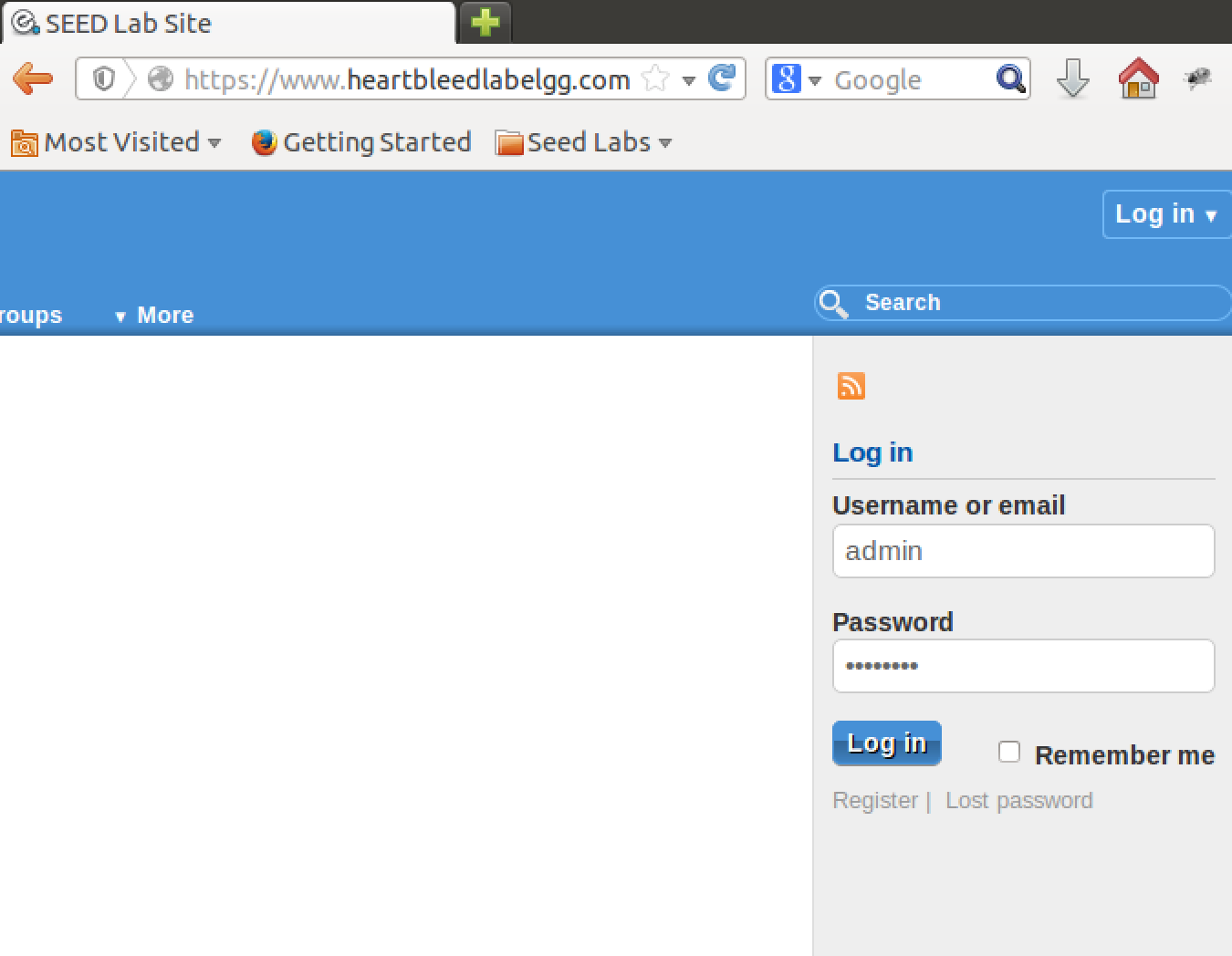
