strongSwan - Feature #1008

FARP for IPv6

24.06.2015 09:59 - Scep CAfail

Status:	Feedback	**Start date:**	24.06.2015
Priority:	Normal	**Due date:**	
Assignee:		**Estimated time:**	0.00 hour
Category:			
Target version:			
Resolution:			

Description

Hello,

I understand that it is advisable to use a prefix that is not assigned to any interfaces already, for being distributed to VPN clients. However when the IPv6 prefix is not routed to strongswan but instead is assigned to an interface already and the VPN pool is a portion of the same prefix, the upstream (ISP) router is sending a neighbor solicitation to strongswan assuming that the VPN client is within the same segment. Since it is not, ISP router does not get a neighbor advertisement and the forwarding is halted.

The solution is to
1) sysctl -w net.ipv6.conf.all.proxy_ndp=1
2) manually add the VPN client ip address in ndp proxy table via
ip -6 neigh add proxy 2a02:XXX:7c24:7e0::5 dev eth0

So that strongswan machine sends NDA on behalf of the VPN client 2a02:XXX:7c24:7e0::5 via the interface (eth0) facing the ISP.

I tried FARP plugin but no dice, I assume it is IPv4 only.
Let me know if what I am explaining makes sense

Regards

Associated revisions

Revision 1de31bcc - 06.08.2015 16:30 - Tobias Brunner

updown: Pass virtual IPs assigned to a peer to the updown script

Previously only received virtual IPs were available.

References #1008.

History

#1 - 24.06.2015 10:48 - Scep CAfail

Note that I am not using the DHCP plugin and I assume FARP should not depend on DHCP.

#2 - 24.06.2015 16:28 - Tobias Brunner

- Status changed from New to Feedback

There are several ways to implement this:

1. Create a *fndp* plugin that does what the *farp* plugin does, i.e. capture NS messages and manually respond with an appropriate NA message.
2. Create a *fndp-netlink* plugin does what you did via iproute2 but does so directly via Netlink (RTM_NEWNEIGH/NTF_PROXY).
3. Just use a custom updown script to automate the ip -6 neigh calls to add/remove the mapping.

The first option would be the most generic and might also work on other platforms (e.g. FreeBSD, however, so far nobody managed to port the *farp* plugin to FreeBSD). The second option would only work on Linux. The third option can be implemented easily (i.e does not require any development efforts) and could probably even be implemented on e.g. FreeBSD via ndp.

#3 - 25.06.2015 14:11 - Scep CAfail

Thanks for the rapid response Tobias!

Even though my traffic selectors (leftsubnet rightsubnet) are IPv6, the $PLUTO_MY_SOURCEIP returns the IPv4 addres of the VPN client (I read Issue #848). What is the correct reference to fetch virtual IPv6 address?

#4 - 25.06.2015 17:23 - Tobias Brunner

> Even though my traffic selectors (leftsubnet rightsubnet) are IPv6, the $PLUTO_MY_SOURCEIP returns the IPv4 addres of the VPN client (I read Issue #848). What is the correct reference to fetch virtual IPv6 address?

It won't work with that variable.

PLUTO_MY_SOURCEIP will contain the first virtual IP that is assigned. But all virtual IPs (including the first one) are passed in PLUTO_MY_SOURCEIP4_x and PLUTO_MY_SOURCEIP6_x to the script (x is an index starting with 1) - at least since 5.0.1.

However, these variables only contain the **local** virtual IPs, i.e. those that were assigned **by** the other peer. What's currently missing are PLUTO_PEER_SOURCEIP* variables that would contain the virtual IPs assigned **to** a peer, if any. It's strange that PLUTO_MY_SOURCEIP is set at all when the script runs on the responder in your case (or did you check on the initiator?). I pushed a commit to the *updown-peer-vip* branch that adds these variables.

Also, please be aware that the script might be called multiple times with the same virtual IPs but different local and remote traffic selectors (PLUTO_MY|PEER_CLIENT).

So in your case you could probably just check whether the remote traffic selector (PLUTO_PEER_CLIENT) ends with /128 and if so interpret that as virtual IP and just ignore PLUTO_*_SOURCEIP* (with the PLUTO_PEER_SOURCEIP* variables these could be compared to PLUTO_PEER_CLIENT to make sure they are virtual IPs).

#5 - 25.06.2015 18:44 - Scep CAfail

using MY_SOURCEIP was a mistake, I am working on the responder.
The traffic selector for remote traffic is a /97 (the entire VPN IPv6 pool) in ipsec.conf.
PLUTO_PEER_CLIENT fetches the /97 not individual /128.
Following is from syslog

assigning virtual IP 2001:470:1:1a5:1::1 to peer 'carol'
CHILD_SA rw{2} established with SPIs c9bd255e_i 194839bc_o and TS ::/0 === 2001:470:1:1a5:1::/97

So it assigns a /128 but establishes the CHILD_SA to /97. So every time a VPN client dials in, the same CHILD_SA will be created as duplicate?

Too bad that add proxy command works with /128 only.

#6 - 25.06.2015 18:56 - Tobias Brunner

> The traffic selector for remote traffic is a /97 (the entire VPN IPv6 pool) in ipsec.conf.

Which is wrong (see #1006#note-3 for my recent explanation why, it's for IPv4 but applies here too if you have *rightsubnet* configured).

> So it assigns a /128 but establishes the CHILD_SA to /97. So every time a VPN client dials in, the same CHILD_SA will be created as duplicate?

Yes, see link above.

> Too bad that add proxy command works with /128 only.

What do you mean?

#7 - 25.06.2015 19:14 - Scep CAfail

I sensed that using rightsubnet could be wrong and removed it. I just saw that you also suggested that way.
Now it all works, I appreciate your help Tobias!
For anyone who may need, following is the script

```
#/etc/strongswan.d/proxyndp.updown
case $PLUTO_VERB in
        up-client-v6)
        ip -6 neigh add proxy ${PLUTO_PEER_CLIENT%????} dev eth0
        ;;
        down-client-v6)
        ip -6 neigh delete proxy ${PLUTO_PEER_CLIENT%????} dev eth0
        ;;
esac
```

#8 - 14.12.2015 20:59 - Kilian Krause

...and in case your default interface for a route is not always eth0:

```
# cat /etc/strongswan.d/proxyndp.updown
OUTDEV=$(ip -6 r get ${PLUTO_PEER_CLIENT%????}|sed -ne 's,^.*dev \(\S\+\) .*,\1,p')
case $PLUTO_VERB in
        up-client-v6)
        ip -6 neigh add proxy ${PLUTO_PEER_CLIENT%????} dev ${OUTDEV:-eth0}
        ;;
        down-client-v6)
        ip -6 neigh delete proxy ${PLUTO_PEER_CLIENT%????} dev ${OUTDEV:-eth0}
        ;;
esac
```