OsmoBSC - Bug #3182

OSMO-BSC: Intermittent Segmentation fault (core dumped)

04/18/2018 04:41 AM - ron.menez@entropysolution.com

Status:	Resolved	**Start date:**	04/18/2018
Priority:	Urgent	**Due date:**	
Assignee:	laforge	**% Done:**	80%
Category:			
Target version:			
Spec Reference:			

Description

Segmentation Fault (core dumped) is experienced intermittently in running OSMO-BSC.

Hardwares used are the following:

- Ettus B210
- UPBOARD

UPBOARD information:

1. lscpu
   Architecture: x86_64
   CPU op-mode(s): 32-bit, 64-bit
   Byte Order: Little Endian
   CPU: 4
   On-line CPU list: 0-3
   Thread(s) per core: 1
   Core(s) per socket: 4
   Socket(s): 1
   NUMA node(s): 1
   Vendor ID: GenuineIntel
   CPU family: 6
   Model: 92
   Model name: Intel(R) Pentium(R) CPU N4200 @ 1.10GHz
   Stepping: 9
   CPU MHz: 800.000
   CPU max MHz: 1101.0000
   CPU min MHz: 800.0000
   BogoMIPS: 2188.79
   Virtualization: VT-x
   L1d cache: 24K
   L1i cache: 32K
   L2 cache: 1024K
   NUMA node0 CPU: 0-3
   Flags: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 ds_cpl vmx est tm2 ssse3 sdbg cx16 xtpr pdcm sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave rdrand lahf_lm 3dnowprefetch intel_pt tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust smep erms mpx rdseed smap clflushopt sha_ni xsaveopt xsavec xgetbv1 dtherm ida arat pln pts

1. lshw (kindly see attached file)

OS used:

1. lsb_release -a
   No LSB modules are available.
   Distributor ID:
   Ubuntu
   Description:
   Ubuntu 16.04.4 LTS
   Release: 16.04

Codename: xenial

The following osmocom elements and libraries were installed last April 14 through git:

- libosmocore
- libosmo-abis
- libosmo-crypt-a53
- libosmo-dsp
- libosmo-netif
- libosmo-sccp
- libsmpp34
- libgtpnl
- osmo-mgw
- openbsc
- osmo-bsc
- osmo-bts
- osmo-hlr
- osmo-msc
- osmo-trx

Also run a backtrace after segfault occurs. Logs, pcap network trace and core dump file are also provided. Kindly see attached.

(gdb) bt
#0 0x000000000041aa51 in rsl_rx_conn_fail (msg=msg@entry=0x8a4090) at abis_rsl.c:1380
#1 0x000000000042172b in abis_rsl_rx_dchan (msg=0x8a4090) at abis_rsl.c:1646
#2 abis_rsl_rcvmsg (msg=0x8a4090) at abis_rsl.c:2853
#3 0x00007ffff710f7e8 in handle_ts1_read (bfd=0x895760) at input/ipaccess.c:282
#4 ipaccess_fd_cb (bfd=0x895760, what=1) at input/ipaccess.c:397
#5 0x00007ffff7329352 in osmo_fd_disp_fds (_eset=0x7fffffffe3b0, _wset=0x7fffffffe330, _rset=0x7fffffffe2b0)
at select.c:216
#6 osmo_select_main (polling=polling@entry=0) at select.c:256
#7 0x00000000004075ef in main (argc=<optimized out>, argv=<optimized out>) at osmo_bsc_main.c:532

