Section 1: Review the SSL/TLS Handshake

Q1: What's the Content-Type for a record containing "Application Data"?

Application Data (23) as shown in Figure 1.

Q2: What's the version of the TLS protocol?

TLS 1.0 (0x0301) as shown in Figure 1.

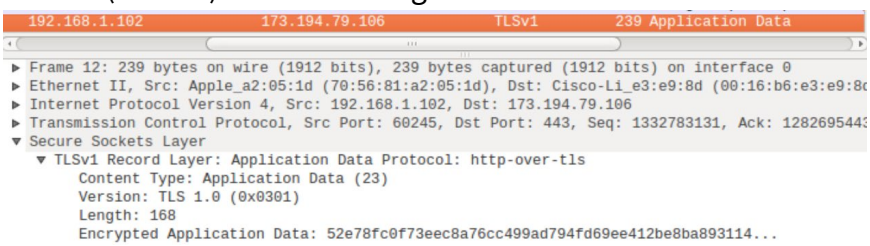

Figure 1: SSL layer of Application Data

<u>Client Hello Messages:</u>

Q3: What are the time (GMT seconds since midnight Jan 1, 1970) and random bytes (size 28) which are used later to generate the symmetric key?

The time, GMT seconds since midnight Jan 1, 1970 to the time and date as shown in Figure 3, 02:18:59 on Jul 31, 2012, is `1343715539.701779000` as shown in Figure 2. The random bytes are `16c25064f7cb0209b336ab332d969b8e091d26d4ccd04b73` as shown in Figure 3.

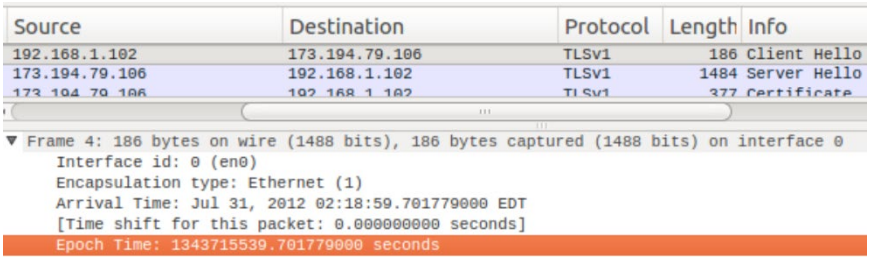

Figure 2: Client Hello Message (Epoch Time)

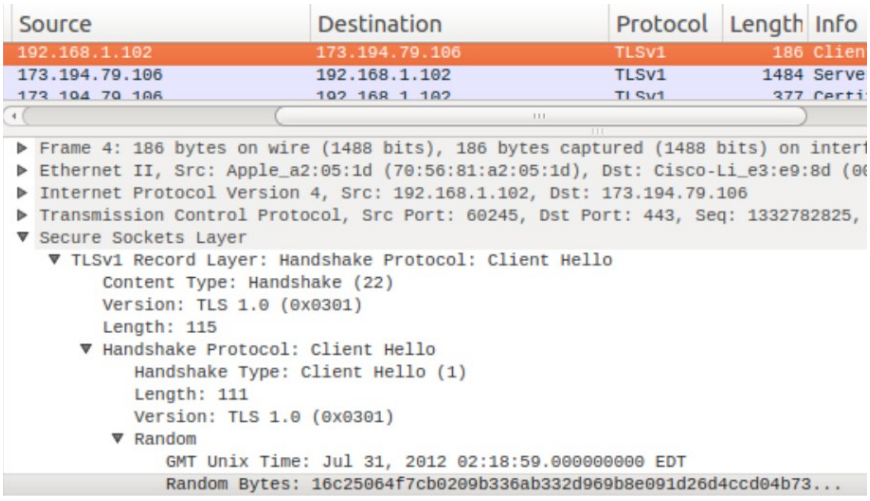

Figure 3: Client Hello Message (Random Bytes)

Q4: What is the list of cipher suites, which dictate the key exchange algorithm, bulk encryption algorithm (with key length), MAC, and a pseudo-random function?

The list of cipher suites can be seen from Figure 4.

