

CS3312 Lab Fuzzing

学号: 522031910439 姓名: 梁俊轩

2025 年 4 月 9 日

1 心脏滴血漏洞 (CVE-2014-0160)

这是一个出现在加密程序库 OpenSSL 的安全漏洞, 该程序库广泛用于实现互联网的传输层安全 (TLS) 协议。

此问题的原因是在实现 TLS 的心跳扩展时没有对输入进行适当验证 (缺少边界检查), 因此漏洞的名称来源于“心跳” (heartbeat)。该程序错误属于缓冲区过滤, 即可以读取的数据比应该允许读取的还多。

HeartBleed 主要存在与 OpenSSL 的 1.0.1 版本到 1.0.1f 版本。

利用该漏洞, 攻击者每次可以远程读取服务器内存中至多 64K 的数据, 获取内存中的敏感信息。

2 实验要求

- 自建虚拟机, 推荐 Ubuntu \geq 18.04 Clang/LLVM \geq 6.0
- 使用 LibFuzzer 或 AFL, 结合 AddressSanitizer 复现该漏洞, 并分析漏洞原因及利用方法
- 手工修复该漏洞, 并对之前得到的 PoC 进行重放 (replay), 以说明修复的正确性。
- 对修复后的代码重新进行 Fuzzing, 持续时间不少于 1 小时, 以确保没有引入新的漏洞。

3 复现

参考代码仓库 <https://gitlab.com/gitlab-org/security-products/demos/coverage-fuzzing/heartbleed-fuzzing-example>, 按照上面 README.md 提供的指示来复现。