(gdb) bt full
#0 0x000000000041aa51 in rsl_rx_conn_fail (msg=msg@entry=0x8a4090) at abis_rsl.c:1380
dh = 0x8a411e
lchan = 0x7ffff7fb3290
tp = {lv = {{len = 0, val = 0x0} <repeats 26 times>, {len = 1, val = 0x8a4124 "\001"}, {len = 0,
val = 0x0} <repeats 229 times>}}
cause = 1 '\001'
#1 0x000000000042172b in abis_rsl_rx_dchan (msg=0x8a4090) at abis_rsl.c:1646
rslh = 0x8a411e
rc = 0
sign_link = 0x8941e0
#2 abis_rsl_rcvmsg (msg=0x8a4090) at abis_rsl.c:2853
sign_link = 0x8941e0
rslh = 0x8a411e
rc = 0
#3 0x00007ffff710f7e8 in handle_ts1_read (bfd=0x895760) at input/ipaccess.c:282
line = 0x894ba0
ts_nr = <optimized out>
link = <optimized out>
e1i_ts = <optimized out>
hh = 0x8a411b
msg = 0x8a4090
ret = <optimized out>
rc = <optimized out>
#4 ipaccess_fd_cb (bfd=0x895760, what=1) at input/ipaccess.c:397
rc = 0
#5 0x00007ffff7329352 in osmo_fd_disp_fds (*eset=0x7fffffffe3b0, _wset=0x7fffffffe330, _rset=0x7fffffffe2b0)*
at select.c:216
flags = 1
ufd = 0x895760
tmp = 0x895d38
work = 1

#6 *osmo_select_main (polling=polling@entry=0) at select.c:256*
readset = {_fds_bits = {0 <repeats 16 times>}}
writeset = {__fds_bits = {0 <repeats 16 times>}}
exceptset = {__fds_bits = {0 <repeats 16 times>}}
rc = <optimized out>
no_time = {tv_sec = 0, tv_usec = 0}
#7 0x00000000004075ef in main (argc=<optimized out>, argv=<optimized out>) at osmo_bsc_main.c:532
msc = 0x6bdab8
data = <optimized out>
rc = <optimized out>

Associated revisions

Revision cc2fb61a - 06/14/2018 12:27 PM - laforge

absi_rsl: Fix segfault in rsl_rx_conn_fail()

When we receive a RSL CONN FAIL IND, it may be that this happens
before any RLL is established (and hence a lchan->conn exists),
or after the RLLs have been shut down (and hence a lchan->conn doesn't
exist anymore).

So in this function, it is not legal to unconditionally dereference
lchan->conn.

Change-Id: I6380f5d2cd9364560ce3947517c84247cf4af0d4
Closes: OS#3182

History

#1 - 04/19/2018 09:05 AM - pespin

I'd say lchan pointer (0x7ffff7fb3290) is not correct in rsl_rx_conn_fail, and it is obtained in caller abis_rsl_rx_dchan(). It is also the only thing which I think can fail in the line causing the segfault.

Interestingly, though, lchan is accessed once in that code path without any issue when calling gsm_lchan_name, where pointer is derreferenced:

```
    msg->lchan = lchan_lookup(sign_link->trx, rslh->chan_nr,
                  "Abis RSL rx DCHAN: ");
    if (!msg->lchan)
        return -1;
    ts_name = gsm_lchan_name(msg->lchan);
```

```
static inline char *gsm_lchan_name(const struct gsm_lchan *lchan)
{
    return lchan->name;
}
```

It is also used in rsl_rx_conn_fail previous to the crash without any issue:

```
    LOGP(DRSL, LOGL_NOTICE, "%s CONNECTION FAIL in state %s ",
         gsm_lchan_name(msg->lchan),
         gsm_lchans_name(msg->lchan->state));
```

Outputing in the log:

```
<0004> abis_rsl.c:1367 (bts=0,trx=0,ts=0,ss=0) CONNECTION FAIL in state ACTIVE CAUSE=0x01(Radio Link Failure)
```

So it seems what is wrong is not lchan pointer, but **lchan->conn**, which is used first in that code path.

It would be interesting to know the value of lchan->conn when getting the segfault, to see if it's NULL or it contains garbage. If you run again into the crash with gdb, can you print the value of the pointer? In gdb cmd line: "print lchan->conn". You can also print the full lchan info: "print *lchan".

