Hi,

In our live production systems we are experiencing link instability on few occasions (sometimes in one direction, server to client direction). Typically DPD is detected and initiator keeps retrying the connection. On one instance, initiator keeps getting COOKIE notifications from server and after 5 retries, it stops the retry. I have attached the log below. I understand that this COOKIE notifications are sent by server when there are more than "threshold" number of half open IKE SA connections.

Couple of questions.
1. Why is the limit 5 here? Is it to prevent some kind of attack?
2. Our DPD/retransmission configuration is as below (~9s). But somehow in the logs below, I see faster retransmissions happening whenever there is a COOKIE notify. Is this expected behaviour.

strongswan.conf:

```
charon {
    load_modular = yes
    make_before_break = yes
    install_routes = no
    retransmit_timeout = 2
    retransmit_tries = 2
    retransmit_base = 1.0

    plugins {
        include strongswan.d/charon/*.conf
    }

    filelog {
        charon {
            path = stdout
            # add a timestamp prefix
            time_format = %Y-%m-%dT%H:%M:%S.0%z
            # prepend connection name, simplifies grepping
            ike_name = yes
            # increase default loglevel for all daemon subsystems
            default = 1
            # flush each line to disk
            flush_line = yes
        }
    }
}

include strongswan.d/*.conf
```

3. Is there any way to prevent this stopping of the initiator retrying the connection? Is increasing the MAX_RETRIES an option?

Threshold for IKE half open SAs is default (10). Is increasing this an option?

IKE initiator stop issue:

```
2020-04-17T17:24:25.0+0000 10[IKE] <server_prod|2412> establishing CHILD_SA server_prod[912]
2020-04-17T17:24:25.0+0000 10[IKE] <server_prod|2412> get_spi: Called
2020-04-17T17:24:25.0+0000 10[ENC] <server_prod|2412> splitting IKE message (2800 bytes) into 3 fragments
```
2020-04-17T24:25:0+0000 11[ENC] <server_prod|2412> received fragment #1 of 2, waiting for complete IKE message
2020-04-17T24:27:0+0000 11[IKE] <server_prod|2412> retransmit 1 of request with message ID 1
2020-04-17T24:29:0+0000 06[IKE] <server_prod|2412> retransmit 2 of request with message ID 1
2020-04-17T24:31:0+0000 06[IKE] <server_prod|2412> giving up after 2 retransmits
2020-04-17T24:31:0+0000 06[IKE] <server_prod|2412> peer not responding, trying again (18/0)
2020-04-17T24:31:0+0000 06[IKE] <server_prod|2412> initiating IKE_SA server_prod[2412] to 103.1.35.57.126

2020-04-17T24:33:0+0000 08[IKE] <server_prod|2412> received COOKIE notify
2020-04-17T24:35:0+0000 11[IKE] <server_prod|2412> retransmit 2 of request with message ID 0
2020-04-17T24:35:0+0000 13[IKE] <server_prod|2412> initiated IKE_SA server_prod[2412] to 103.1.35.57.126

2020-04-17T24:37:0+0000 09[IKE] <server_prod|2412> giving up after 2 retransmits
2020-04-17T24:37:0+0000 09[IKE] <server_prod|2412> peer not responding, trying again (19/0)
2020-04-17T24:37:0+0000 09[IKE] <server_prod|2412> initiating IKE_SA server_prod[2412] to 103.1.35.57.126

2020-04-17T24:39:0+0000 08[IKE] <server_prod|2412> received COOKIE notify
2020-04-17T24:39:0+0000 04[IKE] <server_prod|2412> initiating IKE_SA server_prod[2412] to 103.1.35.57.126

2020-04-17T24:41:0+0000 08[IKE] <server_prod|2412> received COOKIE notify
2020-04-17T24:41:0+0000 13[IKE] <server_prod|2412> initiating IKE_SA server_prod[2412] to 103.1.35.57.126

2020-04-17T24:43:0+0000 08[IKE] <server_prod|2412> giving up after 2 retransmits
2020-04-17T24:43:0+0000 08[IKE] <server_prod|2412> peer not responding, trying again (20/0)
2020-04-17T24:43:0+0000 08[IKE] <server_prod|2412> initiating IKE_SA server_prod[2412] to 103.1.35.57.126

2020-04-17T24:49:0+0000 08[IKE] <server_prod|2412> giving up after 2 retransmits
2020-04-17T24:49:0+0000 08[IKE] <server_prod|2412> peer not responding, trying again (21/0)
2020-04-17T24:49:0+0000 08[IKE] <server_prod|2412> initiating IKE_SA server_prod[2412] to 103.1.35.57.126

2020-04-17T24:51:0+0000 13[IKE] <server_prod|2412> giving up after 2 retransmits
2020-04-17T24:51:0+0000 13[IKE] <server_prod|2412> peer not responding, trying again (22/0)
2020-04-17T24:51:0+0000 13[IKE] <server_prod|2412> initiating IKE_SA server_prod[2412] to 103.1.35.57.126