结果如图1,2,3所示, 我们可以看到在 heap-buffer-overflow 的报错信息里提到了 `tls1_process_heartbeat`, 这也是为什么这个漏洞叫做心跳滴血漏洞。

```
==17695==ERROR: AddressSanitizer: heap-buffer-overflow on address
0x629000009748 at pc 0x5e7ab5ad4387 bp
0x7ffc89e80fb0 sp 0x7ffc89e80780

READ of size 24576 at 0x629000009748 thread T0

#0 0x5e7ab5ad4386 in __asan_memcpy
(/home/whitefork/Desktop/heartbleed-fuzzing-example/handshake-fuzzer+0x2ac386)
(BuildId: 90cc6097ccfdd566e0c90d030a567913b05f2774)
```



```
#1 0x5e7ab5b21be5 in tls1_process_heartbeat
```

```
/home/whitefork/Desktop/heartbleed-fuzzing-example/openssl-1.0.1f/ssl/t1_lib.c:2586:3
```

```

INFO: Running with entropy power schedule (OxKF, 100).
INFO: Loading...
INFO: Loaded 1 modules (35612 inline 8-bit constants): 35612 (hexaf5bafdb3b0, 40495 bytes)
INFO: Loaded 1 PC tables (35612 PCs): 35612 (hexaf5bafdb3b0, 40495 bytes)
INFO: max len for provided libsize will not generate limits larger than 4096 bytes
INFO: A cursor is not needed, starting from an empty cursor
INFO: UNLVED cursor: 148 hexaf5bafdb3b0
INFO: NEW 465 f1: 408 cursor: 272/124 4 execs: 0 rssi: 404b L: 1/5 MS: 5 Changelist-ShuffleBytes-ShuffleBytes-Changelist-
INFO: NEW 466 f1: 418 cursor: 1/4b 144 execs: 0 rssi: 150d L: 2/2 MS: 3 Changelist-ShuffleBytes-CopyPair-
INFO: NEW FUNC[1]: 435 hexaf5bafdb3b0 in f8a not error Phone/htlfork/Desktop/heartsbleed-fuzzing-example/openssl-1.0.1/fcrypto/err.c:708
#451 NEW 467 f1: 442 cursor: 0/6a 0 execs: 0 rssi: 6090 L: 6/6 MS: 4 InsertByte-Erasable-
INFO: NEW 468 f1: 452 cursor: 0/6a 0 execs: 0 rssi: 6090 L: 6/6 MS: 3 InsertByte-Crossover-InsertByte-
INFO: REDUCE cursor: 413 f1: 462 cursor: 0/6a 0 execs: 0 rssi: 1380b L: 1/5 MS: 2 Erases-
INFO: NEW FUNC[1]: 432 hexaf5bafdb3b0 in t151 execs: 0/home/htlfork/Desktop/heartsbleed-fuzzing-example/openssl-1.0.1/fssl/itc.c:605
INFO: NEW FUNC[1]: 432 hexaf5bafdb3b0 in f8a not error Phone/htlfork/Desktop/heartsbleed-fuzzing-example/openssl-1.0.1/fcrypto/obj-lib.c:282
#2385 NEW 469 f1: 461 cursor: 0/513b 21 execs: 0 rssi: 1420d L: 1/5 MS: 4 InsertByte-Erasable-CMP-CopyPair- DE: "(003)(000)(000)000"
INFO: REDUCE cursor: 424 f1: 469 cursor: 0/513b 21 execs: 0 rssi: 1420d L: 1/5 MS: 1 Changelist-
INFO: REDUCE cursor: 428 f1: 471 cursor: 7/25b 121 execs: 0 rssi: 1470d L: 7/7 MS: 2 InsertByte-Crossover-
INFO: REDUCE cursor: 426 f1: 474 cursor: 0/830b 121 execs: 0 rssi: 1450d L: 11/11 MS: 1 InsertRepeatByte-
INFO: REDUCE cursor: 431 f1: 469 cursor: 0/830b 121 execs: 0 rssi: 1470d L: 1/5 MS: 1 Changelist-PersAutoClist-Changelist- DE: "(003)(000)(000)000"
#2514 NEW 470 f1: 477 cursor: 10/55b 121 execs: 0 rssi: 1480d L: 10/11 MS: 1 RepeatByte-
INFO: REDUCE cursor: 433 f1: 477 cursor: 10/55b 121 execs: 0 rssi: 1480d L: 10/11 MS: 1 CopyPair-
#2525 NEW FUNC[1]: 435 hexaf5bafdb3b0 in t151_alert execs: 0/home/htlfork/Desktop/heartsbleed-fuzzing-example/openssl-1.0.1/fssl/itc.c:1214
INFO: NEW FUNC[1]: 432 hexaf5bafdb3b0 in de_ssl_v111 execs: 0/home/htlfork/Desktop/heartsbleed-fuzzing-example/openssl-1.0.1/fssl/ptc_k30-
INFO: REDUCE cursor: 428 f1: 2745b 121 execs: 0 rssi: 1480d L: 1/5 MS: 1 Changelist-PersAutoClist-Changelist- DE: "(003)(000)(000)000"
#2556 NEW 467 f1: 559 cursor: 13/100b 121 execs: 0 rssi: 1515b L: 15/15 MS: 1 PersAutoClist- DE: "(003)(000)(000)000"
INFO: REDUCE cursor: 427 f1: 561 cursor: 13/120b 121 execs: 0 rssi: 1520b L: 13/13 MS: 3 PersAutoClist-Erasables- DE: "(003)(000)(000)000"
#2636 NEW 467 f1: 568 cursor: 13/130b 121 execs: 0 rssi: 1530b L: 15/15 MS: 3 InsertByte-Crossover-InsertByte-
INFO: REDUCE cursor: 467 f1: 570 cursor: 13/130b 121 execs: 0 rssi: 1610d L: 20/20 MS: 3 PersAutoClist-PersAutoClist-InsertByte- DE: "(003)(000)(000)000"
#2829 NEW 467 f1: 581 cursor: 13/120b 121 execs: 0 rssi: 1520b L: 13/13 MS: 2 Changelist-Erasables-
INFO: REDUCE cursor: 467 f1: 570 cursor: 15/120b 121 execs: 0 rssi: 1650d L: 6/20 MS: 3 Changelist-PersAutoClist-Erasables- DE: "(003)(000)(000)000"
#2934 REDUCE cursor: 467 f1: 570 cursor: 15/120b 121 execs: 0 rssi: 1650d L: 6/20 MS: 3 Changelist-PersAutoClist-Erasables- DE: "(003)(000)(000)000"
#3093 REDUCE cursor: 467 f1: 570 cursor: 15/120b 121 execs: 0 rssi: 1650d L: 6/20 MS: 3 Changelist-PersAutoClist-Erasables- DE: "(003)(000)(000)000"
#3019 NEW 467 f1: 580 cursor: 16/140b 141 execs: 0 rssi: 1690d L: 21/21 MS: 3 CopyPair-Changelist-Crossover-
INFO: REDUCE cursor: 467 f1: 580 cursor: 16/140b 141 execs: 0 rssi: 1780d L: 10/21 MS: 1 Erases-
#3046 NEW 467 f1: 581 cursor: 17/120b 121 execs: 0 rssi: 1690d L: 21/21 MS: 1 Changelist-
#4221 NEW 467 f1: 592 cursor: 18/220b 125 execs: 0 rssi: 2190d L: 20/21 MS: 1 PersAutoClist- DE: "(003)(000)(000)000"
INFO: REDUCE cursor: 467 f1: 592 cursor: 18/220b 125 execs: 0 rssi: 2190d L: 20/21 MS: 1 PersAutoClist-PersAutoClist- DE: "(003)(000)(000)000"
#4242 NEW 467 f1: 596 cursor: 20/185b 125 execs: 0 rssi: 2210d L: 21/21 MS: 1 Crossover-
#4363 NEW 467 f1: 611 cursor: 21/255b 130 execs: 0 rssi: 2670d L: 30/30 MS: 1 Crossover-
#4542 REDUCE cursor: 467 f1: 611 cursor: 21/255b 130 execs: 0 rssi: 2670d L: 30/30 MS: 2 Changelist-InsertByte-Changelist-Erasables-
#4563 REDUCE cursor: 467 f1: 611 cursor: 21/255b 130 execs: 0 rssi: 3190d L: 15/30 MS: 2 Changelist-Erasables-
#4568 REDUCE cursor: 467 f1: 616 cursor: 23/270b 130 execs: 0 rssi: 3270d L: 23/23 MS: 2 Changelist-InsertByte-Crossover-
#4706 NEW 467 f1: 619 cursor: 23/200b 130 execs: 0 rssi: 3410b L: 34/34 MS: 1 InsertByte-
#7192 NEW 467 f1: 621 cursor: 24/300b 140 execs: 0 rssi: 3420d L: 42/42 MS: 1 CMP- DE: "(003)(000)(000)000"
#7193 REDUCE cursor: 467 f1: 621 cursor: 24/300b 140 execs: 0 rssi: 3420d L: 42/42 MS: 1 Changelist-Erasables-

