adm6
ip6tables, pf.conf, ipf mit python

Johannes Hubertz

hubertz-it-consulting GmbH

pyCologne, 11. 8. 2010

IPv6 filtering: why for?

IPv6 . . .

- as secure as IPv4
- as insecure as IPv4
- no (questionable) protection by NAT
- always end-to-end communication
- is ready to use implemented, filtering possibilities still often unused
- somtimes used without any notice
- same applications and vulnerabilities like IPv4

So we don't like unfiltered IPv6 in our own networks, do we?

IPv6 filtering: what to use?

system	**filter**	**command**
Linux	NetFilter	ip6tables
OpenBSD	pf	pf, pf.conf, rc.local
Free- u. NetBSD	ipf	ipf
OpenSolaris	ipf	ipf
? Windows ?	? teredo ?	? ipconfig ?

IPv6 filtering: where?

- we filter on the Firewall, everything is secure!
- we filter on Firewall and on routers, everything is secure!
- on Firewall, on routers, on servers, everything is secure!
- really secure?
- Why not on every device?
- too expensive? Not, if:
  1. they have a secure communication channel
  2. they have a reliable configuration change
  3. they belong to a single administation domain

We prefer filtering on every device! **really** . . .

everywhere!

I had a dream . . .

1. flatfile with definitions: (names, address, comment)
2. flatfile(s) with firewall-rules:
   (source, destination, protocol, port, action, comment)
3. **already done for IPv4:** `http://sspe.sourceforge.net`
4. implemented in shell and perl, somehow strange for a newbie
5. no users response since 2003, still some downloads every month

IPv6 is not widely spread

there is very much to be done, let's keep patience, somebody will do . . .

IPv6 will be spread widely tomorrow, let's learn IPv6 today

IPv6 is already implemented, let's learn how to filter IPv6 now!

what we need: one machine to generate that stuff

1. global configuration about everything: **~/.adm6.conf**
2. structure to keep informations: **directorytree, ~/adm6/. . .**
3. sampled devices information: name, os-name, address, routingtable: **~/adm6/desc/name/**
4. simple mechanics to sample and keep the structure up2date: **ssh**[1]
5. elements (grouped by name) of traffic relations: **hostnet6**
6. traffic relations “source destination protocol port action“ : **rules**
7. information about reachability (when filters are applied): **ping6**
8. python-code[2] to produce filter-code
9. some gui for your convienience

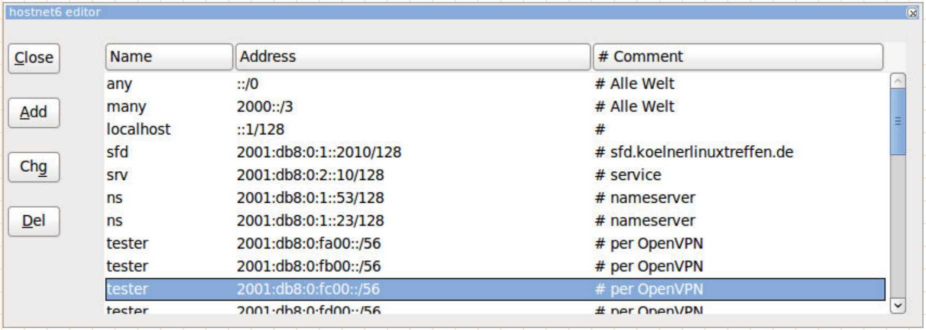
[1] python-paramiko + device-specific commands: ifconfig, . . .
[2] right now started, still α-version

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
```

hubertz-it-consulting GmbH

hostnet6 – 1st dream of a gui

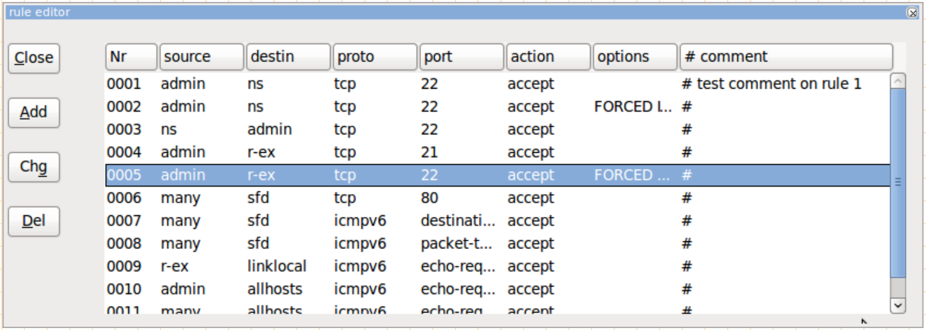


Name	Address	# Comment
any	::/0	# Alle Welt
many	2000::/3	# Alle Welt
localhost	::1/128	#
sfd	2001:db8:0:1::2010/128	# sfd.koelnerlinuxtreffen.de
srv	2001:db8:0:2::10/128	# service
ns	2001:db8:0:1::53/128	# nameserver
ns	2001:db8:0:1::23/128	# nameserver
tester	2001:db8:0:fa00::/56	# per OpenVPN
tester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

rules.admin – filter rules use defs of hostnet6

