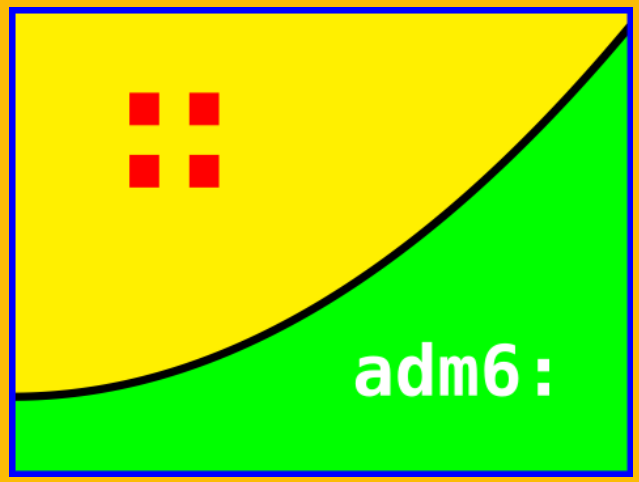
adm6:

IPv6 – packetfiltering managed by Python



Johannes Hubertz

hubertz-it-consulting GmbH

GUUG FFG 2012, München

adm6: Managementsummary

IPv6-Netzwerksicherheit für **alle Systeme** im **Unternehmen**

Zukunft: IPv6 ist die **Zukunft** auch Ihres Netzwerks!

Verteilt: **Alle Geräte** im Unternehmen mit IPv6-Paketfiltern

Flexibel: **Beliebige Betriebssysteme** und Filterarchitekturen dank Python

Zentral: **Einfache Administration** von einem einzelnen Gerät aus

Nutzen: Nur noch erwünschter, **nutzbringender IPv6-Datenverkehr**

Rentabel: Wirtschaftlichkeit und Investitionssicherheit durch **Freie Software**

Fazit: An der Zukunft führt kein Weg vorbei –

mit adm6: wird Ihr Weg etwas sicherer!

Was zu zeigen ist . . .

Vorstellung – Wer zeigt hier was?

Motivation – Warum das alles?

Ein Konzept

Drei Schritte: Lesen, Kreuzprodukt, Generierung

Ausblick

Quellen und Hinweise

Vorstellung: Johannes Hubertz

1954 in Köln-Lindenthal geboren

1973 Studium der Elektrotechnik, RWTH und FH Aachen

1980 Anstellung bei der Bull AG

1981 HW-Reparatur, ASM80, PLM80, Xenix, bourne-shell, C

1994 Erstkontakt mit IPv4

1996 Xlink, root@www.bundestag.de, . . .

1997 X.509 mit SSLeay, ipfwadm mit shell-scripts

1998 „Ins Allerheiligste“, iX 1/1998, Heise Verlag

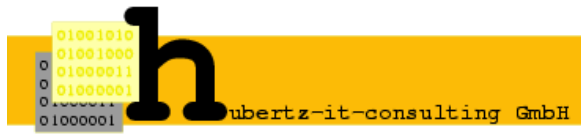
1999 IT-Security Manager Bull D-A-CH

2002 Start der Entwicklung von http://sspe.sourceforge.net

2005 Gründung der hubertz-it-consulting GmbH

seit 1973 Bundesanstalt Technisches Hilfswerk in Köln-Porz

seit 2001 Segeln, am liebsten auf Salzwasser



Vorstellung: hubertz-it-consulting GmbH

Erkenntnisse aus dem Berufsleben

Bellovin and Cheswick: Firewalls and Internet Security, 1994

Fazit: Keep it simple!

Oder mit Einstein: So einfach wie möglich, aber nicht einfacher!

Etwas Erfahrung war Voraussetzung

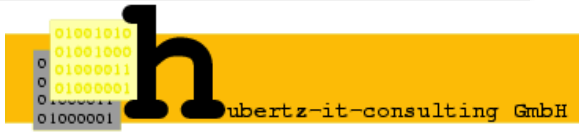
Gründung am 8. August 2005, Sitz in Köln

Geschäftsinhalt: Dienstleistungen im Umfeld der IT-Sicherheit

Logo: Johannes Hubertz Certificate Authority als ASCII-Bitmuster

Diese Bits finden sich in einigen 10^4 X.509 Anwenderzertifikaten bei der Kundschaft in der Seriennummer wieder

Wir sind käuflich ;-)



Einst träumte ich . . . (2002)

Wer Visionen hat, soll zum Arzt gehen
(Helmut Schmidt)

Definitionen in ASCII-Dateien: (Name, Adresse, Kommentar)

Filterregeln in ASCII-Dateien: (src, dest, proto, port, action, comm.)

Erledigt für IPv4: seit März 2003 `http://sspe.sourceforge.net`

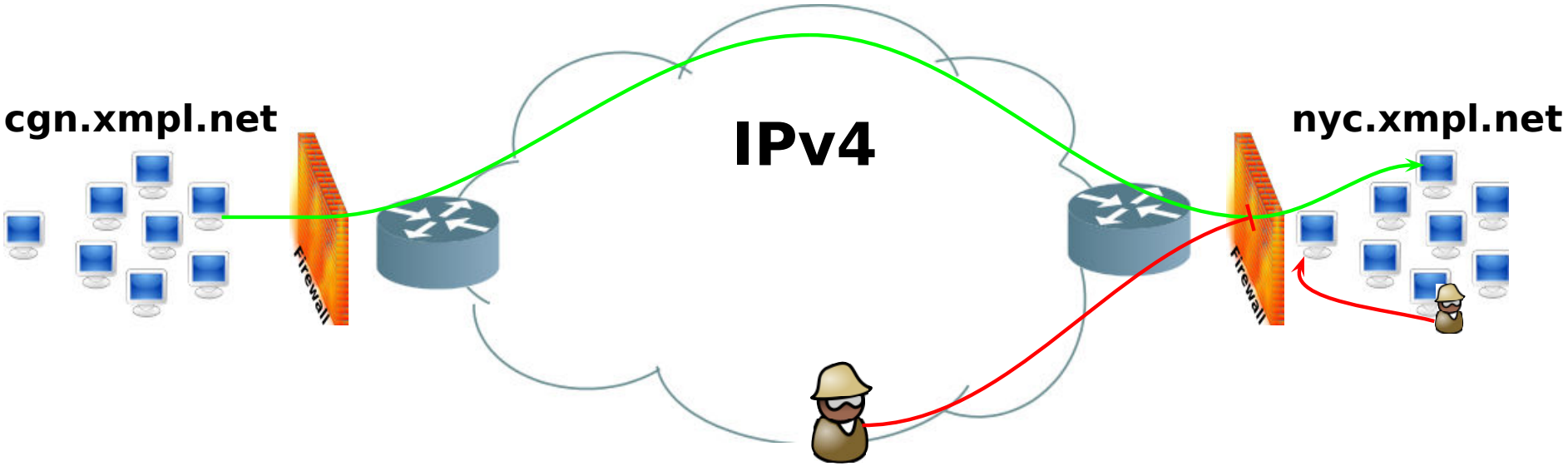
implementiert in Shell und Perl, etwas schwierig für Einsteiger

bei mehreren Kunden erfolgreich im Einsatz

regelmäßig Downloads bei sf.net

Es war einmal ein **IPv4** mit Firewalls und internen . . .

Alles wird gut?



Mit IPv6 wird **alles** anders!

IPv6 . . .

- ist genauso sicher wie IPv4
- ist genauso unsicher wie IPv4
- bietet keinen fragwürdigen Schutz durch NAT
- ist immer Ende zu Ende Kommunikation
- wird genutzt, machmal sogar, ohne dass man es bemerkt
- bietet die gleichen Applikationen und Schwachstellen wie IPv4

Ergo wollen wir keinen ungefilterten Verkehr in unserem Netz!

IPv6 filtern, wo denn?

Menschen mit einer neuen Idee gelten so lange als Spinner,
bis sich die Sache durchgesetzt hat. (Mark Twain)

Wir filtern auf der Firewall, da ist alles sicher!

Wir filtern auf der Firewall und auf den Routern, da ist alles sicher!

auf der Firewall, auf den Routern, auf den Servern, da ist alles sicher!

Wirklich sicher?

Warum nicht auf jedem Gerät?

Zuviel Aufwand? Mit Sicherheit nicht, wenn die Geräte

- über eine sichere Methode verfügen, Kommunikation zu betreiben
- über eine sichere Methode verfügen, Konfiguration zu bearbeiten
- administrativ zu einem Hoheitsbereich gehören

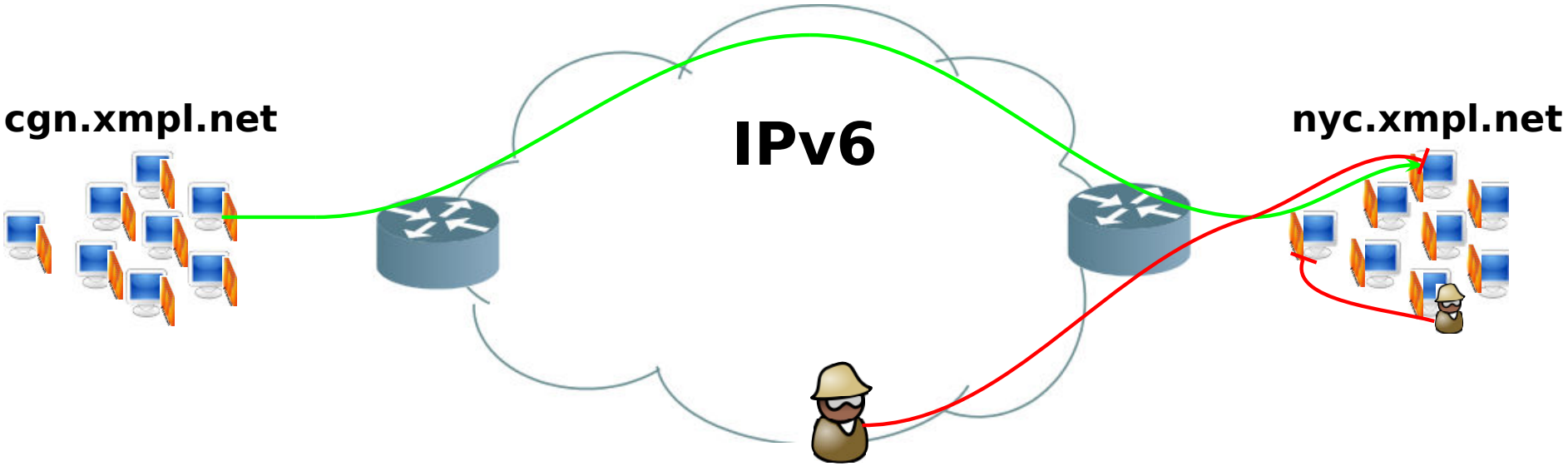
Wir bevorzugen es, auf jedem Gerät zu filtern. . . **wirklich!** . . .