#2 - 04/19/2018 01:27 PM - neels

Just to mention it -- you have tried completely uninstalling all osmo libraries, cleaning all source trees and rebuilding everything from scratch?
If you e.g. install a newer version of a library (with an ABI change) and a dependent program is not rebuilt subsequently, that may cause stack corruption issues.
I hope that's not it and we can uncover a bug here.

#3 - 04/19/2018 01:31 PM - neels

BTW, unrelated: note http://git.osmocom.org/libosmo-crypt-a53/tree/README.md
i.e. you shouldn't need libosmo-crypt-a53

#4 - 04/20/2018 03:01 AM - ron.menez@entropysolution.com

- File gdb_segfault_04202018.log added

pespin wrote:

> I'd say lchan pointer (0x7ffff7fb3290) is not correct in rsl_rx_conn_fail, and it is obtained in caller abis_rsl_rx_dchan(). It is also the only thing which I think can fail in the line causing the segfault.
>
> Interestingly, though, lchan is accessed once in that code path without any issue when calling gsm_lchan_name, where pointer is derreferenced:
> [...]
> [...]
>
> It is also used in rsl_rx_conn_fail previous to the crash without any issue:
> [...]
>
> Outputing in the log:
> [...]
>
> So it seems what is wrong is not lchan pointer, but **lchan->conn**, which is used first in that code path.
>
> It would be interesting to know the value of lchan->conn when getting the segfault, to see if it's NULL or it contains garbage. If you run again into the crash with gdb, can you print the value of the pointer? In gdb cmd line: "print lchan->conn". You can also print the full lchan info: "print *lchan".

