strongSwan - Bug #865

Android >= 5.0.1, Problems when switching to WIFI

25.02.2015 17:33 - M. C.

Status:	Closed	**Start date:**	25.02.2015
Priority:	Normal	**Due date:**	
Assignee:	Tobias Brunner	**Estimated time:**	0.00 hour
Category:	android		
Target version:			
Affected version:	5.2.2	**Resolution:**	Fixed

Description

On Android 5.0.1 and higher I loose connectivity when switching from mobile network to WIFI. The tunnel remains, just no data goes through. It only happens when I switch from the mobile network to wifi, not when I start with WIFI. And I can only reproduce it on Android 5.0.1 and 5.0.2... I browsed the source code, and found the problem is that when the WIFI Interface appears the function is_current_path_valid(this) in ike_sa.c still returns TRUE even when it should not...

The log looks like this when it works (S4, 4.4.3):

Feb 25 17:15:02 03[KNL] interface wlan0 activated
Feb 25 17:15:02 05[KNL] interface p2p0 activated
Feb 25 17:15:04 08[KNL] 195.176.215.109 appeared on wlan0
Feb 25 17:15:04 09[KNL] fe80::400e:85ff:fe13:2936 appeared on p2p0
Feb 25 17:15:04 09[KNL] 10.80.87.109 disappeared from rmnet_usb0
Feb 25 17:15:04 05[KNL] interface rmnet_usb0 deactivated
Feb 25 17:15:05 09[IKE] old path is not available anymore, try to find another
Feb 25 17:15:05 09[IKE] looking for a route to 195.176.211.30 ...
Feb 25 17:15:05 09[IKE] requesting address change using MOBIKE
Feb 25 17:15:05 09[ENC] generating INFORMATIONAL request 6 [ ]
Feb 25 17:15:05 09[IKE] checking path 195.176.215.109[58664] - 195.176.211.30[4500]
Feb 25 17:15:05 09[NET] sending packet: from 195.176.215.109[58664] to 195.176.211.30[4500] (76 bytes)
Feb 25 17:15:05 11[NET] received packet: from 195.176.211.30[4500] to 195.176.215.109[58664] (76 bytes)
Feb 25 17:15:05 11[ENC] parsed INFORMATIONAL response 6 [ ]
Feb 25 17:15:05 11[ENC] generating INFORMATIONAL request 7 [N(UPD_SA_ADDR) N(NATD_S_IP) N(NATD_D_IP) N(COOKIE2) N(NO_ADD_ADDR)]
Feb 25 17:15:05 11[NET] sending packet: from 195.176.215.109[58664] to 195.176.211.30[4500] (172 bytes)
Feb 25 17:15:05 03[NET] received packet: from 195.176.211.30[4500] to 195.176.215.109[58664] (156 bytes)
Feb 25 17:15:05 03[ENC] parsed INFORMATIONAL response 7 [N(NATD_S_IP) N(NATD_D_IP) N(COOKIE2)]
Feb 25 17:15:05 06[IKE] keeping connection path 195.176.215.109 - 195.176.211.30
Feb 25 17:15:05 06[IKE] sending address list update using MOBIKE
Feb 25 17:15:05 06[ENC] generating INFORMATIONAL request 8 [N(NO_ADD_ADDR)]
Feb 25 17:15:05 06[NET] sending packet: from 195.176.215.109[58664] to 195.176.211.30[4500] (76 bytes)

And when it doesn't work (m8, 5.0.1):

Feb 25 17:15:01 09[KNL] interface p2p0 activated
Feb 25 17:15:02 12[KNL] interface wlan0 activated
Feb 25 17:15:03 17[KNL] 195.176.215.118 appeared on wlan0
Feb 25 17:15:04 13[KNL] fe80::2e8a:72ff:feb3:b682 appeared on wlan0
Feb 25 17:15:07 14[IKE] keeping connection path 10.224.142.149 - 195.176.211.30
Feb 25 17:15:07 14[IKE] sending address list update using MOBIKE
Feb 25 17:15:07 14[ENC] generating INFORMATIONAL request 8 [N(ADD_4_ADDR)]
Feb 25 17:15:07 14[NET] sending packet: from 10.224.142.149[34750] to 195.176.211.30[4500] (76 bytes)
Feb 25 17:15:07 17[NET] received packet: from 195.176.211.30[4500] to 195.176.215.118[34750] (76 bytes)
Feb 25 17:15:07 17[ENC] parsed INFORMATIONAL response 8 [ ]
Feb 25 17:15:36 11[KNL] interface rmnet0 deactivated
Feb 25 17:15:36 12[KNL] fe80::e5b7:1371:f173:707e disappeared from rmnet0
Feb 25 17:15:37 15[KNL] 10.224.142.149 disappeared from rmnet0

Whereas the first log shows "interface rmnet_usb0 deactivated" and then "old path is not available anymore, try to find another". In

the second log the deactivated log message comes much later and before in the log it said that it will keep the connection path..

I assume somehow the interface is not getting unregistered in time on lollipop.

Related issues:

Related to Feature #978: Mobike interface priority enforcement (enforce eth0 ...	**Closed**	**01.06.2015**

Associated revisions

Revision f14feed0 - 28.07.2015 13:55 - Tobias Brunner

Merge branch 'android-updates'

Fixes the roaming behavior on Android 5+, a linker issue on Android M,
a few bugs, and adds several new advanced options for VPN profile (MTU,
server port, split tunneling).

Also adds methods and a constructor to parse settings_t from a string
instead of a file.

Fixes #782, #847, #865.

History

#1 - 16.06.2015 11:45 - G. G.

On Android 5.1.1, with a recompiled version of strongSwan 1.4.6 / 5.3.2 and with the described fix (making is_current_path_valid() to return false), it switchs to WIFI properly.

To properly fix it, it looks like the path should be revalidated when an address disappears, not just when a new one appears.

