adm6: ip6tables, pf.conf, ipf mit python

Johannes Hubertz

hubertz-it-consulting GmbH

OpenChaos@C^4, 24. 2. 2011

Was zu zeigen ist

Vorstellung – Wer zeigt hier was?
Motivation – Warum das alles?
Konzept – Wie könnte es funktionieren?
Geräte – Informationen und deren Nutzen
Definitionen – Netze und Hosts
Filter-Regeln in der Sicht des Administrators
Filter-Regeln als generiertes Kreuzprodukt

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 Mgr. 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 ;-)

IPv6 filtern, warum das denn?

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?

- 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	ipf	ipf
OpenSolaris	ipf	ipf
? Other ?	? ? ?	? ? ?

Ich hatte mal einen Traum . . .

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

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

Firewallregeln in einfachen ASCII-Dateien:
(source, destination, protocol, port, action, comment)

Erledigt für IPv4: `http://sspe.sourceforge.net`

implementiert in Shell und Perl, etwas schwierig für Einsteiger

bei mehreren Kunden erfolgreich im Einsatz

keinerlei externe Resonanz seit 2003 außer einer Email,
jedoch weiterhin regelmäßig Downloads bei sf.net

Ich hatte noch einen Traum . . .

IPv6 ist ja noch gar nicht verbreitet

Da ist noch viel zu tun, laßt uns Geduld haben,
irgendwer wirds schon machen. . .

So nicht!

IPv6 ist schon implementiert, es funktioniert
und läßt sich heute schon nutzen und filtern!

Aber wie?

notwendige Voraussetzungen (Auswahl)

eine globale Konfiguration für alles: **~/.adm6.conf**

Strukturen für Informationen: **Verzeichnisbaum: ~/adm6/. . .**

gesammelte Informationen über Geräte: Name, OS-Name, Adresse, Routingtabelle, etc.: **~/adm6/desc/name/**

Vollständige Liste der Adressen aller Datenverkehrsteilnehmer: **hostnet6**

Vollständige Liste der erlaubten Verkehrsbeziehungen: “source destination protocol port action“ : **rules**

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
```

File: ~/.adm6.conf

```
# global adm6 system configuration

[global]
version = 0.1
timestamp = 2010-07-13
home = /home/hans/adm6/
devices = r-ex, ns, obi-wan
software = ['Debian', 'OpenBSD', 'OpenSolaris']

[device#r-ex]
desc = external router via ISP to the world
os = Debian GNU/Linux, Lenny
ip = 2001:db8:f002:1::1
fwd = 1
active = 1

[device#ns]
desc = company dns server
os = Debian GNU/Linux, Lenny
ip = 2001:db8:f002:1::23
fwd = 0
active = 1

[device#obi-wan]
desc = gif-tunnel from company to home
os = OpenBSD 4.5
ip = 2001:db8:f002:1::2
fwd = 0
active = 1
```

class Adm6ConfigParser config-file

```python
import os
from ConfigParser import ConfigParser

"""ugly: module wide variable cfg_file"""
cfg_file = "adm6.conf"


class Adm6ConfigParser(ConfigParser):
    """Read global config from configfile: cfg_file."""

    def __init__(self):
        self.cf = ConfigParser()
        self.filename = os.path.expanduser('~/.'+cfg_file)
        self.cf.read([self.filename])

    def get_adm6_home(self):
        return self.cf.get('global', 'home', False, {})

    def get_adm6_debuglevel(self):
        """get applicationwide debuglevel"""
        level = int(self.cf.get('global', 'debuglevel', False,
                        {}))
        return level

    def get_apply(self, device):
        """give back applyflag (missing flag means true!)"""
        section = "device#" + device.strip()
        value = False
        try:
            return self.cf.getboolean(section, 'active')
        except:
            return False
        return value

    def get_version(self):
        return self.cf.get('global', 'version').strip()

    def get_devices(self):
        """give list of all devices named in global section"""
        return self.cf.get('global', 'devices', False, {})

    def get_software(self):
        """give a list of all os-software named in global
                section"""
        return self.cf.get('global', 'software', False, {})

    def get_device_home(self, device):
        """give directory of device as full pathname"""
        pat = self.get_adm6_home()
        pat = pat.strip() +'desc/' + device.strip()
        return pat

    def get_desc(self, device):
        """give description of named device"""
        section = "device#" + device.strip()
        return self.cf.get(section, 'desc').strip()

    def get_os(self, device):
        """give OS-String of named device"""
        section = "device#" + device.strip()
        return self.cf.get(section, 'os').strip()

    def show_cf(self):
        """show complete content as dict of dicts"""
        for section in self.cf.sections():
            print section, self.cf.items(section)
```

hostnet6.py

hostnet6 – definitions of hosts, networks and groups