Run the following commands requested:

```
(gdb) print lchan->conn
$1 = (struct gsm_subscriber_connection *) 0x0

(gdb) print *lchan
$2 = {ts = 0x7ffff7fb2168, nr = 1 '\001', type = GSM_LCHAN_SDCCH, rsl_cmode = RSL_CMOD_SPD_SIGN,
  tch_mode = GSM48_CMODE_SIGN, csd_mode = LCHAN_CSD_M_NT, state = LCHAN_S_ACTIVE, broken_reason = 0x45a5a5 "",

  bs_power = 0 '\000', ms_power = 14 '\016', encr = {alg_id = 1 '\001', key_len = 0 '\000',
    key = '\000' <repeats 15 times>}, mr_ms_lv = "\000\000\000\000\000\000",
  mr_bts_lv = "\000\000\000\000\000\000", sapis = "\000\000\000\000\000\000\000\000", abis_ip = {bound_ip = 0,
    connect_ip = 0, bound_port = 0, connect_port = 0, conn_id = 0, rtp_payload = 0 '\000',
    rtp_payload2 = 0 '\000', speech_mode = 0 '\000', rtp_socket = 0x0, ass_compl = {rr_cause = 0 '\000',
      valid = false}}, rqd_ta = 0 '\000', name = 0x87ab00 "(bts=0,trx=0,ts=0,ss=1)", T3101 = {node = {
      rb_parent_color = 7067681, rb_right = 0x0, rb_left = 0x0}, list = {next = 0x7ffff7fb3b38,
      prev = 0x7ffff7fb3b38}, timeout = {tv_sec = 1524195320, tv_usec = 830425}, active = 1,
    cb = 0x4205d0 <t3101_expired>, data = 0x7ffff7fb3a90}, T3109 = {node = {rb_parent_color = 0, rb_right = 0x
0,
      rb_left = 0x0}, list = {next = 0x0, prev = 0x0}, timeout = {tv_sec = 0, tv_usec = 0}, active = 0, cb = 0
x0,
    data = 0x0}, T3111 = {node = {rb_parent_color = 0, rb_right = 0x0, rb_left = 0x0}, list = {next = 0x0,
      prev = 0x0}, timeout = {tv_sec = 0, tv_usec = 0}, active = 0, cb = 0x0, data = 0x0}, error_timer = {node
 = {
      rb_parent_color = 0, rb_right = 0x0, rb_left = 0x0}, list = {next = 0x0, prev = 0x0}, timeout = {tv_sec
= 0,
      tv_usec = 0}, active = 0, cb = 0x0, data = 0x0}, act_timer = {node = {rb_parent_color = 7068001,
      rb_right = 0x0, rb_left = 0x0}, list = {next = 0x7ffff7fb3c78, prev = 0x7ffff7fb3c78}, timeout = {
      tv_sec = 1524192324, tv_usec = 829670}, active = 0, cb = 0x41e010 <lchan_act_tmr_cb>,
    data = 0x7ffff7fb3a90}, rel_work = {node = {rb_parent_color = 0, rb_right = 0x0, rb_left = 0x0}, list = {
      next = 0x0, prev = 0x0}, timeout = {tv_sec = 0, tv_usec = 0}, active = 0, cb = 0x0, data = 0x0},
  error_cause = 0 '\000', neigh_meas = {{arfcn = 0, bsic = 0 '\000',
      rxlev = "\000\000\000\000\000\000\000\000\000\000", rxlev_cnt = 0, last_seen_nr = 0 '\000'}, {arfcn = 0,
      bsic = 0 '\000', rxlev = "\000\000\000\000\000\000\000\000\000\000", rxlev_cnt = 0, last_seen_nr = 0 '\000'}
, {
      arfcn = 0, bsic = 0 '\000', rxlev = "\000\000\000\000\000\000\000\000\000\000", rxlev_cnt = 0,
      last_seen_nr = 0 '\000'}, {arfcn = 0, bsic = 0 '\000', rxlev = "\000\000\000\000\000\000\000\000\000\000",
      rxlev_cnt = 0, last_seen_nr = 0 '\000'}, {arfcn = 0, bsic = 0 '\000',
      rxlev = "\000\000\000\000\000\000\000\000\000\000", rxlev_cnt = 0, last_seen_nr = 0 '\000'}, {arfcn = 0,
      bsic = 0 '\000', rxlev = "\000\000\000\000\000\000\000\000\000\000", rxlev_cnt = 0, last_seen_nr = 0 '\000'}
, {
      arfcn = 0, bsic = 0 '\000', rxlev = "\000\000\000\000\000\000\000\000\000\000", rxlev_cnt = 0,
      last_seen_nr = 0 '\000'}, {arfcn = 0, bsic = 0 '\000', rxlev = "\000\000\000\000\000\000\000\000\000\000",
```

```
      rxlev_cnt = 0, last_seen_nr = 0 '\000'}, {arfcn = 0, bsic = 0 '\000',
      rxlev = "\000\000\000\000\000\000\000\000\000\000", rxlev_cnt = 0, last_seen_nr = 0 '\000'}, {arfcn = 0,
      bsic = 0 '\000', rxlev = "\000\000\000\000\000\000\000\000\000\000", rxlev_cnt = 0, last_seen_nr = 0 '\000'}
},
  meas_rep = {{lchan = 0x0, nr = 0 '\000', flags = 0, ul = {full = {rx_lev = 0 '\000', rx_qual = 0 '\000'}, su
b = {
          rx_lev = 0 '\000', rx_qual = 0 '\000'}}, dl = {full = {rx_lev = 0 '\000', rx_qual = 0 '\000'}, sub =
 {
          rx_lev = 0 '\000', rx_qual = 0 '\000'}}, bs_power = 0 '\000', ms_timing_offset = 0, ms_l1 = {
        pwr = 0 '\000', ta = 0 '\000'}, num_cell = 0, cell = {{rxlev = 0 '\000', bsic = 0 '\000',
          neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000