#2 - 16.06.2015 12:18 - Tobias Brunner

- Related to Feature #978: Mobike interface priority enforcement (enforce eth0 over backup ppp0) added

#3 - 16.06.2015 15:10 - Tobias Brunner

- Status changed from New to Feedback

To properly fix it, it looks like the path should be revalidated when an address disappears, not just when a new one appears.

It looks like the problem is that we currently use ConnectivityManager to trigger roam events (instead of address/route changes we receive via Netlink). This is mainly to avoid the noise produced by the many changes that occur at the low-level interface (this was more of a problem in earlier releases before some changes to the MOBIKE code - so it might be possible to change back to Netlink events).

In Android 5.x the existing routes apparently are removed *after* the CONNECTIVITY_ACTION broadcast has been triggered, so when strongSwan checks the current path it is still valid. That the routes and addresses are removed shortly afterwards is not noticed as this does not trigger a CONNECTIVITY_ACTION broadcast.

Ignoring the current address when checking the existing path might work (see the patch in #978) or what you did (basically the same result). But this only works reliably if the priorities of the routes are modified accordingly by the system. Because if the existing routes are not modified before the CONNECTIVITY_ACTION broadcast is triggered (or if all routes have the same priorities/metrics) it all depends on the order in which the kernel returns routes when we dump them (not sure if this undetermined or if there actually is a logic involved). I think Android 5 actually uses multiple routing tables and rules to switch networks. Unfortunately, we currently don't consider rules in our own route lookups (all routing tables are treated equally). Using the Netlink events might therefore work better (even though that will produce a lot more noise).

#4 - 17.06.2015 21:01 - G. G.

Got it for the sync issue with using ConnectivityManager. The lowest level, the better! Thanks for the explanations.

Prevented the registration to the Connectivity/NetworkManager and set kernel netlink to true in the config. Recompiled.
It works and it is stable for my setup.

#5 - 22.06.2015 18:09 - Tobias Brunner

> Prevented the registration to the Connectivity/NetworkManager and set kernel netlink to true in the config. Recompiled.
> It works and it is stable for my setup.

Yes, it works. But...

I noticed that a problem with using *kernel-netlink* is that when switching from a mobile network to WiFi the mobile network connection is not immediately torn down (can take 20 seconds or more). And while the device is still connected the app will send VPN traffic via mobile network. As long as the current path is valid, i.e. the interface and routes are still up, strongSwan will not switch. The routes all have the same priority, but Android uses multiple routing tables, routing rules (ip rule) and fwmarks to direct traffic to the desired routing table. But how exactly is unclear so we can't base any decisions on the rules, which we could theoretically query from the kernel (we currently don't). This means that to the *kernel-netlink* plugin all available default routes are equally valid, and as long as the current route is there no change will occur anyway (as discussed earlier).

I've done some experiments and came up with a pretty simple solution that avoids *kernel-netlink* altogether. The basic implementation of the kernel_net_t interface I pushed to the *android-updates* branch (includes some other stuff for the next release) uses broadcasts via ConnectivityManager to trigger roam events (as we currently do) and connect()/getsockname() to determine a source address to reach the server over the current network connection. In my tests (on Android 5.1.1 and 4.3) this works pretty well, without long delays when switching to WiFi. Would be great if you could test this in the field.

#6 - 24.06.2015 14:00 - G. G.

Yes, the transition WIFI <-> 4G is seamless! Much faster than with netlink.

Just one thing, that does not really apply to MOBIKE, is that that connection is not re-established as before in one case:

Context:
- IPSec server with an internal IP AND an external IP (external traffic is forwarded/DNATed to internal IP)
- Phone switchs from external network to internal network (its IP also changes from an external one to an internal one)

Scenario:
- Phone is connected to an internal network and get the internal IP of the IPSec Server
- Tunnel is established
- Phone switches to external network and get an external IP
- it keeps the internal IP of the IPSec server which is not reachable from the outside and try to update the IPSec Server
- the tunnel should fail
- Phone keeps the tunnel up, no traffic is possible

-> Previously Phone re-established the tunnel by resolving again the hostname of the IPSec Server in order to get the external IP (that was happened before)

#7 - 24.06.2015 14:40 - Tobias Brunner

> Yes, the transition WIFI <-> 4G is seamless! Much faster than with netlink.

OK great, thanks for testing.

- Phone switches to external network and get an external IP
- it keeps the internal IP of the IPSec server which is not reachable from the outside and try to update the IPSec Server

Yes, at this point the app will try to update the SA because its own IP changed. It starts doing routability checks, that is, it sends packets to all known IPs of the server, which probably is only the internal IP at that point (the server doesn't know about the public IP). After 10 retransmits or so it gives up and defers any further updates as there is no known path to the server, the SA stays established, though, as connectivity might change again soon.

However, I don't think this is directly related to the new kernel interface. The behavior should be similar when using *kernel-netlink* (if not I'd like to see a log of what happens there).

A possible solution would be to let the daemon on the server know it is reachable on a public IP (e.g. via a new *charon.mobike_additional_addresses* strongswan.conf option), so it could send those IPs to the clients (this obviously only works with static IPs). A similar solution would be to *exclude* certain IPs from the list of addresses sent to the clients (e.g. via a new *charon.mobike_ignore_addresses* option). That way the server could exclude its private IP address so the client would not switch to it if it enters the server's own subnet (this will only work if the NAT router translates requests from internal clients to the public IP back to the server's private IP, I think most current routers support this).

What actually might work too is adding the public IP (if it is static) to one of the interfaces of the server. That way it should send that address as additional IP to the client, which then should be able to connect to it.

#8 - 25.06.2015 21:02 - G. G.

I have just retried in order to provide you with logs.
Indeed it re-establishes the tunnel properly (after a few seconds). Switching from external to internal network and vice versa.
Don't know why it was stuck the other day.

Nice work! Thanks! :)

#9 - 25.06.2015 23:55 - G. G.