überall!

IPv6 filtern, womit denn?

system	filter	command
Linux	NetFilter	ip6tables
OpenBSD	pf	pf, pf.conf, rc.local
Free- u. NetBSD	ipfilter	ipf
Win XP/SP3	onboard	netsh firewall . . .
Win 7	onboard	netsh advfirewall . . .
IOS 12.+	cisco-acl	access-list . . .
. . .	. . .	. . .

adm6: Eine Idee wird zum Konzept . . .

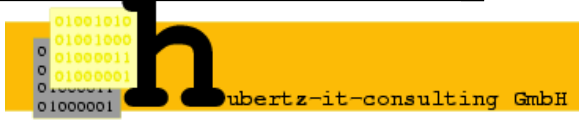


adm6: Fünfpunktekonzept

Jedes Gerät nutzt einen internen Paketfilter!

1. Alle Paketfilter werden zentral erzeugt und verwaltet
2. Alle Hosts, Router, Firewalls im Netz sind mit Paketfiltern zu betreiben
3. Interface- und Routinginformationen sind Berechnungsgrundlage
4. Definition aller beteiligten **Kommunikatoren** (Namen, Adressen)
5. Definition(en) aller **Kommunikationen** im Netz (Regelsatz)

Nur erlaubter Netzwerkverkehr bleibt übrig!



adm6: Fünfpunktekonzept

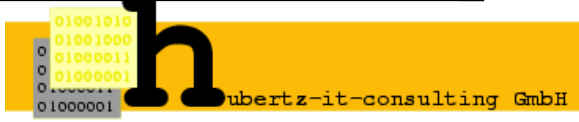
Jedes Gerät nutzt einen internen Paketfilter!

1. Alle Paketfilter werden zentral erzeugt und verwaltet

   Python-Script generiert Filtersequenz, mit ssh wird verteilt

   Ablaufplanung wie folgt:

Nur erlaubter Netzwerkverkehr bleibt übrig!



adm6: Drei Schritte . . .

Lesen aller Parameter

Kreuzprodukt bilden

Generierung pro Gerät

Was ist zu Lesen?

Globale Konfiguration aller Geräte

Interfaces und Routen aller Geräte

Definitionen aus der Datei: **`hostnet6`**

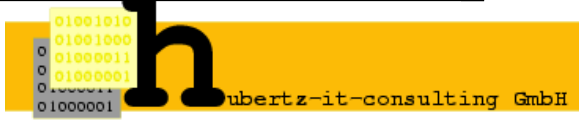
Regeln aus den Dateien: **`nn-rules.*`**

adm6: Fünfpunktekonzept

Jedes Gerät nutzt einen internen Paketfilter!

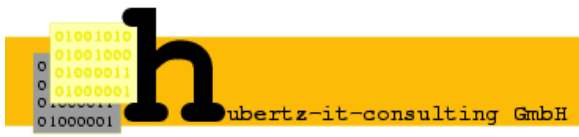
1. Alle Paketfilter werden zentral erzeugt und verwaltet
   Python-Script generiert Filtersequenz, mit ssh wird verteilt
2. Alle Hosts, Router, Firewalls im Netz sind mit Paketfiltern zu betreiben
   Ein Verzeichnis pro Gerät: ~/adm6/desc/gerätename

Nur erlaubter Netzwerkverkehr bleibt übrig!



adm6: Datei- und Verzeichnisstrukturen

```
.adm6.conf
adm6

adm6/bin/
adm6/desc/
adm6/etc/

adm6/desc/adm6/
adm6/desc/ns/
adm6/desc/sfd/
adm6/desc/r-ex/
adm6/desc/obi-lan/

adm6/desc/ns/00-rules.admin
adm6/desc/ns/mangle-startup
adm6/desc/ns/mangle-endup
adm6/desc/ns/hostnet6
adm6/desc/ns/interfaces
adm6/desc/ns/routes

adm6/desc/sfd/00-rules.admin
adm6/desc/sfd/hostnet6
adm6/desc/sfd/interfaces
adm6/desc/sfd/routes

adm6/desc/r-ex/00-rules.admin
adm6/desc/r-ex/hostnet6
adm6/desc/r-ex/interfaces
adm6/desc/r-ex/routes

adm6/desc/obi-lan/00-rules.admin
adm6/desc/obi-lan/mangle-startup
adm6/desc/obi-lan/mangle-endup
adm6/desc/obi-lan/hostnet6
adm6/desc/obi-lan/interfaces
adm6/desc/obi-lan/routes

adm6/etc/00-rules.admin
adm6/etc/Debian-footer
adm6/etc/Debian-header
adm6/etc/hostnet6
adm6/etc/OpenBSD-footer
adm6/etc/OpenBSD-header
```



adm6: Globale Konfiguration

~/.adm.conf liefert:

- **Software Version**
- **Liste aller Gerätenamen**
- **Betriebssystem jeden Gerätes**
- **Aktivitäts-Status jeden Gerätes**
- **ssh-Adresse jeden Gerätes**
- **Forward-Status jeden Gerätes**
- **Asymmetrisches Routing jeden Gerätes**

Was ist zu Lesen?

Globale Konfiguration aller Geräte

Interfaces und Routen aller Geräte

Definitionen aus der Datei: **`hostnet6`**

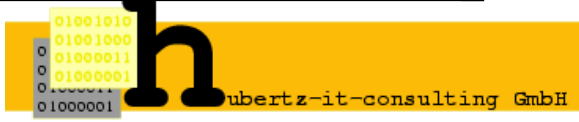
Regeln aus den Dateien: **`nn-rules.*`**

adm6: Fünfpunktekonzept

Jedes Gerät nutzt einen internen Paketfilter!

1. Alle Paketfilter werden zentral erzeugt und verwaltet
   Python-Script generiert Filtersequenz, mit ssh wird verteilt
2. Alle Hosts, Router, Firewalls im Netz sind mit Paketfiltern zu betreiben
   Ein Verzeichnis pro Gerät: ~/adm6/desc/gerätename
3. Interface- und Routinginformationen sind Berechnungsgrundlage
   ~/adm6/desc/gerätename/{interfaces,routes}

Nur erlaubter Netzwerkverkehr bleibt übrig!



Konfiguration der Schnittstelle

ifconfig-Ausgabe in der Linux-Variante (Debian)

```
eth1      Link encap:Ethernet  HWaddr 00:00:24:cc:22:0d                    1
          inet6 addr: 2001:db8:2::23/64 Scope:Global                        2
          inet6 addr: fe80::200:24ff:fecc:220d/64 Scope:Link                3
          UP BROADCAST RUNNING MULTICAST  MTU:1280  Metric:1                4
          RX packets:111977 errors:0 dropped:0 overruns:0 frame:0           5
          TX packets:97028 errors:0 dropped:0 overruns:0 carrier:0          6
          collisions:0 txqueuelen:1000                                      7
          RX bytes:17921992 (17.0 MiB)  TX bytes:10876864 (10.3 MiB)        8
          Interrupt:5 Base address:0xe200                                   9
```

ifconfig-Ausgabe in der OpenBSD-Variante

```
sis0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500          1
        lladdr 00:00:24:c8:6e:b1                                           2
        priority: 0                                                        3
        groups: egress                                                     4
        media: Ethernet autoselect (10baseT half-duplex)                   5
        status: active                                                     6
        inet 192.168.23.177 netmask 0xffffff00 broadcast 192.168.23.255    7
        inet6 fe80::200:24ff:fec8:6eb1%sis0 prefixlen 64 scopeid 0x1       8
        inet6 2001:db8:2::10 prefixlen 64                                  9
```

netsh show interface-Ausgabe in der Win-XP Variante

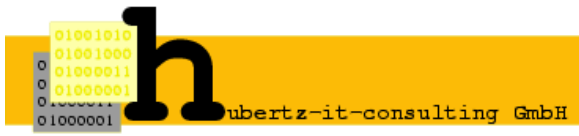
```
very verbose                                                               1
```