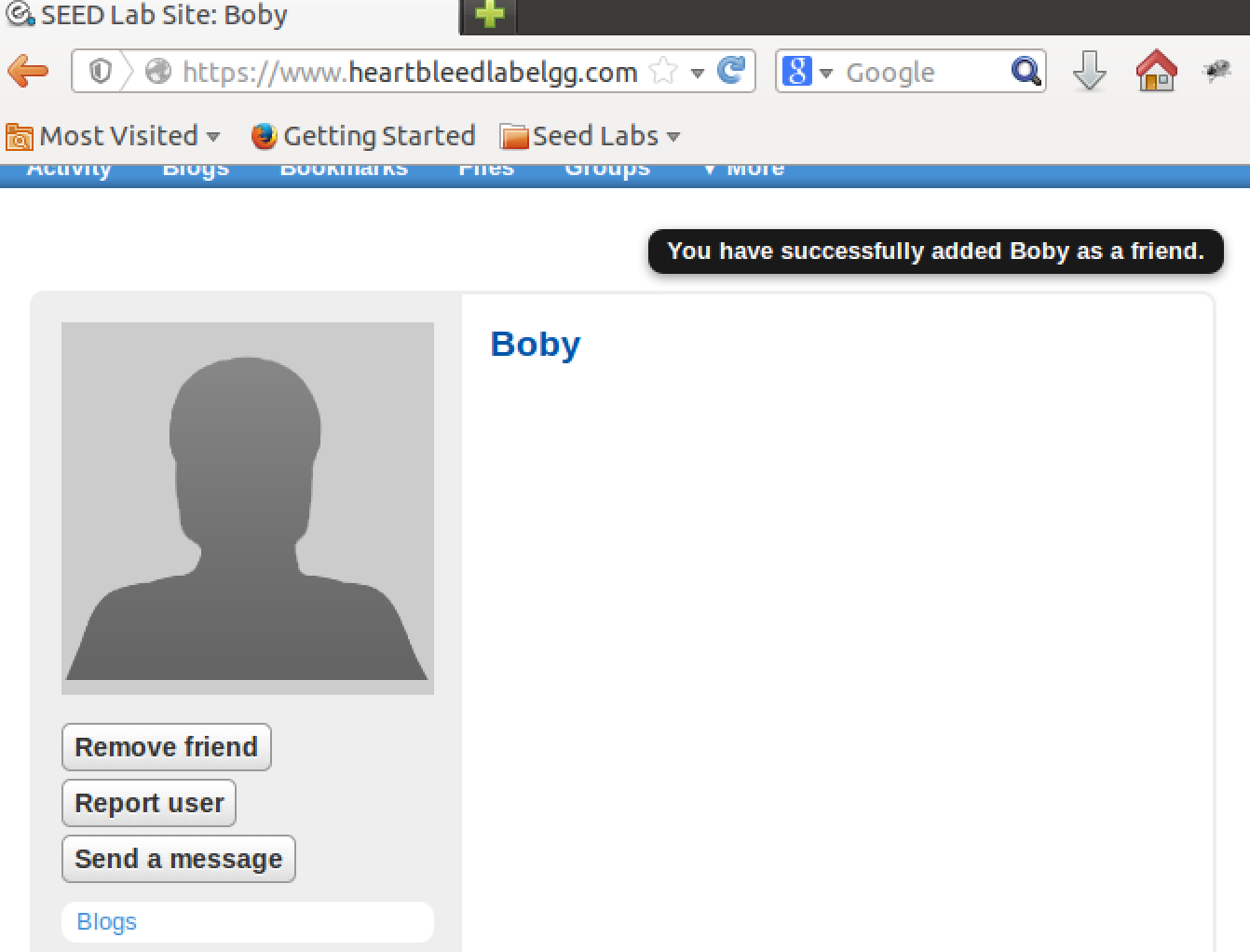
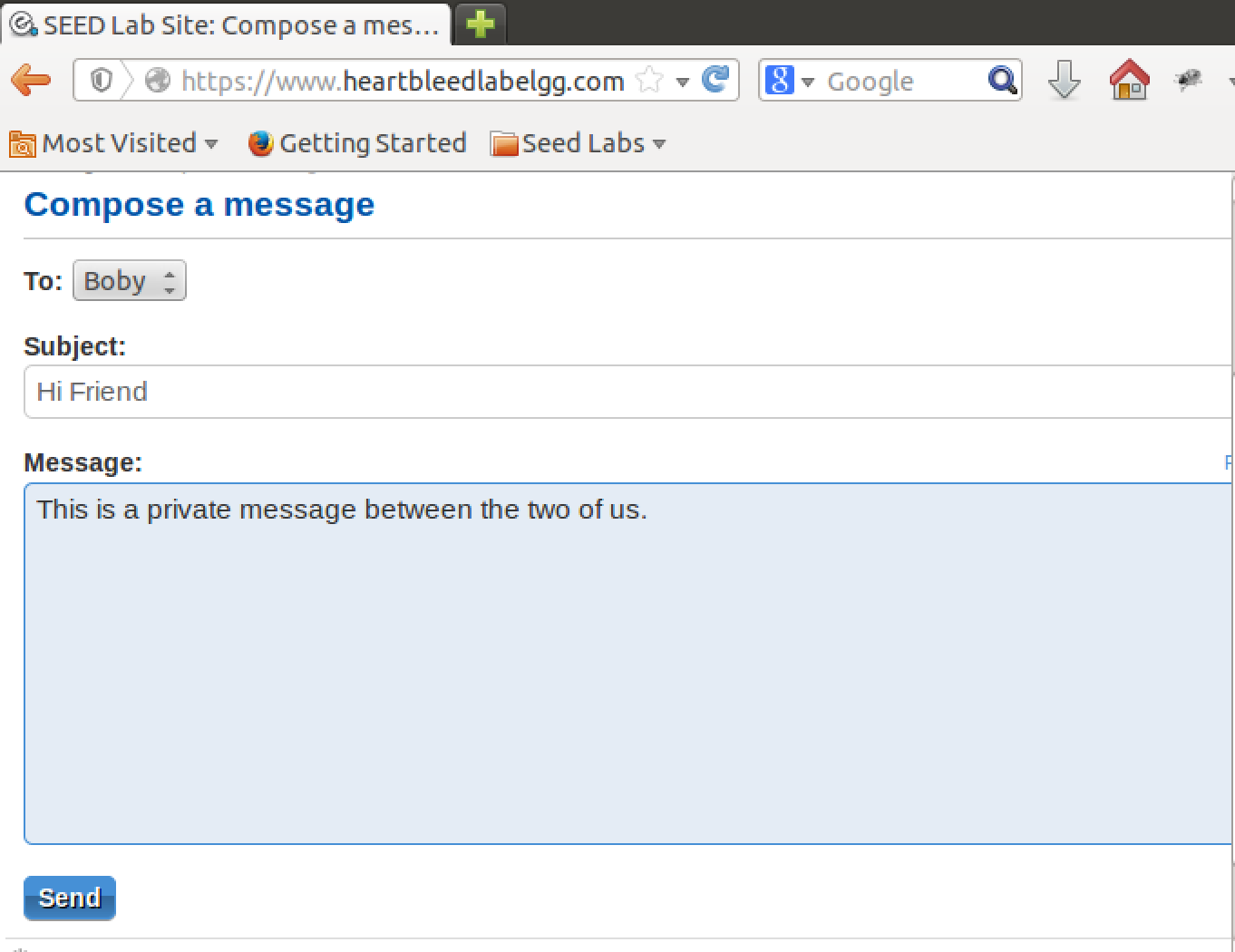
