:52806   0.0.0.0:*               531/avahi-daemon: r
udp        0      0 0.0.0.0:61534   0.0.0.0:*               901/dhclient
udp        0      0 0.0.0.0:867     0.0.0.0:*               695/rpcbind
udp        0      0 0.0.0.0:111     0.0.0.0:*               695/rpcbind
udp        0      0 127.0.0.1:888   0.0.0.0:*               710/rpc.statd
udp        0      0 0.0.0.0:123     0.0.0.0:*               544/chronyd
udp        0      0 0.0.0.0:5353    0.0.0.0:*               531/avahi-daemon: r
udp6       0      0 ::1:323         :::*                    544/chronyd
udp6       0      0 :::37975        :::*                    710/rpc.statd
udp6       0      0 :::867          :::*                    695/rpcbind
udp6       0      0 :::111          :::*                    695/rpcbind
udp6       0      0 :::123          :::*                    544/chronyd
udp6       0      0 :::31692        :::*                    901/dhclient
```


Ausgabe von # netstat -nlptu

"rpcbind" 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 (fec0::/10, fd00::/7)
  - 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 (Februar 2007)
    - Red Hat EL 4 (2.6.9) und 5 (2.6.18): nur stateless!
  - Network Address Translation (NAT) ab Kernel 3.9.0 (April 2013)
    - in Verbindung mit iptables 1.4.18
  - Neues Framework "nftables" seit Kernel 3.13 (April 2014) NEW!
- **Benutzer-Werkzeuge**
  - *iptables* (IPv4) bzw. *ip6tables* (IPv6)
  - *nft* (IPv4 & IPv6) NEW!

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 (1)

- **Distributionseigene Werkzeuge**
  - Red Hat & Clones:
    - bis 5.x: system-config-securitylevel-tui
    - 6.x: system-config-firewall-tui
    - 7.x: firewallD (firewall-cmd)
  - Fedora:
    - bis 16: system-config-firewall-tui
    - ab 17: firewallD (firewall-cmd)
  - openSUSE: SuSEfirewall2

Firewalling Regelsatzgeneratoren (2)

- **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

Beispiel für IPv6 Regelwerk (ip6tables)

- **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 ohne Source-Filter:**

```
-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

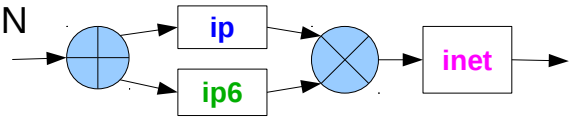
Kann ggf. verfeinert werden durch Einschränkung des Hop-Limits (IPv6: TTL):

-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).

Besonderheiten von nftables

- **Dedizierte Tabellen für IPv4 und IPv6**
  - `table ip`
  - `table ip6`
- **Gemeinsame Tabelle für IPv4 und IP6** 
  - `table inet`
    - Protokol-Auswahl möglich mit
      - `meta nfproto ipv4`
      - `meta nfproto ipv6`
- **Paket durchläuft BEIDE relevanten Tabellen** 
  - Achtung: "accept" in BEIDEN Tabellen notwendig
  - Tipp
    - Verwendung von "marks" in den dedizierten Tabellen
    - "accept" von markierten Paketen in Tabelle "inet"



Beispiel für IPv6 Regelwerk (nftables)

- **Minimales Regelwerk (nur Nutzung von Tabelle "inet"):** 
  - Connection Tracking für IPv4 und IPv6

```
# nft add rule inet filter input ct state established,related counter accept
```

  - Wichtige ICMPv6-Typen

```
# nft add rule inet filter input meta nfproto ipv6 icmpv6 type echo-request accept
# nft add rule inet filter input meta nfproto ipv6 icmpv6 type { nd-neighbor-advert, nd-neighbor-solicit, nd-router-advert} accept
```

  - IPv4-Ping

```
# nft add rule inet filter input meta nfproto ipv4 icmp type { echo-request } accept
```

  - Erlauben eingehend SSH ohne Source-Filter:

```
# nft add rule inet filter input tcp dport 22 ct state new tcp flags \& \(syn \| ack\) == syn accept
```

  - Finale Reject-Regel

```
# nft add rule inet filter input counter reject
```

Eine Regel für IPv4 und IPv6

- **Widrigkeiten**
  - Überlagerung von existierendes ip(6)tables-Regelwerk
  - Kein Loglevel in Log-Option => default: kern.emerg