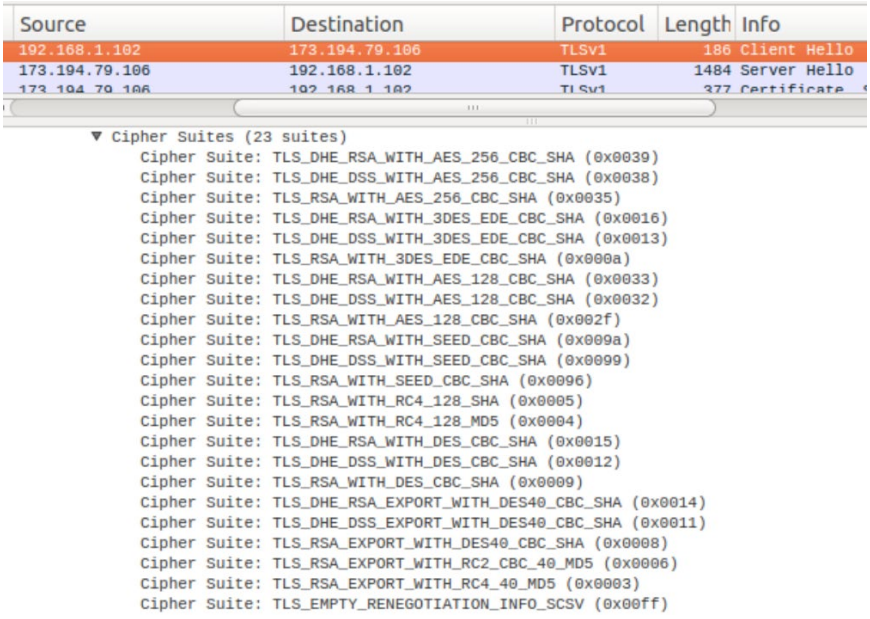

Figure 4: Client Hello Message (Cipher Suites)

Q5: How is the compression methods set? Why is it set like that?

The compression methods are set as in Figure 5.

It is set like this because the DEFLATE compression method allows the sending compressor to select from among several options to provide varying compression ratios, processing speeds, and memory requirements. TLS is a stateful protocol. Compression methods used with TLS can be either stateful or stateless.

Figure 5: Client Hello Message (Compression Methods)

Server Hello Messages:

Q6: What's the Cipher method chosen by the Server?

TLS_RSA_WITH_RC4_128_SHA (0x0005)

Figure 6: Server Hello Message (Cipher Method)

Certificate Messages:

Q7: What's the certificates messages in this step?

The overview of the certificate messages can be seen in Figure 7 and the details of each individual certificate message can be seen in Figures 8 and 9.

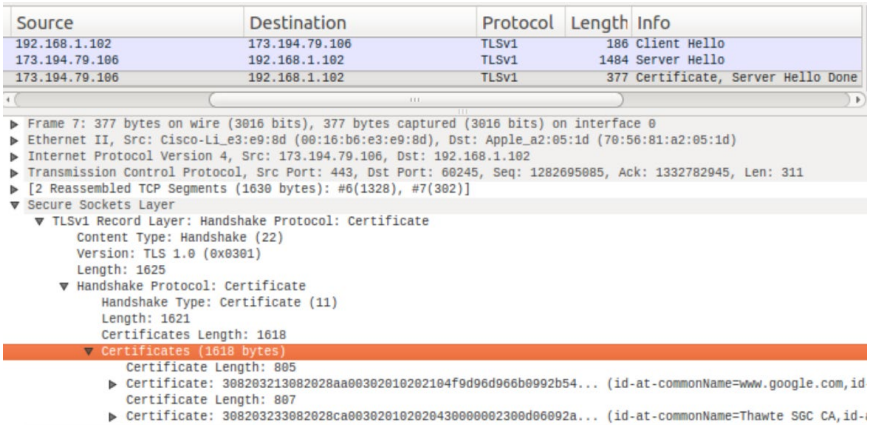

Figure 7: Certificate Messages

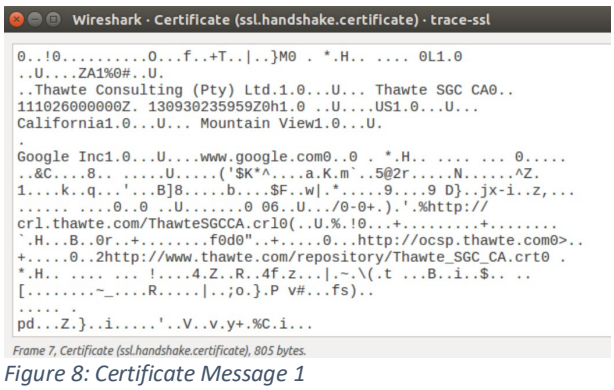

Figure 8: Certificate Message 1

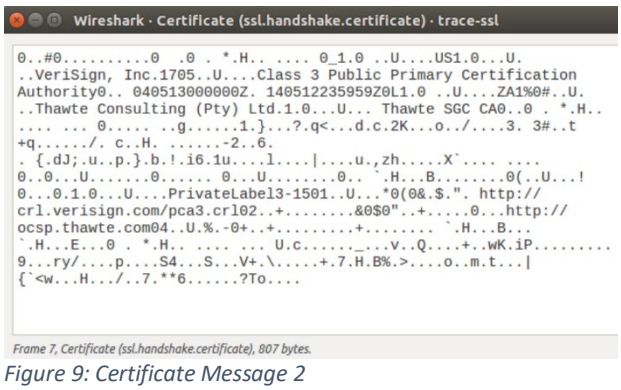

Figure 9: Certificate Message 2

Client Key Exchange and Change Cipher Messages:

Q8: What's the Content-Type for Change Cipher Spec message?