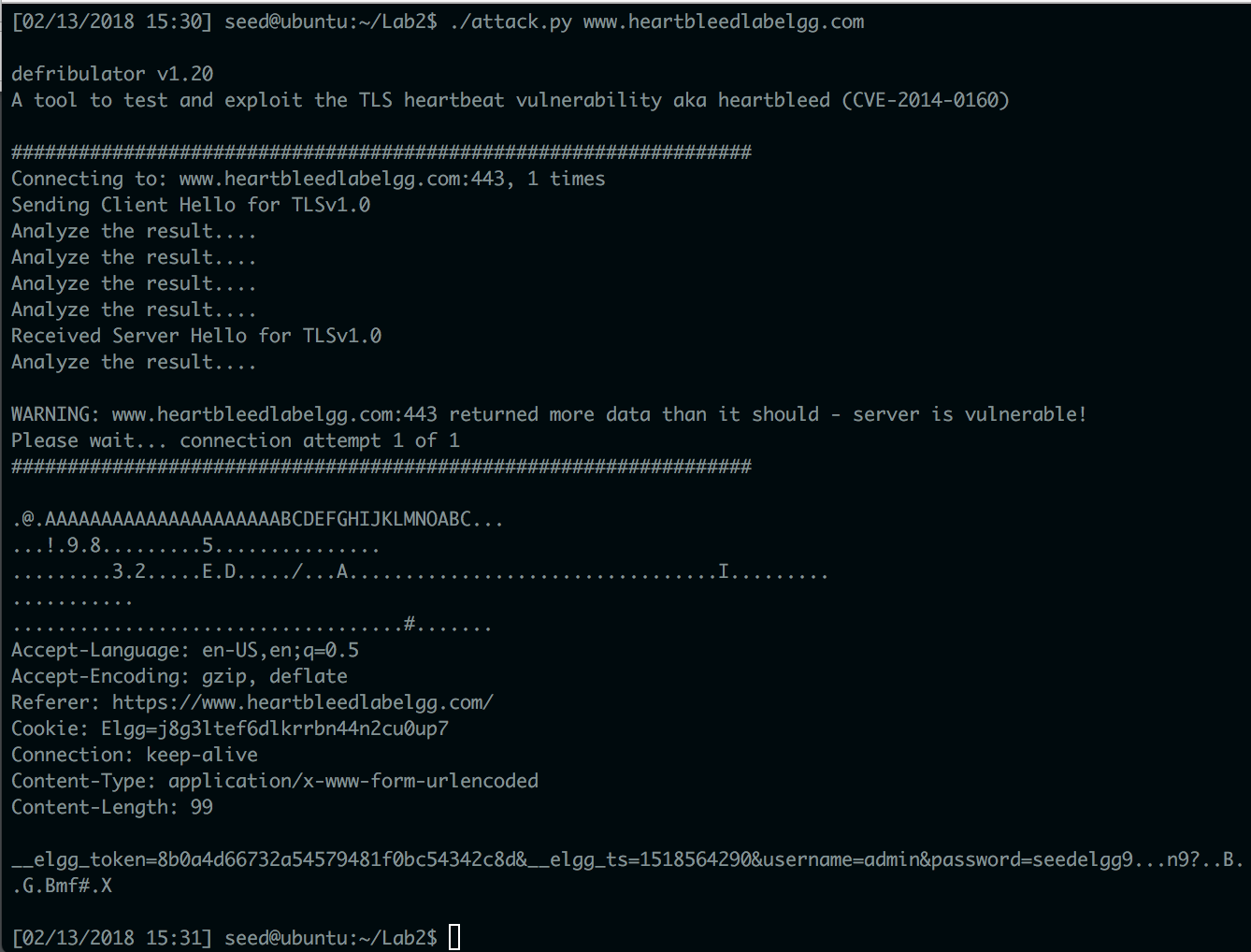