Lesen der Schnittstellen-Konfiguration Zeile für Zeile

```
    def interface_line(self, line):
        """evaluate one line of ifconfig-output store results in self.interfaces = []"""
        if 'Win-XP' in self.device_os:
            """german version only for now"""
            if line.startswith('Schnittstelle '):
                righthalf = line.rsplit(':')
                ifacename = righthalf.pop(-1).strip()
                self.int_name = ifacename
            else:
                items = line.split()
                if len(items) > 1:
                    targ = items.pop(-1)
                    try:
                        target = IPv6Network(targ)
                    except AddressValueError, e:
                        """no IPv6 Address in last column """
                        return
                    self.int_addr = target
                    self.interfaces.append([self.int_name, self.int_addr])
            return
        nam = re.findall('^[a-z]+[ 0-9][ :] ', line, flags=0)
        if nam:
            self.int_name = nam.pop(0).strip()
        add = []
        if 'Linux' in self.device_os:
            add = re.findall('\s*inet6\ .* Scope:*', line, flags=0)
            if add:
                ine = add.pop(0).split()
                adr = ine.pop(2)
                self.int_addr = IPv6Network(adr)
                self.interfaces.append([self.int_name, self.int_addr])
        if 'OpenBSD' in self.device_os:
            if 'inet6' in line:
                if '%' in line:
                    (le, ri) = line.split('%')
                else:
                    le = line
                ine = le.split()
```

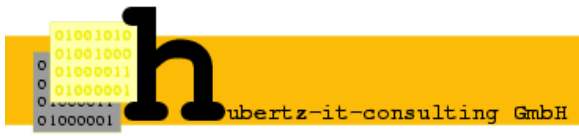
routingtable: Linux Version (Debian)

```
# ip -6 route show                                                                                                    1
2001:db8:23::/64 dev eth3  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295                                       2
2001:db8:23:1::/64 dev eth1  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295                                     3
2001:db8:23:2::/64 dev sit1  metric 1024  mtu 1480 advmss 1420 hoplimit 4294967295                                    4
2001:db8:23:3::/64 via :: dev sit1  metric 256  mtu 1480 advmss 1420 hoplimit 4294967295                              5
2001:db8:23:fa00::/56 via fe80:0:fa00::2 dev tun0  metric 1024  mtu 1500 advmss 1440 hoplimit 4294967295              6
2001:db8:23:fb00::/56 via fe80:0:fb00::2 dev tun1  metric 1024  mtu 1500 advmss 1440 hoplimit 4294967295              7
2001:db8:23:fc00::/56 via fe80:0:fc00::2 dev tun2  metric 1024  mtu 1500 advmss 1440 hoplimit 4294967295              8
2001:db8:23:fd00::/56 via fe80:0:fd00::2 dev tun3  metric 1024  mtu 1500 advmss 1440 hoplimit 4294967295              9
2001:db8:23:fe00::/56 via fe80:0:fe00::2 dev tun4  metric 1024  mtu 1500 advmss 1440 hoplimit 4294967295              10
2001:db8:23:ff00::/56 via fe80:0:ff00::2 dev tun5  metric 1024  mtu 1500 advmss 1440 hoplimit 4294967295              11
unreachable 2001:db8:23::/48 dev lo  metric 1024  error -101 mtu 16436 advmss 16376 hoplimit 4294967295              12
2000::/3 via 2001:db8:23::1 dev eth3  metric 1024  mtu 1500 advmss 1440 hoplimit 4294967295                          13
fe80::/64 dev eth1  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295                                             14
fe80::/64 dev eth0  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295                                             15
fe80::/64 dev eth2  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295                                             16
fe80::/64 dev eth3  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295                                             17
fe80::/64 via :: dev sit1  metric 256  mtu 1480 advmss 1420 hoplimit 4294967295                                      18
fe80::/64 dev tun0  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295                                             19
fe80::/64 dev tun1  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295                                             20
fe80::/64 dev tun2  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295                                             21
fe80::/64 dev tun3  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295                                             22
fe80::/64 dev tun4  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295                                             23
fe80::/64 dev tun5  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295                                             24
fe80:0:fa00::/64 dev tun0  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295                                      25
fe80:0:fb00::/64 dev tun1  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295                                      26
fe80:0:fc00::/64 dev tun2  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295                                      27
fe80:0:fd00::/64 dev tun3  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295                                      28
fe80:0:fe00::/64 dev tun4  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295                                      29
fe80:0:ff00::/64 dev tun5  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295                                      30
#                                                                                                                    31
```



routingtable: BSD Version (OpenBSD)

```
# route -n show
...

Internet6:
Destination                       Gateway                       Flags    Refs      Use   Mtu  Prio Iface
::/104                            ::1                           UGRS        0        0     -     8 lo0
::/96                             ::1                           UGRS        0        0     -     8 lo0
::1                               ::1                           UH         14        0 33204     4 lo0
::127.0.0.0/104                   ::1                           UGRS        0        0     -     8 lo0
::224.0.0.0/100                   ::1                           UGRS        0        0     -     8 lo0
::255.0.0.0/104                   ::1                           UGRS        0        0     -     8 lo0
::ffff:0.0.0.0/96                 ::1                           UGRS        0        0     -     8 lo0
2000::/3                          2001:db8:23:5afe::2           UGS         0    65934     -     8 gif0
2001:db8:23:2::/64                link#1                        UC          1        0     -     4 sis0
2001:db8:23:2::1                  00:00:24:c8:cf:04             UHL         0        6     -     4 lo0
2001:db8:23:2:216:3eff:fe14:4b91  00:16:3e:14:4b:91             UHLc        0    12625     -     4 sis0
2001:db8:23:3::1                  2001:db8:23:3::2              UH          0        4     -     4 gif0
2001:db8:23:3::2                  link#6                        UHL         0        6     -     4 lo0
2001:db8:23:5afe::1               link#6                        UHL         0       12     -     4 lo0
2001:db8:23:5afe::2               2001:db8:23:5afe::1           UH          1      153     -     4 gif0
2002::/24                         ::1                           UGRS        0        0     -     8 lo0
2002:7f00::/24                    ::1                           UGRS        0        0     -     8 lo0
2002:e000::/20                    ::1                           UGRS        0        0     -     8 lo0
2002:ff00::/24                    ::1                           UGRS        0        0     -     8 lo0
fe80::/10                         ::1                           UGRS        0        0     -     8 lo0
fe80::%sis0/64                    link#1                        UC          2        0     -     4 sis0
fe80::200:24ff:fec8:cf04%sis0     00:00:24:c8:cf:04             UHL         1        0     -     4 lo0
fe80::216:3eff:fe14:4b91%sis0     00:16:3e:14:4b:91             UHLc        0    10950     -     4 sis0
fe80::21c:25ff:fed7:c0dd%sis0     00:1c:25:d7:c0:dd             UHLc        0     3502     -     4 sis0
fe80::%lo0/64                     fe80::1%lo0                   U           0        0     -     4 lo0
fe80::1%lo0                       link#5                        UHL         0        0     -     4 lo0
...
#
```



routingtable: windows XP Version (SP3)

```
Der aktive Status wird abgefragt...

Veroeff.  Typ       Met   Praefix                   Idx  Gateway/Schnittstelle
-------  --------  ----  ------------------------  ---  --------------------
no       Autokonf     8  2001:db8:23:2::/64          4  LAN-Verbindung
no       Autokonf   256  ::/0                        4  fe80::200:24ff:fec8:cf04
```

device.py: (_debian_routingtab_line)

```
    def _debian_routingtab_line(self, line):                                  121
        """evaluate one line of debian ipv6 routingtable"""                  122
        words = line.split()                                                  123
        w1 = words.pop(0).strip()                                             124
        if not line.find("unreachable"):                                      125
            return                                                            126
        if not line.find("default") and line.find("via") > 0:                 127
            target = '::/0'                                                   128
            via = words.pop(1)                                                129
            interf = words.pop(2)                                             130
        else:                                                                 131
            target = w1                                                       132
            if line.find("via") == -1:                                        133
                interf = words.pop(1)                                         134
                via = "::/0"                                                  135
            else:                                                             136
                via = words.pop(1)                                            137
                interf = words.pop(2)                                         138
        self.routingtab.append([IPv6Network(target),                          139
                                IPv6Network(via), interf])                    140
```

device.py: (_wxp_routingtab_line)

```
    def _wxp_routingtab_line(self, line):                                  231
        """evaluate one line of WinXP routing-table,                       232
        enter only, if valid IPv6 content                                  233
        """                                                                234
        zeile = line.split()                                               235
        if len(zeile) > 0:                                                 236
            hop = zeile.pop(-1)                                            237
            if len(zeile) > 0:                                             238
                dev = zeile.pop(-1)                                        239
                targ = zeile.pop(-1)                                       240
            else:                                                          241
                return                                                     242
        else:                                                              243
            return                                                         244
        try:                                                               245
            target = IPv6Network(targ)                                     246
            nhp = IPv6Network(hop)                                         247
            self.routingtab.append([target, nhp, dev])                     248
        except:                                                            249
            return                                                         250
        return                                                             251
```

Was ist zu Lesen?

Globale Konfiguration aller Geräte

Interfaces und Routen aller Geräte

Definitionen aus der Datei: **`hostnet6`**

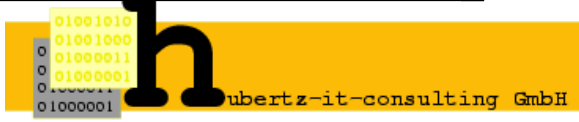
Regeln aus den Dateien: **`nn-rules.*`**

adm6: Fünfpunktekonzept

Jedes Gerät nutzt einen internen Paketfilter!

1. Alle Paketfilter werden zentral erzeugt und verwaltet
   Python-Script generiert Filtersequenz, mit ssh wird verteilt
2. Alle Hosts, Router, Firewalls im Netz sind mit Paketfiltern zu betreiben
   Ein Verzeichnis pro Gerät: ~/adm6/desc/gerätename
3. Interface- und Routinginformationen sind Berechnungsgrundlage
   ~/adm6/desc/gerätename/{interfaces,routes}
4. Definition aller beteiligten **Kommunikatoren** (Namen, Adressen)
   ~/adm6/desc/gerätename/hostnet6

Nur erlaubter Netzwerkverkehr bleibt übrig!