It got stuck again. I can't send the log. But it is stuck here after being connected to internal IP.

```
old path is not available anymore, try to find another looking for a route to EXTERNAL IP ...
no route found to reach EXTERNAL IP, MOBIKE update deferred
error writing to socket: Network is unreachable
```

#10 - 26.06.2015 10:55 - Tobias Brunner

> Indeed it re-establishes the tunnel properly (after a few seconds). Switching from external to internal network and vice versa.

Hm, interesting. The problem you described makes perfect sense, so I'm surprised that it now actually works. Or did you add the external IP to an interface of the server (or change the code there)?

If not, could you please send me a complete log of connecting from an external network, moving to the internal network and then to an external network again? With about 30s in each network before switching again. And perhaps the same thing but starting from the internal network.

> But it is stuck here after being connected to internal IP.
>
> ```
> old path is not available anymore, try to find another looking for a route to EXTERNAL IP ...
> no route found to reach EXTERNAL IP, MOBIKE update deferred
> error writing to socket: Network is unreachable
> ```

When exactly does this happen? When connecting from the internal network and then moving to an external network? Or the other way around?

Anyway, it looks as if the lookup for a suitable route/source IP fails here because connect() returns ENETUNREACH, which indicates the system really has no route to the server's external IP. The third log message is probably from an attempt to send a NAT keep-alive to the currently known server IP. Depending on the situation this could be the external or internal one, so this failing might be expected, but it could also indicate a general connectivity problem.

#11 - 26.06.2015 23:36 - G. G.

Tobias Brunner wrote:

> Indeed it re-establishes the tunnel properly (after a few seconds). Switching from external to internal network and vice versa.

> Hm, interesting. The problem you described makes perfect sense, so I'm surprised that it now actually works. Or did you add the external IP to an interface of the server (or change the code there)?

No change. I'm not running my modified version of StrongSwan.

> If not, could you please send me a complete log of connecting from an external network, moving to the internal network and then to an external network again? With about 30s in each network before switching again. And perhaps the same thing but starting from the internal network.

("Send log" in the app did not work. I got the log from /data/data/org.strongswan...)
I will send you by email the client & server logs, when the phone switches from 4G to Wifi (Internal).
The phone was still stuck with no data after 5 minutes. (Maybe it is the captive portal detection ?!)

> But it is stuck here after being connected to internal IP.
> [...]

> When exactly does this happen? When connecting from the internal network and then moving to an external network? Or the other way around?

I can't identify any deterministic cause at this point, but it occured when switching from 4G to Wifi (Internal IP) or from Wifi (Internal IP) to 4G.
Other than that, it switches seamlessly from 4G to Wifi (External IP) and vice versa.

> Anyway, it looks as if the lookup for a suitable route/source IP fails here because connect() returns ENETUNREACH, which indicates the system really has no route to the server's external IP. The third log message is probably from an attempt to send a NAT keep-alive to the currently known server IP. Depending on the situation this could be the external or internal one, so this failing might be expected, but it could also indicate a general connectivity problem.

#12 - 29.06.2015 11:51 - Tobias Brunner

> No change. I'm not running my modified version of StrongSwan.

So you use the code from the *android-updates* branch as is?

> If not, could you please send me a complete log of connecting from an external network, moving to the internal network and then to an external network again? With about 30s in each network before switching again. And perhaps the same thing but starting from the internal network.

> ("Send log" in the app did not work. I got the log from /data/data/org.strongswan...)

What happened? Any errors in logcat?

> (Maybe it is the captive portal detection ?!)

What do you mean? Is there a captivity portal in the WiFi you connect to? According to the log there simply is no connectivity when the device switches to the WiFi. The NAT keep-alives are sent to the server's external IP and these apparently can't be sent either (these are simply sent over a UDP socket, if that fails it means the system has no route to reach that IP). Interestingly, the server only sends an ADDITIONAL_IP6_ADDRESS notify, so the client does not know an internal IPv4 address of the server (the app can't use IPv6 as external IPs due to the Linux kernel's lack of UDP en-/decapsulation of ESP for IPv6). Is your WiFi by any chance IPv6 only? Also, could you check the routing table after switching to the WiFi (ip route list table all).

#13 - 29.06.2015 19:09 - G. G.

Tobias Brunner wrote:

> No change. I'm not running my modified version of StrongSwan.

> So you use the code from the *android-updates* branch as is?

Yes

> > If not, could you please send me a complete log of connecting from an external network, moving to the internal network and then to an external network again? With about 30s in each network before switching again. And perhaps the same thing but starting from the internal network.

> ("Send log" in the app did not work. I got the log from /data/data/org.strongswan...)

> What happened? Any errors in logcat?

Just a 0 length log. No error in logcat.

> (Maybe it is the captive portal detection ?!)

> What do you mean? Is there a captivity portal in the WiFi you connect to?

No captive portal. I was thinking about the Android captive portal detection that can be disabled with:

settings put global captive_portal_detection_enabled 0

> According to the log there simply is no connectivity when the device switches to the WiFi. The NAT keep-alives are sent to the server's external IP and these apparently can't be sent either (these are simply sent over a UDP socket, if that fails it means the system has no route to reach that IP). Interestingly, the server only sends an ADDITIONAL_IP6_ADDRESS notify, so the client does not know an internal IPv4 address of the server (the app can't use IPv6 as external IPs due to the Linux kernel's lack of UDP en-/decapsulation of ESP for IPv6). Is your WiFi by any chance IPv6 only? Also, could you check the routing table after switching to the WiFi (ip route list table all).

It's dual stack. Indeed, for having taken a look (in Settings > Wifi Advanced) at how IPs appear, I know IPv4 and IPv6 IPs come with up to 5 seconds delay between each other.
I could send you the routes later. However, once the IPSec tunnel is disconnected, the mobile gets access to IPv4 and IPv6 (internal network + internet). So the routes should be fine at the end, maybe not at the exact moment StrongSwan try to re-establish the tunnel.