',
          arfcn = 0, flags = 0}, {rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags =
0}, {
          rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {rxlev = 0 '\000',
          bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {rxlev = 0 '\000', bsic = 0 '\000',
          neigh_idx = 0 '\000', arfcn = 0, flags = 0}}}, {lchan = 0x0, nr = 0 '\000', flags = 0, ul = {full =
{
          rx_lev = 0 '\000', rx_qual = 0 '\000'}, sub = {rx_lev = 0 '\000', rx_qual = 0 '\000'}}, dl = {full =
 {
          rx_lev = 0 '\000', rx_qual = 0 '\000'}, sub = {rx_lev = 0 '\000', rx_qual = 0 '\000'}},
      bs_power = 0 '\000', ms_timing_offset = 0, ms_l1 = {pwr = 0 '\000', ta = 0 '\000'}, num_cell = 0, cell =
 {{
          rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {rxlev = 0 '\000',
          bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {rxlev = 0 '\000', bsic = 0 '\000',
          neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000
',
          arfcn = 0, flags = 0}, {rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags =
0}, {
          rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags = 0}}}, {lchan = 0x0,
      nr = 0 '\000', flags = 0, ul = {full = {rx_lev = 0 '\000', rx_qual = 0 '\000'}, sub = {rx_lev = 0 '\000'
,
          rx_qual = 0 '\000'}}, dl = {full = {rx_lev = 0 '\000', rx_qual = 0 '\000'}, sub = {rx_lev = 0 '\000'
,
          rx_qual = 0 '\000'}}, bs_power = 0 '\000', ms_timing_offset = 0, ms_l1 = {pwr = 0 '\000',
        ta = 0 '\000'}, num_cell = 0, cell = {{rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000', arfcn
= 0,
          flags = 0}, {rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {
          rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {rxlev = 0 '\000',
          bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {rxlev = 0 '\000', bsic = 0 '\000',
          neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000
',
          arfcn = 0, flags = 0}}}, {lchan = 0x0, nr = 0 '\000', flags = 0, ul = {full = {rx_lev = 0 '\000',
          rx_qual = 0 '\000'}, sub = {rx_lev = 0 '\000', rx_qual = 0 '\000'}}, dl = {full = {rx_lev = 0 '\000'
,
          rx_qual = 0 '\000'}, sub = {rx_lev = 0 '\000', rx_qual = 0 '\000'}}, bs_power = 0 '\000',
      ms_timing_offset = 0, ms_l1 = {pwr = 0 '\000', ta = 0 '\000'}, num_cell = 0, cell = {{rxlev = 0 '\000',
          bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {rxlev = 0 '\000', bsic = 0 '\000',
          neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000
',
          arfcn = 0, flags = 0}, {rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags =
0}, {
          rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {rxlev = 0 '\000',
---Type <return> to continue, or q <return> to quit---
          bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags = 0}}}, {lchan = 0x0, nr = 0 '\000', flags =
 0,
      ul = {full = {rx_lev = 0 '\000', rx_qual = 0 '\000'}, sub = {rx_lev = 0 '\000', rx_qual = 0 '\000'}}, dl
 = {
        full = {rx_lev = 0 '\000', rx_qual = 0 '\000'}, sub = {rx_lev = 0 '\000', rx_qual = 0 '\000'}},
      bs_power = 0 '\000', ms_timing_offset = 0, ms_l1 = {pwr = 0 '\000', ta = 0 '\000'}, num_cell = 0, cell =
 {{
          rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {rxlev = 0 '\000',
          bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {rxlev = 0 '\000', bsic = 0 '\000',
          neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000
',
          arfcn = 0, flags = 0}, {rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags =
0}, {
          rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags = 0}}}, {lchan = 0x0,
      nr = 0 '\000', flags = 0, ul = {full = {rx_lev = 0 '\000', rx_qual = 0 '\000'}, sub = {rx_lev = 0 '\000'
,
          rx_qual = 0 '\000'}}, dl = {full = {rx_lev = 0 '\000', rx_qual = 0 '\000'}, sub = {rx_lev = 0 '\000'
,
          rx_qual = 0 '\000'}}, bs_power = 0 '\000', ms_timing_offset = 0, ms_l1 = {pwr = 0 '\000',