Change Cipher Spec (20) as shown in Figure 10.

Q9: What's the Change Cipher Spec message? What's its size?

The Change Cipher Spec message is simply 1 which is the size of a single byte.

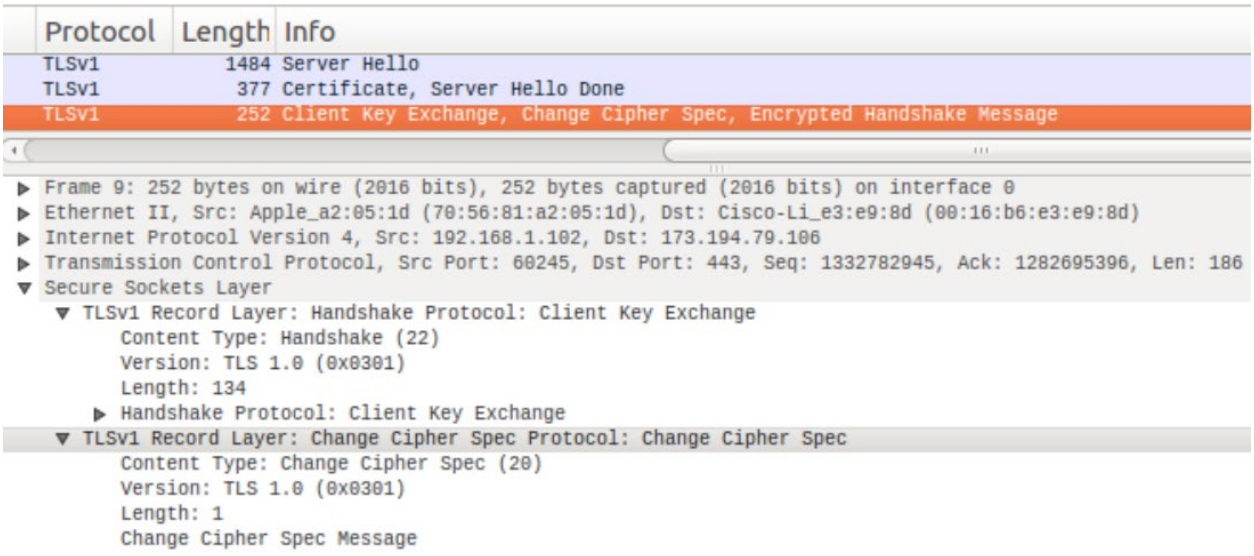

Figure 10: Change Cipher Spec Message

Section 2: The HeartBleed Bug

Task 1: Launch the Heartbleed Attack

In order to make the code executable, I ran `chmod 775 ./attack.py` before running the attack using `./attack.py www.heartbleedlabelgg.com` as shown in Figure 11.

```
[02/22/2020 06:33] seed@ubuntu:~$ chmod 775 ./attack.py
[02/22/2020 06:33] seed@ubuntu:~$ ./attack.py www.heartbleedlabelgg.com
```

Figure 11: Making Heartbleed Attack executable

Upon conducting the attack several times, I managed to obtain the following information from the target server:

- Username and password as shown in Figure 12
  - Username: "admin"
  - Password: "seedelgg"
- User's activity where a private message was composed and sent as shown in Figure 12
- The exact content of the private message as shown in Figure 13
  - The message, "We are now friends without your consent" with the subject "Hello"

Figure 12: Heartbleed Attack (Username, Password, Activity)

Figure 13: Heartbleed Attack (Private Message Content)

Task 2: Find the Cause of the Heartbleed Vulnerability

Q2.1: As the length variable decreases, what kind of difference can you observe? As the length variable decreases, the extra data returned is less and there is a boundary value for the input length variable where no more extra data is returned as the server sends back the reply without extra data.

Q2.2: 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.

The boundary value is 22 as shown in Figure 14. We can see that the request is not benign with a length variable of 23 as shown in Figure 15, showing that the boundary value is 22.