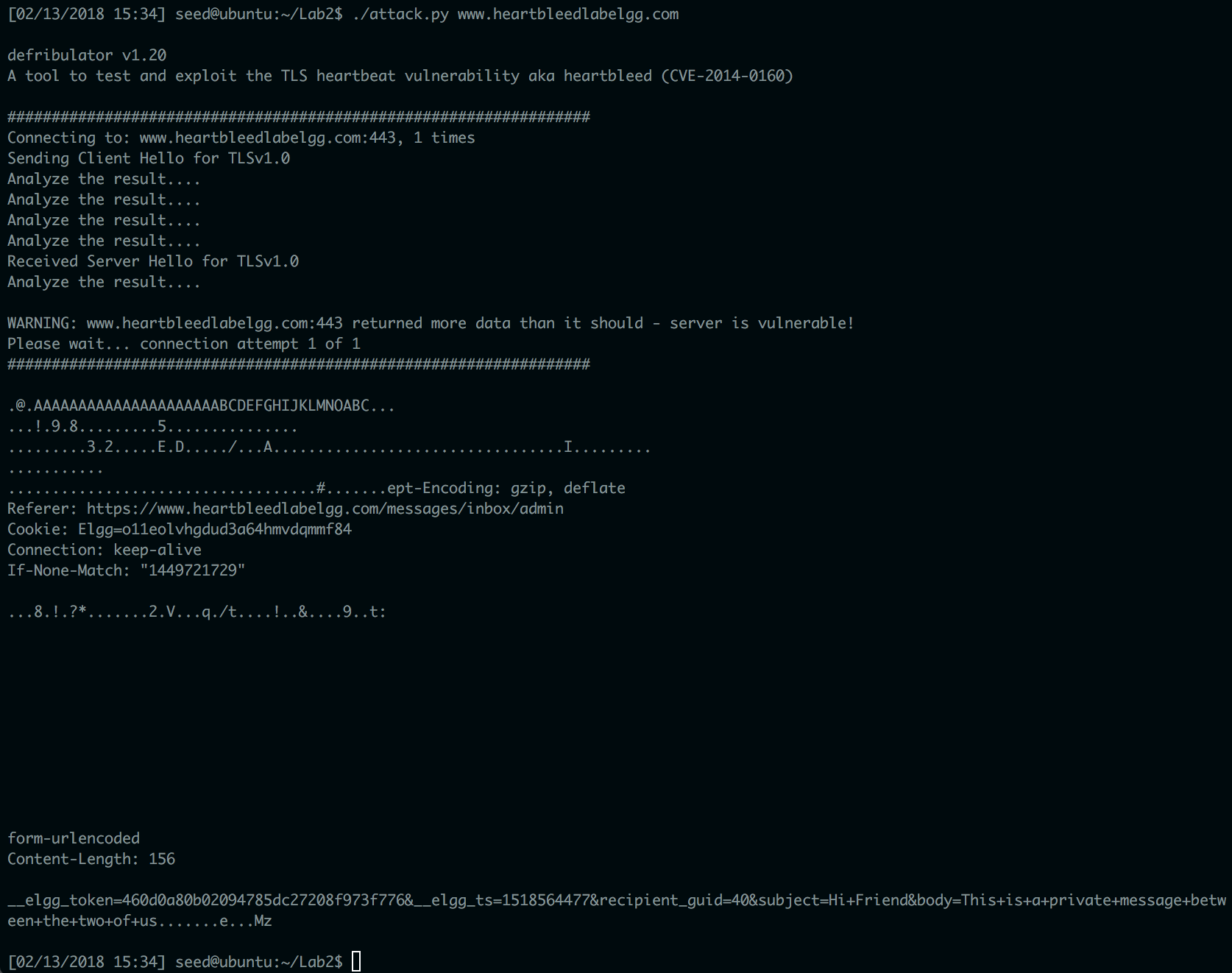
