strongSwan - Bug #3364

stongSwan Android APP no keep-alive / MOBIKE are sent during sleep

06.03.2020 10:45 - Beat Zahnd

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Tobias Brunner	**Estimated time:**	0.00 hour
Category:	android		
Target version:	5.9.0		
Affected version:	5.8.2	**Resolution:**	Fixed

Description

I migrated from IPsec/L2TP to IKEv2 with certificates. Using the Android stronSwan App the connection works well as long as the mobile phone is not put to sleep. The server is libreswan 3.27 on Debian.

After a few minutes on sleep no more traffic is possible. If for about 2 minutes no traffic is exchange the cellular network NAT changes its port. In this tcpdump the port changes from 21977 to 21978. But the server continues to send to 21977. There was no MOBIKE ticket sent.

```
20:43:26.048376 IP 84.75.x.x.4500 > 178.197.x.x.21977: UDP-encap: ESP(spi=0xbe045de8,seq=0x178), l
ength 88
20:43:26.080322 IP 84.75.x.x.4500 > 178.197.x.x.21977: UDP-encap: ESP(spi=0xbe045de8,seq=0x179), l
ength 88
20:43:26.164652 IP 178.197.x.x.21977 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x85bb58ce,seq=0x178), l
ength 88
20:43:26.166635 IP 178.197.x.x.21977 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x85bb58ce,seq=0x179), l
ength 88
20:43:26.168593 IP 178.197.x.x.21977 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x85bb58ce,seq=0x17a), l
ength 88
20:43:26.214647 IP 178.197.x.x.21977 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x85bb58ce,seq=0x17b), l
ength 100
20:43:26.216616 IP 178.197.x.x.21977 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x85bb58ce,seq=0x17c), l
ength 100
20:43:26.218614 IP 178.197.x.x.21977 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x85bb58ce,seq=0x17d), l
ength 100
20:43:26.220576 IP 178.197.x.x.21977 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x85bb58ce,seq=0x17e), l
ength 100
20:43:26.222600 IP 178.197.x.x.21977 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x85bb58ce,seq=0x17f), l
ength 100
20:44:45.725612 IP 178.197.x.x.21977 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x85bb58ce,seq=0x180), l
ength 88
20:44:45.725670 IP 178.197.x.x.21977 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x85bb58ce,seq=0x181), l
ength 88
20:44:45.725681 IP 178.197.x.x.21977 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x85bb58ce,seq=0x182), l
ength 88
20:44:45.725849 IP 84.75.x.x.4500 > 178.197.x.x.21977: UDP-encap: ESP(spi=0xbe045de8,seq=0x17a), l
ength 88
20:44:45.726333 IP 178.197.x.x.21977 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x85bb58ce,seq=0x183), l
ength 88

20:47:52.022713 IP 178.197.x.x.21978 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x85bb58ce,seq=0x184), l
ength 88
20:47:52.022804 IP 178.197.x.x.21978 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x85bb58ce,seq=0x185), l
ength 88
20:47:52.022836 IP 178.197.x.x.21978 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x85bb58ce,seq=0x186), l
ength 88
20:47:52.022858 IP 178.197.x.x.21978 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x85bb58ce,seq=0x187), l
ength 88
20:47:52.022879 IP 178.197.x.x.21978 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x85bb58ce,seq=0x188), l
ength 88
20:47:52.022892 IP 178.197.x.x.21978 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x85bb58ce,seq=0x189), l
ength 88
```

```
20:47:52.022909 IP 178.197.x.x.21978 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x85bb58ce,seq=0x18a), l
ength 88
20:47:52.022922 IP 178.197.x.x.21978 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x85bb58ce,seq=0x18b), l
ength 88
20:47:52.024400 IP 178.197.x.x.21978 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x85bb58ce,seq=0x18c), l
ength 88
20:47:52.041538 IP 178.197.x.x.21978 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x85bb58ce,seq=0x18d), l
ength 96
20:47:52.041633 IP 178.197.x.x.21978 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x85bb58ce,seq=0x18e), l
ength 88
20:47:52.041674 IP 178.197.x.x.21978 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x85bb58ce,seq=0x18f), l
ength 88
20:47:52.041701 IP 178.197.x.x.21978 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x85bb58ce,seq=0x190), l
ength 88
20:47:52.041725 IP 178.197.x.x.21978 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x85bb58ce,seq=0x191), l
ength 88
20:47:52.041745 IP 178.197.x.x.21978 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x85bb58ce,seq=0x192), l
ength 96
20:47:52.041765 IP 178.197.x.x.21978 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x85bb58ce,seq=0x193), l
ength 88
20:47:52.041785 IP 178.197.x.x.21978 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x85bb58ce,seq=0x194), l
ength 88
20:47:52.041961 IP 84.75.x.x.4500 > 178.197.x.x.21977: UDP-encap: ESP(spi=0xbe045de8,seq=0x17b), l
ength 96
20:47:52.042104 IP 84.75.x.x.4500 > 178.197.x.x.21977: UDP-encap: ESP(spi=0xbe045de8,seq=0x17c), l
ength 96
```

Previously with IPsec L2TP there was nat-keep-alive with about 1 min interval. This is not seen now using the strongswan App.

```
10:22:13.362197 IP 178.197.235.175.65141 > 84.75.x.x.4500: isakmp-nat-keep-alive
10:23:14.333471 IP 178.197.235.175.65141 > 84.75.x.x.4500: isakmp-nat-keep-alive
10:24:15.802514 IP 178.197.235.175.65141 > 84.75.x.x.4500: isakmp-nat-keep-alive
```

NAT-T keep-alive time has been set to 15. But this does not help.

MOBIKE works. If i switch on WiFI or off then MOBIKE cookies are sent and the connection works again, as long as the mobile phone is not on sleep.