```
# hostnet6     part of adm6                               # hosts, networks and groups
# name         CIDR address                               # comment
#                                                         #
any            2000::/3                                   # anybody outside and inside
#                                                         #
admin          2001:db8:f002:1::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
```

HostNet6 file to be read by python

```
class HostNet6(IPv6Network):
    """Instance is content of hostnet6-file"""
    def __init__(self,file):
        """read file into self.entries"""
        self.entries = []
        self.append(file)

    def __read_file(self,filename):
        """reads file using filename and fills self.entries"""
        file1 = open(filename,’r’)
        linenr = 0
        for zeile in file1:
            linenr = linenr + 1
            line = str(zeile)
            lefthalf = line.split(’#’)
            try:
                (name, address) =  lefthalf.pop(0).split()
                try:
                    ipad=IPv6Network(address)
                    if self.entries.count([name,ipad]) == 0:
                        self.entries.append([name,ipad])
                except:
                    print "User-Error: file:",filename
                    print "User-Error: line:",linenr
                    print "User-Error: content:",zeile
                    pass
                finally:
                    pass
            except:
                pass
        self.entries.sort(cmp=self.__mycmp__, key=None, reverse=False)
```

HostNet6 entry to be found by python

```
    # class HostNet6(IPv6Network) continued

    def get_addrs(self,name):
        """return list of addresses belonging to a name"""
        addrs = []
        for entry in self.entries:
            (hname,addr) = entry
            if hname == name:
                addrs.append(addr)
        return addrs

    def show_hostnet6(self):
        """show all current entries"""
        nice_print("# hostnet6 contents:",'')
        number = 0
        for entry in self.entries:
            number = number + 1
            (hname,addr) = entry
            nice_print( '#    '+str(hname),str(addr))
        s =  "# hostnet6: %5d entries found" % number
        nice_print(s,'')
        nice_print('#','')
```

device.py

device.py: Gerätespezifisches

- Betriebssystem
- Interfaces, Adressen, Netzmasken
- Routingtabellen
- Angebot einzelner Dienste
- Nutzung einzelner Dienste

device.py: (__init__)

```
class ThisDevice:
    """Object keeps all sampled information about a device,
    information is read from a subdirectory in adm6/desc/
    interface-config (output of ifconfig) and
    routing-table (output of ip -6 route show) and
    filter-rules (plain ascii-filese with defs and actions)
    might be useful for other things than generating filters"""

    def __init__(self, device, confParser, hostnet):
        self.name = device.strip()
        self.confParser = confParser
        self.device_os = confParser.get_os(device)
        self.device_ip = confParser.get_ip(device)
        print "# Device:" + str(device) + " Found IP:" + str(self.device_ip)
        self.hn6 = hostnet
        self.interfaces = []
        self.interfaces_file = confParser.get_device_home(device).strip()
        self.interfaces_file = self.interfaces_file + '/interfaces'
        self.read_interface_file()
        self.routingtab = []
        self.routingtab_file = confParser.get_device_home(device).strip()
        self.routingtab_file = self.routingtab_file + '/routes'
        self.read_routingtab_file(self.device_os)
        self.rules_path = confParser.get_device_home(device).strip()
        self.rule_files = []
        self.rules = []
```

device.py: (interface_file)

```
def read_interface_file(self):
    try:
        f = open(self.interfaces_file, 'r')
        while True:
            line = f.readline()
            if not line:
                break
            else:
                pass
            self.interface_line(line)
        f.close()
    except IOError, e:
        print self.interfaces_file + ": ", e.strerror
        return
```

device.py: (interface_line)