        ta = 0 '\000'}, num_cell = 0, cell = {{rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000', arfcn
= 0,
```

```
          flags = 0}, {rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {
          rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {rxlev = 0 '\000',
          bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {rxlev = 0 '\000', bsic = 0 '\000',
          neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000
',
          arfcn = 0, flags = 0}}}, {lchan = 0x0, nr = 0 '\000', flags = 0, ul = {full = {rx_lev = 0 '\000',
          rx_qual = 0 '\000'}, sub = {rx_lev = 0 '\000', rx_qual = 0 '\000'}}, dl = {full = {rx_lev = 0 '\000'
,
          rx_qual = 0 '\000'}, sub = {rx_lev = 0 '\000', rx_qual = 0 '\000'}}, bs_power = 0 '\000',
      ms_timing_offset = 0, ms_l1 = {pwr = 0 '\000', ta = 0 '\000'}, num_cell = 0, cell = {{rxlev = 0 '\000',
          bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {rxlev = 0 '\000', bsic = 0 '\000',
          neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000
',
          arfcn = 0, flags = 0}, {rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags =
0}, {
          rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {rxlev = 0 '\000',
          bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags = 0}}}, {lchan = 0x0, nr = 0 '\000', flags =
 0,
      ul = {full = {rx_lev = 0 '\000', rx_qual = 0 '\000'}, sub = {rx_lev = 0 '\000', rx_qual = 0 '\000'}}, dl
 = {
        full = {rx_lev = 0 '\000', rx_qual = 0 '\000'}, sub = {rx_lev = 0 '\000', rx_qual = 0 '\000'}},
      bs_power = 0 '\000', ms_timing_offset = 0, ms_l1 = {pwr = 0 '\000', ta = 0 '\000'}, num_cell = 0, cell =
 {{
          rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {rxlev = 0 '\000',
          bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {rxlev = 0 '\000', bsic = 0 '\000',
          neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000
',
          arfcn = 0, flags = 0}, {rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags =
0}, {
          rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags = 0}}}, {lchan = 0x0,
      nr = 0 '\000', flags = 0, ul = {full = {rx_lev = 0 '\000', rx_qual = 0 '\000'}, sub = {rx_lev = 0 '\000'
,
          rx_qual = 0 '\000'}}, dl = {full = {rx_lev = 0 '\000', rx_qual = 0 '\000'}, sub = {rx_lev = 0 '\000'
,
          rx_qual = 0 '\000'}}, bs_power = 0 '\000', ms_timing_offset = 0, ms_l1 = {pwr = 0 '\000',
        ta = 0 '\000'}, num_cell = 0, cell = {{rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000', arfcn
= 0,
          flags = 0}, {rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {
          rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {rxlev = 0 '\000',
          bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {rxlev = 0 '\000', bsic = 0 '\000',
          neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000
',
          arfcn = 0, flags = 0}}}, {lchan = 0x0, nr = 0 '\000', flags = 0, ul = {full = {rx_lev = 0 '\000',
          rx_qual = 0 '\000'}, sub = {rx_lev = 0 '\000', rx_qual = 0 '\000'}}, dl = {full = {rx_lev = 0 '\000'
,
          rx_qual = 0 '\000'}, sub = {rx_lev = 0 '\000', rx_qual = 0 '\000'}}, bs_power = 0 '\000',
      ms_timing_offset = 0, ms_l1 = {pwr = 0 '\000', ta = 0 '\000'}, num_cell = 0, cell = {{rxlev = 0 '\000',
          bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {rxlev = 0 '\000', bsic = 0 '\000',
          neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000
',
          arfcn = 0, flags = 0}, {rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags =
0}, {
          rxlev = 0 '\000', bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags = 0}, {rxlev = 0 '\000',
          bsic = 0 '\000', neigh_idx = 0 '\000', arfcn = 0, flags = 0}}}}, meas_rep_idx = 0, meas_rep_count =
0,
  meas_rep_last_seen_nr = 255 '\377', rqd_ref = 0x0, conn = 0x0, dyn = {act_type = 0 '\000', ho_ref = 0 '\000'
,
    rqd_ref = 0x0, rqd_ta = 0 '\000'}}
```