Homework #2

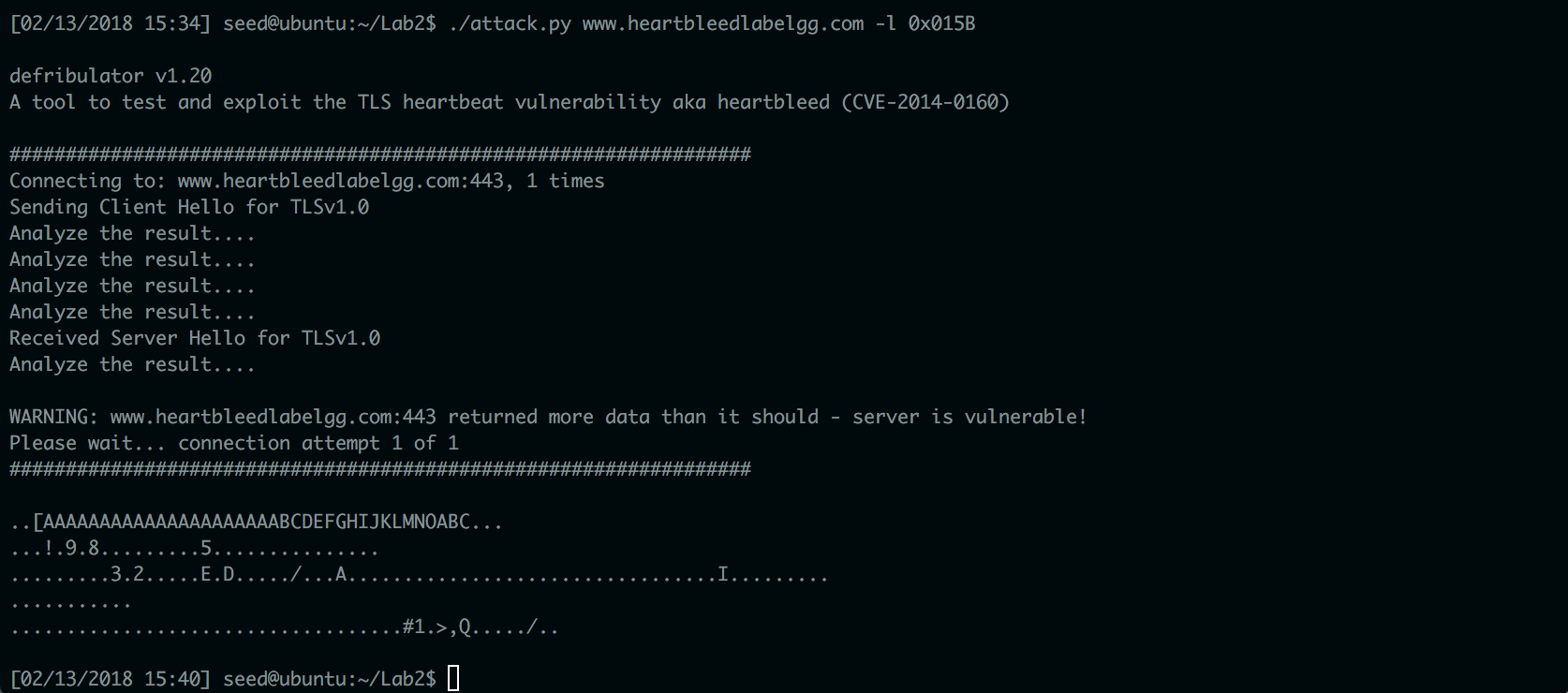