> If not, could you please send me a complete log of connecting from an external network, moving to the internal network and then to an external network again? With about 30s in each network before switching again. And perhaps the same thing but starting from the internal network.

("Send log" in the app did not work. I got the log from /data/data/org.strongswan...)

> What happened? Any errors in logcat?

Just a 0 length log. No error in logcat.

Hm, OK. Will have to look into this.

> (Maybe it is the captive portal detection ?!)

> What do you mean? Is there a captivity portal in the WiFi you connect to?

No captive portal. I was thinking about the Android captive portal detection that can be disabled with:

settings put global captive_portal_detection_enabled 0

I see. I think the captive portal detection is based on random DNS lookups, which normally result in NXDOMAIN results, but if a captive portal is present may result in an A/AAAA record of the portal's IP. So depending on how your internal DNS server behaves (which is assigned to the client in configuration payloads) a captive portal might be detected. However, I don't know how Android would change the network config if this happens. But maybe it cuts connectivity for all apps to prevent outbound traffic when it thinks it's not actually possible.

Is there any change in behavior if you disable the detection?

> According to the log there simply is no connectivity when the device switches to the WiFi. The NAT keep-alives are sent to the server's external IP and these apparently can't be sent either (these are simply sent over a UDP socket, if that fails it means the system has no route to reach that IP). Interestingly, the server only sends an ADDITIONAL_IP6_ADDRESS notify, so the client does not know an internal IPv4 address of the server (the app can't use IPv6 as external IPs due to the Linux kernel's lack of UDP en-/decapsulation of ESP for IPv6). Is your WiFi by any chance IPv6 only? Also, could you check the routing table after switching to the WiFi (ip route list table all).

It's dual stack. Indeed, for having taken a look (in Settings > Wifi Advanced) at how IPs appear, I know IPv4 and IPv6 IPs come with up to 5 seconds delay between each other.
I could send you the routes later. However, once the IPSec tunnel is disconnected, the mobile gets access to IPv4 and IPv6 (internal network + internet). So the routes should be fine at the end, maybe not at the exact moment StrongSwan try to re-establish the tunnel.

I guess it's possible that Android triggers the CONNECTIVITY_SERVICE broadcast before it is connected via IPv4 (i.e. as long as connectivity via IPv6 is established it might consider the link up). However, that does not explain why the app can't send NAT keep-alives over a regular IPv4 UDP socket (i.e. the 02[NET] error writing to socket: Network is unreachable messages). If connectivity via IPv4 is established later this should start to work (even if the daemon think's there is no valid route). Or does that actually start to work after a while?

What might be interesting besides the routes is the output of ip rule (before and after disconnecting the VPN after switching from 4G to WiFi).

#15 - 30.06.2015 14:34 - G. G.

Tobias Brunner wrote:

> (Maybe it is the captive portal detection ?!)

What do you mean? Is there a captivity portal in the WiFi you connect to?

No captive portal. I was thinking about the Android captive portal detection that can be disabled with:

settings put global captive_portal_detection_enabled 0

I see. I think the captive portal detection is based on random DNS lookups, which normally result in NXDOMAIN results, but if a captive portal is present may result in an A/AAAA record of the portal's IP. So depending on how your internal DNS server behaves (which is assigned to the client in configuration payloads) a captive portal might be detected. However, I don't know how Android would change the network config if this happens. But maybe it cuts connectivity for all apps to prevent outbound traffic when it thinks it's not actually possible.

No special DNS config, excepted from the fact that there is an "internal view". So when a DNS lookup is performed from an internal IP, it is the internal IP that is returned. For instance, a DNS lookup on "VPN Server" will return the internal IP instead of the external one.

Well, as far as I can see, it keeps Data and Wifi active. After a DNS lookup, it also gets "http://clients3.google.com/generate_204" (It just generates a 204 and a captive portal should return a HTTP redirect) through the Wifi interface. Depending on the result, it will switch from Data to Wifi or ask the user to sign into the portal.

Is there any change in behavior if you disable the detection?

According to the log there simply is no connectivity when the device switches to the WiFi. The NAT keep-alives are sent to the server's external IP and these apparently can't be sent either (these are simply sent over a UDP socket, if that fails it means the system has no route to reach that IP). Interestingly, the server only sends an ADDITIONAL_IP6_ADDRESS notify, so the client does not know an internal IPv4 address of the server (the app can't use IPv6 as external IPs due to the Linux kernel's lack of UDP en-/decapsulation of ESP for IPv6). Is your WiFi by any chance IPv6 only? Also, could you check the routing table after switching to the WiFi (ip route list table all).

It's dual stack. Indeed, for having taken a look (in Settings > Wifi Advanced) at how IPs appear, I know IPv4 and IPv6 IPs come with up to 5 seconds delay between each other.
I could send you the routes later. However, once the IPSec tunnel is disconnected, the mobile gets access to IPv4 and IPv6 (internal network + internet). So the routes should be fine at the end, maybe not at the exact moment StrongSwan try to re-establish the tunnel.

I guess it's possible that Android triggers the CONNECTIVITY_SERVICE broadcast before it is connected via IPv4 (i.e. as long as connectivity via IPv6 is established it might consider the link up). However, that does not explain why the app can't send NAT keep-alives over a regular IPv4 UDP socket (i.e. the 02[NET] error writing to socket: Network is unreachable messages). If connectivity via IPv4 is established later this should start to work (even if the daemon think's there is no valid route). Or does that actually start to work after a while?

Each time, I waited for more than 5 minutes and the tunnel was still not working.
Might it come from the fact that the phone will send to a packet to the "VPN Server External IP" and get a reply from "VPN Server Internal IP" ?!

What might be interesting besides the routes is the output of ip rule (before and after disconnecting the VPN after switching from 4G to WiFi).