```

图 1 漏洞复现结果 (1)

[illegible]

图 2 漏洞复现结果 (2)

4 原理分析

顺着复现结果，我们去查找 t1 lib.c 的内容：

```
#ifndef OPENSSSL_NO_HEARTBEATS
int
tls1_process_heartbeat(SSL *s)
{
    unsigned char *p = &s->s3->rrec.data[0], *p1;
    unsigned short hbtype;
    unsigned int payload;
    unsigned int padding = 16; /* Use minimum padding */
    /* Read type and payload length first */
    hbtype = *p++;
```



```

SUMMARY: AddressSanitizer: heap-buffer-overflow (/home/htb/fork/Desktop/heartbleed-fuzzing-example/handshake-fuzzer+42ac386) (BuildId: 90cc097ccfd566eac9d030a5e7913b65f774)
shadow bytes around the buggy address:
0x0c527ff92a00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0c527ff92a10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0c527ff92a20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0c527ff92a30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0c527ff92a40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0c527ff92a50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0c527ff92a60: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c527ff92a70: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c527ff92a80: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c527ff92a90: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c527ff92aa0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c527ff92ab0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c527ff92ac0: 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):
Addressables: 00
Partially addressable: 01 02 03 04 05 06 07
Heap left redzone: fa
Fixed heap region: fd
Stack left redzone: fe
Stack mid redzone: ff
Stack right redzone: f3
Stack after return: f5
Stack use after scope: f6
Global redzone: fb
Global init order: f8
Poisoned by user: f7
Container overflow: fc
Array cookie: ce
Intra object redzone: bb
ASAN internal: fa
Left alloca redzone: ca
Right alloca redzone: cb
--libasan=800f0d6c
ms: 5 ChangeByte::PersAutoDict::ShuffleBytes::CrossOver::changeInt::DE: "[274f0b2]"; base unit: 04633c47c5b19fd1c9a9eb5720434c7b0af44f
hex: 0x1,0x3,0x1,0x0,0x5,0x1,0x0,0x0,0x0,0x0,0x0,0x2,
(0x0,0x0,0x1,0x0,0x0,0x0,0x0,0x0,0x1,0x0,0x2,0x0,0x2,0x0)
artifacts_prefix: ./; Test unit written to: ./crash-460af6a13763375bd5105725baf382f22ba84
basefile: G000A00BYAC00=

```

图 3 漏洞复现结果 (3)

```

n2s(p, payload);
pl = p;
if (s->msg_callback)
    s->msg_callback(0, s->version, TLS1_RT_HEARTBEAT,
        &s->s3->rrec.data[0], s->s3->rrec.length,
        s, s->msg_callback_arg);
if (hbtype == TLS1_HB_REQUEST)
{
    unsigned char *buffer, *bp;
    int r;

    /* Allocate memory for the response, size is 1 bytes
     * message type, plus 2 bytes payload length, plus
     * payload, plus padding
     */
    buffer = OPENSSL_malloc(1 + 2 + payload + padding);
    bp = buffer;

    /* Enter response type, length and copy payload */
    *bp++ = TLS1_HB_RESPONSE;
    s2n(payload, bp);
    memcpy(bp, pl, payload);
    bp += payload;

    /* Random padding */
    RAND_pseudo_bytes(bp, padding);
    r = ssl3_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3 + payload + padding);
    if (r >= 0 && s->msg_callback)
        s->msg_callback(1, s->version, TLS1_RT_HEARTBEAT,
            buffer, 3 + payload + padding,
            s, s->msg_callback_arg);
    OPENSSL_free(buffer);
    if (r < 0)
        return r;
}
else if (hbtype == TLS1_HB_RESPONSE)
{
    unsigned int seq;

    /* We only send sequence numbers (2 bytes unsigned int),
     * and 16 random bytes, so we just try to read the
     * sequence number */
    n2s(pl, seq);