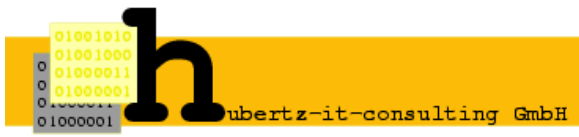


hostnet6 – Namen, Netze und Gruppen

```
# hostnet6      part of adm6                                  # hosts, networks and groups
# name          CIDR address                                  # comment
#                                                             #
any             2000::/3                                      # anybody outside and inside
#                                                             #
admin           2001:db8:f002:2::23/128                       # 1st adminstrators workstation
admin           2001:db8:f002:3::23/128                       # 2nd adminstrators workstation
#
ns              2001:db8:f002:1::53/128                       # 1st domain name server
ns              2001:db8:f002:2::53/128                       # 2nd domain name server
ns              2001:db8:f002:3::53/128                       # 3rd domain name server
www             2001:db8:f002:3::80/128                       # internet web server
intra           2001:db8:f002:1::443/128                      # intranet web server
#
office-cgn      2001:db8:f002:2::/64                          # office cologne
office-muc      2001:db8:f002:3::/64                          # office munich
office-bln      2001:db8:f002:7::/64                          # office berlin
#
fw-i            2001:db8:f002:2::1/128                        # firewall internal view
fw-e            2001:db8:f002:1::2/128                        # firewall external view
#
r-mine          2001:db8:f002::2/128                          # my router to r-isp
r-mine-i        2001:db8:f002:1::1/128                        # my router to r-isp
r-isp-e         2001:db8:abba::1/128                          # ISP routers ISP-side
r-isp           2001:db8:f002::1/128                          # ISP router to r-mine
#
ripe-net        2001:610:240:22::c100:68b/128                 # ripe.net web-server
www-kame-net    2001:200:dff:fff1:216:3eff:feb1:44d7/128      # orange.kame.net
#
# EOF
```

class HostNet6: Definitionen lesen

```
class HostNet6(IPv6Network):                                                    1
    """Instance is content of hostnet6-file"""                                  2
    def __init__(self,file):                                                    3
        """read file into self.entries"""                                       4
        self.entries = []                                                       5
        self.append(file)                                                       6
                                                                                7
    def __read_file(self,filename):                                             8
        """reads file using filename and fills self.entries"""                  9
        file1 = open(filename,'r')                                              10
        linenr = 0                                                              11
        for zeile in file1:                                                     12
            linenr = linenr + 1                                                 13
            line = str(zeile)                                                   14
            lefthalf = line.split('#')                                          15
            try:                                                                16
                (name, address) =  lefthalf.pop(0).split()                      17
                try:                                                            18
                    ipad=IPv6Network(address)                                   19
                    if self.entries.count([name,ipad]) == 0:                    20
                        self.entries.append([name,ipad])                        21
                except:                                                         22
                    print "User-Error: file:",filename                          23
                    print "User-Error: line:",linenr                            24
                    print "User-Error: content:",zeile                          25
                    pass                                                        26
                finally:                                                        27
                    pass                                                        28
            except:                                                             29
                pass                                                            30
        self.entries.sort(cmp=self.__mycmp__, key=None, reverse=False)          31
```



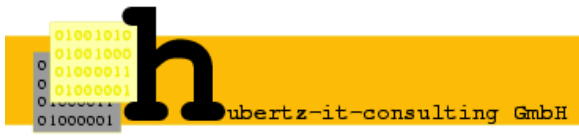
Was ist zu Lesen?

Globale Konfiguration aller Geräte

Interfaces und Routen aller Geräte

Definitionen aus der Datei: **`hostnet6`**

Regeln aus den Dateien: **`nn-rules.*`**



adm6: Fünfpunktekonzept

Jedes Gerät nutzt einen internen Paketfilter!

1. Alle Paketfilter werden zentral erzeugt und verwaltet
   Python-Script generiert Filtersequenz, mit ssh wird verteilt
2. Alle Hosts, Router, Firewalls im Netz sind mit Paketfiltern zu betreiben
   Ein Verzeichnis pro Gerät: ~/adm6/desc/gerätename
3. Interface- und Routinginformationen sind Berechnungsgrundlage
   ~/adm6/desc/gerätename/{interfaces,routes}
4. Definition aller beteiligten **Kommunikatoren** (Namen, Adressen)
   ~/adm6/desc/gerätename/hostnet6
5. Definition(en) aller **Kommunikationen** im Netz (Regelsatz)
   ~/adm6/desc/gerätename/XX-rules.{admin,users, . . . }

Nur erlaubter Netzwerkverkehr bleibt übrig!

00-rules.admin – Filterregeln (nutzt **hostnet6**)

```
# rules.admin    part          of        adm6
#
# source         destin        proto     port    action    options # comment or not
#
admin            ns            tcp       ssh     accept
admin            ns            udp       53      accept    INSEC NOSTATE # for debug
any              ns            udp       53      accept    NOSTATE # faster without
admin            www           tcp       80      accept
#
office-cgn       any           tcp       80      accept
office-cgn       any           tcp       443     accept
office-cgn       office-muc    ipv6      all     accept
#
office-muc       office-cgn    ipv6      all     accept
any              office-cgn    icmpv6    all     accept
#
# EOF
```

class ThisDevice: Eine Regelzeile lesen

```
    def read_one_rule(self, line):
        """take one line of rules-file and do the appropriate"""
        line = line.strip()
        line = line.replace("\t", " ")
        try:
            if line.__len__() < 8:
                #print "#Line to small"
                return
            if '#' in line:
                if line.startswith('#'):
                    return
                (left, right) = line.split('#')
                rule = left.split()
            else:
                rule = line.split()
        except:
            rule = line.split()
        try:
            src = rule.pop(0)
        except:
            # found empty line, no fault!
            return
        if src.startswith('#'):
            # still a comment line, no fault!
            return
        rule.insert(0, src)
        self.rules.append(rule)
```

adm6: Drei Schritte . . .

Lesen aller Parameter

Kreuzprodukt bilden

Generierung pro Gerät

Kreuzprodukt bilden

Quellen und Ziele

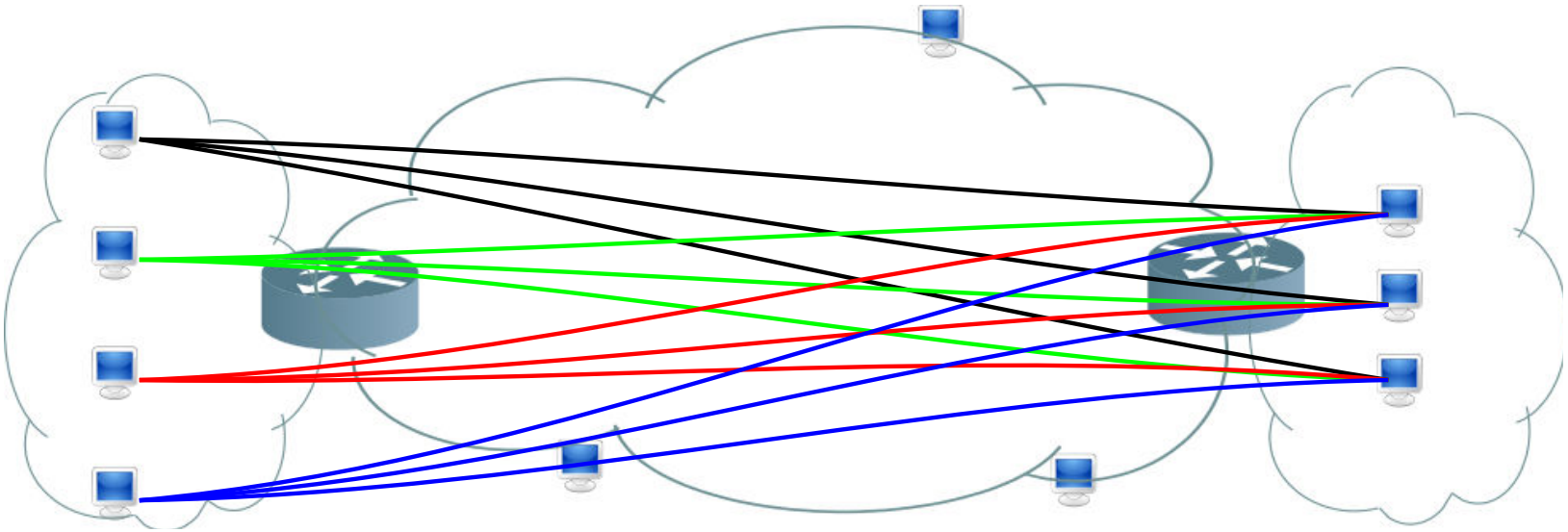
Wege durchs Netz

Gruppen auflösen

Protokolle

Optionen

IPv6: Firma mit zwei Standorten



Wollen Sie das händisch konfigurieren?