Dr. Peter Bieringer – IPv6 mit Linux (Tutorial) – IPv6-Kongress – 22. - 23. Mai 2014, Frankfurt/Main, Deutschland 23.05.14 20:16:24 57

Erlauben aller eingehender ICMPv6 -Pakete ist bei nftablesaktuell NICHT möglich, "any" scheint nicht implementiert zu sein, somit muß ICMPv6 Type explizit angegeben werden, kann ggf. verfeinert werden durch Einschränkung des Hop-Limits (IPv6: TTL):

nft add rule ip6 filter input icmpv6 type { nd-neighbor-solicit, nd-router-advert, nd-neighbor-advert } **ip6 hoplimit 1** counter accept

nft add rule ip6 filter input icmpv6 type { nd-neighbor-solicit, nd-router-advert, nd-neighbor-advert } **ip6 hoplimit 255** counter accept

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

Bei ersten Tests vor Aktivieren der Log-Option Syslog-Konfiguration prüfen und folgende Zeile deaktivieren (/etc/rsyslog.conf) und syslog neu starten

```
#*.emerg                       :omusrmsg:*
```

(ansonsten kann das Logging die Konsole fluten und das System unbrauchbar machen!)

Beispiel für IPv6 netfilter NAT

- **Funktion gleichwertig zu IPv4**
  - IPv6 Masquerading

```
# ip6tables -t nat -I POSTROUTING -o sixxs -s fec0::/64 -j MASQUERADE
```

    - Maskierung interner IPv6-Clients durch Router-Adresse

  - IPv6 Ziel Weiterleitung

```
# ip6tables -t nat -A PREROUTING
  -d 2001:db8:0:1:5054:ff:fe01:2345 -i sixxs
  -j DNAT --to-destination fec0::5054:ff:fe01:2345
```

    - Weiterleitung externer IPv6-Adresse an interne Adresse

  - IPv6 Ziel/Port Weiterleitung

```
# ip6tables -t nat -A PREROUTING -i sixxs
  -p tcp --dport 8080
  -j DNAT --to-destination [fec0::1234]:80
```

    - Weiterleitung von Port 8080 via Router nach intern auf Port 80

Dr. Peter Bieringer – IPv6 mit Linux (Tutorial) – IPv6-Kongress – 22. - 23. Mai 2014, Frankfurt/Main, Deutschland 23.05.14 20:16:24 58

Weitere Möglichkeiten

```
$ man iptables-extensions
```

Beispiele für nftables
http://kernelnewbies.org/nftables_examples

Demo IPv6-NAT

- **Fedora 20 als Router**
- **Fedora 20 (2. Installation)**
  - als Host: 1:1 NAT
  - als Client: Masquerading
- **OpenSUSE**
  - als Host: 1:1 NAT
  - als Client: Masquerading
  - als Server: Portweiterleitung von 8080 auf 80

IPv6 in virtualisierten Umgebungen (libvirt, VirtualBox)

Status von libvirt / virt-manager

- **Allgemein**
  - IPv6 ist immer noch ein Stiefkind
- **libvirt 1.1.3.4 (Fedora 20)**
  - IPv6-Konfiguration via virtsh (bzw. XML-Datei)
  - libvirt erzeugt passenden ip6tables-Einträge für Bridges NUR wenn IPv6 explizit im XML konfiguriert
    - inkonsistent zu IPv4 
- **virt-manager 1.0.1 (Fedora 20)**
  - IPv6-Unterstützung fehlt (noch immer) 

Dr. Peter Bieringer – IPv6 mit Linux (Tutorial) – IPv6-Kongress – 22. - 23. Mai 2014, Frankfurt/Main, Deutschland 23.05.14 20:16:24 61

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
```

Einfache Variante für Firewall-Daemon:

```
# firewall-cmd --zone dmz --add-interface=virbr+
```

Je nach Version des Firewall-Daemon:

```
# firewall-cmd --direct --add-rule ipv6
    filter FWDO_ZONE_dmz 1 -j ACCEPT
# firewall-cmd --direct --add-rule ipv6
    filter FWDI_ZONE_dmz 1 -j ACCEPT
```

oder

