OsmoGGSN (former OpenGGSN) - Bug #3288

osmo-ggsn hangs on certain CREATE PDP CONTEXT gtp packets

05/24/2018 04:25 PM - dexter

Status:	Resolved	**Start date:**	05/24/2018
Priority:	Normal	**Due date:**	
Assignee:	dexter	**% Done:**	100%
Category:			
Target version:			
Spec Reference:			

Description

The function ipcp_contains_option() in ggsn.c hangs because some options seem to contain zero length. This causes cur += cur_opt->len to stay at a constant value. Unfortunately cur is also used in the abortion condition of the while loop. This causes osmo-ggsn to hang with 100% CPU load eventually.

The problem can be reproduced locally by a simple osmo-ggsn/osmo-sgsn/osmo-nitb setup and a blackberry 9780.

Related issues:

Related to OsmoGGSN (former OpenGGSN) - Bug #3319: also handle PCOs that cont...	**Resolved**	**06/04/2018**

History

#1 - 05/24/2018 04:28 PM - dexter

- File test.c added

Attached one finds a piece of test code. I first I had it reproducible, unfortunately I found out that I just gave the test function the data with a wrong offset. Now the code passes, but since I can now reproduce the problem very reliable in the lab setup I can just have a look directly. I will do that tomorrow

#2 - 05/24/2018 05:25 PM - pespin

To start with, in the pcap frame you sent me, frame 23, I see we can receive several IPCP frames in one PCO message.

As a result, we should change build_ipcp_pco() to iterate over them to gather the information instead of just picking the first IPCP frame in pco_ipcp = pco_contains_proto(&pdp->pco_req, PCO_P_IPCP))

I'm unable to see the zero length you are saying in the pcap trace so far.
Also, sharing a gdb output in here would be nice for everybody to understand the issue better.

#3 - 05/25/2018 08:24 AM - dexter

- File problematic_packet.pcapng added

- File test.c added

I think I have a better understanding now of what goes wrong here. The abort condition of the loop is still problematic, but I think the root of the problem is the endianess in memory. Also I think the structs should get an *attribute* ((packed));

The main problem is that ipcp->len is an uint16_t, and when we parse the function by just assigning the memory blob to a pointer of type struct ipcp_hdr the endianess becomes byteswapped and 0x000a becomes 0x0a00, which is far longer than intended. This does not matter as long as we find the option we are looking for. When we find it the function returns and everything is fine. If we do not find the option, the function will advance the pointer and land into memory that contains zeros by chance, this eventually causes the endless loop.

I have updated test.c. The endianess is corrected, when the fix is commented out the crash can be reproduced. The problem only occurs with

IPCP_OPT_SECONDARY_DNS which is not in the ipcp. Normally one would find the IPCP_OPT_PRIMARY_DNS and IPCP_OPT_SECONDARY_DNS option in the same ipcp, but the blackberry 9780 does it in two separate ipcp. This also explains why this problem does not occur with other phones.

#4 - 05/28/2018 12:59 PM - dexter

For me the most important question is now how this should be fixed. Simply twist the endieness of ipcp->len is probably not that good. Maybe we should have a parser function that returns a pointer to the (correctly) parsed struct and does the trick inside. The next question is if we have to create a copy of the memory before we change it. In general its not so good when a parser alters the input data. In my option we could afford the memory copy since it is not done very often. Just once when the PDP context is established.

#5 - 05/28/2018 04:02 PM - dexter

- % Done changed from 0 to 80

I now access the struct member ipcp->len through ntohs(). I also adde an attribute ((packed)); to the two structs.

https://gerrit.osmocom.org/#/c/osmo-ggsn/+/9354 ggsn: fix misinterpreted length field in ipcp_contains_option()
https://gerrit.osmocom.org/#/c/osmo-ggsn/+/9355 ggsn: make sure ipcp_option_hdr and and ipcp_hdr are packed

#6 - 05/28/2018 04:02 PM - dexter

- Status changed from New to In Progress

#7 - 05/28/2018 04:30 PM - laforge

please also include a TTCN-3 test that reproduces the original problem. Should be rather easy to send that particular PDP CTX ACT which triggered the problem.

Thanks!

#8 - 05/30/2018 03:33 PM - dexter

- % Done changed from 80 to 100

I have now added a test that provokes the problematic behavior of osmo-sgsn. I have added the test at the bottom of the control section, so it should not mess up all other tests by freezing the ggsn. I did not test it in docker yet.

https://gerrit.osmocom.org/#/c/osmo-ttcn3-hacks/+/9394 GGSN_Tests: test what happens when PCO contains only one DNS entry

#9 - 06/11/2018 06:57 AM - dexter

- Status changed from In Progress to Resolved

#10 - 07/05/2018 08:15 AM - dexter

- Status changed from Resolved to In Progress

#11 - 07/05/2018 09:45 AM - dexter

Updated testcase, now also the situation where Primary and Secondary DNS are in separate IPCP containers is tested. However, this test will fail. At the moment only PCOs with either a primary or a secondary DNS in one IPCP will pass. The testcase also now checks the responses in order to

make sure that we get the DNS we expect.

See also: https://gerrit.osmocom.org/#/c/osmo-ttcn3-hacks/+/9394/

#12 - 07/09/2018 08:57 AM - dexter

- Status changed from In Progress to Resolved

#13 - 07/19/2018 04:00 PM - stsp

- Related to Bug #3319: also handle PCOs that contain primary and secondary DNS in two separate IPCP containers added

Files

test.c	1.26 KB	05/24/2018	dexter
problematic_packet.pcapng	548 Bytes	05/25/2018	dexter
test.c	1.59 KB	05/25/2018	dexter