The mobile is a Sony with Android 8

```
Mar  3 22:52:58 00[DMN] +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Mar  3 22:52:58 00[DMN] Starting IKE service (strongSwan 5.8.2dr1, Android 8.0.0 - 34.4.A.2.118/20
18-09-01, F5321 - Sony/F5321/Sony, Linux 3.10.84-perf-g51b663cc723-04742-g5bcfaca9e61, aarch64)
Mar  3 22:52:58 00[LIB] loaded plugins: androidbridge charon android-log openssl fips-prf random n
once pubkey chapoly curve25519 pkcs1 pkcs8 pem xcbc hmac socket-default revocation eap-identity ea
p-mschapv2 eap-md5 eap-gtc eap-tls x509
Mar  3 22:52:58 00[JOB] spawning 16 worker threads
Mar  3 22:52:58 08[CFG] loaded user certificate 'CN=bz' and private key
Mar  3 22:52:58 08[CFG] loaded CA certificate 'C=CH, ST=ZH, O=hackersGarden'
Mar  3 22:52:58 08[IKE] initiating IKE_SA android[7] to 84.75.x.x
Mar  3 22:52:58 08[ENC] generating IKE_SA_INIT request 0 [ SA KE No N(NATD_S_IP) N(NATD_D_IP) N(FR
AG_SUP) N(HASH_ALG) N(REDIR_SUP) ]
Mar  3 22:52:58 08[NET] sending packet: from 10.125.64.234[37422] to 84.75.x.x[500] (716 bytes)
Mar  3 22:52:58 07[NET] received packet: from 84.75.x.x[500] to 10.125.64.234[37422] (38 bytes)
Mar  3 22:52:58 07[ENC] parsed IKE_SA_INIT response 0 [ N(INVAL_KE) ]
Mar  3 22:52:58 07[IKE] peer didn't accept DH group ECP_256, it requested MODP_2048
Mar  3 22:52:58 07[IKE] initiating IKE_SA android[7] to 84.75.x.x
Mar  3 22:52:58 07[ENC] generating IKE_SA_INIT request 0 [ SA KE No N(NATD_S_IP) N(NATD_D_IP) N(FR
AG_SUP) N(HASH_ALG) N(REDIR_SUP) ]
Mar  3 22:52:58 07[NET] sending packet: from 10.125.64.234[37422] to 84.75.x.x[500] (908 bytes)
Mar  3 22:52:58 12[NET] received packet: from 84.75.x.x[500] to 10.125.64.234[37422] (457 bytes)
Mar  3 22:52:58 12[ENC] parsed IKE_SA_INIT response 0 [ SA KE No N(FRAG_SUP) N(NATD_S_IP) N(NATD_D
_IP) CERTREQ ]
Mar  3 22:52:58 12[CFG] selected proposal: IKE:AES_GCM_16_256/PRF_HMAC_SHA2_512/MODP_2048
Mar  3 22:52:58 12[IKE] local host is behind NAT, sending keep alives
Mar  3 22:52:58 12[IKE] received cert request for "C=CH, ST=ZH, O=hackersGarden"
Mar  3 22:52:58 12[IKE] sending cert request for "C=CH, ST=ZH, O=hackersGarden"
Mar  3 22:52:59 12[IKE] authentication of 'CN=bz' (myself) with RSA signature successful
Mar  3 22:52:59 12[IKE] sending end entity cert "CN=bz"
Mar  3 22:52:59 12[IKE] establishing CHILD_SA android{7}
Mar  3 22:52:59 12[ENC] generating IKE_AUTH request 1 [ IDi CERT N(INIT_CONTACT) CERTREQ AUTH CPRQ
(ADDR ADDR6 DNS DNS6) N(ESP_TFC_PAD_N) SA TSi TSr N(MOBIKE_SUP) N(NO_ADD_ADDR) N(EAP_ONLY) N(MSG_I
D_SYN_SUP) ]
Mar  3 22:52:59 12[ENC] splitting IKE message (2022 bytes) into 2 fragments
Mar  3 22:52:59 12[ENC] generating IKE_AUTH request 1 [ EF(1/2) ]
Mar  3 22:52:59 12[ENC] generating IKE_AUTH request 1 [ EF(2/2) ]
Mar  3 22:52:59 12[NET] sending packet: from 10.125.64.234[41633] to 84.75.x.x[4500] (1368 bytes)
Mar  3 22:52:59 12[NET] sending packet: from 10.125.64.234[41633] to 84.75.x.x[4500] (719 bytes)
Mar  3 22:52:59 16[NET] received packet: from 84.75.x.x[4500] to 10.125.64.234[41633] (535 bytes)
Mar  3 22:52:59 16[ENC] parsed IKE_AUTH response 1 [ EF(1/4) ]
Mar  3 22:52:59 16[ENC] received fragment #1 of 4, waiting for complete IKE message
Mar  3 22:52:59 13[NET] received packet: from 84.75.x.x[4500] to 10.125.64.234[41633] (535 bytes)
Mar  3 22:52:59 13[ENC] parsed IKE_AUTH response 1 [ EF(2/4) ]
Mar  3 22:52:59 13[ENC] received fragment #2 of 4, waiting for complete IKE message
Mar  3 22:52:59 14[NET] received packet: from 84.75.x.x[4500] to 10.125.64.234[41633] (535 bytes)
Mar  3 22:52:59 14[ENC] parsed IKE_AUTH response 1 [ EF(3/4) ]
Mar  3 22:52:59 14[ENC] received fragment #3 of 4, waiting for complete IKE message
Mar  3 22:52:59 15[NET] received packet: from 84.75.x.x[4500] to 10.125.64.234[41633] (366 bytes)
Mar  3 22:52:59 15[ENC] parsed IKE_AUTH response 1 [ EF(4/4) ]
Mar  3 22:52:59 15[ENC] received fragment #4 of 4, reassembled fragmented IKE message (1784 bytes)
Mar  3 22:52:59 15[ENC] parsed IKE_AUTH response 1 [ IDr CERT AUTH CPRP(ADDR DNS) SA TSi TSr ]
Mar  3 22:52:59 15[IKE] received end entity cert "CN=hackersgarden.dyn.ch"
Mar  3 22:52:59 15[CFG]   using trusted ca certificate "C=CH, ST=ZH, O=hackersGarden"
Mar  3 22:52:59 15[CFG] checking certificate status of "CN=bz"
Mar  3 22:52:59 15[CFG] certificate status is not available
Mar  3 22:52:59 15[CFG]   reached self-signed root ca with a path length of 0
Mar  3 22:52:59 15[CFG]   using trusted certificate "CN=bz"
Mar  3 22:52:59 15[IKE] signature validation failed, looking for another key
Mar  3 22:52:59 15[CFG]   using certificate "CN=hackersgarden.dyn.ch"
Mar  3 22:52:59 15[CFG]   using trusted ca certificate "C=CH, ST=ZH, O=hackersGarden"
Mar  3 22:52:59 15[CFG] checking certificate status of "CN=hackersgarden.dyn.ch"
Mar  3 22:52:59 15[CFG] certificate status is not available
Mar  3 22:52:59 15[CFG]   reached self-signed root ca with a path length of 0
Mar  3 22:52:59 15[IKE] authentication of 'hackersgarden.dyn.ch' with RSA signature successful
Mar  3 22:52:59 15[IKE] IKE_SA android[7] established between 10.125.64.234[CN=bz]...84.75.x.x[hac
kersgarden.dyn.ch]
Mar  3 22:52:59 15[IKE] scheduling rekeying in 35506s
Mar  3 22:52:59 15[IKE] maximum IKE_SA lifetime 36106s
Mar  3 22:52:59 15[IKE] installing DNS server 10.76.1.1
Mar  3 22:52:59 15[IKE] installing new virtual IP 192.168.1.100
Mar  3 22:52:59 15[IKE] CHILD_SA android{7} established with SPIs dd2dd5ff_i 1acc2969_o and TS 192
.168.1.100/32 === 0.0.0.0/0
Mar  3 22:52:59 15[DMN] setting up TUN device for CHILD_SA android{7}
Mar  3 22:52:59 15[DMN] successfully created TUN device
```

Any ideas why the keep-alive and MOBIKE does not work on the cellular network?

Associated revisions

Revision a22a1493 - 02.06.2020 14:34 - Tobias Brunner

Merge branch 'android-scheduler'

Starting with Android 6, the system will aggressively suspend apps when
the device is idle (Doze mode). With Android 10 on a Pixel 4 this seems
to happen after about 70 minutes. Then the scheduler thread in our
default scheduler is only woken rarely, combined with our previous use
of the monotonic clock it meant that events were executed with severe
delays and noticing that there was such a delay. This was particularly
bad in regards to NAT keepalives as it usually meant that the device was
not reachable anymore from the outside.

Some changes here try to improve that situation, e.g. the clock is switched
to CLOCK_REALTIME (Bionic doesn't support CLOCK_BOOTTIME for condvars) so we
can measure the actual difference e.g. since the last outbound message,

other changes try to ensure that connectivity is restored after being asleep
for a while (send DPD instead of keepalive after a long delay, send DPD even
if path to peer stays the same).