Will try to provide you that next time I get the issue.

> According to the log there simply is no connectivity when the device switches to the WiFi. The NAT keep-alives are sent to the server's external IP and these apparently can't be sent either (these are simply sent over a UDP socket, if that fails it means the system has no route to reach that IP). Interestingly, the server only sends an ADDITIONAL_IP6_ADDRESS notify, so the client does not know an internal IPv4 address of the server (the app can't use IPv6 as external IPs due to the Linux kernel's lack of UDP en-/decapsulation of ESP for IPv6). Is your WiFi by any chance IPv6 only? Also, could you check the routing table after switching to the WiFi (ip route list table all).

> It's dual stack. Indeed, for having taken a look (in Settings > Wifi Advanced) at how IPs appear, I know IPv4 and IPv6 IPs come with up to 5 seconds delay between each other.
> I could send you the routes later. However, once the IPSec tunnel is disconnected, the mobile gets access to IPv4 and IPv6 (internal network + internet). So the routes should be fine at the end, maybe not at the exact moment StrongSwan try to re-establish the tunnel.

> I guess it's possible that Android triggers the CONNECTIVITY_SERVICE broadcast before it is connected via IPv4 (i.e. as long as connectivity via IPv6 is established it might consider the link up). However, that does not explain why the app can't send NAT keep-alives over a regular IPv4 UDP socket (i.e. the 02[NET] error writing to socket: Network is unreachable messages). If connectivity via IPv4 is established later this should start to work (even if the daemon thinks there is no valid route). Or does that actually start to work after a while?

> Each time, I waited for more than 5 minutes and the tunnel was still not working.

But do you see those error writing to socket: Network is unreachable errors all this time? Or are the keep-alives sent successfully after a while?

> Might it come from the fact that the phone will send to a packet to the "VPN Server External IP" and get a reply from "VPN Server Internal IP" ?!

If sending the keep-alives already fails I don't think it's related to that (as no packet will be leaving the app during that time).

You mentioned the VPN server's internal IP. As can be seen in the log you sent earlier the server didn't send that IP address to the client. The reason for that is that the MOBIKE code currently doesn't consider the IP address used to communicate with the client as additional address. This is, of course, not the case if the server is behind a NAT. To the server it seems the client communicates with the internal IP but it obviously does not when it is outside the NAT. I guess we could send the current IP too if we notice that we are behind a NAT, which would allow the client to try to connect to the internal IP when it moves behind the NAT. However, this does not help if the client then moves outside the NAT again as the server does not know its public IP that it would have to send to the client to allow such a switch (unless that IP is static and installed on a local interface of the server, or configured somehow as I mentioned before in #865-7).

#17 - 30.06.2015 18:49 - G. G.

Tobias Brunner wrote:

> According to the log there simply is no connectivity when the device switches to the WiFi. The NAT keep-alives are sent to the server's external IP and these apparently can't be sent either (these are simply sent over a UDP socket, if that fails it means the system has no route to reach that IP). Interestingly, the server only sends an ADDITIONAL_IP6_ADDRESS notify, so the client does not know an internal IPv4 address of the server (the app can't use IPv6 as external IPs due to the Linux kernel's lack of UDP en-/decapsulation of ESP for IPv6). Is your WiFi by any chance IPv6 only? Also, could you check the routing table after switching to the WiFi (ip route list table all).

It's dual stack. Indeed, for having taken a look (in Settings > Wifi Advanced) at how IPs appear, I know IPv4 and IPv6 IPs come with up to 5 seconds delay between each other.
I could send you the routes later. However, once the IPSec tunnel is disconnected, the mobile gets access to IPv4 and IPv6 (internal network + internet). So the routes should be fine at the end, maybe not at the exact moment StrongSwan try to re-establish the tunnel.

> I guess it's possible that Android triggers the CONNECTIVITY_SERVICE broadcast before it is connected via IPv4 (i.e. as long as connectivity via IPv6 is established it might consider the link up). However, that does not explain why the app can't send NAT keep-alives over a regular IPv4 UDP socket (i.e. the 02[NET] error writing to socket: Network is unreachable messages). If connectivity via IPv4 is established later this should start to work (even if the daemon thinks there is no valid route). Or does that actually start to work after a while?

All logs are taken after more than 5 mins after the phone switched to Wifi Internal.
It does not change after a while. It stays this way (unable to ping the server IP or anything else).
Indeed it looks like StrongSwan behaves like everything is setup properly but that nothing (any event) will trigger a check or change of state.

> Each time, I waited for more than 5 minutes and the tunnel was still not working.

But do you see those error writing to socket: Network is unreachable errors all this time? Or are the keep-alives sent successfully after a while?

> Might it come from the fact that the phone will send to a packet to the "VPN Server External IP" and get a reply from "VPN Server Internal IP" ?!

If sending the keep-alives already fails I don't think it's related to that (as no packet will be leaving the app during that time).

You mentioned the VPN server's internal IP. As can be seen in the log you sent earlier the server didn't send that IP address to the client. The reason for that is that the MOBIKE code currently doesn't consider the IP address used to communicate with the client as additional address. This is, of course, not the case if the server is behind a NAT. To the server it seems the client communicates with the internal IP but it obviously does not when it is outside the NAT. I guess we could send the current IP too if we notice that we are behind a NAT, which would allow the client to try to connect to the internal IP when it moves behind the NAT. However, this does not help if the client then moves outside the NAT again as the server does not know its public IP that it would have to send to the client to allow such a switch (unless that IP is static and installed on a local interface of the server, or configured somehow as I mentioned before in #865-7).

As stated earlier, here it is just expected the tunnel to fail and to redo the setup process (by resolving the VPN server address and all IPSec phases).
The same as it does with netlink.
For me it is out of the scope of MOBIKE as all IPs will change (mobile and VPN server).