```
def interface_line(self, line):
    """evaluate one line of ifconfig-output store results in self.interfaces = []"""
    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()
            adr = ine.pop(1)
            self.int_addr = IPv6Network(adr)
            self.interfaces.append([self.int_name, self.int_addr])
        return
```

device.py: (read routingtable)

```
    def read_routingtab_file(self, os):
        """read plain file containg output of
        Debian:     ip -6 route show
        OpenBSD:    route  -n   show
        """
        try:
            f = open(self.routingtab_file, 'r')
            while True:
                line = f.readline()
                if not line:
                    break
                self.routingtab_line(line, os)
            f.close()
        except IOError, e:
            print self.routingtab_file + ": ", e.strerror
        return

    def routingtab_line(self, line, os):
        """read a line using os-spcific version
        """
        if os == "Debian GNU/Linux, Lenny":
            self._debian_routingtab_line(line)
        elif os == "OpenBSD 4.5":
            self._bsd_routingtab_line(line)
        else:
            raise "# error: Attempt to read routingtable for unknown OS"
        return
```

routingtable: Debian version

```
# ip -6 route show
2001:db8:23::/64 dev eth3  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295
2001:db8:23:1::/64 dev eth1  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295
2001:db8:23:2::/64 dev sit1  metric 1024  mtu 1480 advmss 1420 hoplimit 4294967295
2001:db8:23:3::/64 via :: dev sit1  metric 256  mtu 1480 advmss 1420 hoplimit 4294967295
2001:db8:23:fa00::/56 via fe80:0:fa00::2 dev tun0  metric 1024  mtu 1500 advmss 1440 hoplimit 4294967295
2001:db8:23:fb00::/56 via fe80:0:fb00::2 dev tun1  metric 1024  mtu 1500 advmss 1440 hoplimit 4294967295
2001:db8:23:fc00::/56 via fe80:0:fc00::2 dev tun2  metric 1024  mtu 1500 advmss 1440 hoplimit 4294967295
2001:db8:23:fd00::/56 via fe80:0:fd00::2 dev tun3  metric 1024  mtu 1500 advmss 1440 hoplimit 4294967295
2001:db8:23:fe00::/56 via fe80:0:fe00::2 dev tun4  metric 1024  mtu 1500 advmss 1440 hoplimit 4294967295
2001:db8:23:ff00::/56 via fe80:0:ff00::2 dev tun5  metric 1024  mtu 1500 advmss 1440 hoplimit 4294967295
unreachable 2001:db8:23::/48 dev lo  metric 1024  error -101 mtu 16436 advmss 16376 hoplimit 4294967295
2000::/3 via 2001:db8:23::1 dev eth3  metric 1024  mtu 1500 advmss 1440 hoplimit 4294967295
fe80::/64 dev eth1  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295
fe80::/64 dev eth0  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295
fe80::/64 dev eth2  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295
fe80::/64 dev eth3  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295
fe80::/64 via :: dev sit1  metric 256  mtu 1480 advmss 1420 hoplimit 4294967295
fe80::/64 dev tun0  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295
fe80::/64 dev tun1  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295
fe80::/64 dev tun2  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295
fe80::/64 dev tun3  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295
fe80::/64 dev tun4  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295
fe80::/64 dev tun5  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295
fe80:0:fa00::/64 dev tun0  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295
fe80:0:fb00::/64 dev tun1  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295
fe80:0:fc00::/64 dev tun2  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295
fe80:0:fd00::/64 dev tun3  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295
fe80:0:fe00::/64 dev tun4  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295
fe80:0:ff00::/64 dev tun5  metric 256  mtu 1500 advmss 1440 hoplimit 4294967295
#
```

device.py: (_debian_routingtab_line)

```
    def _debian_routingtab_line(self, line):
        """evaluate one line of debian ipv6 routingtable"""
        words = line.split()
        w1 = words.pop(0).strip()
        if not line.find("unreachable"):
            return
        if not line.find("default") and line.find("via") > 0:
            target = '::/0'
            via = words.pop(1)
            interf = words.pop(2)
        else:
            target = w1
            if line.find("via") == -1:
                interf = words.pop(1)
                via = "::/0"
            else:
                via = words.pop(1)
                interf = words.pop(2)
        self.routingtab.append([IPv6Network(target),
                                IPv6Network(via), interf])
```

routingtable: OpenBSD version