However, the most significant change is the replacement of the default
scheduler with one specifically designed for Android. It schedules
long-term events via AlarmManager, which allows waking up the app even
if the system put it to sleep. The latter requires adding the app to the
system's battery optimization whitelist, which is requested from the
user automatically if necessary. With this, NAT keepalives and rekeyings
are now scheduled accurately, with little changes to the battery usage.
If the app is not whitelisted (there is a setting to ignore this), events
are delayed by up to 15 minutes after about 70 minutes, so behind a NAT
the device won't be reachable from the outside afterwards (connectivity
should be restored as soon as the device is woken from deep sleep by the
user).

Fixes #3364.

History

#1 - 06.03.2020 11:53 - Tobias Brunner

- Status changed from New to Feedback

> Using the Android stronSwan App the connection works well as long as the mobile phone is not put to sleep.

What do you mean with "put to sleep" **exactly**?

> There was no MOBIKE ticket sent.

MOBIKE will only cause an exchange if the client's IP changes. (Although, there is a patch somewhere that triggers a DPD, with NAT-T payloads to detect NAT mapping changes, even if the IP stays the same after addresses have changed - whether that would be the case here and it would help, is another question, though.)

> Previously with IPsec L2TP there was nat-keep-alive with about 1 min interval. This is not seen now using the strongswan App.

You can't compare the two. The former runs as Android system service, the latter is a simple app. If Android suspends it, no NAT keepalives can be sent (and obviously no inbound traffic from the server could be processed either). The keepalives are also sent from the C part of the app, which Android might not know much about (if that even would be a consideration).

> NAT-T keep-alive time has been set to 15. But this does not help.

So the app is already suspended after 15 seconds? Or is the whole device kind of off?

> The mobile is a Sony with Android 8

Could be a factor if their system image does force suspension of any app/process after a short while, no idea.

> Any ideas why the keep-alive and MOBIKE does not work on the cellular network?

The app does not keep any explicit wake locks, in case that would make a difference (Android can ignore them). However, it is registered as foreground service and it has a VpnService instance open. I wonder what would happen if the device had some traffic to send that required the VPN connection.

By the way, the log is not really helpful as it only shows the initial connection setup.

#2 - 06.03.2020 14:04 - Beat Zahnd

Tobias Brunner wrote:

> > Using the Android stronSwan App the connection works well as long as the mobile phone is not put to sleep.
>
> What do you mean with "put to sleep" **exactly**?

Either by just letting it unused until the screen is switched off and the phone is locked or by pressing the power button. But not switched off.

> MOBIKE will only cause an exchange if the client's IP changes. (Although, there is a patch somewhere that triggers a DPD, with NAT-T payloads to detect NAT mapping changes, even if the IP stays the same after addresses have changed - whether that would be the case here and it would help, is another question, though.)

Sure, after IP change it is anyway needed. But it would be also needed if the cellular link is lost for more than the NAT timeout off my cellular provider. During this time no communication is possible and therefore NAT keep-alives are useless. Such interruptions could occur often and there is no way to force cellular providers to increase the NAT timeout...

> > Previously with IPsec L2TP there was nat-keep-alive with about 1 min interval. This is not seen now using the strongswan App.
>
> You can't compare the two. The former runs as Android system service, the latter is a simple app. If Android suspends it, no NAT keepalives can be sent (and obviously no inbound traffic from the server could be processed either). The keepalives are also sent from the C part of the app, which Android might not know much about (if that even would be a consideration).

Not sure what I could do more. The nasty Sony Stamina Mode is disabled here and the strongSwan VPN Client app is excepted from power saving.

> > NAT-T keep-alive time has been set to 15. But this does not help.
>
> So the app is already suspended after 15 seconds? Or is the whole device kind of off?

The mobile phone is in sleep as explained above. Other apps e.g. messengers, or mail are able to receive messages. If it is active (on) then keep alives are received on the server (again using the 45 sec default):

```
13:18:59.693162 IP 178.197.x.x.32061 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x9853ad46,seq=0x1b01), length 88
13:18:59.740103 IP 178.197.x.x.32061 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x9853ad46,seq=0x1b02), length 88
13:19:44.906099 IP 178.197.x.x.32061 > 84.75.x.x.4500: isakmp-nat-keep-alive

13:20:27.137180 IP 178.197.x.x.32061 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x9853ad46,seq=0x1b0a), length 88
13:20:27.347205 IP 178.197.x.x.32061 > 84.75.x.x.4500: UDP-encap: ESP(spi=0x9853ad46,seq=0x1b0b), length 100
13:21:12.982128 IP 178.197.x.x.32061 > 84.75.x.x.4500: isakmp-nat-keep-alive
```

> > The mobile is a Sony with Android 8
>
> Could be a factor if their system image does force suspension of any app/process after a short while, no idea.

Only know about the Stamina Mode which is disabled and having the app excepted from battery saving.

> > Any ideas why the keep-alive and MOBIKE does not work on the cellular network?
>
> The app does not keep any explicit wake locks, in case that would make a difference (Android can ignore them). However, it is registered as foreground service and it has a VpnService instance open. I wonder what would happen if the device had some traffic to send that required the VPN connection.

If I ping the mobile phone from the server to its VPN assigned IP then the connection persists even if pings are not answered after about 2 min. Seem to keep the NAT alive at least

> By the way, the log is not really helpful as it only shows the initial connection setup.