#18 - 01.07.2015 16:55 - Tobias Brunner

- File 0001-WIP-Some-tests-with-registerNetworkCallback.patch added

- File 0001-WIP-Recheck-connectivity-if-reported-up-but-no-IP-fo.patch added

> According to the log there simply is no connectivity when the device switches to the WiFi. The NAT keep-alives are sent to the server's external IP and these apparently can't be sent either (these are simply sent over a UDP socket, if that fails it means the system has no route to reach that IP). Interestingly, the server only sends an ADDITIONAL_IP6_ADDRESS notify, so the client does not know an internal IPv4 address of the server (the app can't use IPv6 as external IPs due to the Linux kernel's lack of UDP en-/decapsulation of ESP for IPv6). Is your WiFi by any chance IPv6 only? Also, could you check the routing table after switching to the WiFi (ip route list table all).

It's dual stack. Indeed, for having taken a look (in Settings > Wifi Advanced) at how IPs appear, I know IPv4 and IPv6 IPs come with up to 5 seconds delay between each other.
I could send you the routes later. However, once the IPSec tunnel is disconnected, the mobile gets access to IPv4 and IPv6 (internal network + internet). So the routes should be fine at the end, maybe not at the exact moment StrongSwan try to re-establish the tunnel.

> I guess it's possible that Android triggers the CONNECTIVITY_SERVICE broadcast before it is connected via IPv4 (i.e. as long as connectivity via IPv6 is established it might consider the link up). However, that does not explain why the app can't send NAT keep-alives over a regular IPv4 UDP socket (i.e. the 02[NET] error writing to socket: Network is unreachable messages). If connectivity via IPv4 is established later this should start to work (even if the daemon thinks there is no valid route). Or does that actually start to work after a while?

All logs are taken after more than 5 mins after the phone switched to Wifi Internal.
It does not change after a while. It stays this way (unable to ping the server IP or anything else).
Indeed it looks like StrongSwan behaves like everything is setup properly but that nothing (any event) will trigger a check or change of state.

Looking at the log again, we see the first switch to WiFi at 22:27:12, there is no successful MOBIKE update. Then at 22:27:16 it looks like the device is disconnected from the WiFi again and a successful MOBIKE update via 4G occurs (new client IP). Until 22:33:12 we don't see any messages, but then a NAT keep-alive is sent, which indicates that there was outbound traffic in the mean time (otherwise a keep-alive would be sent every 20 seconds during those 6 minutes) and since we don't see errors those packets were sent successfully, but obviously via 4G. Until 22:45:45 there are no messages (i.e. there is outbound traffic via 4G again).

At that point the connection via 4G seems gone again and the app is once more not able to determine a path to the external IP via WiFi. The last message is at 22:45:47, so if there was outbound traffic afterwards (and NAT keep-alives were therefore suppressed) then this really seems to be caused by a delay to add the IPv4 address and a lack of Android to notify broadcast listeners when that happens.

I did some experiments with registerNetworkCallback but it doesn't look like it would help. While onLost is called once the 4G connection disappears after the switch, which we could use as trigger for another roam event, this happened 30 seconds after the switch in my tests (this is actually announced via onLosing, where maxMsToLive is 30000). The onAvailable call comes twice, before and after the broadcast (which is also triggered twice) but I'm not sure if the slight delay of a few ms for the second call would help (I don't have a dual-stack WiFi to test this on though, perhaps there the delay is different). You could try the attached patch, which adds some logcat output (filter via tag:asdf) when the callback or broadcast is called.

If using registerNetworkCallback doesn't help, a possible workaround would perhaps be to do a recheck when connectivity is reported up but there seems no usable address. For instance, we could store the connectivity state we get in connectivity_cb and then trigger a delayed roam event (e.g. a second later) if get_source_addr fails to find a source IP but connectivity is reported up. The second patch adds that, so that might be something to try too.

> Each time, I waited for more than 5 minutes and the tunnel was still not working.

> But do you see those error writing to socket: Network is unreachable errors all this time? Or are the keep-alives sent successfully after a while?

> Might it come from the fact that the phone will send to a packet to the "VPN Server External IP" and get a reply from "VPN Server Internal IP" ?!

> If sending the keep-alives already fails I don't think it's related to that (as no packet will be leaving the app during that time).
>
> You mentioned the VPN server's internal IP. As can be seen in the log you sent earlier the server didn't send that IP address to the client. The reason for that is that the MOBIKE code currently doesn't consider the IP address used to communicate with the client as additional address. This is, of course, not the case if the server is behind a NAT. To the server it seems the client communicates with the internal IP but it obviously does not when it is outside the NAT. I guess we could send the current IP too if we notice that we are behind a NAT, which would allow the client to try to connect to the internal IP when it moves behind the NAT. However, this does not help if the client then moves outside the NAT again as the server does not know its public IP that it would have to send to the client to allow such a switch (unless that IP is static and installed on a local interface of the server, or configured somehow as I mentioned before in #865-7).

> As stated earlier, here it is just expected the tunnel to fail and to redo the setup process (by resolving the VPN server address and all IPSec phases). The same as it does with netlink.
> For me it is out of the scope of MOBIKE as all IPs will change (mobile and VPN server).

I don't agree, I think that's exactly what MOBIKE is for. Although, we haven't focused much on situations where the server is behind a NAT and the client actually enters the same subnet (which isn't that much of a problem if the server is not natted as the client will know the server's public and private IP depending on which side it is on).

That the tunnel actually fails with Netlink is strange in the first place. If it should be possible to reach your server on the external IP also from the internal network (as you claim) MOBIKE should not fail when switching from 4G to WiFi. You should see some path probing message from the device's new internal IP to the external IP of the server and that should result in a response. The following update of the SAs should not fail either. I really wonder why the SA is reestablished in that case. Could you send me a log that shows this?

#19 - 01.07.2015 22:02 - G. G.

