strongSwan - Issue #455

strongSwan and sleeping states for laptops

22.11.2013 11:04 - Noel Kuntze

Status:	Closed		
Priority:	Normal		
Assignee:	Tobias Brunner		
Category:			
Affected version:	5.1.1	**Resolution:**	

Description

Hello,

I use strongSwan 5.1.1 on a notebook and I noticed, that when my laptop goes to sleep (S1 or S3) and wakes up again, strongSwan goes nuts and can't be stopped by starter.
When I invoke "ipsec statusall", it doesn't show any conns and networking doesn't work at all.
Terminating charon and starter with SIGTERM doesn't help. I have to remove /var/run/charon.{pid,ctl} by hand.
XFRM policies and states aren't there. Flushing the XFRM state and policies doesn't help at restoring network connectivity.
When I start strongSwan again, no conns are shown. Only rebooting helps and fixes all the things.
Is this known? If it is, how can it be fixed?

Regards
Noel Kuntze

Related issues:

Related to Feature #3300: support dpd and dpdaction = restart in the network ...	**Closed**	**20.12.2019**

History

#1 - 22.11.2013 18:15 - Noel Kuntze

- File charon_2.log added

After the laptop wakes up again, "ipsec statusall" shows this:

```
Status of IKE charon daemon (strongSwan 5.1.1, Linux 3.12.0-1-ARCH,
x86_64):
  uptime: 47 seconds, since Nov 22 17:09:09 2013
  malloc: sbrk 2420736, mmap 0, used 436560, free 1984176
  worker threads: 11 of 16 idle, 5/0/0/0 working, job queue: 0/0/0/0,
scheduled: 4
  loaded plugins: charon test-vectors curl random nonce x509 revocation
constraints pubkey pkcs1 pem openssl af-alg gmp xcbc cmac hmac ccm attr
kernel-netlink socket-default farp stroke updown eap-identity eap-gtc
eap-mschapv2 eap-radius xauth-generic xauth-eap unity
Listening IP addresses:
  192.168.178.84
Connections:
        home:  %any...cdgsthermi.no-ip.org  IKEv2, dpddelay=10s
        home:   local:  [C=DE, ST=Baden-W??rttemberg, O=ThermiCorp,
OU=Users, CN=Thermi Thinkpad, E=Thermi_Thinkpad@cdgsthermi.no-ip.org]
uses public key authentication
        home:    cert:  "C=DE, ST=Baden-W??rttemberg, O=ThermiCorp,
OU=Users, CN=Thermi Thinkpad, E=Thermi_Thinkpad@cdgsthermi.no-ip.org"
        home:   remote: [cdgsthermi.no-ip.org] uses public key
authentication
        home:   child:  dynamic === ::/0 192.168.178.0/24 TUNNEL,
dpdaction=restart
     tunnel6:   child:  dynamic === 0.0.0.0/0 ::/0 TUNNEL,
dpdaction=restart
      server:  %any...192.168.178.48  IKEv2, dpddelay=10s
      server:   local:  [C=DE, ST=Baden-W??rttemberg, O=ThermiCorp,
OU=Users, CN=Thermi Thinkpad, E=Thermi_Thinkpad@cdgsthermi.no-ip.org]
uses public key authentication
      server:    cert:  "C=DE, ST=Baden-W??rttemberg, O=ThermiCorp,
OU=Users, CN=Thermi Thinkpad, E=Thermi_Thinkpad@cdgsthermi.no-ip.org"
      server:   remote: [cdgsthermi.no-ip.org] uses public key
```

```
authentication
      server:   child:  dynamic === dynamic TUNNEL, dpdaction=restart
        test:  %any...192.168.178.43  IKEv2, dpddelay=10s
        test:   local:  [C=DE, ST=Baden-W??rttemberg, O=ThermiCorp,
OU=Users, CN=Thermi Thinkpad, E=Thermi_Thinkpad@cdgsthermi.no-ip.org]
uses pre-shared key authentication
        test:    cert:  "C=DE, ST=Baden-W??rttemberg, O=ThermiCorp,
OU=Users, CN=Thermi Thinkpad, E=Thermi_Thinkpad@cdgsthermi.no-ip.org"
        test:   remote: [thermi-desktop] uses pre-shared key
authentication
        test:   child:  dynamic === 0.0.0.0/0 TUNNEL, dpdaction=restart
Security Associations (1 up, 0 connecting):
      server[1]: ESTABLISHED 38 seconds ago, 192.168.178.84[C=DE,
ST=Baden-W??rttemberg, O=ThermiCorp, OU=Users, CN=Thermi Thinkpad,
E=Thermi_Thinkpad@cdgsthermi.no-ip.org]...192.168.178.48[cdgsthermi.no-ip.org]
      server[1]: IKEv2 SPIs: 7779a1365e1338fe_i* 9af1cd8a98c0395b_r,
rekeying in 53 minutes
      server[1]: IKE proposal: AES_GCM_16_256/PRF_HMAC_SHA2_256/ECP_521
      server[1]: Tasks active: IKE_DPD
      server{1}:  INSTALLED, TUNNEL, ESP SPIs: c686746c_i ca1c4406_o
      server{1}:  AES_GCM_16_256, 53502 bytes_i (37 pkts, 637s ago),
6024 bytes_o (46 pkts, 637s ago), rekeying in 13 minutes
      server{1}:   172.16.19.1/32 === 192.168.178.48/32
```