In this case nothing more was logged even after the connection stalled for some time. Seems that MOBIKE was not enabled on the server. Here a log with MOBIKE enabled on the server and the connection stalled for 5 min. The last communication was at about 14:54. But there is no further logging as well.

```
Mar  6 13:49:37 00[DMN] +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Mar  6 13:49:37 00[DMN] Starting IKE service (strongSwan 5.8.2dr1, Android 8.0.0 - 34.4.A.2.118/2018-09-01, F5
321 - Sony/F5321/Sony, Linux 3.10.84-perf-g51b663cc723-04742-g5bcfaca9e61, aarch64)
Mar  6 13:49:37 00[LIB] loaded plugins: androidbridge charon android-log openssl fips-prf random nonce pubkey
chapoly curve25519 pkcs1 pkcs8 pem xcbc hmac socket-default revocation eap-identity eap-mschapv2 eap-md5 eap-g
tc eap-tls x509
Mar  6 13:49:37 00[JOB] spawning 16 worker threads
Mar  6 13:49:37 09[CFG] loaded user certificate 'CN=bz' and private key
Mar  6 13:49:37 09[CFG] loaded CA certificate 'C=CH, ST=ZH, O=hackersGarden'
Mar  6 13:49:37 09[IKE] initiating IKE_SA android[26] to 84.75.x.x
Mar  6 13:49:37 09[ENC] generating IKE_SA_INIT request 0 [ SA KE No N(NATD_S_IP) N(NATD_D_IP) N(FRAG_SUP) N(HA
SH_ALG) N(REDIR_SUP) ]
Mar  6 13:49:37 09[NET] sending packet: from 10.87.147.26[40213] to 84.75.x.x[500] (716 bytes)
Mar  6 13:49:38 10[NET] received packet: from 84.75.x.x[500] to 10.87.147.26[40213] (38 bytes)
Mar  6 13:49:38 10[ENC] parsed IKE_SA_INIT response 0 [ N(INVAL_KE) ]
Mar  6 13:49:38 10[IKE] peer didn't accept DH group ECP_256, it requested MODP_2048
Mar  6 13:49:38 10[IKE] initiating IKE_SA android[26] to 84.75.x.x
Mar  6 13:49:38 10[ENC] generating IKE_SA_INIT request 0 [ SA KE No N(NATD_S_IP) N(NATD_D_IP) N(FRAG_SUP) N(HA
SH_ALG) N(REDIR_SUP) ]
Mar  6 13:49:38 10[NET] sending packet: from 10.87.147.26[40213] to 84.75.x.x[500] (908 bytes)
Mar  6 13:49:38 11[NET] received packet: from 84.75.x.x[500] to 10.87.147.26[40213] (457 bytes)
Mar  6 13:49:38 11[ENC] parsed IKE_SA_INIT response 0 [ SA KE No N(FRAG_SUP) N(NATD_S_IP) N(NATD_D_IP) CERTREQ
 ]
Mar  6 13:49:38 11[CFG] selected proposal: IKE:AES_GCM_16_256/PRF_HMAC_SHA2_512/MODP_2048
Mar  6 13:49:38 11[IKE] local host is behind NAT, sending keep alives
Mar  6 13:49:38 11[IKE] received cert request for "C=CH, ST=ZH, O=hackersGarden"
Mar  6 13:49:38 11[IKE] sending cert request for "C=CH, ST=ZH, O=hackersGarden"
Mar  6 13:49:38 11[IKE] authentication of 'CN=bz' (myself) with RSA signature successful
Mar  6 13:49:38 11[IKE] sending end entity cert "CN=bz"
Mar  6 13:49:38 11[IKE] establishing CHILD_SA android{38}
Mar  6 13:49:38 11[ENC] generating IKE_AUTH request 1 [ IDi CERT N(INIT_CONTACT) CERTREQ AUTH CPRQ(ADDR ADDR6
DNS DNS6) N(ESP_TFC_PAD_N) SA TSi TSr N(MOBIKE_SUP) N(NO_ADD_ADDR) N(EAP_ONLY) N(MSG_ID_SYN_SUP) ]
Mar  6 13:49:38 11[ENC] splitting IKE message (2022 bytes) into 2 fragments
Mar  6 13:49:38 11[ENC] generating IKE_AUTH request 1 [ EF(1/2) ]
Mar  6 13:49:38 11[ENC] generating IKE_AUTH request 1 [ EF(2/2) ]
Mar  6 13:49:38 11[NET] sending packet: from 10.87.147.26[41218] to 84.75.x.x[4500] (1368 bytes)
Mar  6 13:49:38 11[NET] sending packet: from 10.87.147.26[41218] to 84.75.x.x[4500] (719 bytes)
Mar  6 13:49:38 12[NET] received packet: from 84.75.x.x[4500] to 10.87.147.26[41218] (535 bytes)
Mar  6 13:49:38 12[ENC] parsed IKE_AUTH response 1 [ EF(1/4) ]
Mar  6 13:49:38 12[ENC] received fragment #1 of 4, waiting for complete IKE message
Mar  6 13:49:38 13[NET] received packet: from 84.75.x.x[4500] to 10.87.147.26[41218] (535 bytes)
Mar  6 13:49:38 13[ENC] parsed IKE_AUTH response 1 [ EF(2/4) ]
Mar  6 13:49:38 13[ENC] received fragment #2 of 4, waiting for complete IKE message
Mar  6 13:49:38 14[NET] received packet: from 84.75.x.x[4500] to 10.87.147.26[41218] (535 bytes)
Mar  6 13:49:38 14[ENC] parsed IKE_AUTH response 1 [ EF(3/4) ]
Mar  6 13:49:38 14[ENC] received fragment #3 of 4, waiting for complete IKE message
Mar  6 13:49:38 15[NET] received packet: from 84.75.x.x[4500] to 10.87.147.26[41218] (374 bytes)
Mar  6 13:49:38 15[ENC] parsed IKE_AUTH response 1 [ EF(4/4) ]
Mar  6 13:49:38 15[ENC] received fragment #4 of 4, reassembled fragmented IKE message (1792 bytes)
Mar  6 13:49:38 15[ENC] parsed IKE_AUTH response 1 [ N(MOBIKE_SUP) IDr CERT AUTH CPRP(ADDR DNS) SA TSi TSr ]
Mar  6 13:49:38 15[IKE] received end entity cert "CN=example.dyn.ch"
Mar  6 13:49:38 15[CFG]   using trusted ca certificate "C=CH, ST=ZH, O=hackersGarden"
Mar  6 13:49:38 15[CFG] checking certificate status of "CN=bz"
Mar  6 13:49:38 15[CFG] certificate status is not available
Mar  6 13:49:38 15[CFG]   reached self-signed root ca with a path length of 0
Mar  6 13:49:38 15[CFG]   using trusted certificate "CN=bz"
Mar  6 13:49:38 15[IKE] signature validation failed, looking for another key
Mar  6 13:49:38 15[CFG]   using certificate "CN=example.dyn.ch"
Mar  6 13:49:38 15[CFG]   using trusted ca certificate "C=CH, ST=ZH, O=hackersGarden"
Mar  6 13:49:38 15[CFG] checking certificate status of "CN=example.dyn.ch"
Mar  6 13:49:38 15[CFG] certificate status is not available
Mar  6 13:49:38 15[CFG]   reached self-signed root ca with a path length of 0
Mar  6 13:49:38 15[IKE] authentication of 'example.dyn.ch' with RSA signature successful
Mar  6 13:49:38 15[IKE] IKE_SA android[26] established between 10.87.147.26[CN=bz]...84.75.x.x[example.dyn.ch]
Mar  6 13:49:38 15[IKE] scheduling rekeying in 35441s
Mar  6 13:49:38 15[IKE] maximum IKE_SA lifetime 36041s
Mar  6 13:49:38 15[IKE] installing DNS server 10.76.1.1
Mar  6 13:49:38 15[IKE] installing new virtual IP 192.168.1.100
Mar  6 13:49:38 15[IKE] CHILD_SA android{38} established with SPIs 4456f303_i ce8f9ab6_o and TS 192.168.1.100/
32 === 0.0.0.0/0
Mar  6 13:49:38 15[DMN] setting up TUN device for CHILD_SA android{38}
Mar  6 13:49:38 15[DMN] successfully created TUN device
Mar  6 13:49:38 15[IKE] peer supports MOBIKE
```

#3 - 06.03.2020 14:41 - Tobias Brunner

> The mobile phone is in sleep as explained above. Other apps e.g. messengers, or mail are able to receive messages.

Push notifications are based on a totally different mechanism (probably Firebase Cloud Messaging, which uses a single persistent connection for all messaging apps).

> If I ping the mobile phone from the server to its VPN assigned IP then the connection persists even if pings are not answered after about 2 min. Seem to keep the NAT alive at least

Probably depends on the NAT implementation whether packets from the "outside" are considered when deciding whether to keep a mapping alive.

> Here a log with MOBIKE enabled on the server and the connection stalled for 5 min. The last communication was at about 14:54. But there is no further logging as well.

Was that with constant pinging? The app will only send keepalives (and log about them) if no other outbound traffic was sent. However, if the system suspends the app even if it recently was actively processing network traffic, I doubt we can do much about it.

Maybe wake locks could force the device from suspending the app, but besides being quite invasive they can apparently be ignored too by Android with its Doze/App Standby system. Not sure about Doze but App Standby should not be used for our app while a VPN connection is active as it has a foreground service active.

#4 - 06.03.2020 15:10 - Beat Zahnd

Then it seems the app is suspended and no keep alives are sent while the device is on sleep. This leads to NAT timeout and then the connection is lost forever. Just tried it with a different Android mobile phone with the same effect. Only a MOBIKE update would be able to bring it back again.

This renders the use of this kind of app based VPN almost unusable IMHO. In normal life a mobile phone is on sleep for most of the time. Or how is this intended to be handled in the strongswan app?

Or the VPN on the server shall accept UDP-encap ESP from all source ports even if they change almost randomly. But no idea how this could be feasible.