Zwei Lösungen bieten sich an

1. In hostnet6 alle Geräte eines Standortes mit einem Namen und einer Netzadresse anlegen:

links	2001:db8:8000:1::/64	# Linker Standort
rechts	2001:db8:8000:2::/64	# Rechter Standort

2. In hostnet6 jedes Gerät in einem Standort als Gruppe mit einem Namen und jeweiliger Adresse anlegen:

links	2001:db8:8000:1::1/128	# Linker Standort
links	2001:db8:8000:1::2/128	# Linker Standort
links	2001:db8:8000:1::3/128	# Linker Standort
links	2001:db8:8000:1::4/128	# Linker Standort
rechts	2001:db8:8000:2::1/128	# Rechter Standort
rechts	2001:db8:8000:2::2/128	# Rechter Standort
rechts	2001:db8:8000:2::3/128	# Rechter Standort

Kreuzprodukt bilden

Quellen und Ziele

Wege durchs Netz

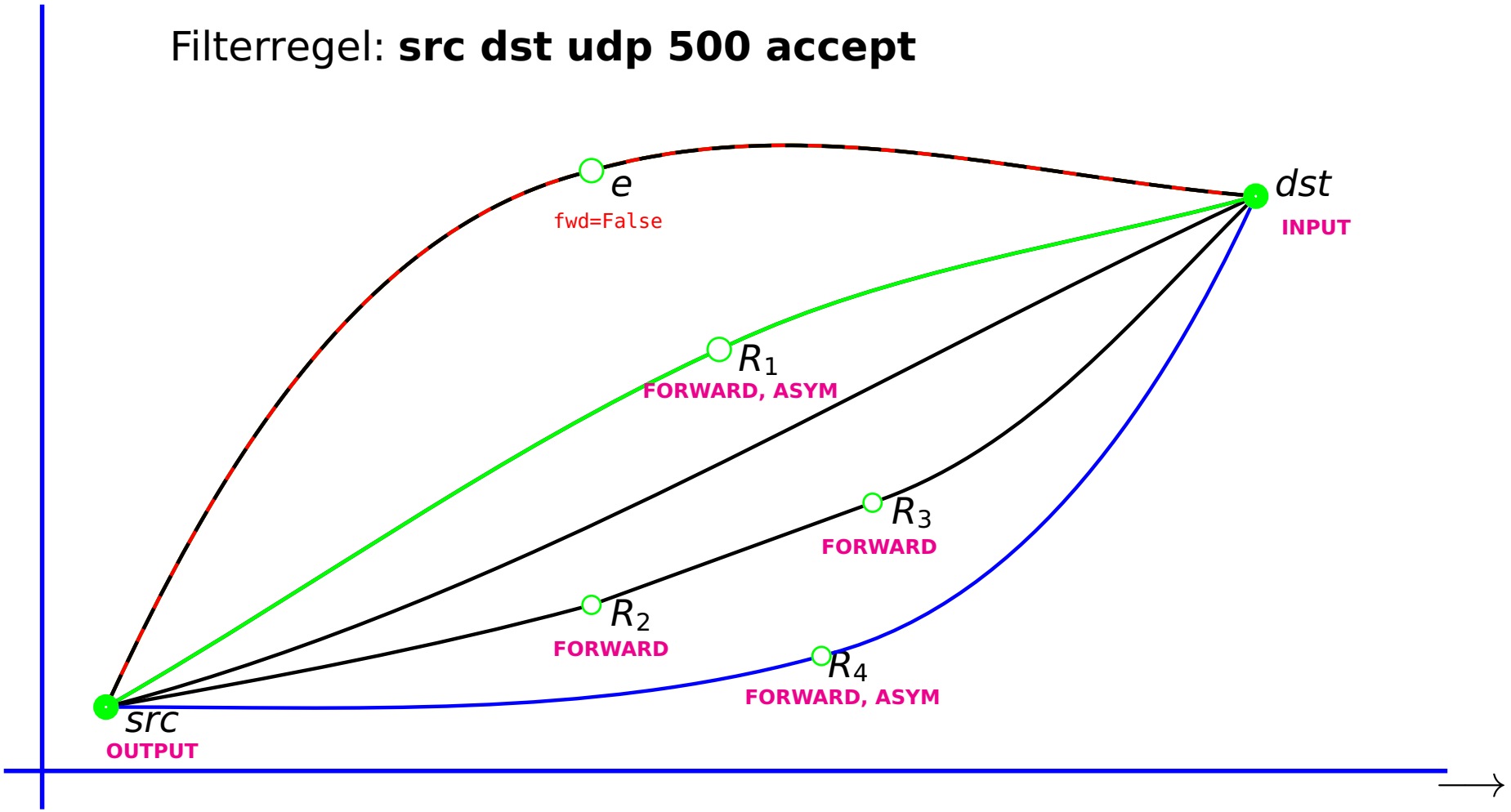
Gruppen auflösen

Protokolle

Optionen

adm6: Wege durchs Netz

Filterregel: **src dst udp 500 accept**



Kreuzprodukt bilden

Quellen und Ziele

Wege durchs Netz

Gruppen auflösen

Protokolle

Optionen

Gruppen auflösen: (do_this_rule I)

```
    def do_this_rule(self, clone, rn, filter6,                                  166
                     rh, sr, ds, pr, po, ac, op):                               167
        """build os-independant detailed rule without options,                  168
           which are very os-specific                                           169
        Step 1: find IP-Addresses of Sources and Destinations, 'de-grouping' """ 170
        srcs = self.hn6.get_addrs(sr)                                           171
        dsts = self.hn6.get_addrs(ds)                                           172
        rule_start_text = rh                                                    173
        nice_print(rule_start_text +u'has  '+str(len(srcs))+" source(s) and "   174
                             +str(len(dsts))+" destination(s) in hostnet6", '') 175
        pair = 0                                                                176
        """Step 2: Loop over all Source and Destination pairs"""                177
        for source in srcs:                                                     178
            i_am_source = self.address_is_own(source)                           179
            for destin in dsts:                                                 180
                pair += 1                                                       181
                i_am_destin = self.address_is_own(destin)                       182
                (ifs, ros) = self.look_for(rn, source)                          183
                (ifd, rod) = self.look_for(rn, destin)                          184
                """Step 3: Which traffic is it?"""                              185
                if i_am_source:                                                 186
                    """Step 3a: This is outgoing traffic"""                     187
                    nice_print(rule_start_text,                                 188
                        '  outgoing traffic!')                                  189
                elif i_am_destin:                                               190
                    """Step 3b: This is incoming traffic"""                     191
                    nice_print(rule_start_text,                                 192
                        '  incoming traffic!')                                  193
                else:                                                           194
                    """Step 3c: This is possibly traversing traffic"""          195
                    ...                                                         196
```

Gruppen auflösen: (do_this_rule II)

```
            ...                                                                193
            else:                                                              194
                """Step 3c: This is possibly traversing traffic"""             195
                if ros == rod:                                                 196
                    if not 'FORCED' in op:                                     197
                        nice_print(rule_start_text                             198
                            +u'bypassing traffic, nothing done!', '')          199
                        continue                                               200
                        nice_print(rule_start_text                             201
                            +u'bypassing traffic but FORCED', '')              202
                else:                                                          203
                    """We are sure about traversing traffic now"""             204
                    nice_print(rule_start_text                                 205
                        +u'traversing traffic, action needed', '')             206
            """Step 4: append appropriate filter"""                            207
            filter6.append([clone, rn, pair, i_am_source, i_am_destin,         208
                            source, destin, ifs, ros, ifd, rod,                209
                            pr, po, ac, op])                                   210
```

Kreuzprodukt bilden

Quellen und Ziele

Wege durchs Netz

Gruppen auflösen

Protokolle

Optionen

adm6: Protokolle

1. Manche Protokolle sind unidirektional:

Protokoll	**Besonderheiten**
IPv6	beliebige Quellen und Ziele **::**
ICMPv6	beliebige Quellen und Ziele **::**
Multicasts	Ziel immer auf Linklocal **ff00::/8**
ipencap, ipip	Routingheader, Ziele aus **2000::/3**

2. Andere Protokolle sind bidirektional:

Protokoll	**Besonderheiten**
tcp, udp	beliebige Quellen und Ziele **::**

d.h. es gibt zugehörige Antwortpakete.

Kreuzprodukt bilden

Quellen und Ziele

Wege durchs Netz

Gruppen auflösen

Protokolle

Optionen

adm6: Optionen auf Regeln

Option	**Bedeutung**	fertig
NOIF	Unterdrückung der Interfaceangabe	✓
NONEW	nur „ESTABLISHED, RELATED“ generieren	✓
NOSTATE	Unterdrückung Statefull Inspection	✓
FORCED	IN, OUT, FORWARD unabhängig von Interface- und Routing-Informationen	✓
INSEC	Quellport kleiner 1024 zulassen	✓
NOFORWARD	keinesfalls FORWARD	
LOG	zusätzlich Pakete loggen	
20110615	Ab 15. 6. 2011 nicht mehr generieren	

adm6: Drei Schritte . . .

Lesen aller Parameter

Kreuzprodukt bilden

Generierung pro Gerät

Generierung pro Gerät

Abstrakte Zwischenschicht

Umsetzung auf Zielplattform OS

Header und Footer je nach OS

Generierter Filter je nach OS

Möglichst **systemunabhängige** Programmierung

Notwendige Informationen pro Gerät:

1.) OS-Name bzw. Filterarchitektur

2.) OS-spezifische Header- und Footer

3.) Interface- und Routinginformationen

4.) Host- und Netzdefinitionen

5.) Regelsatz (evtl. Geräteabhängig)

6.) evtl. Zusätze fürs Shellscript (paket-mangling, QoS)

Realisierung in zwei Objektklassen:

IPv6_Filter: generiert Shellscript aus den Bausteinen (2,6)

IPv6_Filter_Rule: erzeugt jeweiligen Filter pro Regel (1,3,4,5) (System-abhängig)!

