Heap-buffer-overflow was found at ./src/core/ddsc/src/dds_stream.c:310:9 #501

Closed **luckyzfl** opened this issue on Apr 21, 2020 · 3 comments

luckyzfl commented on Apr 21, 2020

I used Peach Fuzzer to fuzz the HelloworldSubscriber at ./build/bin/HelloworldSubscriber.
After a period of time, A heap-buffer-overflow crash was found by AddressSanitizer. Next is the full crash information.

```
=================================================================
==51931==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x61100001a2b0 at pc 0x0000004c3c0c bp 0x7ffdab96aaa0 sp 0x7ffdab96a250
READ of size 786432 at 0x61100001a2b0 thread T0
    #0 0x4c3c0b in __asan_memcpy /root/CL/llvm/projects/compiler-rt/lib/asan/asan_interceptors.cc:466
    #1 0x7fa9a66b2a99 in dds_stream_reuse_string /root/fouzhe/cyclonedds-0.1.0-coverage/src/core/ddsc/src/dds_stream.c:310:9
    #2 0x7fa9a66b2a99 in dds_stream_read /root/fouzhe/cyclonedds-0.1.0-coverage/src/core/ddsc/src/dds_stream.c:841
    #3 0x7fa9a66b17c8 in dds_stream_read_sample /root/fouzhe/cyclonedds-0.1.0-coverage/src/core/ddsc/src/dds_stream.c:395:5
    #4 0x7fa9a651eab5 in serdata_default_to_sample_cdr /root/fouzhe/cyclonedds-0.1.0-coverage/src/core/ddsi/src/ddsi_serdata_default.c:506:5
    #5 0x7fa9a66754b9 in ddsi_serdata_to_sample /root/fouzhe/cyclonedds-0.1.0-coverage/src/core/ddsi/include/ddsi/ddsi_serdata.h:157:10
    #6 0x7fa9a66754b9 in dds_rhc_read_w_qminv /root/fouzhe/cyclonedds-0.1.0-coverage/src/core/ddsc/src/dds_rhc.c:1819
    #7 0x7fa9a66754b9 in dds_rhc_read /root/fouzhe/cyclonedds-0.1.0-coverage/src/core/ddsc/src/dds_rhc.c:2593
    #8 0x7fa9a66a855a in dds_read_impl /root/fouzhe/cyclonedds-0.1.0-coverage/src/core/ddsc/src/dds_read.c:166:29
    #9 0x7fa9a66a73c3 in dds_read /root/fouzhe/cyclonedds-0.1.0-coverage/src/core/ddsc/src/dds_read.c:265:12
    #10 0x511288 in main /root/fouzhe/cyclonedds-0.1.0-coverage/src/examples/helloworld/subscriber.c:53:14
feilong: iterationFinished status: 10559False_messageExit: False FaultOnEarlyExit:True _IsRunning:True
    #11 0x7fa9a559982f in __libc_start_main /build/glibc-LK5gWL/glibc-2.23/csu/../csu/libc-start.c:291
    #12 0x419f38 in _start (/root/fouzhe/cyclonedds-0.1.0-coverage/build/bin/HelloworldSubscriber+0x419f38)

0x61100001a2b0 is located 0 bytes to the right of 240-byte region [0x61100001a1c0,0x61100001a2b0)
allocated by thread T5 (recv) here:
    #0 0x4daa38 in __interceptor_malloc /root/CL/llvm/projects/compiler-rt/lib/asan/asan_malloc_linux.cc:67
    #1 0x7fa9a6709bf7 in os_malloc_s /root/fouzhe/cyclonedds-0.1.0-coverage/src/os/src/posix/../snippets/code/os_heap.c:30:12
    #2 0x7fa9a6709bf7 in os_malloc /root/fouzhe/cyclonedds-0.1.0-coverage/src/os/src/posix/../snippets/code/os_heap.c:39
    #3 0x7fa9a651e5e5 in serdata_default_from_ser /root/fouzhe/cyclonedds-0.1.0-coverage/src/core/ddsi/src/ddsi_serdata_default.c:283:31

Thread T5 (recv) created by T0 here:
    #0 0x431fdd in pthread_create /root/CL/llvm/projects/compiler-rt/lib/asan/asan_interceptors.cc:317
    #1 0x7fa9a6714464 in os_threadCreate /root/fouzhe/cyclonedds-0.1.0-coverage/src/os/src/posix/../snippets/code/os_posix_thread.c:275:21
    #2 0x7fa9a66215fe in create_thread /root/fouzhe/cyclonedds-0.1.0-coverage/src/core/ddsi/src/q_thread.c:272:7
    #3 0x7fa9a6593e98 in setup_and_start_recv_threads /root/fouzhe/cyclonedds-0.1.0-coverage/src/core/ddsi/src/q_init.c:838:34
    #4 0x7fa9a6593e98 in rtps_init /root/fouzhe/cyclonedds-0.1.0-coverage/src/core/ddsi/src/q_init.c:1312
    #5 0x7fa9a665fce1 in dds_init /root/fouzhe/cyclonedds-0.1.0-coverage/src/core/ddsc/src/dds_init.c:133:7
    #6 0x7fa9a6651a42 in dds_create_participant /root/fouzhe/cyclonedds-0.1.0-coverage/src/core/ddsc/src/dds_participant.c:160:11
    #7 0x511106 in main /root/fouzhe/cyclonedds-0.1.0-coverage/src/examples/helloworld/subscriber.c:24:19
    #8 0x7fa9a559982f in __libc_start_main /build/glibc-LK5gWL/glibc-2.23/csu/../csu/libc-start.c:291

SUMMARY: AddressSanitizer: heap-buffer-overflow /root/CL/llvm/projects/compiler-rt/lib/asan/asan_interceptors.cc:466 in __asan_memcpy
Shadow bytes around the buggy address:
  0x0c227fffb400: 00 00 00 00 00 00 fa fa fa fa fa fa fa fa fa fa
  0x0c227fffb410: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c227fffb420: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 fa fa
  0x0c227fffb430: fa fa fa fa fa fa fa fa 00 00 00 00 00 00 00 00
  0x0c227fffb440: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0c227fffb450: 00 00 00 00 00 00[fa]fa fa fa fa fa fa fa fa fa
  0x0c227fffb460: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c227fffb470: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 fa fa
  0x0c227fffb480: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c227fffb490: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c227fffb4a0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
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
==51931==ABORTING
```