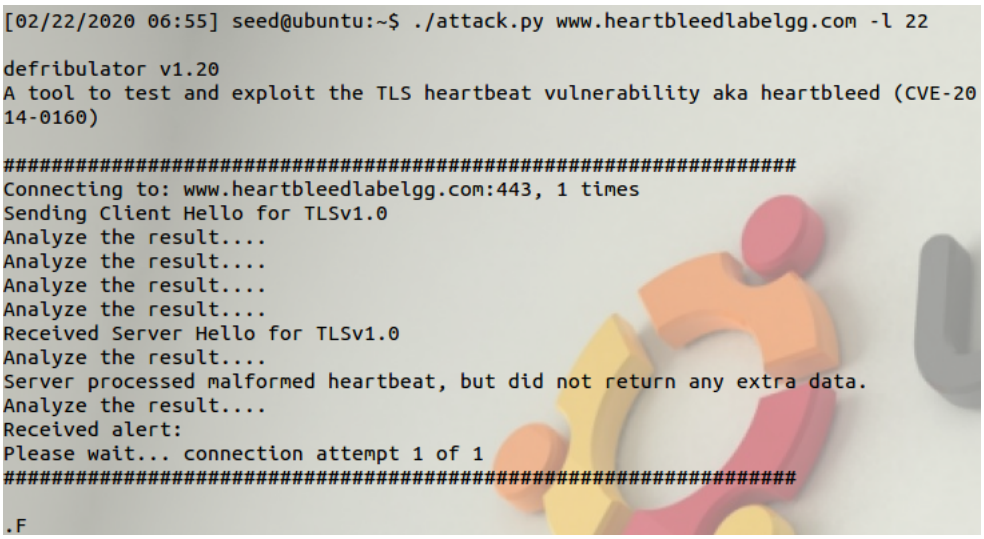

Figure 14: Length Variable Boundary (22)

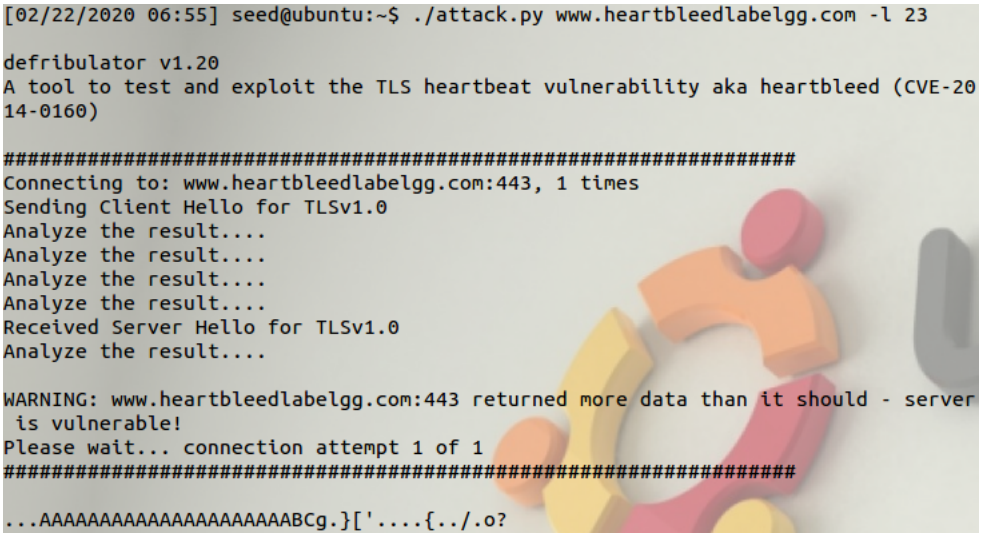

Figure 15: Length Variable (23)

Task 3: Countermeasure and Bug Fix

Q3.1: Try your attack again after you have updated the OpenSSL library. Please describe your observations.

As shown in Figure 16, the attack is unsuccessful, and no results are returned.

```
[02/22/2020 07:40] seed@ubuntu:~$ ./attack.py www.heartbleedlabelgg.com

defribulator v1.20
A tool to test and exploit the TLS heartbeat vulnerability aka heartbleed (CVE-20
14-0160)

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

Figure 16: Unsuccessful Heartbleed Attack after OpenSSL Library Update

Q3.2: Problem from the code and solution to fix the bug.

The problem from the code is in line 40 as shown in Figure 17, where there is no check conducted to ensure that the amount of data in `pl` is equal to the value given of `payload` to ensure that the data copied starting from the beginning of the `payload` content as determined by the pointer `pl`, is actually the data that should be copied and not more.

```
39      // copy payload
40      memcpy(bp, pl, payload); /* pl is the pointer which
41                               * points to the beginning
42                              * of the payload content */
```

Figure 17: Problem of Heartbeat request/response Packet Formation

To fix the bug, a check should be run to ensure that the amount of data in `pl` is equal to the value given of `payload`.

"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."

Alice is incorrect in stating that the error stems from the missing boundary checking during the buffer copy as it is the missing boundary checking during the payload copy that is the problem. Bob is incorrect in stating that user input validation would fix the issue as it is a good practice but would not resolve the issue since more than the expected payload can still be copied. Eva is incorrect in thinking that we can simply delete the length value from the packet as the length value is needed for the boundary checking during the payload copy to fix the problem.