Generierung pro Gerät

Abstrakte Zwischenschicht

Umsetzung auf Zielplattform OS

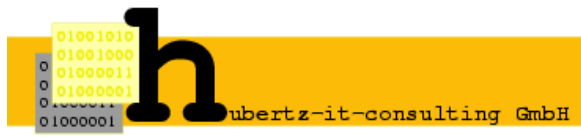
Header und Footer je nach OS

Generierter Filter je nach OS

class IP6_Filter_Rule: (produce_Debian I)

```
    def produce_Debian(self, outfile, commented):                                  77
        """do one pair of src-dst out of a rule for Debian"""                      78
        print u"# producing ip6tables commands for rule:", self['Rule-Nr'],        79
        print u"Pair: ", self['Pair-Nr']                                           80
        #outfile.write(u'echo -n "."; ')                                           81
        #print u"# commented: ", commented                                         82
        answer_packets = False                                                     83
        icmp_type = False                                                          84
        proto = str(self['Protocol']).strip()                                      85
        # tcp, udp, and with esp we want bidirectional traffic, too!               86
        if proto in ['tcp', 'udp', 'esp']:                                         87
            answer_packets = True                                                  88
        #icmpv6 has no states!                                                     89
        if proto in ['icmpv6']:                                                    90
            self['nostate'] = True                                                 91
            icmp_type = True                                                       92
        st_new = " -m state --state NEW,ESTABLISHED,RELATED"                       93
        st_ans = " -m state --state     ESTABLISHED,RELATED"                       94
        if self['nonew']:                                                          95
            st_new = st_ans                                                        96
        if self['nostate']:                                                        97
            st_new = ""                                                            98
            st_ans = ""                                                            99
        if u'accept' in self['Action']:                                            100
            act = u' -j ACCEPT'                                                    101
        elif u'reject' in self['Action']:                                          102
            act = u' -j REJECT'                                                    103
            addition = u''' nyr '''                                                104
        else:                                                                      105
            act = u' -j DROP'                                                      106
```

und so weiter . . .



Generierung pro Gerät

Abstrakte Zwischenschicht

Umsetzung auf Zielplattform OS

Header und Footer je nach OS

Generierter Filter je nach OS

Debian Header

```
#!/bin/bash
echo "*************************************************************************"
echo "##                                                                     ##"
echo "##   a d m 6    -  a device manager for IPv6 packetfiltering            ##"
echo "##                                                                     ##"
echo "##   version:      0.1                                                 ##"
echo "##                                                                     ##"
echo "##   device-name:  cccccc                                              ##"
echo "##   device-type:  Debian GNU/Linux                                    ##"
echo "##                                                                     ##"
echo "##   date:         dddddd                                              ##"
echo "##   author:       Johannes Hubertz, hubertz-it-consulting GmbH        ##"
echo "##                                                                     ##"
echo "##   license:      GNU general public license version 3                ##"
echo "##                     or any later  version                           ##"
echo "##                                                                     ##"
echo "*************************************************************************"
POLICY_D='DROP'
I6='/sbin/ip6tables '
IP6I='/sbin/ip6tables -A   input___new '
IP6O='/sbin/ip6tables -A   output__new '
IP6F='/sbin/ip6tables -A   forward_new '
CHAINS="$CHAINS input___"
CHAINS="$CHAINS output_"
CHAINS="$CHAINS forward"
for chain in $CHAINS
do
        /sbin/ip6tables -N ${chain}_act >/dev/null 2>/dev/null
        /sbin/ip6tables -N ${chain}_new
done
$I6 -P   INPUT $POLICY_D
$I6 -P  OUTPUT $POLICY_D
$I6 -P FORWARD $POLICY_D
# do local and multicast on every interface
LOCAL="fe80::/10"
MCAST="ff02::/10"
$IP6I -p ipv6-icmp -s ${LOCAL} -d ${LOCAL} -j ACCEPT
$IP6O -p ipv6-icmp -s ${LOCAL} -d ${LOCAL} -j ACCEPT
$IP6I -p ipv6-icmp -s ${MCAST} -j ACCEPT
$IP6I -p ipv6-icmp -d ${MCAST} -j ACCEPT
$IP6O -p ipv6-icmp -s ${MCAST} -j ACCEPT
# all prepared now, individual mangling and rules following
```

Debian Footer part I

```
#ICMPv6types="${ICMPv6types} destination-unreachable"
ICMPv6types="${ICMPv6types} echo-request"
ICMPv6types="${ICMPv6types} echo-reply"
ICMPv6types="${ICMPv6types} neighbour-solicitation"
ICMPv6types="${ICMPv6types} neighbour-advertisement"
ICMPv6types="${ICMPv6types} router-solicitation"
ICMPv6types="${ICMPv6types} router-advertisement"
for icmptype in $ICMPv6types
do
        $IP6I -p ipv6-icmp --icmpv6-type $icmptype -j ACCEPT
        $IP6O -p ipv6-icmp --icmpv6-type $icmptype -j ACCEPT
done
$IP6I -p ipv6-icmp --icmpv6-type destination-unreachable -j LOG  --log-prefix "unreach: " \
                   -m limit --limit 30/second --limit-burst 60
$IP6I -p ipv6-icmp --icmpv6-type destination-unreachable -j ACCEPT
#
CHAINS=""
CHAINS="$CHAINS input__"
CHAINS="$CHAINS output_"
CHAINS="$CHAINS forward"
#set -x
for chain in $CHAINS
do
        /sbin/ip6tables -E "${chain}_act" "${chain}_old"
        /sbin/ip6tables -E "${chain}_new" "${chain}_act"
done
#
$I6 -F INPUT
$I6 -A INPUT   -m rt --rt-type 0 -j LOG --log-prefix "rt-0: " -m limit --limit 3/second --limit-burst 6
$I6 -A INPUT   -m rt --rt-type 0 -j DROP
$I6 -A INPUT   -m rt --rt-type 2 -j LOG --log-prefix "rt-2: " -m limit --limit 3/second --limit-burst 6
$I6 -A INPUT   -m rt --rt-type 2 -j DROP
$I6 -A INPUT  -i lo -j ACCEPT
$I6 -A INPUT   --jump input___act
#
```

Debian Footer part II

```
#
$I6 -F OUTPUT
$I6 -A OUTPUT -o lo -j ACCEPT
$I6 -A OUTPUT  --jump output__act
#
$I6 -F FORWARD
$I6 -A FORWARD -m rt --rt-type 0 -j LOG --log-prefix "rt-0: " -m limit --limit 3/second --limit-burst 6
$I6 -A FORWARD -m rt --rt-type 0 -j DROP
$I6 -A FORWARD --jump forward_act
#
for chain in $CHAINS
do
        /sbin/ip6tables -F "${chain}_old"
        /sbin/ip6tables -X "${chain}_old"
done
$I6 -F logdrop
$I6 -X logdrop
$I6 -N logdrop
$I6 -A   INPUT   --jump logdrop
$I6 -A  OUTPUT   --jump logdrop
$I6 -A FORWARD   --jump logdrop
$I6 -A logdrop -j LOG --log-prefix "drp: " -m limit --limit 3/second --limit-burst 6
$I6 -A logdrop -j DROP
#
/sbin/ip6tables-save -c >/root/last-filter
echo "************************************************************************"
echo "************************************************************************"
echo "##                                                                    ##"
echo "##    All rules applied, thanks for your patience ...                 ##"
echo "##    cu                                                              ##"
echo "##                                                                    ##"
echo "************************************************************************"
echo "************************************************************************"
# EOF
```

OpenBSD Header

```
#!/bin/sh                                                                       1
#                                                                               2
echo "*************************************************************************"  3
echo "*************************************************************************"  4
echo "##                                                                     ##"  5
echo "##   a d m 6    -  a device manager for IPv6 packetfiltering           ##"  6
echo "##                                                                     ##"  7
echo "##   version:      0.1                                                 ##"  8
echo "##                                                                     ##"  9
echo "##   device-name:  cccccc                                              ##"  10
echo "##   device-type:  OpenBSD pf.conf                                     ##"  11
echo "##                                                                     ##"  12
echo "##   date:         dddddd                                              ##"  13
echo "##   author:       Johannes Hubertz, hubertz-it-consulting GmbH        ##"  14
echo "##                                                                     ##"  15
echo "##   license:      GNU general public license version 3                ##"  16
echo "##                     or any later  version                           ##"  17
echo "##                                                                     ##"  18
echo "*************************************************************************"  19
echo "*************************************************************************"  20
echo "##                                                                     ##"  21
echo "##   some magic abbreviations follow                                   ##"  22
echo "##                                                                     ##"  23
#                                                                               24
cat << EOFEOFEOFEOF  > /tmp/new-pf-conf                                         25
# set default policy first                                                      26
block all                                                                       27
#                                                                               28
```

OpenBSD Footer

```
echo "********************************************************************************"
echo "********************************************************************************"
echo "##                                                                            ##"
echo "##    End of generated /tmp/new-pf-conf                                       ##"
echo "##                                                                            ##"
echo "********************************************************************************"
EOFEOFEOFEOF
echo "********************************************************************************"
echo "********************************************************************************"
echo "##                                                                            ##"
echo "##    End of generated filter-rules                                           ##"
echo "##                                                                            ##"
echo "********************************************************************************"
echo "********************************************************************************"
# EOF
```

Windows XP Header

```
@echo off                                                                                    1
echo "adm6: test-batch header file for device: wxp-jh                                 "    2
echo "adm6: created on:  2011-06-02 15:29                                          "      3
rem ##############################################################################         4
rem disable privacy extensions                                                               5
rem                                                                                          6
netsh interface ipv6 set global randomizeidentifiers=disabled                                7
netsh interface ipv6 set global randomizeidentifiers=disabled store=persistent               8
netsh interface ipv6 set privacy state=disabled store=persistent                             9
rem ##############################################################################         10
rem disable teredo tunnels                                                                   11
rem                                                                                          12
netsh interface 6to4 set state state=disabled undoonstop=disabled                            13
netsh interface isatap set state state=disabled                                              14
netsh interface teredo set state type=disabled                                               15
rem                                                                                          16
rem ##############################################################################         17
rem ##############################################################################         18
```

