gtp only

Trying to set up the GTP tunnel with GTP-U from the kernel by using libgtpnl's tools. It seems the documentation for these got lost, but I found this presentation and tried to set up the example from there.

Script that I'm running, based on the example, also enables dynamic_debug for gtp:

https://gitea.osmocom.org/osmith/ipsec-gtpu-poc/src/branch/master/gtp-ns.sh

<table>
<thead>
<tr>
<th>Server</th>
<th>Client (ns2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>veth</td>
<td>172.0.0.1</td>
</tr>
<tr>
<td>gtp</td>
<td>172.99.0.1</td>
</tr>
</tbody>
</table>

Output:

```bash
--- start ---
+ start
+ set -e
+ modprobe gtp
+ echo -n module gtp +p
+ ip link add veth1 type veth peer name veth2
+ ip addr add 172.0.0.1/24 dev veth1
+ ip link set veth1 up
+ ip addr add 172.99.0.1/32 dev lo
+ gtp-tunnel add gtp1 v1 200 100 172.99.0.2 172.0.0.2
+ gtp-link add gtp1
+ ip route add 172.99.0.2/32 dev gtp1
+ ip netns add ns2
+ ip netns exec ns2 ip addr add 172.0.0.2/24 dev veth2
+ ip netns exec ns2 ip link set veth2 up
+ ip netns exec ns2 ip addr add 172.99.0.2/32 dev lo
+ ip netns exec ns2 ip link set lo up
+ ip netns exec ns2 gtp-tunnel add gtp2 v1 100 200 172.99.0.1 172.0.0.1
+ ip netns exec ns2 gtp-link add gtp2
+ ip netns exec ns2 ip route add 172.99.0.1/32 dev gtp2
+ gtp-tunnel list
  version 1 tei 200/100 ms_addr 172.99.0.2 sgsn_addr 172.0.0.2
  + ip netns exec ns2 gtp-tunnel list
version 1 tei 100/200 ms_addr 172.99.0.1 sgsn_addr 172.0.0.1
+ tail -F /tmp/log-gtp-link1 /tmp/log-gtp-link2
  ===>/tmp/log-gtp-link1 <==
WARNING: attaching dummy socket descriptors. Keep this process running for testing purposes.

  ===>/tmp/log-gtp-link2 <==
WARNING: attaching dummy socket descriptors. Keep this process running for testing purposes.
dmesg:
```
gtp: GTP module loaded (pdp ctx size 104 bytes)
gtp: enable gtp on 3, 4
gtp: enable gtp on 4, 5
gtp1: registered new GTP interface
gtp1: GTPv1-U: new PDP ctx id=c8/64 ssgn=172.0.0.2 ms=172.99.0.2 (pdp=00000000f12af c19)
gtp1: GTPv1-U: new PDP ctx id=64/c8 ssgn=172.0.0.1 ms=172.99.0.1 (pdp=0000000075cab 55a)

Ping in veth works:

```
# ping 172.0.0.2
PING 172.0.0.2 (172.0.0.2) 56(84) bytes of data.
64 bytes from 172.0.0.2: icmp_seq=1 ttl=64 time=0.154 ms
64 bytes from 172.0.0.2: icmp_seq=2 ttl=64 time=0.143 ms
# ip netns exec ns2 ping 172.0.0.1
PING 172.0.0.1 (172.0.0.1) 56(84) bytes of data.
64 bytes from 172.0.0.1: icmp_seq=1 ttl=64 time=0.184 ms
64 bytes from 172.0.0.1: icmp_seq=2 ttl=64 time=0.087 ms

Initially ping over gtp does not work in both directions (hangs):

```
# ping 172.99.0.2
PING 172.99.0.2 (172.99.0.2) 56(84) bytes of data.
# ip netns exec ns2 ping 172.99.0.1
PING 172.99.0.1 (172.99.0.1) 56(84) bytes of data.
```

Initially also wget doesn't work in both directions (hangs):

```
# ip netns exec ns2 python3 -m http.server &
# wget http://172.99.0.2:8000
Connecting to 172.99.0.2:8000... Nothing in dmesg while attempting these, and no packets on the gtp device in tcpdump/wireshark.
```

However suddenly ping and wget started working. After some trail and error, I found that attempting to connect with wget a couple of times from both directions reliably makes the gtp-tunnel work.

Running wget in one direction a couple of times makes the gtp-tunnel work in one direction:

```
# timeout 1 ip netns exec ns2 wget http://172.99.0.1
```

Ping packets get sent into one direction (answer is not coming back):