Also attached is the complete gdb logs.

#5 - 04/20/2018 03:07 AM - ron.menez@entropysolution.com

neels wrote:

> Just to mention it -- you have tried completely uninstalling all osmo libraries, cleaning all source trees and rebuilding everything from scratch? If you e.g. install a newer version of a library (with an ABI change) and a dependent program is not rebuilt subsequently, that may cause stack corruption issues.
> I hope that's not it and we can uncover a bug here.

Hi Neels,

We installed all the osmo elements from scratch to a newly and updated installation of Ubuntu 16.04 last April 14, 2018 using the latest git version

that time.

We will try to reinstall all of the osmo elements again today using the latest versions from git and removing the "libosmo-crypt-a53" from the installation.

We'll let you know if we will experience the segfault again.

#6 - 04/20/2018 06:32 AM - ron.menez@entropysolution.com

ron.menez@entropysolution.com wrote:

> neels wrote:
>
> > Just to mention it -- you have tried completely uninstalling all osmo libraries, cleaning all source trees and rebuilding everything from scratch?
> > If you e.g. install a newer version of a library (with an ABI change) and a dependent program is not rebuilt subsequently, that may cause stack corruption issues.
> > I hope that's not it and we can uncover a bug here.
>
> Hi Neels,
>
> We installed all the osmo elements from scratch to a newly and updated installation of Ubuntu 16.04 last April 14, 2018 using the latest git version that time.
>
> We will try to reinstall all of the osmo elements again today using the latest versions from git and removing the "libosmo-crypt-a53" from the installation.
>
> We'll let you know if we will experience the segfault again.

Hi Neels,

We installed the latest version today and still we experience segfault.

It seems that every time a "Radio Link Failure" occurs, segfault will be triggered. Kindly see logs below for your reference:

```
<0004> abis_rsl.c:1367 (bts=0,trx=0,ts=0,ss=0) CONNECTION FAIL in state ACTIVE CAUSE=0x01(Radio Link Failure)
Segmentation fault (core dumped)
```

#7 - 04/23/2018 10:04 PM - neels

@Ron, thanks for the clarification.

The next thing to do to fix this issue is to try reproducing the failure with ttcn3 tests:
Trigger the CONNECTION FAIL with cause Radio Link Failure as seen in the logs and ensure graceful handling.

So far we haven't assigned or prioritized this issue.
A segfault is inherently important, but currently not sure when we'll get a chance to investigate in detail.

#8 - 06/14/2018 07:40 AM - laforge

- Assignee set to laforge

- Priority changed from Normal to Urgent

#9 - 06/14/2018 07:47 AM - laforge

Looking at this in more detail.

- rxl_rx_conn_fail() blindly dereferences lchan->conn
- lchan->conn is set in gsm0408_rcvmsg() when the first DATA IND (i.e. Layer 3 data) arrives

However, what if a CONN FAIL is received even before we receive the first DATA IND is received?

#10 - 06/14/2018 07:55 AM - laforge

- % Done changed from 0 to 20

or even more so: What if the RLL connection (equals LAPDm link) has already been closed by means of RLL REL IND / RLL REL REQ? handle_release() will e.g. set lchan->conn to NULL. As will do lchan_release().

I think it's fundamentally wrong to assume that a lchan will always have a "conn" associated.

#11 - 06/14/2018 07:58 AM - laforge

- Checklist item [ ] do a thorough audit of all lchan->conn dereferences added
Checklist item [ ] fix the actual bug reported added
Checklist item [ ] implement TTCN3 tests to trigger the problem added

#12 - 06/14/2018 11:46 AM - laforge

- Checklist item [x] implement TTCN3 tests to trigger the problem set to Done

- % Done changed from 20 to 40

We have two new test cases in https://gerrit.osmocom.org/9630 which reporduce the problem on current OsmoBSC master

#13 - 06/14/2018 12:34 PM - laforge

- Checklist item [x] fix the actual bug reported set to Done

- % Done changed from 40 to 80

fix has been committed to osmo-bsc master in http://git.osmocom.org/osmo-bsc/commit/?id=cc2fb61a1639b5237d2271f2789cfbe951471d78

feel free to either upgrade to latest osmo-bsc master, or to back-port the fix using

```
git cherry-pick cc2fb61a1639b5237d2271f2789cfbe951471d78
```

to your currently used version of osmo-bsc.