Tobias Brunner wrote:

> According to the log there simply is no connectivity when the device switches to the WiFi. The NAT keep-alives are sent to the server's external IP and these apparently can't be sent either (these are simply sent over a UDP socket, if that fails it means the system has no route to reach that IP). Interestingly, the server only sends an ADDITIONAL_IP6_ADDRESS notify, so the client does not know an internal IPv4 address of the server (the app can't use IPv6 as external IPs due to the Linux kernel's lack of UDP en-/decapsulation of ESP for IPv6). Is your WiFi by any chance IPv6 only? Also, could you check the routing table after switching to the WiFi (ip route list table all).

> It's dual stack. Indeed, for having taken a look (in Settings > Wifi Advanced) at how IPs appear, I know IPv4 and IPv6 IPs come with up to 5 seconds delay between each other.
> I could send you the routes later. However, once the IPSec tunnel is disconnected, the mobile gets access to IPv4 and IPv6 (internal network + internet). So the routes should be fine at the end, maybe not at the exact moment StrongSwan try to re-establish the tunnel.

> I guess it's possible that Android triggers the CONNECTIVITY_SERVICE broadcast before it is connected via IPv4 (i.e. as long as connectivity via IPv6 is established it might consider the link up). However, that does not explain why the app can't send NAT keep-alives over a regular IPv4 UDP socket (i.e. the 02[NET] error writing to socket: Network is unreachable messages). If connectivity via IPv4 is established later this should start to work (even if the daemon thinks there is no valid route). Or does that actually start to work after a while?

> All logs are taken after more than 5 mins after the phone switched to Wifi Internal.
> It does not change after a while. It stays this way (unable to ping the server IP or anything else).
> Indeed it looks like StrongSwan behaves like everything is setup properly but that nothing (any event) will trigger a check or change of state.

> Looking at the log again, we see the first switch to WiFi at 22:27:12, there is no successful MOBIKE update. Then at 22:27:16 it looks like the device is disconnected from the WiFi again and a successful MOBIKE update via 4G occurs (new client IP). Until 22:33:12 we don't see any messages, but then a NAT keep-alive is sent, which indicates that there was outbound traffic in the mean time (otherwise a keep-alive would be sent every 20 seconds during those 6 minutes) and since we don't see errors those packets were sent successfully, but obviously via 4G. Until 22:45:45 there are no messages (i.e. there is outbound traffic via 4G again).
>
> At that point the connection via 4G seems gone again and the app is once more not able to determine a path to the external IP via WiFi. The last message is at 22:45:47, so if there was outbound traffic afterwards (and NAT keep-alives were therefore suppressed) then this really seems to be caused by a delay to add the IPv4 address and a lack of Android to notify broadcast listeners when that happens.
>
> I did some experiments with registerNetworkCallback but it doesn't look like it would help. While onLost is called once the 4G connection disappears after the switch, which we could use as trigger for another roam event, this happened 30 seconds after the switch in my tests (this is actually announced via onLosing, where maxMsToLive is 30000). The onAvailable call comes twice, before and after the broadcast (which is also triggered twice) but I'm not sure if the slight delay of a few ms for the second call would help (I don't have a dual-stack WiFi to test this on though, perhaps there the delay is different). You could try the attached patch, which adds some logcat output (filter via tag:asdf) when the callback or broadcast is called.
>
> If using registerNetworkCallback doesn't help, a possible workaround would perhaps be to do a recheck when connectivity is reported up but there seems no usable address. For instance, we could store the connectivity state we get in connectivity_cb and then trigger a delayed roam event (e.g. a second later) if get_source_addr fails to find a source IP but connectivity is reported up. The second patch adds that, so that might be something to try too.

I will try the patches later, as I can't right now.
However, dealing with an issue by using delays to check states, don't look good to me.

> Each time, I waited for more than 5 minutes and the tunnel was still not working.

> But do you see those error writing to socket: Network is unreachable errors all this time? Or are the keep-alives sent successfully after a while?
>
> > Might it come from the fact that the phone will send to a packet to the "VPN Server External IP" and get a reply from "VPN Server Internal IP" ?!

> If sending the keep-alives already fails I don't think it's related to that (as no packet will be leaving the app during that time).
>
> You mentioned the VPN server's internal IP. As can be seen in the log you sent earlier the server didn't send that IP address to the client. The reason for that is that the MOBIKE code currently doesn't consider the IP address used to communicate with the client as additional address. This is, of course, not the case if the server is behind a NAT. To the server it seems the client communicates with the internal IP but it obviously does not when it is outside the NAT. I guess we could send the current IP too if we notice that we are

> behind a NAT, which would allow the client to try to connect to the internal IP when it moves behind the NAT. However, this does not help if the client then moves outside the NAT again as the server does not know its public IP that it would have to send to the client to allow such a switch (unless that IP is static and installed on a local interface of the server, or configured somehow as I mentioned before in #865-7).

> As stated earlier, here it is just expected the tunnel to fail and to redo the setup process (by resolving the VPN server address and all IPSec phases). The same as it does with netlink.
> For me it is out of the scope of MOBIKE as all IPs will change (mobile and VPN server).

> I don't agree, I think that's exactly what MOBIKE is for. Although, we haven't focused much on situations where the server is behind a NAT and the client actually enters the same subnet (which isn't that much of a problem if the server is not natted as the client will know the server's public and private IP depending on which side it is on).

OK if you find it falls into the scope of MOBIKE.

> That the tunnel actually fails with Netlink is strange in the first place. If it should be possible to reach your server on the external IP also from the internal network (as you claim) MOBIKE should not fail when switching from 4G to WiFi. You should see some path probing message from the device's new internal IP to the external IP of the server and that should result in a response. The following update of the SAs should not fail either. I really wonder why the SA is reestablished in that case. Could you send me a log that shows this?