```
[ 966.736504] gtp1: found PDP context 00000000f12afc19
[ 966.741549] gtp1: gtp -> IP src: 172.99.0.1 dst: 172.99.0.2
[ 966.747456] gtp2: encap_recv sk=00000000a5c172e
[ 966.752176] gtp2: received GTPU packet
[ 966.756102] gtp2: forwarding packet from GGSN to uplink
[ 967.370018] gtp1: found PDP context 00000000f12afc19
```

Running wget in both directions a couple of times opens the gtp-tunnel into the both direction, and the pings get answered:
# timeout 1 ip netns exec ns2 wget http://172.99.0.1
# timeout 1 wget http://172.99.0.2

dmesg:

```
[ 1292.626534] gtp1: found PDP context 00000000f12afc19
[ 1292.631696] gtp1: gtp -> IP src: 172.99.0.1 dst: 172.99.0.2
[ 1292.637484] gtp2: encap_recv sk=00000000a5c172e
[ 1292.642343] gtp2: received GTP1U packet
[ 1292.646293] gtp2: forwarding packet from GGSN to uplink
[ 1292.651826] gtp2: found PDP context 0000000075cab55a
[ 1292.656978] gtp2: gtp -> IP src: 172.99.0.2 dst: 172.99.0.1
[ 1292.662739] gtp1: encap_recv sk=000000007355b1b2
[ 1292.667511] gtp1: received GTP1U packet
[ 1292.671529] gtp1: forwarding packet from GGSN to uplink
[ 1293.628333] gtp1: found PDP context 00000000f12afc19
```...

=> So this sort of works, at least with the workaround with timeout wget. Might be enough for the proof of concept.

## ipsec only

I have connected two APUs directly with one patch cable, and I'm trying to run strongswan like this:

```
<table>
<thead>
<tr>
<th></th>
<th>Server</th>
<th>Client</th>
</tr>
</thead>
<tbody>
<tr>
<td>lan</td>
<td>10.0.0.1</td>
<td>10.0.0.2</td>
</tr>
<tr>
<td>ipsc</td>
<td>10.1.0.1</td>
<td>10.1.0.2</td>
</tr>
</tbody>
</table>
```


- swanctl --initiate --child home successfully initiates the connection.
- ping and wget from client to server works
- ping and wget from server to client does not work (hangs)

Not sure why it hangs in server -> client direction, maybe I should analyze this further. I guess for the PoC it would be good enough if it works in one direction.

## ipsec + gtp

Combining both:

```
<table>
<thead>
<tr>
<th></th>
<th>Server</th>
<th>Client</th>
</tr>
</thead>
<tbody>
<tr>
<td>lan</td>
<td>10.0.0.1</td>
<td>10.0.0.2</td>
</tr>
<tr>
<td>ipsec</td>
<td>10.1.0.1</td>
<td>10.1.0.2</td>
</tr>
<tr>
<td>gtp</td>
<td>10.2.0.1</td>
<td>10.2.0.2</td>
</tr>
</tbody>
</table>
```

After establishing ipsec, I'm running client.gtp.sh and server.gtp.sh:


I need to double check what I found here, and get proper logs and packet traces. But from previous tries:
- directly when starting server.gtp.sh, ipsec packets were sent between server and client
- ping + wget did not work in any direction
- I wasn't able to make it work with the wget workaround above

laforge: FYI, this is the current status. while writing this, I found this workaround that makes the gtp tunnel work, I'm looking into it some more.

09/22/2022
My strongswan configs were wrong for this PoC, that's why I could only send ipsec traffic in one direction. Fixed here: https://gitea.osmocom.org/osmith/ipsec-gtpu-poc/commit/161cadfd3b8e7cde3f7e15259d9c74ab47a41375

New status:

- ipsec: traffic goes in both directions between client and server
- ipsec + gtp: behaves the same as just gtp now, I need to do the trick with wget described above, but then it works!
- so I believe I've completed the proof of concept, I have icmp/ping and http/wget packets getting encapsulated in gtp first, and then ipsec, and then decapsulated on the other end and getting answered the same way. it works in both directions.

pcaps and wireshark screenshots attached.

(I've used null encryption in the strongswan config, and in theory wireshark should be able to decrypt the ESPs. But when enabling this in the protocol options, it only shows "Malformed Packet" in wireshark 3.4.10; maybe it works in newer versions.)

laforge: anything else left to do here? :)

EDIT: screenshots and pcap show 2 pings from client -> server, and then client doing a HTTP get towards python3 -m http.server running on server.