```
# 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
...
#
```

device.py: (_bsd_routingtab_line)

```
    def _bsd_routingtab_line(self, line):
        """evaluate one line of OpenBSD routing-table, enter only, if useful content"""
        zeile = line.split()
        if len(zeile) > 0:
            targ = zeile.pop(0)
            if not ":" in targ:
                return
            try:
                target = IPv6Network(targ)
            except:
                """no IPv6 Address in column one"""
                return
            try:
                hop = zeile.pop(0)
                nhp = IPv6Network(hop.strip())
                nhp._prefixlen = 128
                dev = zeile.pop(-1)
                self.routingtab.append([target, nhp, dev])
                #print "APPEND:",str([target, nhp, dev])
                return
            except:
                #print " something wrong reading bsd-routingtable"
                return
```

device.py: (do_this_rule I)

```
    def do_this_rule(self, clone, rn, filter6,
                     rh, sr, ds, pr, po, ac, op):
        """build os-independant detailed rule without options,
           which are very os-specific
        Step 1: find IP-Addresses of Sources and Destinations, 'de-grouping' """
        srcs = self.hn6.get_addrs(sr)
        dsts = self.hn6.get_addrs(ds)
        rule_start_text = rh
        nice_print(rule_start_text +u'has  '+str(len(srcs))+" source(s) and "
                         +str(len(dsts))+" destination(s) in hostnet6", '')
        pair = 0
        """Step 2: Loop over all Source and Destination pairs"""
        for source in srcs:
            i_am_source = self.address_is_own(source)
            for destin in dsts:
                pair += 1
                i_am_destin = self.address_is_own(destin)
                (ifs, ros) = self.look_for(rn, source)
                (ifd, rod) = self.look_for(rn, destin)
                """Step 3: Which traffic is it?"""
                if i_am_source:
                    """Step 3a: This is outgoing traffic"""
                    nice_print(rule_start_text,
                        '  outgoing traffic!')
                elif i_am_destin:
                    """Step 3b: This is incoming traffic"""
                    nice_print(rule_start_text,
                        '  incoming traffic!')
                else:
                    """Step 3c: This is possibly traversing traffic"""
                    ...
```

device.py: (do_this_rule II)

```
                ...
                else:
                    """Step 3c: This is possibly traversing traffic"""
                    if ros == rod:
                        if not 'FORCED' in op:
                            nice_print(rule_start_text
                                +u'bypassing traffic, nothing done!', '')
                            continue
                            nice_print(rule_start_text
                                +u'bypassing traffic but FORCED', '')
                    else:
                        """We are sure about traversing traffic now"""
                        nice_print(rule_start_text
                            +u'traversing traffic, action needed', '')
                """Step 4: append appropriate filter"""
                filter6.append([clone, rn, pair, i_am_source, i_am_destin,
                               source, destin, ifs, ros, ifd, rod,
                               pr, po, ac, op])
```

device.py: (do_rules)

```
    def do_rules(self, filter):
        """invocation: dev.do_rules(filter)"""
        nice_print('# begin on rules expecting interface and routing for:',
            self.device_os)
        nice_print("#"*76, '#')
        rn = 0
        for rule in self.rules:
            rn += 1
            clone = str(rule)
            rule_header = u'# Rule '+str(rn)+u': '
            lstart = rule_header + "has  "+str(len(rule))+" items : "
            nice_print(lstart, '')
            nice_print(u'# '+str(rule), '')
            if len(rule) > 0:
                src = rule.pop(0)
            if len(rule) > 0:
                dst = rule.pop(0)
            if len(rule) > 0:
                pro = rule.pop(0)
            if len(rule) > 0:
                prt = rule.pop(0)
            if len(rule) > 0:
                act = rule.pop(0)
            else:
                nice_print(rule_header +"has insufficient parametercount", '')
                nice_print(rule_header + str(rule), '')
                continue
            self.do_this_rule(clone, rn, filter, rule_header,
                src, dst, pro, prt, act, rule)
            nice_print("#"*76, '#')
        nice_print('# '+self.name+u': ready, '+str(rn)+u' rules found', '')
        filter.mach_output()
```

device.py (__main__)

