9th Netfilter Workshop
10th to 14th March 2013
Copenhagen (Denmark)

IPv6 NAT

Open Source Days
9th-10th March 2013
Copenhagen, Denmark

Patrick McHardy <kaber@trash.net>

Netfilter and IPv6 NAT historically

http://lists.netfilter.org/pipermail/netfilter/2005-March/059463.html

Harald Welte laforge at netfilter.org
Wed Mar 30 17:23:16 CEST 2005
...

> When is support NAT table for Ip6tables?

Only over my dead body. We will never implement ipv6-to-ipv6 network address translation as long as I have any say in netfilter/iptables development. NAT is evil and causes horrible breakage of end-to-end on the internet. IPv6 has enough addresses and therefore no justification for NAT.

What’s wrong with NAT?

- Most cases of NAT need statefulness
- Commonly used scenario breaks end-to-end (masquerading)
- Network addresses are usually included as "pseudo header" in transport layer checksums
- New transport layer protocols need support from NAT
- Network addresses and port numbers may be included in application layer protocol payload
- Need ALGs (helpers) for every such application layer protocol
- Need to intercept new connections negotiated through NATed control protocols and direct them to proper destination (FTP data, SIP RTP, ...)
- Fragments, ICMP translation, IPsec, ...

Why IPv6 NAT?

- Security?

No. It’s a wide spread misconception that NAT provides security.

The only reason why the internal hosts of a NATed network are perceived to be "more secure" is because they often use globally unroutable addresses.

These addresses are just *globally* unroutable though, anything connected directly to the NAT router on the ouside (next hop router, cable modem neighbours, ...) can reach the internal network just fine, unless you use packet filtering.

Why IPv6 NAT?

- Dynamic IPv6 Prefixes

Some ISPs assign dynamic IPv6 prefixes, which means your internal networking addresses change at every prefix change.

If you want static internal addresses, you have little choice but to choose a different ISP or use NAT.

NAT can be performed bi-directional if desired, so incoming connections can be established to the internal hosts using the assigned prefix.

Why IPv6 NAT?

- Server Load Balancing

A very easy way to implement server load balancing is to perform destination

NAT on incoming requests to multiple addresses of a server farm.

There are other options, one advantage of using NAT is that the servers don’t need any awareness of being load balanced. As long as NAT doesn’t interfere with the proxied protocol, there’s nothing wrong with it.

Why IPv6 NAT?

- Proxy redirection

Transparent proxy redirection is often implemented using destination NAT to the proxy server/port.

There are other options how to implement this, but these are harder to set up. Similar to server load balancing, as long as you know it won’t interfere with the proxied protocol, there’s nothing wrong with it.

Why IPv6 NAT?

- Easy test network setup

When setting up a test network between virtual hosts that need to reach other networks, its very easy to just masquerade the outgoing traffic instead of requesting your own prefix that has to be known to other routers.

Why IPv6 NAT?

- People are doing it anyway
  - Users are asking for IPv6 NAT
  - Linux based router vendors have implemented it
  - Various out of tree patches for netfilter exist
  - Other operating systems have implemented it

Providing a single, well debugged implementation for Linux is better than having many different

implementations.

  - Users have less bugs
  - Users already know how it works
  - Application developers can predict the behaviour of the NAT

Why IPv6 NAT?

- End-to-end is not always negatively affected
- End-to-end is not always required or desired
- Many protocols are not negatively affected by NAT
- The address space is large enough not to need NAT, that doesn’t solve administrative problems
- Sometimes no other choice
- It seems unavoidable since users want it

Netfilter IPv6 NAT overview

Implementation started at the end of 2011, merged in 2.6.37.

- Largely based on the old IPv4-only implementation
- Address family independant core based on old IPv4-only NAT
- Address family specific layer 3 protocol modules
- Address family independant layer 4 protocol modules
- Address family specific NAT tables
- Address family independant SNAT/DNAT/NETMAP/REDIRECT targets
- New MASQUERADE target for IPv6
- NAT helpers adjusted to support IPv6
- Small improvements to conntrack fragmentation handling

Netfilter IPv6 NAT

- Address family independant core
  - Protocol book keeping
  - NAT mapping setup
  - Address/Port selection core
  - Packet mangling dispatch
  - IPsec session decoding dispatch
  - IPsec policy recomputation
- Address family specific layer 3 protocol modules (IPv4, IPv6)
  - Address selection
  - Address replacement in packets
  - Checksum updates
  - IPsec session decoding
  - ICMP/ICMPv6 error translations

Netfilter IPv6 NAT

- Address family independant layer 4 protocol modules (TCP, UDP, ...)
  - Port selection
  - Port replacement in packets
  - Checksum updates
- Address family specific NAT tables (iptable_nat, ip6table_nat)
  - Provide user-visible NAT table and chains
  - Bind NAT code to packet processing
  - Invoke IPsec policy recomputation after NAT
- NAT targets (MASQUERADE, SNAT, DNAT, ...)
  - Set up NAT mappings

Netfilter IPv6 NAT

- NAT helpers (FTP, SIP, ...)
  - Mangle addresses in application layer protocols
  - Redirect negotiated connections to the proper destination (FTP data, SIP RTP, ...)

Netfilter IPv6 User perspective

- User perspective similar to IPv4 NAT
  - IPv6 NAT table
  - PREROUTING/INPUT/OUTPUT/POSTROUTING chains
  - All IPv4 NAT targets supported
  - NAT rules invoked for NEW connections
  - Behaviour wrt. routing and IPsec similar to IPv4

Netfilter IPv6 User perspective

```
Chain PREROUTING (policy ACCEPT 13 packets, 3782 bytes)
    pkts      bytes target     prot opt in     out     source          destination
       0        0 REDIRECT   tcp      eth0   *       ::/0            ::/0          tcp dpt:80 redir ports 3128

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
    pkts      bytes target     prot opt in     out     source          destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
    pkts      bytes target     prot opt in     out     source          destination

Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
    pkts      bytes target     prot opt in     out     source          destination
       0        0 MASQUERADE  all      *      wlan0   ::/0            ::/0
```

Stateless Network Prefix Translation (RFC 6296)

- Stateless method of doing IPv6 NAT
- Only capable of mapping prefixes
- Performs checksum neutral address updates
- Preserves end-to-end reachability
- Transport agnostic

- Implemented as ip6tables target (SNPT/DNPT)
- Setup more complicated than stateful NAT

Stateless Network Prefix Translation (RFC 6296)

Example of prefix translation for packets from eth0 to eth1:

- If connection tracking is used, packets using NPT must be exempt
  - ip6tables -t raw -A PREROUTING -i eth0 -s 2001:db8::/64 -j NOTRACK
  - ip6tables -t raw -A PREROUTING -i eth1 -d 2001:f00f::/64 -j NOTRACK
- Prefix translation needs to be configured for both directions
  - ip6tables -t mangle -A POSTROUTING -o eth1 -s 2001:db8::/64 -j SNPT --src-pfx 2001:db8::/64 --dst-pfx 2001:f00f::/64
  - ip6tables -t mangle -A PREROUTING -i eth1 -d 2001:f00f::/64 -j DNPT --src-pfx 2001:f00f::/64 --dst-pfx 2001:db8::/64

Future plans

Current work in progress is NAT64,

- which allows to translate between IPv6 and IPv4.
- This allows you to run your network using IPv6 only while still being able to reach IPv4 only hosts.

Bulk work of the implementation is complete,
some tricky parts remain to be done.

Hope to finish the implementation during
the hacking days of next weeks Netfilter Workshop.

Questions?

http://goo.gl/MhV7M (YouTube) will answer all your quesions :)