```
# rules.admin   part         of        adm6
# src           dst          proto     port   action   options
admin           ns           tcp        ssh   accept
admin           ns           udp         53   accept   INSEC NOSTATE # for debug
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
```

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		#

class Adm6ConfigParser config-file

```

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 show_cf(self):
        """show_complete_content_as_dict_of_dicts"""
        for section in self.cf.sections():
            print section, self.cf.items(section)

    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 set_adm6_debuglevel(self, level):
        """set_applicationwide_debuglevel"""
        self.cf.set('global', 'debuglevel', str(level))
        with open(self.filename, 'wb') as configfile:
            self.cf.write(configfile)
        configfile.close()
        return True

    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_a_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.cf.get('global', 'home', False, {})
        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()
```

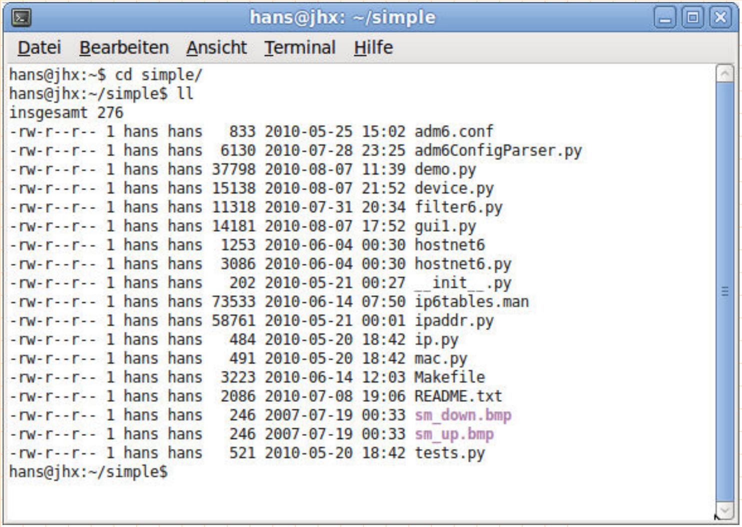
File: ~/.adm6.conf

```
# 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
```

device.py

```python
#
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()
            filter = Filter6(debuglevel,
                             path,
                             device_name,
                             device_os,
                             dev.interfaces)
            dev.do_rules(filter)
            #filter.mach_output(version)
    print "#"*80

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

source view – live !



```
hans@jhx:~$ cd simple/
hans@jhx:~/simple$ ll
insgesamt 276
-rw-r--r-- 1 hans hans   833 2010-05-25 15:02 adm6.conf
-rw-r--r-- 1 hans hans  6130 2010-07-28 23:25 adm6ConfigParser.py
-rw-r--r-- 1 hans hans 37798 2010-08-07 11:39 demo.py
-rw-r--r-- 1 hans hans 15138 2010-08-07 21:52 device.py
-rw-r--r-- 1 hans hans 11318 2010-07-31 20:34 filter6.py
-rw-r--r-- 1 hans hans 14181 2010-08-07 17:52 gui1.py
-rw-r--r-- 1 hans hans  1253 2010-06-04 00:30 hostnet6
-rw-r--r-- 1 hans hans  3086 2010-06-04 00:30 hostnet6.py
-rw-r--r-- 1 hans hans   202 2010-05-21 00:27 __init__.py
-rw-r--r-- 1 hans hans 73533 2010-06-14 07:50 ip6tables.man
-rw-r--r-- 1 hans hans 58761 2010-05-21 00:01 ipaddr.py
-rw-r--r-- 1 hans hans   484 2010-05-20 18:42 ip.py
-rw-r--r-- 1 hans hans   491 2010-05-20 18:42 mac.py
-rw-r--r-- 1 hans hans  3223 2010-06-14 12:03 Makefile
-rw-r--r-- 1 hans hans  2086 2010-07-08 19:06 README.txt
-rw-r--r-- 1 hans hans   246 2007-07-19 00:33 sm_down.bmp
-rw-r--r-- 1 hans hans   246 2007-07-19 00:33 sm_up.bmp
-rw-r--r-- 1 hans hans   521 2010-05-20 18:42 tests.py
hans@jhx:~/simple$
```

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 2_ε
and PSTricks