```
def do_all_configured_devices():
    confParser = Adm6ConfigParser()
    version = confParser.get_version()
    confParser.print_header()
    debuglevel = confParser.get_adm6_debuglevel()
    #confParser.show_cf()
    my_devices = confParser.get_devices().split(',')
    for device_name in my_devices:
        if confParser.get_apply(device_name):
            device_os = confParser.get_os(device_name)
            confParser.print_head(device_name)
            path = str(confParser.get_device_home(device_name))
            h_path = path+'/hostnet6'
            hn6 = HostNet6(h_path)
            dev = ThisDevice(device_name, confParser, hn6)
            dev.read_rules()
            #hn6.show_hostnet6()
            #dev.show_interfaces()
            #dev.show_routingtab()
            #dev.show_rules()
            filter6 = IP6_Filter(debuglevel,
                                 path,
                                 device_name,
                                 device_os,
                                 dev.interfaces)
            dev.do_rules(filter6)
            #filter6.mach_output(version)
    print "#"*80

if __name__ == "__main__":
    do_all_configured_devices()
```

filter6.py

rules.admin – filter rules use defs of 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 IP6_Filter (__init__)

```
class IP6_Filter():
    os = ''
    me = None
    """Devicetype mostly independant Filter"""

    def __init__(self, debuglevel, path, name, os, interfaces):
        """start with an empty filter"""
        self.rules = []
        self.debuglevel = debuglevel
        self.path = path
        self.name = name
        if 'Debian' in os:
            self.os = 'Debian'
        elif 'OpenBSD' in os:
            self.os = 'OpenBSD'
        elif 'OpenSolaris' in os:
            self.os = 'OpenSolaris'
        else:
            print "# try to create filter object for unknown OS"
        return

    def append(self, rule):
        """append a rule to the creation list"""
        #print "APPENDING to filter rule: "+str(rule)
        self.rules.append(rule)
        return
```

class IP6_Filter (mangle_file)

```
    def mangle_file(self,outfile,mangleinclude):
        mangle_filename = self.path + u'/' + mangleinclude
        #print "#"
        #outfile.write("#")
        try:
            mang = open(mangle_filename)
            print "# mangle-file: %s inclusion starts" % mangle_filename
            outfile.write("# mangle-file: %s inclusion starts\n" % mangle_filename)
            for line in mang:
                print line,
                outfile.write(line)
            mang.close()
            print "# mangle-file: %s inclusion successfully ended" % mangle_filename
            outfile.write("# mangle-file: %s inclusion successfully ended\n" % mangle_filename)
        except:
            print "# mangle-file: %s for inclusion not found\n" % mangle_filename
            outfile.write("# mangle-file: %s for inclusion not found\n" % mangle_filename)
```

class IP6_Filter (mach_output)

```
    def mach_output(self):
        """construct header, rules and footer altogether"""
        fname = self.path + '/output'
        header_file = self.path + "/../../etc/" + str(self.os) + "-header"
        footer_file = self.path + "/../../etc/" + str(self.os) + "-footer"
        outfile = open(fname, 'w')
        head = open(header_file, 'r')
        header_name = u"%-25s" %(self.name)
        date = time.localtime()
        header_date = time.strftime("%Y-%m-%d %H:%M")
        # beautify header, device-name, date,
        for line in head:
            l = line.replace('cccccc                   ', header_name)
            line = l.replace('dddddd          ', header_date)
            outfile.write(line)
        head.close()
        # read mangle-start if present
        self.mangle_file(outfile,u'mangle-startup')
        # every rule could do an output now
        for rule in self.rules:
            self.final_this_rule(rule, outfile)
        # some finalization, get ready
        # read mangle-end if present
        self.mangle_file(outfile,u'mangle-endup')
        foot = open(footer_file, 'r')
        outfile.writelines(foot.readlines())
        outfile.close()
        return
```

class IP6_Filter (final_this_rule) I