2020-04-17T24:53:0+0000 15[IKE] <server_prod|2412> giving up after 2 retransmits
2020-04-17T24:53:0+0000 15[IKE] <server_prod|2412> peer not responding, trying again (23/0)
2020-04-17T24:53:0+0000 15[IKE] <server_prod|2412> initiating IKE_SA server_prod[2412] to 103.1.35.57.126

2020-04-17T24:55:0+0000 15[IKE] <server_prod|2412> giving up after 2 retransmits
2020-04-17T24:55:0+0000 15[IKE] <server_prod|2412> peer not responding, trying again (24/0)
2020-04-17T24:55:0+0000 15[IKE] <server_prod|2412> initiating IKE_SA server_prod[2412] to 103.1.35.57.126

2020-04-17T24:57:0+0000 15[IKE] <server_prod|2412> giving up after 2 retransmits
2020-04-17T24:57:0+0000 15[IKE] <server_prod|2412> peer not responding, trying again (25/0)
2020-04-17T24:57:0+0000 15[IKE] <server_prod|2412> initiating IKE_SA server_prod[2412] to 103.1.35.57.126

2020-04-17T24:59:0+0000 14[IKE] <server_prod|2412> giving up after 5 retries
1. Why is the limit 5 here? Is it to prevent some kind of attack?

The option intends to limit DoS attacks. But the default value is pretty much arbitrary.

2. Our DPD/retransmission configuration is as below (~9s). But somehow in the logs below, I see faster retransmissions happening whenever there is a COOKIE notify. Is this expected behaviour.

These are unrelated. The response/retry to a COOKIE notify is sent out immediately, not when retransmits are scheduled.

```
retransmit_timeout = 2
retransmit_tries = 2
retransmit_base = 1.0
```

That doesn't allow for much network interruption. If the responder is in a condition to reply with COOKIE notifies, it will do so constantly because the client will not be able to get its request with COOKIE through. So it will abort and then restart without COOKIE, just to receive a COOKIE again.

3. Is there any way to prevent this stopping of the initiator retrying the connection? Is increasing the MAX RETRIES an option? Threshold for IKE half open SAs is default (10). Is increasing this an option?

As discussed before (#3388) try using a newer release.

#2 - 21.04.2020 13:30 - Krishnamurthy Daulatabad

Tobias Brunner wrote:

That doesn't allow for much network interruption. If the responder is in a condition to reply with COOKIE notifies, it will do so constantly because the client will not be able to get its request with COOKIE through. So it will abort and then restart without COOKIE, just to receive a COOKIE again.

I did not get why the client will not be able to get the request through with COOKIE. Are you saying that client SA INIT request with COOKIE is not reaching the server due to link issues? After how many attempts, does the client abort and restart without COOKIE?

3. Is there any way to prevent this stopping of the initiator retrying the connection? Is increasing the MAX RETRIES an option? Threshold for IKE half open SAs is default (10). Is increasing this an option?

As discussed before (#3388) try using a newer release.

I have already picked the fix done in 5.7.2 (https://wiki.strongswan.org/projects/strongswan/repository/revisions/01f462f0d327bc2b88c0e056be2). Are there any other fixes in newer releases related to this?

#3 - 21.04.2020 15:08 - Tobias Brunner

- Resolution set to Can't reproduce

Are you saying that client SA INIT request with COOKIE is not reaching the server due to link issues?

That's what it looks like, but you are the only one who can check (read the logs, capture traffic).

After how many attempts, does the client abort and restart without COOKIE?

Whatever you configure as retransmission settings.

I have already picked the fix done in 5.7.2 (
Possible, I've no idea. I really don't care for debugging old versions.

#4 - 22.04.2020 19:57 - Krishnamurthy Daulatabad

One quick question regarding sending of COOKIES from the responder. I see from the code at src/libcharon/network/receiver.c that responder keeps sending cookie notifications as long as:
1. It has more than 10 (default) half open SAs
2. For COOKIE_CALMDOWN_DELAY which is 10s

So, in both the cases, if a valid client sends a SA_INIT request, it will get a COOKIE notification. This will trigger an immediate retry of SA_INIT and rx of COOKIE notification. So this will continue and client will stop after 5 retries.

Is there any backoff in client to protect this condition? Or am I missing something here?

#5 - 23.04.2020 09:31 - Tobias Brunner

So this will continue and client will stop after 5 retries.

Only if the responder does not accept the IKE_SA_INIT with the COOKIE and responds with another (which it should not and if the same COOKIE is received that should be ignored). But if the client's request does not get through and the responder's IKE_SA_INIT messages are delayed too, there might be several responses with different COOKIE notifies around, triggering this issue. Again, if you have bad connectivity it might not be ideal to have such low retransmission settings.