Ping: its just sending ICPM ping to the IP assigned to the child, 192.168.1.100 in the above case. Since this is sent inside the tunnel there is continous UDP traffic preventing NAT timeout.

#5 - 06.03.2020 15:19 - Tobias Brunner

> Or how is this intended to be handled in the strongswan app?

No idea. We can't really change how Android suspends apps.

> Or the VPN on the server shall accept UDP-encap ESP from all source ports even if they change almost randomly. But no idea how this could be feasible.

They should do that anyway (and notify the IKE daemon about the change, see RFC 7296, section 2.23). More problematic is sending traffic if the client didn't send anything for a while.

#6 - 07.03.2020 10:01 - Beat Zahnd

> No idea. We can't really change how Android suspends apps.

Seems to be a tricky issue. I have no android experience. Just found the following:
https://stackoverflow.com/questions/32774437/prevent-android-app-in-background-getting-suspended

In CharonVpnService.java:185 the return value is START_NOT_STICKY instead of the recommended START_STICKY. Not sure if this applies here.

> They should do that anyway (and notify the IKE daemon about the change, see RFC 7296, section 2.23). More problematic is sending traffic if the client didn't send anything for a while.

Will check on the libreswan mailing list if this a configuration issue.

#7 - 09.03.2020 10:45 - Tobias Brunner

> In CharonVpnService.java:185 the return value is START_NOT_STICKY instead of the recommended START_STICKY. Not sure if this applies here.

I don't think the process is actually killed here. That return value would only have an effect then. And our VpnService instance is already running as foreground service.

#8 - 13.03.2020 14:23 - Beat Zahnd

Still trying to find a solution...

Some statements from the libreswan mailing list:

- So that looks like the strongswan bug doing SHA1 for RFC7427 connections that RFC 8472 says should never use SHA1 and which libreswan didn’t advertise

- I know iphone for example just ALWAYS sends a MOBIKE ADDRESS_UPDATE, even if its addrress did not change, so that the port change can be picked up by the server side. Strongswan should really do the same.

#9 - 13.03.2020 14:32 - Tobias Brunner

> - So that looks like the strongswan bug doing SHA1 for RFC7427 connections that RFC 8472 says should never use SHA1 and which libreswan didn’t advertise

A wrong statement (we never propose SHA-1 for that) and completely irrelevant.

> - I know iphone for example just ALWAYS sends a MOBIKE ADDRESS_UPDATE, even if its addrress did not change, so that the port change can be picked up by the server side. Strongswan should really do the same.

That is a possibility (there is actually a patch somewhere that does that). However, it would only have any effect if on your device there is actually an event that would trigger this process in the first place and only once the screen is on again. While the screen is off, the device would probably still not be reachable.

By the way, I wasn't able to reproduce this issue with a Pixel 4 (Android 10) behind a NAT router. Even after hours of the screen being off the device was reachable from the server.

#10 - 19.03.2020 16:50 - Tobias Brunner

> By the way, I wasn't able to reproduce this issue with a Pixel 4 (Android 10) behind a NAT router. Even after hours of the screen being off the device was reachable from the server.

Unfortunately, I was wrong on that. While the device generally stayed reachable the whole time, there were intermittent periods in which it actually wasn't, because Android prevented the app from sending NAT keep-alives (in our test environment the NAT router was not very busy, so the same NAT mapping was assigned once the app was allowed to send a packet again, so it was still reachable afterwards). Basically, without further measures the app/device won't be reachable continuously from the outside.

One option to prevent this from happening, and this seems to work perfectly, is holding a partial wake lock (preventing the CPU from going to sleep) and whitelisting the app in the system settings (otherwise the wake lock is ignored, whitelisting has otherwise no effect). But that will probably have a significant impact on the power usage (I've not measured it yet, though), so we could only enable such a feature optionally and after having made the user aware of this.

To improve the situation without wake lock we could send a DPD after a connectivity check even when the IP stays the same and e.g. use the screen getting turned on as additional trigger for such a check. But that would obviously not help improve the reachability while the screen is turned off longer.

A partial fix for the latter would be to send a DPD instead of a NAT keep-alive if the time difference since the last keep-alive exceeds a certain margin beyond the scheduled interval. That would at least restore connectivity if the NAT router creates new mapping for the sent packet (a keep alive would create a new mapping but without triggering a MOBIKE update and updating the IPsec SAs). The problem with that is that we currently use monotonic time for all events/timestamps, which is suspended while the CPU is not running (i.e. to the app it looks like it sent keep-alives in the expected interval). We could avoid that and use the system's real time clock instead, but that's affected by time adjustments (e.g. daylight savings time or time zone changes), so that's not ideal. Luckily, on Linux (and Android) we can use CLOCK_BOOTTIME instead of CLOCK_MONOTONIC (or CLOCK_REALTIME), which does include the times the system is suspended. So we can measure the actual interval since the last sent message and trigger a DPD if the NAT interval has been exceeded by some margin. This might also not require us to use the screen turning on as trigger for a connectivity check as we should be able to send regular keep-alives then, and the use of DPDs when the IP stays the same after a connectivity check might also be redundant (I guess we have to see how the behavior is in practice).

Using DPDs does have its disadvantages (which is why they are generally disabled in the app) as they produce more overhead (they are encrypted, larger, might have to be retransmitted, require processing a response, could trigger a MOBIKE update and may even cause the recreation of the complete connection if the server is not reachable), but it's probably still lower than keeping a wake lock.

I released a beta version of the app (see AndroidVPNClient for the link to opt-in) that includes the same IP/DPD change, the clock switch and the keep-alive/DPD change mentioned above.

Adding a global option to whitelist the app and keep a wake lock while a connection is established would be something I could add too if there is interest in such a feature (might be interesting to do some measurements first, though). With that enabled, the app would always be reachable from the outside as long as the device itself has network connectivity and the workarounds above would generally not come into use.

#11 - 19.03.2020 22:18 - Beat Zahnd

Thanks for the detailed investigation. Regarding Android stuff I have no coding experience at all to contribute something.

But I played a lot with the server settings (after moving form libreswan to strongswan...). I achieved full day connectivity by sending DPDs from the server:

connections.<conn>.dpd_delay = 60s

charon.retransmit_limit = 20
charon.retransmit_tries = 500

Seems to work with no changes on the app. Not sure what happens, if the mobile is loosing cellular connection for some time.

I first tried to have the NAT-T keepalives sent by the server. But this seems to be not possible. Had the same problem with libreswan. But DPDs do the job...

#12 - 20.03.2020 08:58 - Tobias Brunner

> Regarding Android stuff I have no coding experience at all to contribute something.

No problem. But if you could try the beta version that would help.

> Seems to work with no changes on the app.

I see. Could be interesting to see if there are any retransmits or if the device is really woken with every inbound DPD packet, or if it doesn't go to sleep at all (I guess either would have some impact on the power usage). Whether it works also depends on the NAT router (i.e. the timeout for UDP mappings has to be 60 seconds or more).

> Not sure what happens, if the mobile is loosing cellular connection for some time.

Unless it loses the connectivity for a long time, probably not much with the retransmission settings you used. It will take over two and a half hours until the server kills the SA due to failed retransmits. However, if the retransmitted message is a CREATE_CHILD_SA to rekey the IKE/CHILD_SA this might not work that well as the SA would expire long before the number of retransmits hits the limit (obviously depends on the configured rekey/lifetime settings). Either way, the client should send a DPD when it regains connectivity (with the beta version also if the IP stayed the same) so it would notice if the IKE_SA on the server was gone and recreate the SAs (it wouldn't notice it if only the CHILD_SA was gone, though).