The bug was originally introduced [by me] in git commit 3561bd48976dbee8dbd4659dad15be85a3e79ace in January 2018

as we now have automatic tests for this bug, it should never be re-introduced unnoticed.

I'll also continue to audit the code for other similar bugs.

#14 - 06/14/2018 12:34 PM - laforge

- Status changed from New to In Progress

#15 - 06/14/2018 01:56 PM - laforge

- Checklist item [x] do a thorough audit of all lchan->conn dereferences set to Done

- Status changed from In Progress to Resolved

A manual audit of the code concludes that there appear to be no other bugs that dereference lchan->conn or even lchan->conn->fi without proper checks. It might be useful to re-audit the upcoming lchan_fsm changes by neels

#16 - 07/26/2018 06:41 AM - ron.menez@entropysolution.com

Hi Neels / Support,

As of now the segfault error is resolved. But we are encountering another issue in SDCCH channel not being release immediately if "CONNECTION FAIL in state ACTIVE CAUSE=0x01(Radio Link Failure)" error is experience.

May we know how long does the OSMO-BSC holds the SDCCH channel if this error occurs? or Does it really release the SDCCH channel or not?

#17 - 07/26/2018 12:35 PM - neels

ron.menez@entropysolution.com wrote:

> Hi Neels / Support,

Hi there -- osmocom.org is an open community, if you want to address dedicated support, you need to contact a commercial support vendor. Also, rather direct questions and issues to the general community instead of individual members. Thanks!

> As of now the segfault error is resolved. But we are encountering another issue in SDCCH channel not being release immediately if "CONNECTION FAIL in state ACTIVE CAUSE=0x01(Radio Link Failure)" error is experience.

Please try to avoid mixing separate issues in the same ticket. Feel free to create a new ticket for any problem you encounter.

> May we know how long does the OSMO-BSC holds the SDCCH channel if this error occurs? or Does it really release the SDCCH channel or not?

While I'm here, even though it does not belong here, let me mention that osmo-bsc is currently being refactored (almost done), to use well-defined FSMs and catch a few holes of missing cleanup/release. You may try to run osmo-bsc from branch neels/inter_bsc_ho to see whether the failure to release still occurs on that branch, most likely it is fixed there. The proper course of action would be to find a ticket that describes your error or create a new one, then report there whether the branch fixes it or not. Thanks!

Files

segfault_osmo-bsc.pcap	50.7 KB	04/18/2018	ron.menez@entropysolution.com
segfault_osmo-bsc.log	128 KB	04/18/2018	ron.menez@entropysolution.com
osmo-bts_standalone_demo.cfg	2.26 KB	04/18/2018	ron.menez@entropysolution.com
osmo-bsc_standalone_demo.cfg	4.57 KB	04/18/2018	ron.menez@entropysolution.com
osmo-mgw_standalone_demo.cfg	1.16 KB	04/18/2018	ron.menez@entropysolution.com
osmo-stp_standalone_demo.cfg	424 Bytes	04/18/2018	ron.menez@entropysolution.com
osmo-trx_standalone_demo.cfg	903 Bytes	04/18/2018	ron.menez@entropysolution.com
osmo-hlr_standalone_demo.cfg	305 Bytes	04/18/2018	ron.menez@entropysolution.com
osmo-msc_standalone_demo.cfg	1.78 KB	04/18/2018	ron.menez@entropysolution.com
core	3.3 MB	04/18/2018	ron.menez@entropysolution.com
gdb_segfault_04202018.log	88.4 KB	04/20/2018	ron.menez@entropysolution.com