```
    def final_this_rule(self, rule, outfile):
        """do output for one pair out of rule-nr into file: outfile,
        convert simple list-structure in rule into Rule-UserDict-Object"""
        r = Ip6_Filter_Rule()
        r['debuglevel'] = self.debuglevel
        r['Output'] = outfile
        r['OS'] = self.os
        r['System-Name'] = self.name.strip()
        r['RuleText'] = rule.pop(0)    # Orig. Rule Text as List
        r['Rule-Nr'] = rule.pop(0)
        r['Pair-Nr'] = rule.pop(0)
        r['i_am_s'] = rule.pop(0)
        r['i_am_d'] = rule.pop(0)
        if 'NOIF' in rule[-1]:
            r['noif'] = True
        if 'NONEW' in rule[-1]:
            r['nonew'] = True
        if 'NOSTATE' in rule[-1]:
            r['nostate'] = True
        if 'INSEC' in rule[-1]:
            r['insec'] = True
        r['Source'] = rule.pop(0)
        r['Destin'] = rule.pop(0)
        r['source-if'] = rule.pop(0)
        r['source-rn'] = rule.pop(0)
        r['destin-if'] = rule.pop(0)
        r['destin-rn'] = rule.pop(0)
        r['Protocol'] = rule.pop(0)
        r['dport'] = rule.pop(0)
        r['Action'] = rule.pop(0)
        ...
```

class IP6_Filter (final_this_rule) II

```
        ...
        r['Action'] = rule.pop(0)
        r['src-multicast'] = r['Source'].is_multicast
        r['src-linklocal'] = r['Source'].is_link_local
        r['dst-multicast'] = r['Destin'].is_multicast
        r['dst-linklocal'] = r['Destin'].is_link_local
        if r['source-rn'] <> r['destin-rn']:
            r['travers'] = True
        if r['source-if'] <> r['destin-if']:
            r['travers'] = True
        # source or destin doesn't do forwarding except FORCED
        if r['i_am_s']:
            r['travers'] = False
        if r['i_am_d']:
            r['travers'] = False
        # option FORCED overrides some calculations
        if 'FORCED' in rule[-1]:
            r['i_am_s'] = True
            r['i_am_d'] = True
            r['travers'] = True
        s = "# "+'-'*76 + " #"
        print s
        outfile.write(s+'\n')
        print "%s" % (r),
        outfile.write(str(r))
        r.produce(outfile)
```

class IP6_Filter_Rule – I

```python
class Ip6_Filter_Rule(UserDict):
    """IP6_Filter_Rule is a container with all the neccessary stuff
    for device-type independant filter-generation.
    It is filled by reading all the specific device-files of one device,
    device-type is one out of (Debian, OpenBSD, OpenSolaris)
    interfaces, routing-table, hostnet6 and all device-rules"""

    def __init__(self, dict=None, **kwargs):
        """set initial params valid for all instances, and create a
        DisplayList for representation of this Object"""
        UserDict.__init__(self, dict, **kwargs)
        self['travers'] = False
        self['i_am_s'] = False
        self['i_am_d'] = False
        self['noif'] = False
        self['nonew'] = False
        self['nostate'] = False
        self['insec'] = False
        self['sport'] = u'1024:'
        # we cannot print a filedescriptor
        self.NeverDisplay = ['Output', 'debuglevel']
        self.DisplayList = [
            # meta-info
            'Rule-Nr', 'Pair-Nr', 'System-Name', 'OS',
            # user-given rule-info
            'RuleText',
            'Source', 'Destin', 'Protocol', 'sport', 'dport', 'Action',
            'nonew', 'noif', 'nostate', 'insec',
            # caclulated info
            'i_am_s', 'i_am_d', 'travers',
            'source-if', 'source-rn', 'src-linklocal', 'src-multicast',
            'destin-if', 'destin-rn', 'dst-linklocal', 'dst-multicast',
            ]
        return
```

class IP6_Filter_Rule – II

```python
    def __repr__(self):
        """representaion of Rule-Object for printouts"""
        retStr = u''
        if self['debuglevel']:
            reprList = self.DisplayList
        else:
            reprList = self.CommentList
        # sample the wellknown keys of DisplayList first
        for key in reprList:
            try:
                s = u"# %-15s: %-59s #\n" % (key, self[key])
            except:
                continue
            retStr += s
        # unsorted keys at last
        for key in dict(self):
            s = u''
            try:
                if key in self.NeverDisplay:
                    s = u"# %-15s: %-59s #\n" % (key, str(self['key']))
                elif not key in self.DisplayList:
                    s = u"# %-15s: %-59s #\n" % (key, self[key])
            except:
                continue
            retStr +=s
        return retStr
```

class IP6_Filter_Rule – III