> I first tried to have the NAT-T keepalives sent by the server. But this seems to be not possible.

No, these are currently only sent from hosts behind a NAT (as per RFC 7296). Depending on the NAT routers involved, lone packets from the outside would not keep a NAT mapping alive anyway.

#13 - 20.03.2020 11:59 - Beat Zahnd

There seems to be a reply coming back:

Server log

```
Mar 20 11:38:48 core charon: 01[IKE] sending DPD request
Mar 20 11:38:48 core charon: 01[ENC] generating INFORMATIONAL request 88 [ ]
Mar 20 11:38:48 core charon: 01[NET] sending packet: from 84.75.x.x[4500] to 178.197.235.253[58553] (80 bytes)
Mar 20 11:38:48 core charon: 11[NET] received packet: from 178.197.235.253[58553] to 84.75.x.x[4500] (80 bytes
)
Mar 20 11:38:48 core charon: 11[ENC] parsed INFORMATIONAL response 88 [ ]
Mar 20 11:39:48 core charon: 15[IKE] sending DPD request
Mar 20 11:39:48 core charon: 15[ENC] generating INFORMATIONAL request 89 [ ]
Mar 20 11:39:48 core charon: 15[NET] sending packet: from 84.75.x.x[4500] to 178.197.235.253[58553] (80 bytes)
Mar 20 11:39:49 core charon: 05[NET] received packet: from 178.197.235.253[58553] to 84.75.x.x[4500] (80 bytes
)
Mar 20 11:39:49 core charon: 05[ENC] parsed INFORMATIONAL response 89 [ ]
Mar 20 11:40:48 core charon: 13[IKE] sending DPD request
Mar 20 11:40:48 core charon: 13[ENC] generating INFORMATIONAL request 90 [ ]
Mar 20 11:40:48 core charon: 13[NET] sending packet: from 84.75.x.x[4500] to 178.197.235.253[58553] (80 bytes)
Mar 20 11:40:49 core charon: 10[NET] received packet: from 178.197.235.253[58553] to 84.75.x.x[4500] (80 bytes
)
Mar 20 11:40:49 core charon: 10[ENC] parsed INFORMATIONAL response 90 [ ]
```

tcpdump:

```
11:38:48.340499 IP 84.75.x.x.4500 > 178.197.235.253.58553: NONESP-encap: isakmp: child_sa  inf2[I]
11:38:48.899857 IP 178.197.235.253.58553 > 84.75.x.x.4500: NONESP-encap: isakmp: child_sa  inf2[R]
11:39:48.340899 IP 84.75.x.x.4500 > 178.197.235.253.58553: NONESP-encap: isakmp: child_sa  inf2[I]
11:39:49.054939 IP 178.197.235.253.58553 > 84.75.x.x.4500: NONESP-encap: isakmp: child_sa  inf2[R]
11:40:48.341193 IP 84.75.x.x.4500 > 178.197.235.253.58553: NONESP-encap: isakmp: child_sa  inf2[I]
11:40:49.094963 IP 178.197.235.253.58553 > 84.75.x.x.4500: NONESP-encap: isakmp: child_sa  inf2[R]
```

App log:

```
Mar 20 11:38:49 07[NET] received packet: from 84.75.x.x[4500] to 10.80.171.170[38228] (80 bytes)
Mar 20 11:38:49 07[ENC] parsed INFORMATIONAL request 88 [ ]
Mar 20 11:38:49 07[ENC] generating INFORMATIONAL response 88 [ ]
Mar 20 11:38:49 07[NET] sending packet: from 10.80.171.170[38228] to 84.75.x.x[4500] (80 bytes)
Mar 20 11:39:50 09[NET] received packet: from 84.75.x.x[4500] to 10.80.171.170[38228] (80 bytes)
Mar 20 11:39:50 09[ENC] parsed INFORMATIONAL request 89 [ ]
Mar 20 11:39:50 09[ENC] generating INFORMATIONAL response 89 [ ]
Mar 20 11:39:50 09[NET] sending packet: from 10.80.171.170[38228] to 84.75.x.x[4500] (80 bytes)
Mar 20 11:40:49 13[NET] received packet: from 84.75.x.x[4500] to 10.80.171.170[38228] (80 bytes)
Mar 20 11:40:49 13[ENC] parsed INFORMATIONAL request 90 [ ]
Mar 20 11:40:49 13[ENC] generating INFORMATIONAL response 90 [ ]
Mar 20 11:40:49 13[NET] sending packet: from 10.80.171.170[38228] to 84.75.x.x[4500] (80 bytes)
```

It seems that Swisscom has 120s NAT timeout. This is why I have chosen 60s dpd_delay. No impact on battery usage is observed.

Will try the beta app later.

#14 - 21.03.2020 14:25 - Beat Zahnd

With the beta app and the server DPD disabled it keeps connectivity but there are interruptions. I saw several minutes of inactivity. In this example I sent a message at 14:07:47 to the Telegram messenger on the mobile after 2.5 min inactivity:

When the messenger message was delivered, the server tried to send data to the mobile client to port 53000, and gave up after 75s. 107s later the client sent data from a new port 53003. Therefore NAT timed out in between. Connection was reestablished with a MOBIKE cookie at 14:10:20.

At this time the message was delivered to the messenger app as well.