Generierung pro Gerät

Abstrakte Zwischenschicht

Umsetzung auf Zielplattform OS

Header und Footer je nach OS

Generierter Filter je nach OS

Eine Regel und was daraus wird: **Debian**

```
# ------------------------------------------------------------------------------ #  1
# Rule-Nr        : 3                                                             #  2
# Pair-Nr        : 1                                                             #  3
# System-Name    : r-ex                                                          #  4
# OS             : Debian                                                        #  5
# RuleText       : ['any', 'ns', 'udp', '53', 'accept', 'NOSTATE']               #  6
# Source         : 2000::/3                                                      #  7
# Destin         : 2001:db8:23:1::23/128                                         #  8
# Protocol       : udp                                                           #  9
# sport          : 1024:                                                         # 10
# dport          : 53                                                            # 11
# Action         : accept                                                        # 12
# nonew          : False                                                         # 13
# noif           : False                                                         # 14
# nostate        : True                                                          # 15
# insec          : False                                                         # 16
# i_am_s         : None                                                          # 17
# i_am_d         : None                                                          # 18
# travers        : True                                                          # 19
# source-if      : eth3                                                          # 20
# source-rn      : 10                                                            # 21
# src-linklocal  : False                                                         # 22
# src-multicast  : False                                                         # 23
# destin-if      : eth1                                                          # 24
# destin-rn      : 1                                                             # 25
# dst-linklocal  : False                                                         # 26
# dst-multicast  : False                                                         # 27
/sbin/ip6tables -A   forward_new  -i eth3 -s 2000::/3 -d 2001:db8:23:1::23/128 \ 28
                         -p udp --sport 1024: --dport 53 -j ACCEPT               29
/sbin/ip6tables -A   forward_new  -o eth1 -d 2000::/3 -s 2001:db8:23:1::23/128 \ 30
                         -p udp --dport 1024: --sport 53 -j ACCEPT               31
```

Eine Regel und was daraus wird: **OpenBSD**

```
# ------------------------------------------------------------------------ #
# Rule-Nr        : 3                                                       #
# Pair-Nr        : 1                                                       #
# System-Name    : obi-lan                                                 #
# OS             : OpenBSD                                                 #
# RuleText       : ['any', 'ns', 'udp', '53', 'accept', 'NOSTATE']         #
# Source         : 2000::/3                                                #
# Destin         : 2001:db8:23:1::23/128                                   #
# Protocol       : udp                                                     #
# sport          : 1024:                                                   #
# dport          : 53                                                      #
# Action         : accept                                                  #
# nonew          : False                                                   #
# noif           : False                                                   #
# nostate        : True                                                    #
# insec          : False                                                   #
# i_am_s         : None                                                    #
# i_am_d         : None                                                    #
# travers        : True                                                    #
# source-if      : undef                                                   #
# source-rn      : 17                                                      #
# src-linklocal  : False                                                   #
# src-multicast  : False                                                   #
# destin-if      : gif0                                                    #
# destin-rn      : 7                                                       #
# dst-linklocal  : False                                                   #
# dst-multicast  : False                                                   #
pass  in quick  from 2000::/3 to   2001:db8:23:1::23/128 port    53 proto udp
pass out quick  to   2000::/3 from 2001:db8:23:1::23/128 proto udp
# n o t   y e t   r e a d y
```

Shellscript strukturiert generiert

adm6 generiert pro Gerät ein Shellscript aus:

- ⟹ **header** (1 je OS)
- ⟹ **mangle-startup** (0 oder 1 je Gerät)
- ⟹ **Filterregeln** (1 .. n je Gerät)
- ⟹ **mangle-endup** (0 oder 1 je Gerät)
- ⟹ **footer** (1 je OS)

Flexibilität, die sich auszahlt!

Betrieb – Erfahrung

adm6: Betrieb – 24/7

If the facts don't fit the theory, change the facts.
(Albert Einstein)

Start im September 2010 auf zwei Linuxsystemen (web, dns, mail)