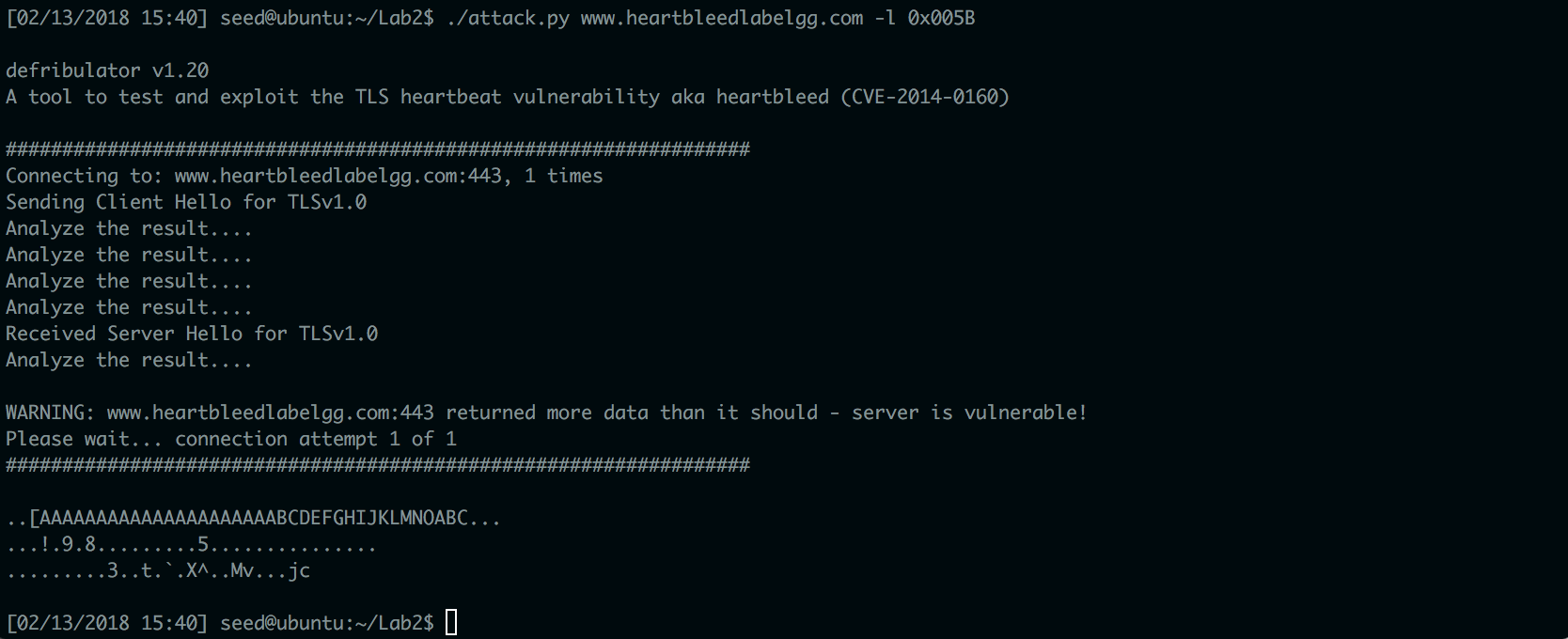
