Memory leak in client induced by malicious server(s) without CURVE/ZAP

Low **bluca** published **GHSA-wfr2-29gj-5w87** on Sep 7, 2020

Package

libzmq

Affected versions	Patched versions
<= 4.3.2	4.3.3

Description

Impact

Client connecting to compromised servers without CURVE/ZAP

Patches

#3918

Workarounds

No workarounds

References

Found thanks to oss-fuzz:

https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=22037
https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=22123

For more information

When a pipe processes a delimiter and is already not in active state but still has an unfinished message, the message is leaked.

The following input from a server causes the leak to appear ~70% of the time when running test_connect_null_fuzzer under valgrind:

```
fffefefd0101010108ff01010108ff010101010101c0a15b0100010001000100010001000100010001000000000000000001000100010001000100010001000100010001000100010001000100ffff01000100010001000100010061
```

The following reproduces 100% of the time:

```
01f301033672630000c0a15b010001000100010001000100010001000100616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161
```

```
==31504== Memcheck, a memory error detector
==31504== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==31504== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==31504== Command: tests/.libs/test_connect_null_fuzzer
==31504==
tests/test_connect_null_fuzzer.cpp:141:test_connect_null_fuzzer:PASS

-----------------------
1 Tests 0 Failures 0 Ignored
OK
==31504==
==31504== HEAP SUMMARY:
==31504==     in use at exit: 18,160 bytes in 1 blocks
==31504==   total heap usage: 76 allocs, 75 frees, 169,230 bytes allocated
==31504==
==31504== 18,160 bytes in 1 blocks are definitely lost in loss record 1 of 1
==31504==    at 0x483577F: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==31504==    by 0x48DC506: zmq::shared_message_memory_allocator::allocate() (decoder_allocators.cpp:84)
==31504==    by 0x48CE1D9: zmq::decoder_base_t<zmq::v2_decoder_t, zmq::shared_message_memory_allocator>::decoder_base_t(unsigned long) (decoder.hpp:66)
==31504==    by 0x48CDA60: zmq::v2_decoder_t::v2_decoder_t(unsigned long, long, bool) (v2_decoder.cpp:47)
==31504==    by 0x48E1A3F: zmq::zmtp_engine_t::handshake_v2_0() (zmtp_engine.cpp:344)
==31504==    by 0x48E0C99: zmq::zmtp_engine_t::handshake() (zmtp_engine.cpp:134)
==31504==    by 0x48BED7D: zmq::stream_engine_base_t::in_event_internal() (stream_engine_base.cpp:253)
==31504==    by 0x48BECDB: zmq::stream_engine_base_t::in_event() (stream_engine_base.cpp:243)
==31504==    by 0x48E0B8C: zmq::zmtp_engine_t::plug_internal() (zmtp_engine.cpp:116)
==31504==    by 0x48BEB03: zmq::stream_engine_base_t::plug(zmq::io_thread_t*, zmq::session_base_t*) (stream_engine_base.cpp:196)
==31504==    by 0x48AC0A7: zmq::session_base_t::process_attach(zmq::i_engine*) (session_base.cpp:417)
==31504==    by 0x488D051: zmq::object_t::process_command(zmq::command_t const&) (object.cpp:97)
==31504==
==31504== LEAK SUMMARY:
==31504==    definitely lost: 18,160 bytes in 1 blocks
==31504==    indirectly lost: 0 bytes in 0 blocks
==31504==      possibly lost: 0 bytes in 0 blocks
==31504==    still reachable: 0 bytes in 0 blocks
==31504==         suppressed: 0 bytes in 0 blocks
==31504==
==31504== For lists of detected and suppressed errors, rerun with: -s
==31504== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

The leak scales with the number of servers, which makes it worth a low severity advisory. Probably very hard to exploit in a real application, since you'd need to trick the client into connecting to multiple servers.

Severity

Low

CVE ID

No known CVE

Weaknesses

No CWEs