Nutzung von `he.net/certification` und `lg.he.net` zu Tests

Regeln seit Oktober 2010 unverändert

Verbesserungen in Header und Footer bis Dezember 2010

Start auf Linuxrouter im Januar 2011 mit gleichen Regeln

Experimente mit OpenBSD seit Februar 2011

Experimente mit Win (xp) seit Juni 2011

Asymmetrisches Routing ab August 2011

Jan. 2012: python-paramiko zum Lesen, scp zur Verteilung

Jan. 2012: ~/adm6/.git, commit zu jeder Erzeugung / Verteilung

adm6: Ausblick

Mehrwert — Erweiterungen

GUI – einfache Benutzung

Mehrwert – Erweiterungen

GnuPG-Verschlüsselung der Filterscripts

Client-Pull aus zentralem git-repository

Validitätszeitraum pro Regel

Doppelte Regeln finden

Dokumentation erstellen mit LaTeX 2ε

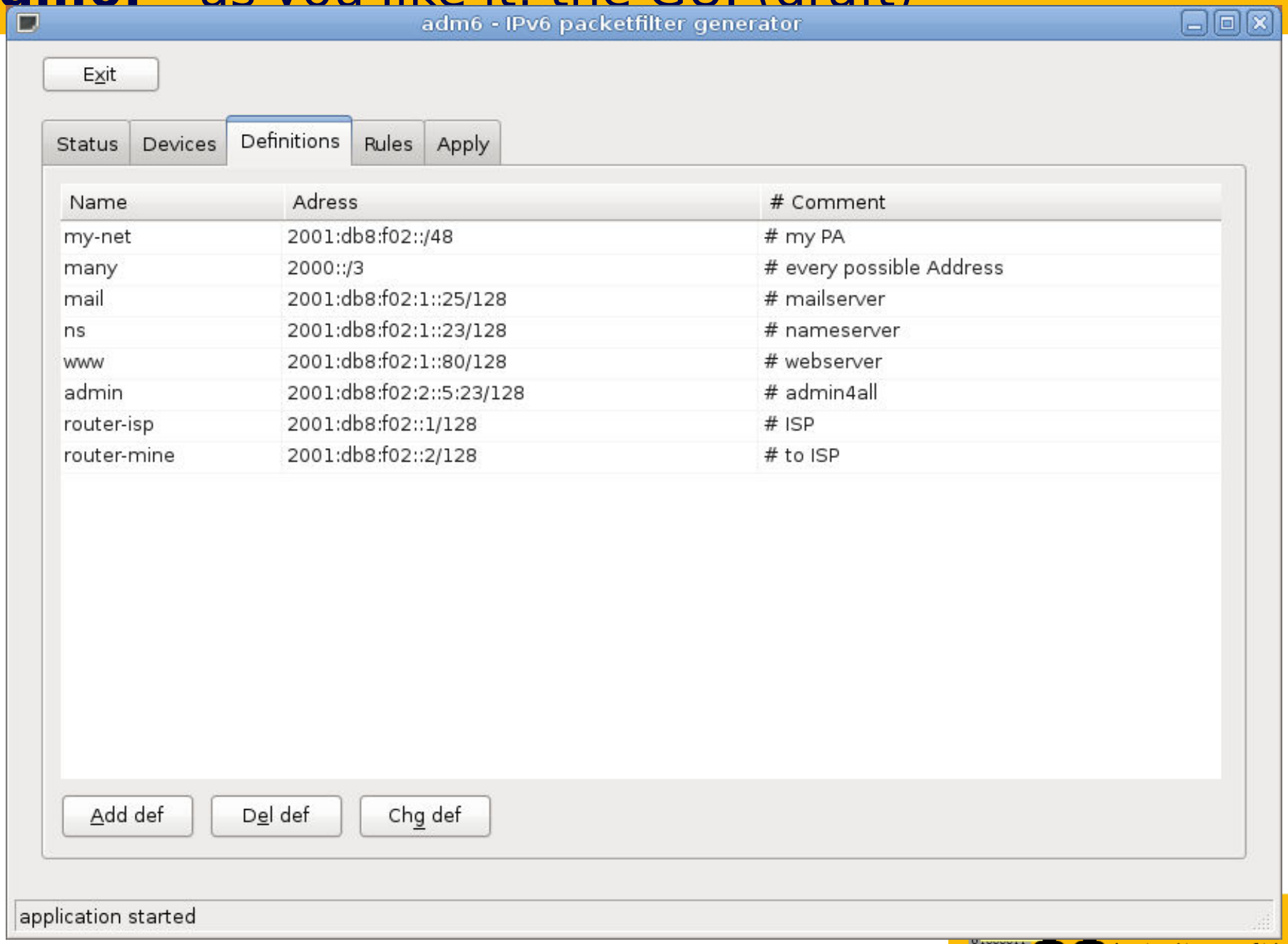
Byte- und Paketzähler auswerten

adm6: Ausblick

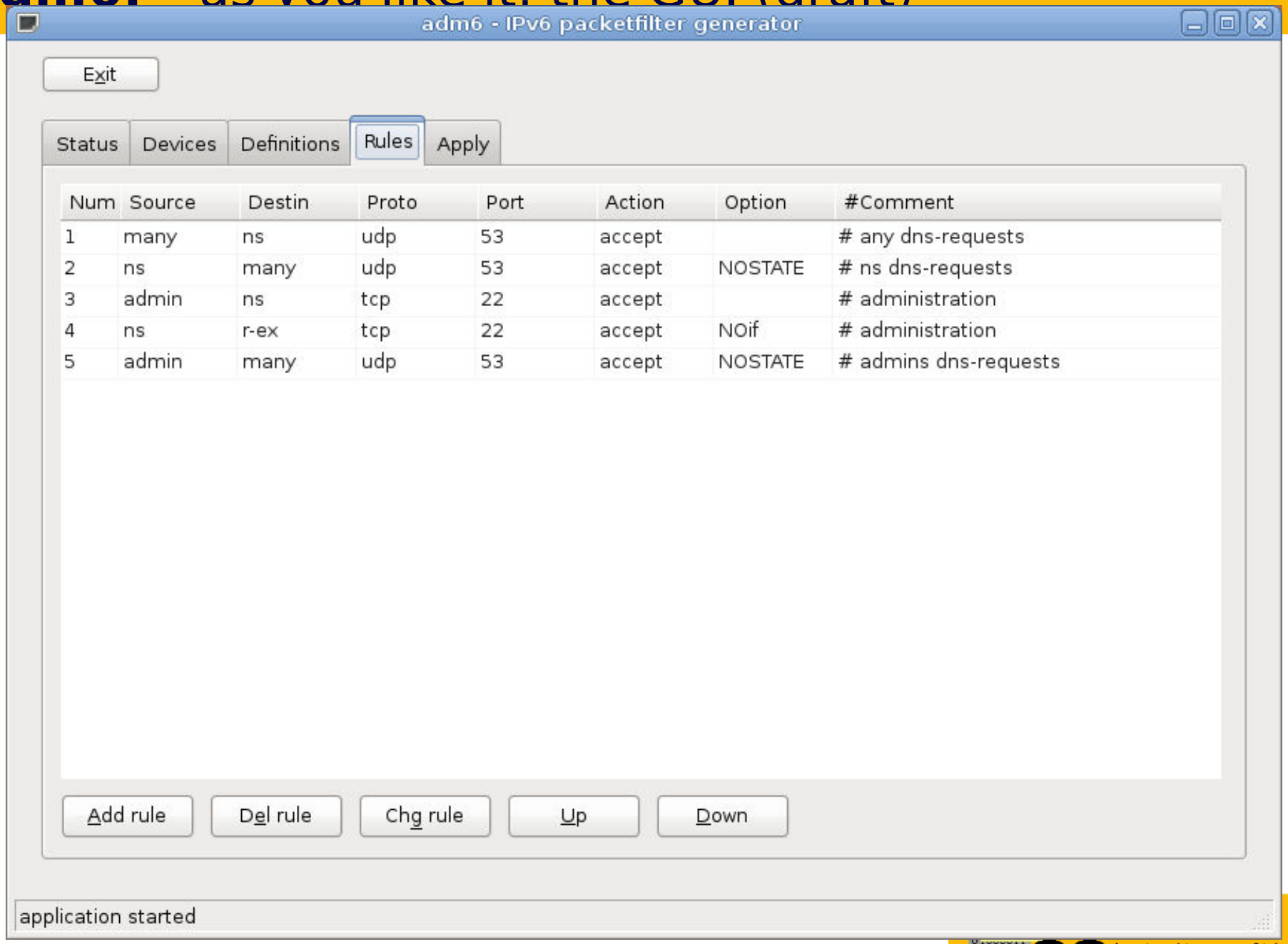
Mehrwert — Erweiterungen

GUI – einfache Benutzung

adm6: – as you like it: the GUI (draft)

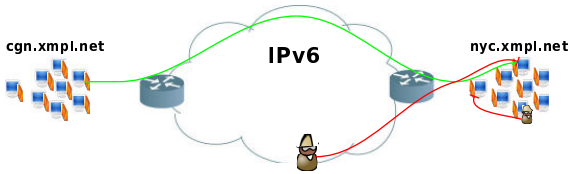


adm6: – as you like it: the GUI (draft)



adm6: Zusammenfassung

adm6: Eine Idee wird zum Konzept . . .



adm6: Fünfpunktekonzept

Jedes Gerät nutzt einen internen Paketfilter!

1. Alle Paketfilter werden zentral erzeugt und verwaltet
2. Alle Hosts, Router, Firewalls im Netz sind mit Paketfiltern zu betreiben
3. Interface- und Routinginformationen sind Berechnungsgrundlage
4. Definition aller beteiligten **Kommunikatoren** (Namen, Adressen)
5. Definition(en) aller **Kommunikationen** im Netz (Regelsatz)

Nur erlaubter Netzwerkverkehr bleibt übrig!

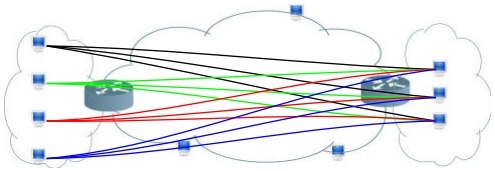
adm6: Drei Schritte . . .

Lesen aller Parameter

Kreuzprodukt bilden

Generierung pro Gerät

IPv6: Firma mit zwei Standorten



Wollen Sie das händisch konfigurieren?

Möglichst **systemunabhängige** Programmierung

Notwendige Informationen pro Gerät:

1.) OS-Name bzw. Filterarchitektur
2.) OS-spezifische Header- und Footer
3.) Interface- und Routinginformationen
4.) Host- und Netzdefinitionen
5.) Regelsatz (evtl. Geräteabhängig)
6.) evtl. Zusätze fürs Shellscript (paket-mangling, QoS)

Realisierung in zwei Objektklassen:

IPv6_Filter: generiert Shellscript aus den Bausteinen (2,6)

IPv6_Filter_Rule: erzeugt jeweiligen Filter pro Regel (1,3,4,5) (System-abhängig)!

Shellscript strukturiert generiert

adm6 generiert pro Gerät ein Shellscript aus:

⟹ **header**	(1 je OS)
⟹ **mangle-startup**	(0 oder 1 je Gerät)
⟹ **Filterregeln**	(1 .. n je Gerät)
⟹ **mangle-endup**	(0 oder 1 je Gerät)
⟹ **footer**	(1 je OS)

Flexibilität, die sich auszahlt!

hubertz-it-consulting GmbH

adm6: – roadmap

Windows XP (sp3), Windows 7

crypto: encryption, signature, distribution, applying on the clients

integration into the gui, simplified usage

scaling, database vs. config-files

scripted documentation by LaTeX 2_ε and PSTricks

Quellen und Anregungen (Auszug)

... only a few of more than 200 ...
RFC 2460 Internet Protocol, Version 6 (IPv6) Specification
RFC 2461 Neighbor Discovery for IP Version 6 (IPv6)
RFC 2462 IPv6 Stateless Address Autoconfiguration
RFC 2463 Internet Control Message Protocol for the Internet Protocol Version 6 (IPv6) Specification
RFC 2464 Transmission of IPv6 Packets over Ethernet Networks
RFC 3315 Dynamic Host Configuration Protocol for IPv6 (DHCPv6)
RFC 3484 Default Address Selection for Internet Protocol version 6 (IPv6)
RFC 3756 IPv6 Neighbor Discovery (ND) Trust Models and Threats
RFC 3775 Mobility Support in IPv6
RFC 3971 SEcure Neighbor Discovery (SEND)
RFC 3972 Cryptographically Generated Addresses (CGA)
RFC 4429 Optimistic Duplicate Address Detection (DAD) for IPv6
RFC 4443 Internet Control Message Protocol for the Internet Protocol Version 6 (IPv6) Specification
RFC 4861 Neighbor Discovery for IPv6
RFC 4890 Recommendations for Filtering ICMPv6 Messages in Firewalls
RFC 5095 Deprecation of RH0

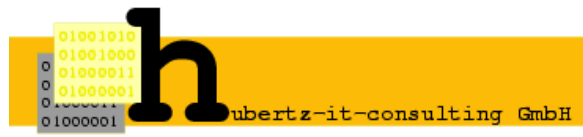
Linux:
http://www.bieringer.de/linux/IPv6/IPv6-HOWTO/IPv6-HOWTO.html
OpenVPN-tunnelbroker: http://blog.ghitr.com/index.php/archives/673
http://www.6net.org/publications/presentations/strauf-openvpn.pdf

Books:
IPv6, Sylvia Hagen, Sunny Edition, 2. Auflage, ISBN 978-3-9522842-2-2
IPv6 in Practice, Benedikt Stockebrand, Springer, ISBN 978-3-540-24524-7
IPv6 Security, Scott Hogg, Eric Vyncke, Cisco Press, ISBN 1587055942
Deploying IPv6 Networks, Ciprian Popoviciu et.al., Cisco Press, ISBN 1587052105

Tests:
http://freeworld.thc.org/thc-ipv6/
http://lg.he.net/

Security:
http://www.wecon.net/files/48/GUUG-RT_WEST2010-SvI.pdf
http://seanconvery.com/ipv6.html

Lernen:
http://ipv6.he.net/certification/



adm6: Zeit zum Dialog!

Kompetente und kompatible

Schlangenbändiger gesucht!

Noch Fragen?

Ich bedanke mich für Ihre Aufmerksamkeit

hubertz-it-consulting GmbH jederzeit zu Ihren Diensten

Ihre Sicherheit ist uns wichtig!

Frohes Schaffen

Johannes Hubertz

it-consulting _at_ hubertz dot de

$H\alpha \in \{$ kompetenzspektrum.de $\}$

powered by LaTeX 2ε
and PSTricks

adm6: EVOLVIS Repository

```
git clone https://evolvis.org/anonscm/git/adm6/adm6.git
```

http://www.hubertz.de/papers/20120301-p.pdf

hubertz-it-consulting GmbH