Note the active task.

And when I try to down the conn, then this happens:

```
ipsec down server
retransmit 2 of request with message ID 3
sending packet: from 192.168.178.84[4500] to 192.168.178.48[4500] (57
bytes)
retransmit 3 of request with message ID 3
sending packet: from 192.168.178.84[4500] to 192.168.178.48[4500] (57
bytes)
retransmit 4 of request with message ID 3
sending packet: from 192.168.178.84[4500] to 192.168.178.48[4500] (57
bytes)
retransmit 5 of request with message ID 3
sending packet: from 192.168.178.84[4500] to 192.168.178.48[4500] (57
bytes)
giving up after 5 retransmits
installing new virtual IP 172.16.19.1
restarting CHILD_SA server
initiating IKE_SA server[2] to 192.168.178.48
generating IKE_SA_INIT request 0 [ SA KE No N(NATD_S_IP) N(NATD_D_IP) V
]
sending packet: from 192.168.178.84[500] to 192.168.178.48[500] (1060
bytes)
closing IKE_SA [1] failed
```

This indicates that there are DPD related tasks queued up. Optimizing dpddelay, dpdtimeout and dpdaction on the initiator and responder sides might help with this problem.
A better, but more complex solution is to try to detect if the computer goes to sleep (you can do this by listening to ACPI events) and when waking up, check if the dpdtimeout of any conn could have caused any conn to be torn down on the remote side. If any have been, reestablish them.

#2 - 22.11.2013 19:02 - Tobias Brunner

- Status changed from New to Feedback

- Assignee set to Tobias Brunner

The reason for ipsec down not working properly is documented in #429 (actually, even if the DELETE task would get queued there would be a delay if the other peer is not reachable).

> Optimizing dpddelay, dpdtimeout and dpdaction on the initiator and responder sides might help with this problem.

The problem is that no matter what *dpddelay* you configure it could happen that the task is queued just before the host goes to sleep (or that DPDs are already being sent, but not answered, before that happens).

> A better, but more complex solution is to try to detect if the computer goes to sleep (you can do this by listening to ACPI events) and when waking up, check if the dpdtimeout of any conn could have caused any conn to be torn down on the remote side. If any have been, reestablish them.

Since we don't know what DPD and retransmission timeouts the other peer has configured, we don't know for sure if it lost the state or not. We could use our own configuration as estimate but currently timeouts are determined using a monotonic clock and timestamps, so they might not have changed at all during the suspend, so this wouldn't work that easily. I'm not sure either if listening to ACPI events would be all that practical.

But there are certainly things we could do to try to remedy this. For instance, what we currently don't do is send back INVALID_SPI notifications. The responder may send those back, if it receives a packet with unknown SPIs (IKE or ESP). Since those are always sent unencrypted in the IKE case the client can't just act on them. But what it could do, for example, is reduce the retransmission timeouts (or the number of retransmits).

Even better would be to implement RFC 6290, as that would allow the client to learn that the other peer lost the state and so it could immediately reestablish the SA, without any more retransmits.

#3 - 16.04.2015 10:30 - Tobias Brunner

- Status changed from Feedback to Closed

#4 - 18.05.2020 13:59 - Tobias Brunner

- Related to Feature #3300: support dpd and dpdaction = restart in the network manager app added

Files

charon_2.log	907 KB	22.11.2013	Noel Kuntze