```
    def produce(self, outfile):
        if 'Debian' in self['OS']:
            self.produce_Debian(outfile, False)
        elif 'OpenBSD' in self['OS']:
            self.produce_OpenBSD(outfile, False)
        elif 'BSD' in self['OS']:
            self.produce_IPF(outfile, False)
        elif 'OpenSolaris' in self['OS']:
            #self.produce_Debian(outfile, True)
            self.produce_IPF(outfile, False)
        else:
            print "# cannot make filter commands for unknown OS"
        return
```

filter-rules

generated output: Debian Header

```
#!/bin/bash
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
# but ignore all the boring fault-messages
$I6 -P   INPUT $POLICY_D
$I6 -P  OUTPUT $POLICY_D
$I6 -P FORWARD $POLICY_D
#
# some things need to pass, even if you don't like them
# 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
#
```

generated output: Debian Footer 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
#
```

generated output: Debian Footer 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
```

Endprodukt einer Regel (Version für Debian)

```
# ---------------------------------------------------------------------------- #
# Rule-Nr        : 3                                                           #
# Pair-Nr        : 1                                                           #
# System-Name    : r-ex                                                        #
# OS             : Debian                                                      #
# 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      : eth3                                                        #
# source-rn      : 10                                                          #
# src-linklocal  : False                                                       #
# src-multicast  : False                                                       #
# destin-if      : eth1                                                        #
# destin-rn      : 1                                                           #
# dst-linklocal  : False                                                       #
# dst-multicast  : False
/sbin/ip6tables -A   forward_new  -i eth3 -s 2000::/3 -d 2001:db8:23:1::23/128 \
                           -p udp --sport 1024: --dport 53 -j ACCEPT
/sbin/ip6tables -A   forward_new  -o eth1 -d 2000::/3 -s 2001:db8:23:1::23/128 \
                           -p udp --dport 1024: --sport 53 -j ACCEPT
```

Endprodukt einer Regel (Version für OpenBSD)

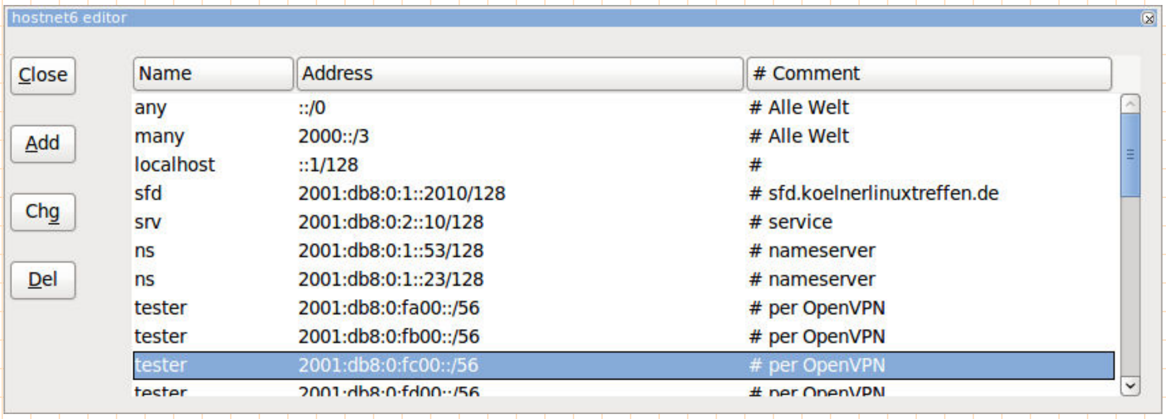
```