Conclusion: works, but still with interruptions longer than the cellular provider NAT timeout.

```
tcpdump:
14:03:44.085715 IP 178.197.233.188.53000 > 84.75.x.x.4500: NONESP-encap: isakmp: child_sa  inf2[I]
14:03:44.086604 IP 84.75.x.x.4500 > 178.197.233.188.53000: NONESP-encap: isakmp: child_sa  inf2[R]
14:03:44.414748 IP 178.197.233.188.53000 > 84.75.x.x.4500: NONESP-encap: isakmp: child_sa  inf2[I]
14:03:44.416601 IP 84.75.x.x.4500 > 178.197.233.188.53000: NONESP-encap: isakmp: child_sa  inf2[R]

14:05:08.902711 IP 178.197.233.188.53000 > 84.75.x.x.4500: NONESP-encap: isakmp: child_sa  inf2[I]
14:05:08.903651 IP 84.75.x.x.4500 > 178.197.233.188.53000: NONESP-encap: isakmp: child_sa  inf2[R]

14:07:47.301035 IP 84.75.x.x.4500 > 178.197.233.188.53000: UDP-encap: ESP(spi=0x7a6ed202,seq=0xaa), length 564
14:07:48.065678 IP 84.75.x.x.4500 > 178.197.233.188.53000: UDP-encap: ESP(spi=0x7a6ed202,seq=0xab), length 564
14:07:49.474166 IP 84.75.x.x.4500 > 178.197.233.188.53000: UDP-encap: ESP(spi=0x7a6ed202,seq=0xac), length 564
14:07:52.225630 IP 84.75.x.x.4500 > 178.197.233.188.53000: UDP-encap: ESP(spi=0x7a6ed202,seq=0xad), length 564
14:07:57.793612 IP 84.75.x.x.4500 > 178.197.233.188.53000: UDP-encap: ESP(spi=0x7a6ed202,seq=0xae), length 564
14:08:09.057901 IP 84.75.x.x.4500 > 178.197.233.188.53000: UDP-encap: ESP(spi=0x7a6ed202,seq=0xaf), length 564
14:08:31.073645 IP 84.75.x.x.4500 > 178.197.233.188.53000: UDP-encap: ESP(spi=0x7a6ed202,seq=0xb0), length 564

14:10:18.657704 IP 178.197.233.188.53003 > 84.75.x.x.4500: UDP-encap: ESP(spi=0xcf02f2c9,seq=0x92), length 112
14:10:18.680341 IP 84.75.x.x.4500 > 178.197.233.188.53003: UDP-encap: ESP(spi=0x7a6ed202,seq=0xb1), length 76
14:10:18.745860 IP 178.197.233.188.53003 > 84.75.x.x.4500: UDP-encap: ESP(spi=0xcf02f2c9,seq=0x93), length 96
14:10:18.768525 IP 84.75.x.x.4500 > 178.197.233.188.53003: UDP-encap: ESP(spi=0x7a6ed202,seq=0xb2), length 144
14:10:18.825486 IP 178.197.233.188.53003 > 84.75.x.x.4500: UDP-encap: ESP(spi=0xcf02f2c9,seq=0x94), length 96
14:10:18.848559 IP 84.75.x.x.4500 > 178.197.233.188.53003: UDP-encap: ESP(spi=0x7a6ed202,seq=0xb3), length 96
14:10:18.899545 IP 178.197.233.188.53003 > 84.75.x.x.4500: UDP-encap: ESP(spi=0xcf02f2c9,seq=0x95), length 88

syslog:
Mar 21 14:03:44 core charon: 01[NET] received packet: from 178.197.233.188[53000] to 84.75.x.x[4500] (128 byte
s)
Mar 21 14:03:44 core charon: 01[ENC] parsed INFORMATIONAL request 29 [ N(NATD_S_IP) N(NATD_D_IP) ]
Mar 21 14:03:44 core charon: 01[ENC] generating INFORMATIONAL response 29 [ N(NATD_S_IP) N(NATD_D_IP) ]
Mar 21 14:03:44 core charon: 01[NET] sending packet: from 84.75.x.x[4500] to 178.197.233.188[53000] (128 bytes
)
Mar 21 14:03:44 core charon: 13[NET] received packet: from 178.197.233.188[53000] to 84.75.x.x[4500] (160 byte
s)
Mar 21 14:03:44 core charon: 13[ENC] parsed INFORMATIONAL request 30 [ N(UPD_SA_ADDR) N(NATD_S_IP) N(NATD_D_IP
) N(COOKIE2) ]
```

```
Mar 21 14:03:44 core charon: 13[ENC] generating INFORMATIONAL response 30 [ N(NATD_S_IP) N(NATD_D_IP) N(COOKIE
2) ]
Mar 21 14:03:44 core charon: 13[NET] sending packet: from 84.75.x.x[4500] to 178.197.233.188[53000] (160 bytes
)

Mar 21 14:05:08 core charon: 14[NET] received packet: from 178.197.233.188[53000] to 84.75.x.x[4500] (128 byte
s)
Mar 21 14:05:08 core charon: 14[ENC] parsed INFORMATIONAL request 31 [ N(NATD_S_IP) N(NATD_D_IP) ]
Mar 21 14:05:08 core charon: 14[ENC] generating INFORMATIONAL response 31 [ N(NATD_S_IP) N(NATD_D_IP) ]
Mar 21 14:05:08 core charon: 14[NET] sending packet: from 84.75.x.x[4500] to 178.197.233.188[53000] (128 bytes
)

Mar 21 14:10:18 core charon: 06[KNL] NAT mappings of CHILD_SA ESP/0xcf02f2c9/84.75.x.x changed to 178.197.233.
188[53003], queuing update job

Mar 21 14:10:20 core charon: 08[NET] received packet: from 178.197.233.188[53003] to 84.75.x.x[4500] (128 byte
s)
Mar 21 14:10:20 core charon: 08[ENC] parsed INFORMATIONAL request 32 [ N(NATD_S_IP) N(NATD_D_IP) ]
Mar 21 14:10:20 core charon: 08[ENC] generating INFORMATIONAL response 32 [ N(NATD_S_IP) N(NATD_D_IP) ]
Mar 21 14:10:20 core charon: 08[NET] sending packet: from 84.75.x.x[4500] to 178.197.233.188[53003] (128 bytes
)
Mar 21 14:10:20 core charon: 12[NET] received packet: from 178.197.233.188[53003] to 84.75.x.x[4500] (160 byte
s)
Mar 21 14:10:20 core charon: 12[ENC] parsed INFORMATIONAL request 33 [ N(UPD_SA_ADDR) N(NATD_S_IP) N(NATD_D_IP
) N(COOKIE2) ]
Mar 21 14:10:20 core charon: 12[ENC] generating INFORMATIONAL response 33 [ N(NATD_S_IP) N(NATD_D_IP) N(COOKIE
2) ]
Mar 21 14:10:20 core charon: 12[NET] sending packet: from 84.75.x.x[4500] to 178.197.233.188[53003] (160 bytes
)
```

#15 - 06.04.2020 18:32 - Tobias Brunner

Thanks for testing.

During the last two weeks I did some experiments on a Pixel 4 with Android 10 and only our app installed (besides all the Google apps/services that are preinstalled). No account is configured so the device does pretty much nothing other than occasional DNS queries and contacting some Google services (e.g. for FCM).

TL;DR

I let the device run in different configurations for 22 hours at a time, the server pinged the device every 5 minutes, the NAT keepalive interval was at the default of 45 seconds, the device was connected via WiFi:

Config	Battery Level (after 22 hours)	Packet Loss (5min pings)	Notes
No VPN	93%		
No patches	93%	26%	keepalives were sent 4 or more minutes after the last outbound packet, so the NAT occasionally removed the mapping
CLOCK_BOOTTIME	55%	0%	keepalives were sent ~1min (~61s) apart (occasionally 10-20s more)
CLOCK_BOOTTIME + DPDs	55%	0%	often 3 keepalives were sent ~1min (~61s) apart, then followed a slightly longer delay (10-20s), which triggered a DPD (configured to occur if the NAT keepalive interval was exceeded by more than 20s)
Partial WakeLock	58% (after 4 hours)	0%	keepalives were sent exactly in 45s intervals

As you can see, the clock change makes a major difference (and keeping a WakeLock is definitely not an option). When using CLOCK_MONOTONIC it takes quite a while until the keepalive interval is reached (the clock does not advance when the device is asleep). With CLOCK_BOOTTIME the interval (45s) is apparently exceeded whenever the app is woken up and a keepalive is sent. While this did keep the NAT mapping alive, it had quite a significant impact on the battery usage. Sending DPDs did not change that very much here as the same NAT mapping was always reaquired (its advantage would be seen more in your scenario where the delay is longer and the NAT mapping changes).

The impact on battery usage seemed extreme, though. In particular after I read your statement here:

> This is why I have chosen 60s dpd_delay. No impact on battery usage is observed.

Looking at the data above, I was surprised that sending a DPD every minute from the server would not have a similar impact on the battery usage as sending keepalives/DPDs more frequently from the client. So I tested that scenario too and to my surprise this was the result:

Config	Battery Level (after 22 hours)	Packet Loss (5min pings)	Notes
No patches, DPD from server	93%	0%	DPDs were sent sent and answered every minute, no retransmits

That seemed strange, especially since I also tested sending NAT keepalives in larger intervals (60s and 120s, no DPDs), which made no difference in battery usage or packet loss (55% and 0%, respectively). But it also showed that sending regular DPDs from the server could be a viable workaround if the device needs to be reachable constantly.

So I dug a bit deeper and found the problem: While the Android kernel supports CLOCK_BOOTTIME (e.g. for clock_gettime(), which we use to get timestamps), the POSIX threads (pthreads) implementation in Android's *bionic* C library actually doesn't. Its conditional variables (pthread_cond_t) only support CLOCK_REALTIME and CLOCK_MONOTONIC (see here).

