

Figure Demonstration of Heartbleed attack revealing user activity

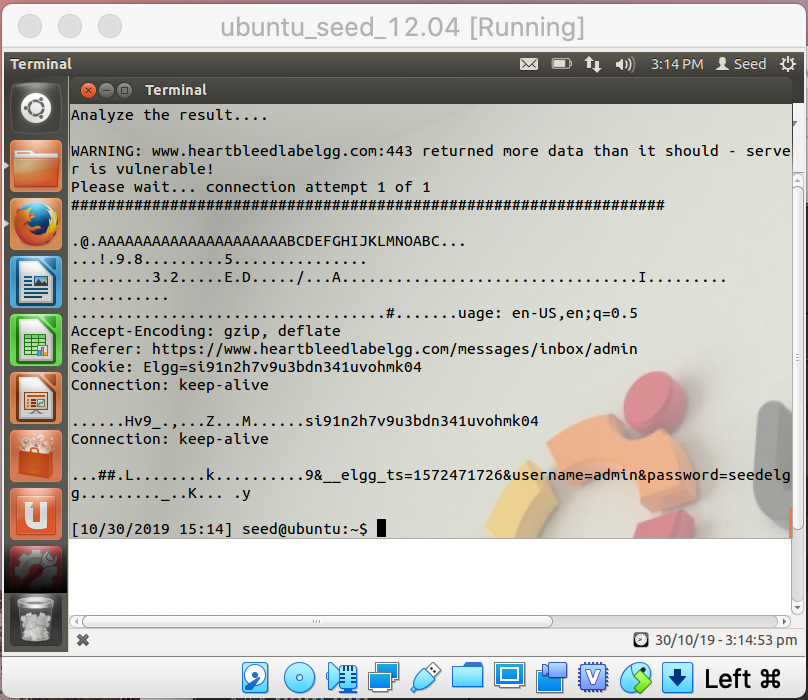


Figure Demonstrating of Heartbleed attack revealing admin credentials

**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**

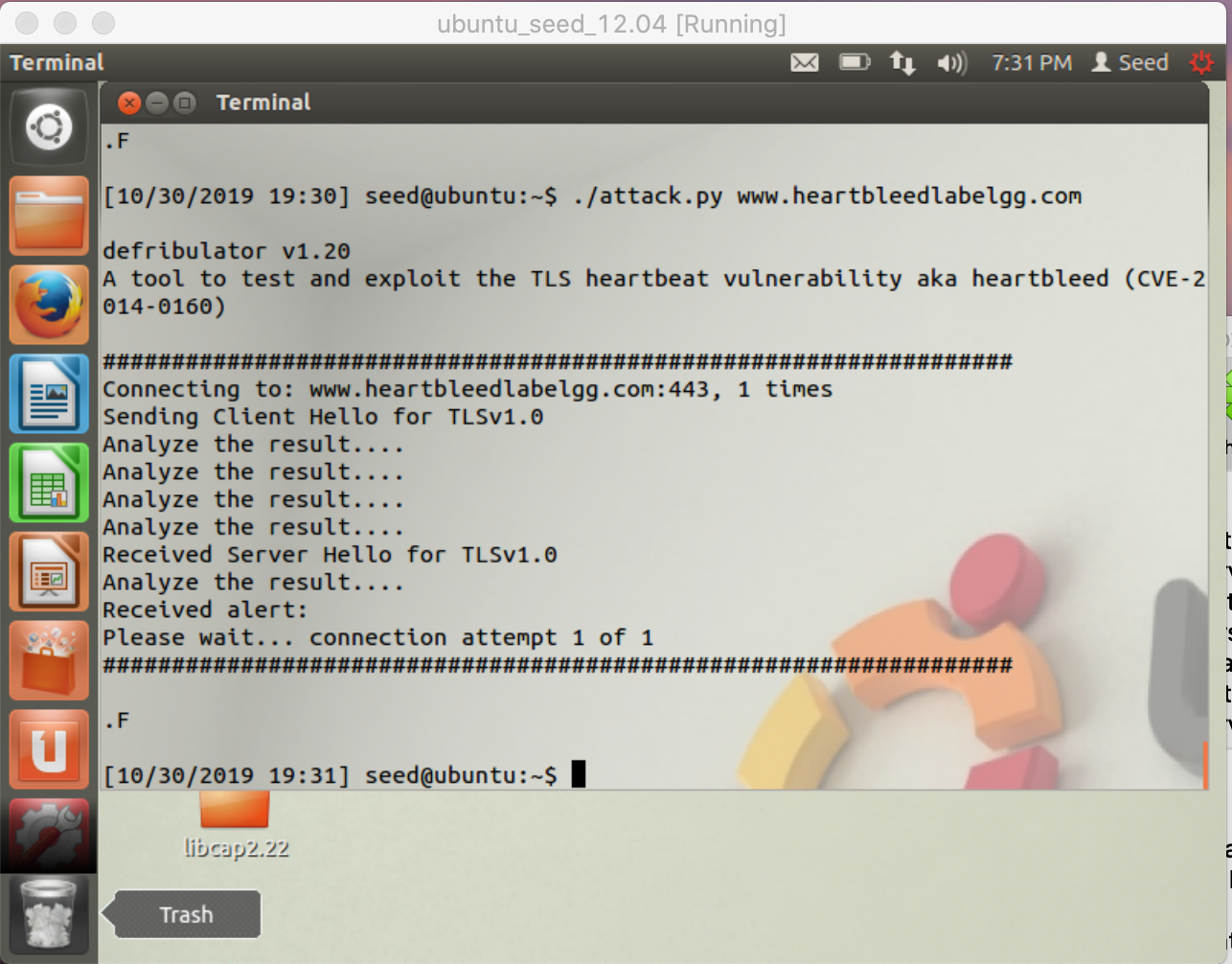


Figure Demonstration of a failed Heartbleed attack

**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 could be less than the client-specified size in , 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:

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 .