# ---------------------------------------------------------------------------- #
# Rule-Nr        : 3                                                           #
# Pair-Nr        : 1                                                           #
# System-Name    : obi-lan                                                     #
# OS             : OpenBSD                                                     #
# RuleText       : ['any', 'ns', 'udp', '53', 'accept', 'NOSTATE']             #
# Source         : ::/0                                                        #
# 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 ::/0 to   2001:db8:23:1::23/128 port    53 proto udp
pass out quick  to   ::/0 from 2001:db8:23:1::23/128 proto udp
# n o t   y e t   r e a d y
```

hubertz-it-consulting GmbH

gui – a design example

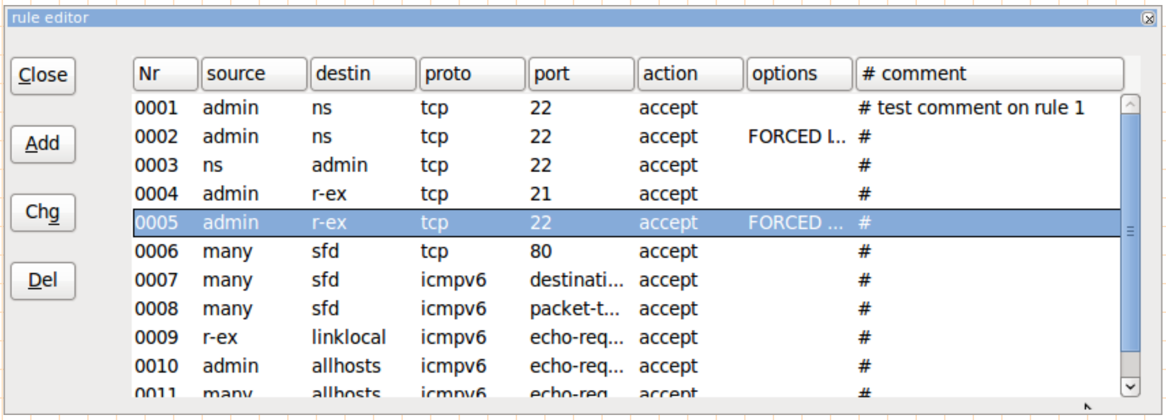
hubertz-it-consulting GmbH

hostnet6 – 1st dream of a gui



Name	Address	# Comment
any	::/0	# Alle Welt
many	2000::/3	# Alle Welt
localhost	::1/128	#
sfd	2001:db8:0:1::2010/128	# sfd.koelnerlinuxtreffen.de
srv	2001:db8:0:2::10/128	# service
ns	2001:db8:0:1::53/128	# nameserver
ns	2001:db8:0:1::23/128	# nameserver
tester	2001:db8:0:fa00::/56	# per OpenVPN
tester	2001:db8:0:fb00::/56	# per OpenVPN
tester	2001:db8:0:fc00::/56	# per OpenVPN
tester	2001:db8:0:fd00::/56	# per OpenVPN

rule editor – 1st dream of a gui



Nr	source	destin	proto	port	action	options	# comment
0001	admin	ns	tcp	22	accept		# test comment on rule 1
0002	admin	ns	tcp	22	accept	FORCED I...	#
0003	ns	admin	tcp	22	accept		#
0004	admin	r-ex	tcp	21	accept		#
0005	admin	r-ex	tcp	22	accept	FORCED ...	#
0006	many	sfd	tcp	80	accept		#
0007	many	sfd	icmpv6	destinati...	accept		#
0008	many	sfd	icmpv6	packet-t...	accept		#
0009	r-ex	linklocal	icmpv6	echo-req...	accept		#
0010	admin	allhosts	icmpv6	echo-req...	accept		#
0011	many	allhosts	icmpv6	echo-req...	accept		#

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 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 in Practice, Benedikt Stockebrand, Springer, ISBN 978-3-540-24524-7
IPv6, Sylvia Hagen, Sunny Edition, 2. Auflage, ISBN 978-3-9522842-2-2
Deploying IPv6 Networks, Ciprian Popoviciu et.al., Cisco Press, ISBN 1587052105

Tests:                                  Security:
http://freeworld.thc.org/thc-ipv6/      http://www.wecon.net/files/48/GUUG-RT_WEST2010-SvI.pdf
http://lg.he.net/                       http://seanconvery.com/ipv6.html
```

Jez hammer et geschaff . . .

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 ∈ { `kompetenzspektrum.de` }

Python *es e* ***Geföhl!***

powered by $\LaTeX\,2_\varepsilon$
and PSTricks

Besonderer Dank gilt Markus | Markus ∈ { `kompetenzspektrum.de` }
für seine stetige Geduld, meine Python-Unkenntnis zu [*lm*]indern.