And this problem was found in version v0.1.0.
I guess the problem was in function dds_realloc().
I did the similar fuzz operation on version v0.5.1 too, but the new version does not have this vulnerability.
But I found some similar code around the vulnerability position in the new version.
I am not sure why v0.1.0 has the vulnerability and v0.5.1 doe not have it.

eboasson commented on Apr 21, 2020 Contributor

This:

```
==51931==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x61100001a2b0 at pc 0x0000004c3c0c bp 0x7ffdab96aaa0 sp 0x7ffdab96a250
READ of size 786432 at 0x61100001a2b0 thread T0
    #0 0x4c3c0b in __asan_memcpy /root/CL/llvm/projects/compiler-rt/lib/asan/asan_interceptors.cc:466
    #1 0x7fa9a66b2a99 in dds_stream_reuse_string /root/fouzhe/cyclonedds-0.1.0-coverage/src/core/ddsc/src/dds_stream.c:310:9
```

is trying to `memcpy` a 786432 bytes long string (including terminating 0) from a sample that has a content of of less than 240 bytes (those 240 bytes are a header + payload).

It looks to me like your fuzzer flipped two bytes in the length:

- I believe address sanitizer only works on x64, so it would be little endian machine and most likely the input sample was also in little-endian encoding
- "Hello world" including the terminating 0 requires 12 bytes, so 0c.00.00.00 in a little-endian 32-bit int representation
- 786432 = 0xc0000, so 00.00.0c.00 in a little-endian 32-bit int representation

but regardless of how that length field got to have this value that is way too large, the cause of the crash is that it did not validate the input is well-formed.

What changed between 0.1.0 and 0.5.0 is a rewrite of the input processing (see `3067a69`), precisely because of insufficient input validation. Now, as soon as it receives the sample from the network it validates the length fields and string terminators embedded in it and rejects it if anything is wrong. The result is that you only get a `struct ddsi_serdata` if the input is well-formed and hence there is no need for changes to where it deserializes into an application sample (which is where it crashed).

nluedtke commented on Aug 31, 2021

This was assigned CVE-2020-18735.

k0ekk0ek commented on Aug 31, 2021 Contributor

Hi **@nluedtke**! Thanks for creating (or letting us know) 👍 For future reference, I think it's good to mention that this issue only existed in Cyclone DDS 0.1.0 and has been fixed by the commit indicated by **@eboasson**. I think we simply forgot to close this issue (I'll do so now). I've quickly looked into backporting this, but it would entail copying the serializer from 0.5.0, and that makes 0.1.0 be 0.5.0 in practice. Users are therefore simply encouraged to upgrade instead. Hope that makes sense(?) **@nluedtke** (or anyone else for that matter), if you need additional support or have any questions, please let us know.

👍 1

k0ekk0ek closed this as completed on Sep 1, 2021

k0ekk0ek mentioned this issue on Sep 1, 2021

Stack-buffer-overflow was found at ./src/core/ddsi/include/ddsi/q_bitset.h:50:13 in nn_bitset_one. #476

Closed

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

4 participants