What happened was that none of the condvars were initialized properly (defaulting to CLOCK_REALTIME) so our scheduler never actually waited until the next event. The absolute time it was trying to wait for was based on CLOCK_BOOTTIME, but CLOCK_REALTIME used in pthread_cond_timedwait() was way ahead already so it immediately returned. This resulted in busy-looping, the scheduler was basically constantly checking if the time for the next event was already reached, causing that massive drain on the battery.

So the CLOCK_BOOTTIME patch is a no-go on Android. We could use CLOCK_REALTIME, which is not ideal but at least would allow sending a DPD if the interval between keepalives is too long. I tested that too for a little while and the behavior is quite similar to CLOCK_MONOTONIC, i.e. the check whether a keepalive has to be sent is delayed for minutes, but since the interval is exceed by quite some margin each time, it resulted in lots of DPDs getting sent.

I also experimented a bit with timers (timer_create()), but an app evidently is not permitted to create a timer with CLOCK_BOOTTIME_ALARM (CLOCK_BOOTTIME works, but the device is not woken when the timer expires so that doesn't change very much compared to using CLOCK_REALTIME via condvar).

However, it's possible to use CLOCK_REALTIME_ALARM and CLOCK_BOOTTIME_ALARM via android.app.AlarmManager (setExactAndAllowWhileIdle() is available since Android 6/API level 23). So I tried to leverage that to send keepalives. I used a 60s interval in this test (to compare the result to the server-side DPDs) and that initially worked out very well. The keepalives were sent pretty much in the configured interval - until they weren't! After more than an hour (~70 minutes) the device went into idle mode (triggers the ACTION_DEVICE_IDLE_MODE_CHANGED broadcast) and timers scheduled with setExactAndAllowWhileIdle() are delayed by several minutes (according to the docs up to more than 15).

Then I tried the same with the app whitelisted in the battery optimization system settings. Again, using a NAT keepalive interval of 60s (plus the patches above with CLOCK_REALTIME instead of CLOCK_BOOTTIME):

Config	Battery Level (after 22 hours)	Packet Loss (5min pings)	Notes
AlarmManager + patches	96%	0%	keepalives were sent ~61s after the last outbound packet the whole time

While the docs warn developers from using setExactAndAllowWhileIdle() due to the potential impact on power usage, it apparently works very well for this use case. That there was even less power usage than without VPN seems weird (but that might just be some natural variance). However, besides only being a PoC-like hack there was still a major issue: Rekeyings were not scheduled on time. The thread waiting in the scheduler's condvar slept through all the "wakeups" for the keepalives (which were sent by the thread handling onReceive in the BroadcastReceiver) and when it eventually woke up, it was hours later and the soft and hard lifetimes both expired concurrently. This scheduled two jobs, one for rekeying and one for deletion, which works for CHILD_SA, but is currently problematic for IKE_SAs (especially if the server requested a reauthentication as the client uses break-before-make reauthenticaiton and if the old IKE_SA is deleted locally while waiting for a reply from the server no replacement is created).

So I expanded the patch and instead of directly sending a keepalive, I scheduled a job for it without a delay and waited for its completion. The idea was that it would signal/wake the scheduler thread, queue any delayed jobs to the processor (thread pool), and also signal/wake up the latters threads. This was the result (again with 60s NAT keepalive interval):

Config	Battery Level (after 22 hours)	Packet Loss (5min pings)	Notes
AlarmManager expanded	96%	~0%	keepalives were sent ~61s after the last outbound packet, rekeyings were scheduled on time, there was packet loss during the first 30 minutes of two back-to-back runs (2 and 3 pings, respectively)

I also tried the patch without whitelisting:

Config	Battery Level (after 22 hours)	Packet Loss (5min pings)	Notes
AlarmManager, no whitelisting	94%	68%	after 70 min. keepalives (or more often DPDs) were sent about 9 min. after the last outbound message, rekeyings happened regularly but delayed (so that would have to be accounted for in the configuration)

Anyway, this seems like a pretty good solution as the overhead on battery usage seems negligible. But it requires asking the user to whitelist the app (there is an Intent for that, though) and only works on newer Android releases (but on older ones we don't have this issue in the first place). So there needs to be some kind of fallback to e.g. using just CLOCK_REALTIME (allowing us to send DPDs for longer delays).

The ideal solution would obviously be to use Android 10's android.net.SocketKeepalive. Sadly it currently can only be used with Android's IpSecManager.UdpEncapsulationSocket, which is of no use to us as the kernel would fail at processing UDP-encapsulated ESP packets received on such sockets.

Conclusions:

- The beta version needs an update as CLOCK_BOOTTIME can't be used like that.
- The proposed patches together with using CLOCK_REALTIME should at least help restore the connectivity if the interval between keepalives is exceeded.
- Using DPDs from the server might not be the worst workaround if the device should stay reachable.
- Using AlarmManager to send keepalives and wake the scheduler will probably be the best long-term solution (but requires whitelisting to fully work in low-power idle states). I'll try to implement this properly in the coming days/weeks.

By the way, I also found a problem with setting charon.retransmit_tries to a high value (as you did). With the default settings only 60 retransmits were sent with the correct interval (charon.retransmit_limit), the rest were sent immediately afterwards in one row. I pushed a fix for that to the *retransmission-max* branch. I also fixed a bug that incorrectly stored the timestamps of retransmission checks when no packet was actually sent because a response was received in the meantime (not pushed yet).

#16 - 07.04.2020 13:03 - Beat Zahnd

Awesome work. Thanks for the detailed investigation. Seems that your observations are fully consistent with mine.

Regarding the high value retransmit_tries I made the same observation but did not try to reproduce (mostly working at home due to Covid...).

Let me know if you find time to implement the AlarmManager approach. Currently the server DPD works for me.

Cheers from Winti, Beat

#17 - 11.05.2020 16:11 - Tobias Brunner

I've just released a new beta version of the app. It includes a new scheduler that uses AlarmManager to schedule events that are to be executed more than 3 seconds in the future (which includes NAT-keepalives, rekeyings but not roaming events or initial retransmits). This is to avoid the overhead of using AlarmManager (broadcasts, wake locks etc.) for events that may occur more often (e.g. every sent IKE request requires at least one retransmit job and roaming events are scheduled just 100 ms in the future). The new scheduler is only used on Android 6 and newer but also if the app is not whitelisted (the user can always change that manually anyway), in which case events will be delayed several minutes after the device has been idle for about 70 minutes.

When a connection is started, the user is asked to add the app to the device's power whitelist (unless the app already is whitelisted, or this check is permanently disabled in the settings). Note that accepting this request is not reflected in the system UI (app info and whitelist itself) until the device is restarted (at least with Android 10 on my Pixel 4).

#18 - 27.05.2020 21:37 - Monty Muth

The Beta Version 2.2.2 solved my Problem of stalling VPN Connections. 6 days without disconnect with a Android 10 Phone. Also no abnormal battery usage with the Beta.

Thank you.

#19 - 29.05.2020 09:43 - Tobias Brunner

> The Beta Version 2.2.2 solved my Problem of stalling VPN Connections. 6 days without disconnect with a Android 10 Phone. Also no abnormal battery usage with the Beta.

Thanks for the feedback. I'll probably release these changes next week.

#20 - 02.06.2020 15:07 - Tobias Brunner

- Tracker changed from Issue to Bug

- Status changed from Feedback to Closed

- Assignee set to Tobias Brunner

- Target version set to 5.9.0

- Resolution set to Fixed

These changes are now released with version 2.3.0 of the app.