Heartbleed Attack Lab

**Task 1: Launch the Heartbleed Attack**

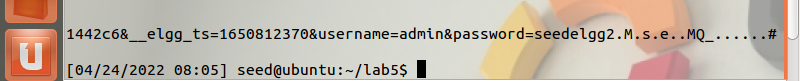
I have logged in as administrator, added boby as friend, and send a message. Then I run the attack.py code to execute a heartbleed attack.

Text

Description automatically generated

Text

Description automatically generated



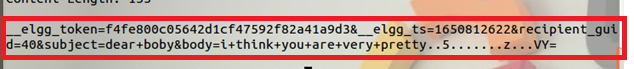
It is difficult to determine exactly what information has been returned. Some data appears to be encoded or hidden. But I did find the username and password of the account used in this round.



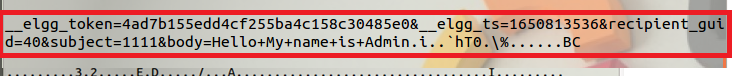
I played around with the system a bit, sending a few messages and adding a few friends. Then ran the heartbleed attack and found this:

Text

Description automatically generated with medium confidence



Bingo! I found a message send to recipient d = 40 with subject “dear boby” and the body of “I think you are very pretty”. I found that precise information regarding the actions of the user are difficult to determine, as the server does not explicitly state “admin has become friends with boby” or other plaintext that is easy to understand.



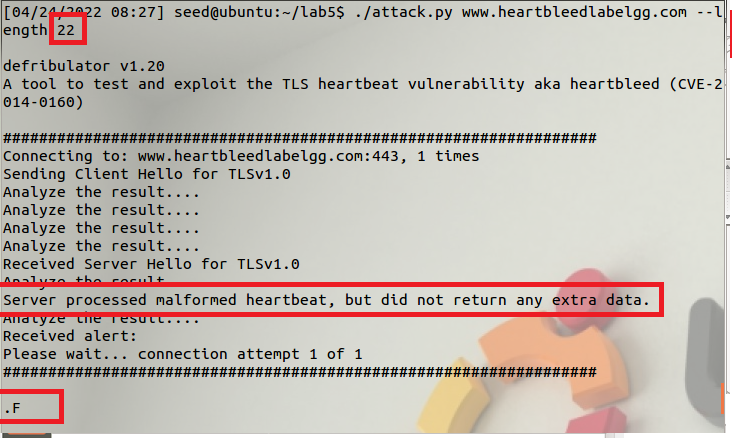
With enough tries however, the messages sent out are easy to distinguish.

**Task 2: Find the Cause of the Heartbleed Vulnerability**

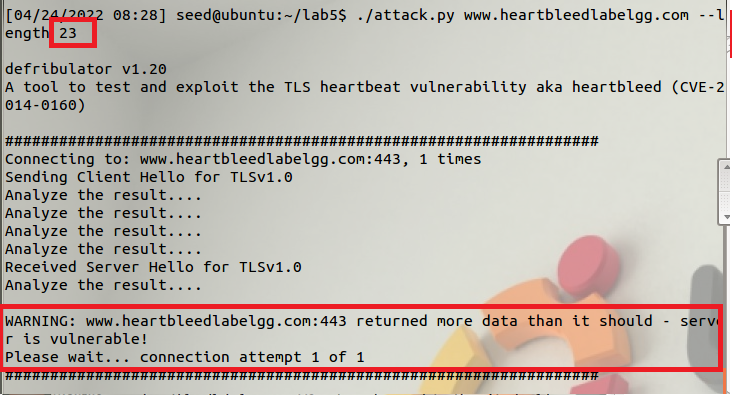
**Question 2.1:** As the length variable decreases, the amount of information being returned by the server decreases. Therefore, the payload length variable should be kept as large as possible to ensure the server returns all of its memory.

**Question 2.2:** The boundary length of the server is 22. At length = 22 and below, the server stops sending back extra data. At length >= 23, the server sends back extra data.

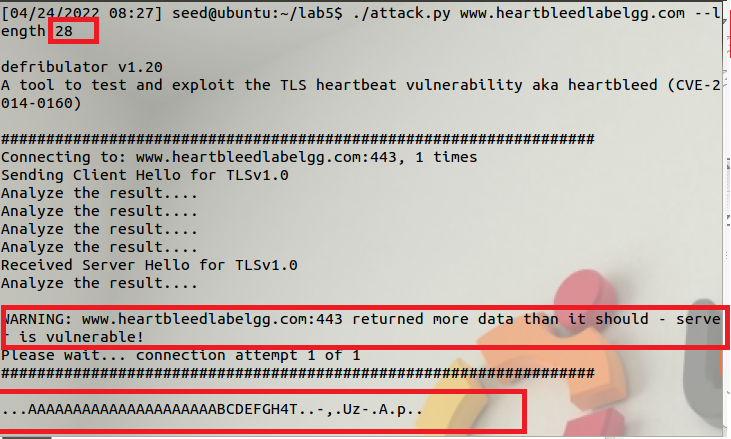
Screenshots shown below.

****

Length = 22, no extra data is returned. The server processed malformed heartbeat, but did not return any extra data.

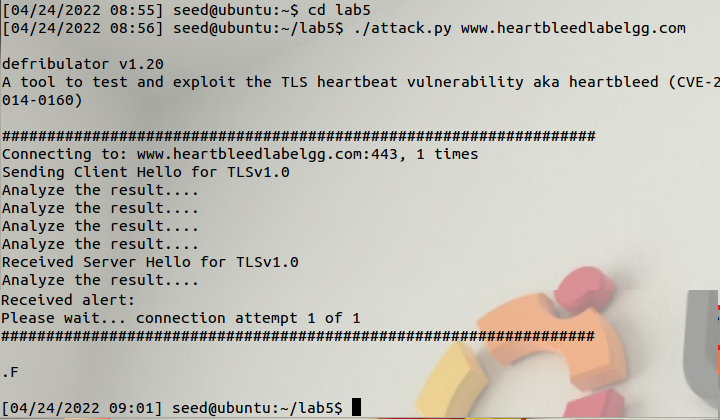


Length = 23. The server returns extra data and is vulnerable to heartbleed attack.



Length = 28. The server returned extra data, which is also circled in a red box.

**Task 3: Countermeasure and Bug Fix**



**Task 3.1:** After applying the patch, the heartbleed attack no longer returns any server information. No more stealing private messages from fake social media sites.

**Task 3.2:** At the line “buffer = OPENSSL\_malloc(1+2+payload+padding);”, we allocate memory equal to 3+payload+padding without doing any boundary checking on these values. The heartbleed attack sets the payload size to a very large number, which is never checked before memory allocation. The payload copying mechanism copies from a pointer location and continues to copy until the buffer is full. To solve this, input validation must be implemented to check what the size of the payload should be before blindly allocating more memory than is needed.

For the discussion between Alice, Bob, and Eva: Bob is correct. Alice’s solution is a less efficient workaround, boundary checking during buffer copy could fix the problem but not at the root. Eva’s solution might work at a basic level but leaves the vulnerability unpatched. Bob’s solution is the best solution because it solves the problem at its root and ensures that it will be impossible for a bogus payload length to be inserted.