testing ip6tables

ip6tables – Setup

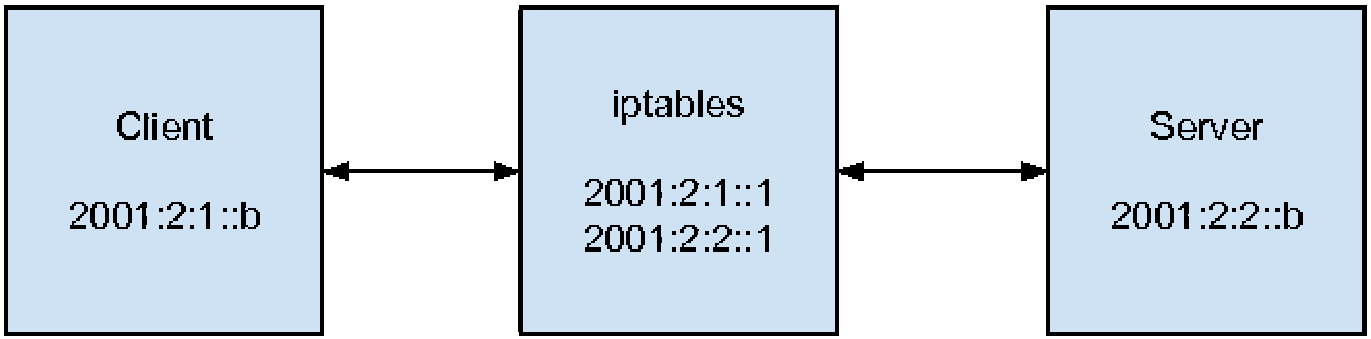

Abbildung : Network Setup

- Linux grml 3.7.1-grml-amd64 Debian 3.7.9+grml.1 x86_64
- ip6tables 1.4.18
- ft6 SVN-Revision 12442
- Python 2.7.3
- Scapy 2.2.0

ip6tables – Results

Test	Basic Rules	Advanced Rules
ICMPv6 Filtering	✗	✓
Routing Header	✗	✓
Header Chain	✗	✓[1]
Overlapping Fragments	✓	✓
Tiny IPv6 Fragments A	✗	✗
Tiny IPv6 Fragments B	✗	✗
Excessive HBH Options	✗	✓[1]
PadN Covert Channel	✗	✗
Address Scope	✓	✓

[1] not very elegant

ip6tables – Results

Test	Basic Rules	Advanced Rules
ICMPv6 Filtering	✗	✓
Routing Header	✗	✓
Header Chain	✗	✓[1]
Overlapping Fragments	✓	✓
Tiny IPv6 Fragments A	✗	✗
Tiny IPv6 Fragments B	✗	✗
Excessive HBH Options	✗	✓[1]
PadN Covert Channel	✗	✗
Address Scope	✓	✓

[1] not very elegant

ip6tables – Configuration

```
ip6tables -A FORWARD -p tcp --dport 80 -j ACCEPT
ip6tables -A FORWARD -p udp --dport 80 -j ACCEPT
ip6tables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
ip6tables -P FORWARD DROP
```

ip6tables – Routing Header

Check whether the firewall correctly forwards or discards packets containing a routing header.

default config

- ✓ Type = 0, segments-left = 0: forwarded
- ✗ Type = 0, segments-left $\neq$ 0: forwarded
- ✗ Type = 2, segments-left $\neq$ 1: forwarded
- !! Type = 2, segments-left = 1: forwarded
- !! Type = 200, segments-left = 0: forwarded
- ✗ Type = 200, segments-left $\neq$ 0: forwarded

ip6tables – Routing Header

Problem:

Packets are targeted at port 80, ip6tables doesn't check for routing headers by default.

Solution:

Use the `rt` module. But not like this:

```
ip6tables -A FORWARD -m rt --rt-type 0 -j ACCEPT
```

ip6tables would accept all packets with routing headers without checking the port! Rather: inspect routing headers in a separate chain.

ip6tables – Routing Header

advanced configuration:

```
ip6tables -N routinghdr
ip6tables -A routinghdr -m rt --rt-type 0 ! --rt-segsleft 0 -j DROP
ip6tables -A routinghdr -m rt --rt-type 2 ! --rt-segsleft 1 -j DROP
ip6tables -A routinghdr -m rt --rt-type 0 --rt-segsleft 0 -j RETURN
ip6tables -A routinghdr -m rt --rt-type 2 --rt-segsleft 1 -j RETURN
ip6tables -A routinghdr -m rt ! --rt-segsleft 0 --j DROP

ip6tables -A FORWARD -m ipv6header --header ipv6-route --soft \
     -j routinghdr
```

ip6tables – Routing Header

advanced config

- ✓ Type = 0, segments-left = 0: forwarded
- ✓ Type = 0, segments-left $\neq$ 0: dropped
- ✓ Type = 2, segments-left $\neq$ 1: dropped
- !! Type = 2, segments-left = 1: forwarded
- !! Type = 200, segments-left = 0: forwarded
- ✓ Type = 200, segments-left $\neq$ 0: dropped

ip6tables – Extension Header Chain

Check whether the firewall correctly filters packets based on number and order of header chains present.

default config

- ✗ HBH: dropped
- ✓ HBH-HBH: dropped
- ✓ HBH-DSTOPT-RH-HBH: dropped
- ✓ DSTOPT: forwarded
- ✗ DSTOPT-HBH: forwarded
- ✗ DSTOPT-DSTOPT: forwarded
- ✓ DSTOPT-RH-DSTOPT: forwarded

ip6tables – Extension Header Chain

Problem:

Packets are targeted at port 80, ip6tables doesn't check for extension headers by default.

Solution:

Use modules `ipv6header` or `ipv6headerorder` and check for extension headers in separate chain.
But: you would have to enumerate all possible combinations and write one rule for each of them. This wasn't performed in our test. We tested the modules for basic functionality though.
So, no advanced config.

ip6tables – Results

Test	Basic Rules	Advanced Rules
ICMPv6 Filtering	✗	✓
Routing Header	✗	✓
Header Chain	✗	✓[1]
Overlapping Fragments	✓	✓
Tiny IPv6 Fragments A (dropped)	✗	✗
Tiny IPv6 Fragments B (dropped)	✗	✗
Excessive HBH Options (see header chain)	✗	✓[1]
PadN Covert Channel	✗	✗
Address Scope	✓	✓

[1] not very elegant