```

```
if (payload == 18 && seq == s->tlsext_hb_seq)
{
    s->tlsext_hb_seq++;
    s->tlsext_hb_pending = 0;
}
return 0;
}
```

在客户端发送完心跳包之后，服务端会短暂的存储心跳包中的内容，分配一个内存缓冲区（一个物理内存区域用以存储信息），该区域的存储空间和心跳请求信号里的长度一致。接下来，它会存储请求信号的加密数据到内存缓冲区，然后读取数据并将其发送回你的浏览器，来证明连接仍然存在。

关键是，OpenSSL 官方编写的库函数里没有检查这个客户端心跳包的实际内容和它给定的长度，也就是说本身是一个 10 字节的数据，客户端声明了 20 字节的内容，服务端就会开辟 20 字节的空间，导致服务端返回了 20 字节的数据。

算机接受心跳请求时从不检查该请求和它声称的内容是否一致，及从不检查所请求的数据长度是否和声称的数据长度一致，导致响应方返回额外长度的数据。

导致心脏出血漏洞的编程错误可以归于 `tl_lib.c` 中第 2586 行的代码：

```
memcpy(bp, pl, payload);
```

5 修复思路

那么针对此，我们首先去修改 `ssl/d1_both.c` 的内容：

原文部分如下，这是存在问题的：

```
hbtype = *p++;
n2s(p, payload);
pl = p;
```

修改之后，在拷贝之前先检查，若不符合则直接返回 0：

```
if (1 + 2 + 16 > s->s3->rrec.length)
    return 0;
hbtype = *p++;
n2s(p, payload);
if (1 + 2 + payload + 16 > s->s3->rrec.length) 2Has conversations. Original line has conversations.
    return 0;
pl = p;
```

同时先声明一个 `write_length`，用来替代后面的 `buffer` 长度，同时需要保证 `write_length` 不要太长：

```
unsigned int write_length = 1 /* heartbeat type */ +
    2 /* heartbeat length */ + 4Has conversations. Original line has conversations.
    payload + padding;

if (write_length > SSL3_RT_MAX_PLAIN_LENGTH)
    return 0;
```

给 `buffer` 分配空间时不再是 `3 + payload + padding`，而是 `write_length`，避免溢出：

```
buffer = OPENSSL_malloc(write_length);
```

下面与 3 + payload + padding 对应的 buffer 长度也要修改成 write_length:

```
if (r >= 0 && s->msg_callback)
    s->msg_callback(1, s->version, TLS1_RT_HEARTBEAT,
        buffer, write_length,
        s, s->msg_callback_arg);
```

在 ssl/tl_lib.c 中也要对应修改内容,就是在拷贝之前要先判断一遍长度:

```
if (1 + 2 + 16 > s->s3->rrec.length) return 0;
hbtype = *p++;
n2s(p, payload);
if (1 + 2 + payload + 16 > s->s3->rrec.length) return 0;
p1 = p;
```

6 修复结果

将第一次 Fuzzing 时得到 crash 作为输入进行重放,结果如图4所示,此时成功修复了此漏洞。

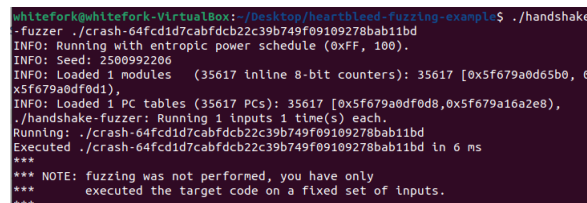


图 4 重放结果

对修复后的代码重新进行 Fuzzing,如图5所示,此时可以发现没有出现 heap-buffer-over-flow 的问题,但是出现了 memory leak 的问题,初步推测不是心脏滴血漏洞或者修改后的代码所引起的,因此可以认为此时漏洞修复成功了。

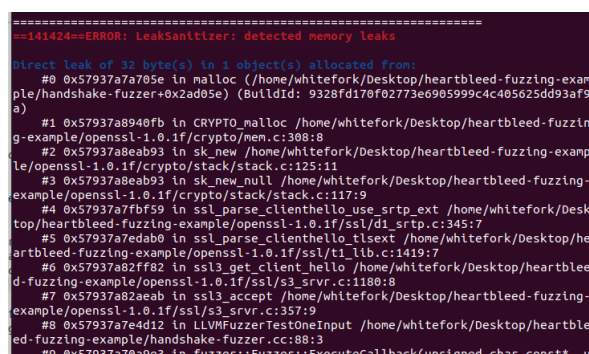


图 5 memory leak