You are right. From 4G to Wifi (Internal) it should not fail.
I sent you a log with only netlink activated.
You will see that when switching from 4G to Wifi (Internal) it sends packets to the VPN Server External IP and get a response from the VPN Server Internal IP. So that works.
But, when it switches from Wifi (Internal) to 4G, it keeps trying sending packets to VPN Server Internal IP. At this stage, it will always fail because this internal address is unreachable from outside and it never tries to resolve again the address of the VPN Server (that was perfectly OK for me).
And then, it restarts everything. It resolves again the VPN Server address and establishes again IPSec.This creates another new connection where MOBIKE is not involved (if it could be, that would be great! Just by trying to resolv again the VPN Server address or adding the IP to look for in the config :))

> I did some experiments with registerNetworkCallback but it doesn't look like it would help. While onLost is called once the 4G connection disappears after the switch, which we could use as trigger for another roam event, this happened 30 seconds after the switch in my tests (this is actually announced via onLosing, where maxMsToLive is 30000). The onAvailable call comes twice, before and after the broadcast (which is also triggered twice) but I'm not sure if the slight delay of a few ms for the second call would help (I don't have a dual-stack WiFi to test this on though, perhaps there the delay is different). You could try the attached patch, which adds some logcat output (filter via tag:asdf) when the callback or broadcast is called.
>
> If using registerNetworkCallback doesn't help, a possible workaround would perhaps be to do a recheck when connectivity is reported up but there seems no usable address. For instance, we could store the connectivity state we get in connectivity_cb and then trigger a delayed roam event (e.g. a second later) if get_source_addr fails to find a source IP but connectivity is reported up. The second patch adds that, so that might be something to try too.

> I will try the patches later, as I can't right now.
> However, dealing with an issue by using delays to check states, don't look good to me.

I agree that's certainly not ideal. If Android really triggers that broadcast before connectivity is fully established (i.e. before an IPv4 route/address is available) then this might be a bug and get fixed in an upcoming Android release and that additional delayed check will never even be queued.

> That the tunnel actually fails with Netlink is strange in the first place. If it should be possible to reach your server on the external IP also from the internal network (as you claim) MOBIKE should not fail when switching from 4G to WiFi. You should see some path probing message from the device's new internal IP to the external IP of the server and that should result in a response. The following update of the SAs should not fail either. I really wonder why the SA is reestablished in that case. Could you send me a log that shows this?

> You are right. From 4G to Wifi (Internal) it should not fail.
> I sent you a log with only netlink activated.
> You will see that when switching from 4G to Wifi (Internal) it sends packets to the VPN Server External IP and get a response from the VPN Server Internal IP. So that works.

Thanks a lot for the log. Yes, this is what I expected. What we also see is that the IPv4 address actually appears before the IPv6 addresses on the wlan0 interface. And we see that delay, where the app continues to use the 4G address until it disappears 30 seconds later. But I still not fully get what goes wrong when using ConnectivityManager (however the kernel-netlink plugin does not attempt to switch to WiFi, so perhaps the routes appear quite some time after the addresses appeared). Another possibility is that the bypass_socket call is also required on your system (in my tests it wasn't on systems that used at least Android 4.4). You could try to remove the if (android_sdk_version <= ANDROID_JELLY_BEAN_MR2) statement in get_source_addr so that call always happens.

> But, when it switches from Wifi (Internal) to 4G, it keeps trying sending packets to VPN Server Internal IP. At this stage, it will always fail because this internal address is unreachable from outside and it never tries to resolve again the address of the VPN Server (that was perfectly OK for me).

I've described that basic problem in #865-7. As soon as the client updates the SA to the internal IP it has no knowledge of the external IP anymore as the server won't send it as ADDITIONAL_IP4_ADDRESS (it simply doesn't know about that address). As mentioned before, you might be able to workaround this by adding the external IP (you haven't mentioned yet whether it's static or not) to one of the interfaces of the server, so it can send that address to the client as additional address (we could perhaps also write a UPNP plugin that queries the router for its public IP and then add that to the list of known IPs sent to the client).

By the way, I think if you do the same thing (start in the internal network and then switch to 4G) with the "new" kernel interface this should result in exactly the same behavior as seen in the first part of the log (i.e. it should start path probing to the internal IP and then restart the SA after 10 failures).

#21 - 10.07.2015 22:37 - G. G.

Tobias Brunner wrote:

> Another possibility is that the bypass_socket call is also required on your system (in my tests it wasn't on systems that used at least Android 4.4). You could try to remove the if (android_sdk_version <= ANDROID_JELLY_BEAN_MR2) statement in get_source_addr so that call always happens.

Tried and it does not help. Got stuck when switching from 4G to Wifi (Internal). Indeed, it looks weird that it does not find a path in this case...

Also, applied both patches (without ANDROID_JELLY_BEAN_MR2 removals) and currently testing it.
Will let you know.

#22 - 11.07.2015 19:15 - G. G.

With the 2 patches, it works (after 1 second delay performed by the patches) with and without "captive portal detection".

#23 - 13.07.2015 14:41 - Tobias Brunner

> With the 2 patches, it works (after 1 second delay performed by the patches) with and without "captive portal detection".

OK, thanks a lot for testing. Do you have the logcat output of a move from 4G to WiFi with these patches applied? The first patch basically adds a bunch of additional log messages to see if the registerNetworkCallback API might help in this situation. If not, I guess I'll go with that reckeck workaround (what's missing there is some code to determine the initial connectivity state).

#24 - 28.07.2015 14:22 - Tobias Brunner

- Tracker changed from Issue to Bug

- Status changed from Feedback to Closed

- Assignee set to Tobias Brunner

- % Done set to 0

- Resolution set to Fixed

These changes are now released with version 1.5.0 of the app.

Thanks again for your assistance in resolving this.

Files

0001-WIP-Some-tests-with-registerNetworkCallback.patch	7.59 KB	01.07.2015	Tobias Brunner
0001-WIP-Recheck-connectivity-if-reported-up-but-no-IP-fo.patch	2.97 KB	01.07.2015	Tobias Brunner