OsmoMSC - Bug #2916

asan failure on debian unstable in msc_vlr_test_gsm_ciph

02/09/2018 04:20 PM - laforge

Status:	New	**Start date:**	02/09/2018
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:			
Target version:			
Resolution:		**Spec Reference:**	

Description

The ./msc_vlr_test_gsm_ciph unit test fails on debian unstable when compiled using asan:

DRLL Dispatching 04.08 message GSM48_MT_CC_SETUP (0x3:0x5)
DRLL subscr IMSI:901700000004620: Message not permitted for initial conn: GSM48_MT_CC_SETUP
DRLL Dispatching 04.08 message unknown 0x33 (0x5:0x33)
DRLL subscr IMSI:901700000004620: Message not permitted for initial conn: unknown 0x33
DRLL Dispatching 04.08 message GSM48_MT_RR_SYSINFO_1 (0x6:0x19)
DRLL subscr IMSI:901700000004620: Message not permitted for initial conn: GSM48_MT_RR_SYSINFO_1
DRLL Dispatching 04.08 message SMS:0x01 (0x9:0x1)
DRLL subscr IMSI:901700000004620: Message not permitted for initial conn: SMS:0x01
lu_result_sent 0
DREF VLR subscr IMSI:901700000004620 usage increases to: 2
vsub->imeisv[0] 0
DREF VLR subscr IMSI:901700000004620 usage decreases to: 1
- MS sends Ciphering Mode Complete with IMEISV, VLR accepts and sends GSUP LU Req to HLR
MSC <--RAN_GERAN_A-- MS: GSM48_MT_RR_CIPH_M_COMPL
DRR IMSI:901700000004620: CIPHERING MODE COMPLETE
DVLR vlr_lu_fsm(901700000004620){VLR_ULA_S_WAIT_CIPH}: Received Event VLR_ULA_E_CIPH_RES
===
28515ERROR: AddressSanitizer: stack-use-after-scope on address 0x7ffe6af2f8e0 at pc 0x7f1d52773203 bp 0x7ffe6af2dc70 sp 0x7ffe6af2d420
READ of size 18 at 0x7ffe6af2f8e0 thread T0
#0 0x7f1d52773202 (/usr/lib/x86_64-linux-gnu/libasan.so.4+0x71202)
#1 0x7f1d527e5d07 (/usr/lib/x86_64-linux-gnu/libasan.so.4+0xe3d07)
#2 0x7f1d5277338b (/usr/lib/x86_64-linux-gnu/libasan.so.4+0x7138b)
#3 0x7f1d5279f015 in vsnprintf (/usr/lib/x86_64-linux-gnu/libasan.so.4+0x9d015)
#4 0x7f1d51c22cff in _output /home/laforge/projects/git/libosmocore/src/logging.c:419
#5 0x7f1d51c23172 in osmo_vlogp /home/laforge/projects/git/libosmocore/src/logging.c:520
#6 0x7f1d51c23297 in logp2 /home/laforge/projects/git/libosmocore/src/logging.c:553
#7 0x563682cb0c73 in lu_fsm_wait_ciph /home/laforge/projects/git/osmo-msc/src/libvlr/vlr_lu_fsm.c:1135
#8 0x7f1d51c1fe7e in _osmo_fsm_inst_dispatch /home/laforge/projects/git/libosmocore/src/fsm.c:481
#9 0x563682c9d196 in vlr_subscr_rx_ciph_res /home/laforge/projects/git/osmo-msc/src/libvlr/vlr.c:1097
#10 0x563682c65ff9 in msc_cipher_mode_compl /home/laforge/projects/git/osmo-msc/src/libmsc/osmo_msc.c:200
#11 0x563682c1fea5 in rx_from_ms /home/laforge/projects/git/osmo-msc/tests/msc_vlr/msc_vlr_tests.c:230
#12 0x563682c20141 in ms_sends_msg /home/laforge/projects/git/osmo-msc/tests/msc_vlr/msc_vlr_tests.c:245
#13 0x563682c02fc2 in test_ciph_imeisv /home/laforge/projects/git/osmo-msc/tests/msc_vlr/msc_vlr_test_gsm_ciph.c:637
#14 0x563682c1dd6c in run_tests /home/laforge/projects/git/osmo-msc/tests/msc_vlr/msc_vlr_tests.c:841
#15 0x563682bf2070 in main /home/laforge/projects/git/osmo-msc/tests/msc_vlr/msc_vlr_tests.c:898
#16 0x7f1d4fd79f29 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x20f29)
#17 0x563682bf2979 in _start (/space/home/laforge/projects/git/osmo-msc/tests/msc_vlr/msc_vlr_test_gsm_ciph+0x10b979)

Address 0x7ffe6af2f8e0 is located in stack of thread T0 at offset 96 in frame
#0 0x563682c65ccf in msc_cipher_mode_compl /home/laforge/projects/git/osmo-msc/src/libmsc/osmo_msc.c:158

```
This frame has 3 object(s):
    [32, 48) 'ciph_res'
    [96, 128) 'imeisv' <== Memory access at offset 96 is inside this variable
    [160, 4256) 'tp'
HINT: this may be a false positive if your program uses some custom stack unwind mechanism or swap
context
      (longjmp and C++ exceptions are supported)
```

```
SUMMARY: AddressSanitizer: stack-use-after-scope (/usr/lib/x86_64-linux-gnu/libasan.so.4+0x71202)
Shadow bytes around the buggy address:
  0x10004d5ddec0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10004d5dded0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10004d5ddee0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10004d5ddef0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10004d5ddf00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x10004d5ddf10: f1 f1 f1 f1 00 00 f2 f2 f2 f2 f2 f2[f8]f8 f8 f8
  0x10004d5ddf20: f2 f2 f2 f2 f8 f8 f8 f8 f8 f8 f8 f8 f8 f8 f8 f8
  0x10004d5ddf30: f8 f8 f8 f8 f8 f8 f8 f8 f8 f8 f8 f8 f8 f8 f8 f8
  0x10004d5ddf40: f8 f8 f8 f8 f8 f8 f8 f8 f8 f8 f8 f8 f8 f8 f8 f8
  0x10004d5ddf50: f8 f8 f8 f8 f8 f8 f8 f8 f8 f8 f8 f8 f8 f8 f8 f8
  0x10004d5ddf60: f8 f8 f8 f8 f8 f8 f8 f8 f8 f8 f8 f8 f8 f8 f8 f8
Shadow byte legend (one shadow byte represents 8 application bytes):
  Addressable:           00
  Partially addressable: 01 02 03 04 05 06 07
  Heap left redzone:       fa
  Freed heap region:       fd
  Stack left redzone:      f1
  Stack mid redzone:       f2
  Stack right redzone:     f3
  Stack after return:      f5
  Stack use after scope:   f8
  Global redzone:          f9
  Global init order:       f6
  Poisoned by user:        f7
  Container overflow:      fc
  Array cookie:            ac
  Intra object redzone:    bb
  ASan internal:           fe
  Left alloca redzone:     ca
  Right alloca redzone:    cb
28515ABORTING
```

I did some initial investigation and didn't really seme to be able to find out why the imeisv would be used after the scope has been left. Maybe a false positive? Or I'm too stupid?