```
# firewall-cmd --direct --add-rule ipv6
    filter FWDO_dmz 1 -j ACCEPT
# firewall-cmd --direct --add-rule ipv6
    filter FWDI_dmz 1 -j ACCEPT
```

möglicherweise ist zusätzlich noch notwendig:

```
# ip6tables -t raw -I PREROUTING -j ACCEPT
```

Status von VirtualBox

- **Aktuell verfügbar: 4.3.10**
  - Host-Only-Netzwerk “vboxnetX”
    - Zuweisung einer IPv6-Adresse möglich 
  - Interne Netzwerke sind **nicht** als Linux-Bridges realisiert
    - Keine Probleme mit Host-Firewall 

Weitere Informationen

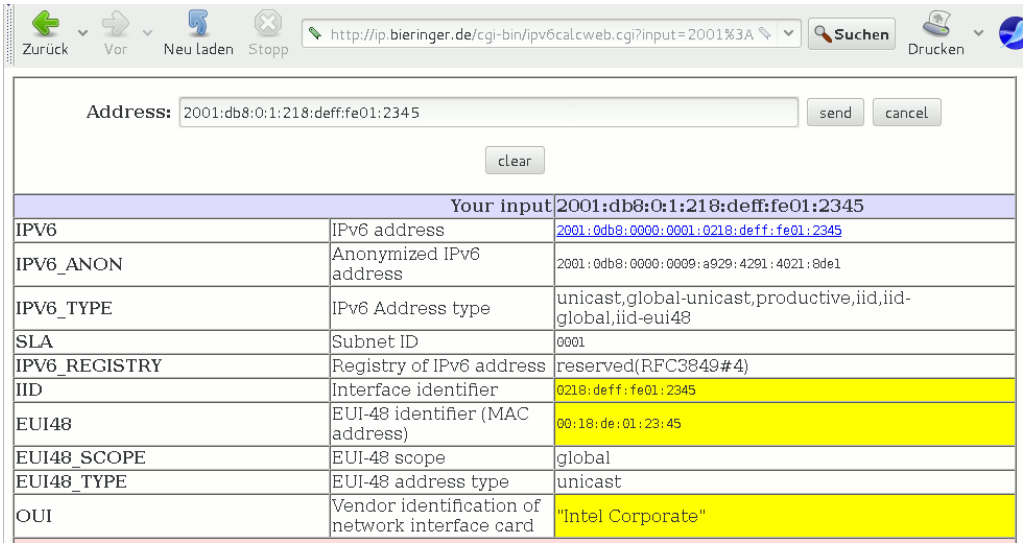
IPv6 & Linux bezogene Information

- ***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)
    http://mirrors.deepspace6.net/howtos/ (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

ipv6calc Online Tool

- **URL: http://ip.bieringer.de/cgi-bin/ipv6calcweb.cgi**
  - Betrieb von ipv6calcweb.cgi im “Form”-Modus
  - Information über MAC, IPv4 & IPv6-Adressen



Your input	2001:db8:0:1:218:deff:fe01:2345
IPV6	IPv6 address	2001:0db8:0000:0001:0218:deff:fe01:2345
IPV6_ANON	Anonymized IPv6 address	2001:0db8:0000:0009:a929:4291:4021:8de1
IPV6_TYPE	IPv6 Address type	unicast,global-unicast,productive,iid,iid-global,iid-eui48
SLA	Subnet ID	0001
IPV6_REGISTRY	Registry of IPv6 address	reserved(RFC3849#4)
IID	Interface identifier	0218:deff:fe01:2345
EUI48	EUI-48 identifier (MAC address)	00:18:de:01:23:45
EUI48_SCOPE	EUI-48 scope	global
EUI48_TYPE	EUI-48 address type	unicast
OUI	Vendor identification of network interface card	"Intel Corporate"

Dr. Peter Bieringer – IPv6 mit Linux (Tutorial) – IPv6-Kongress – 22. - 23. Mai 2014, Frankfurt/Main, Deutschland 23.05.14 20:16:24 65

„Form“-Modus ist neues Feature in „ipv6calcweb.cgi“ seit Version 0.93

Projekt-Homepage: http://www.deepspace6.net/projects/ipv6calc.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 bzw. über Veranstalter erhältlich!

Dankeschön an

Jürgen Seeger, iX (Einladung)