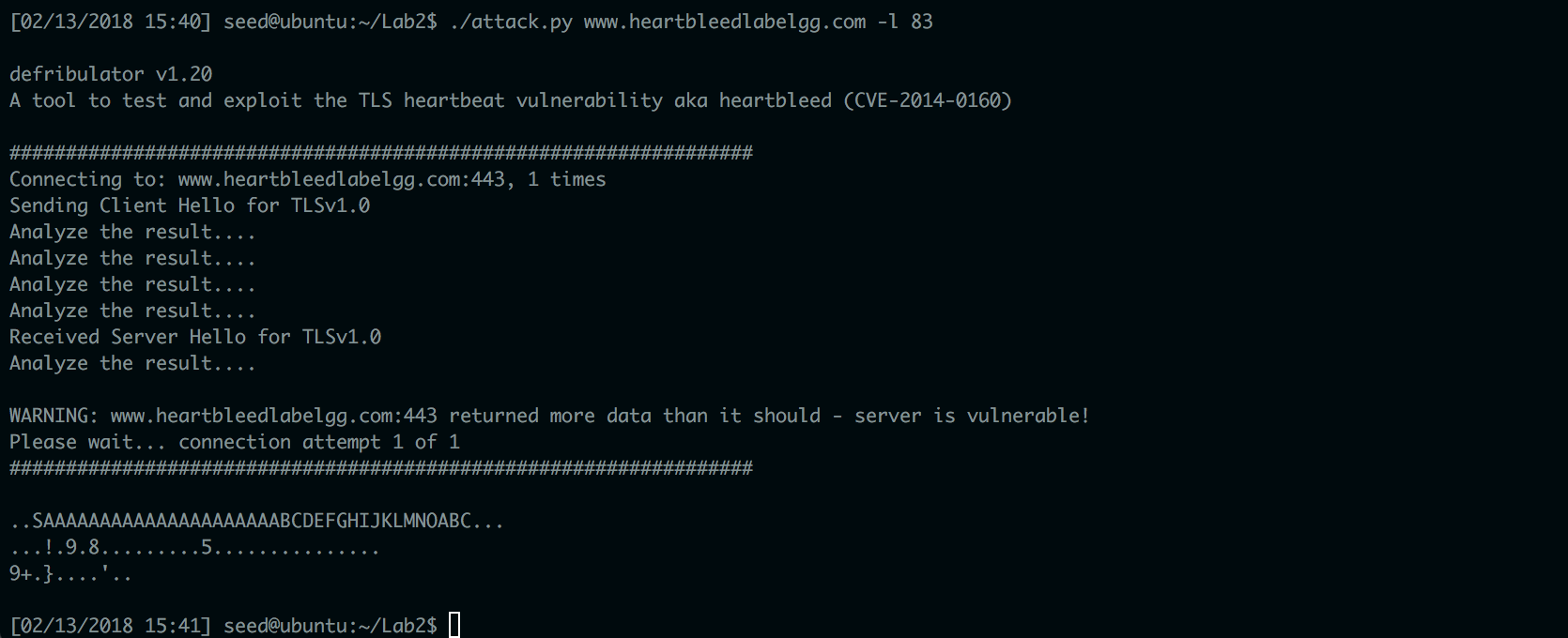
Heartbleed Attack Lab

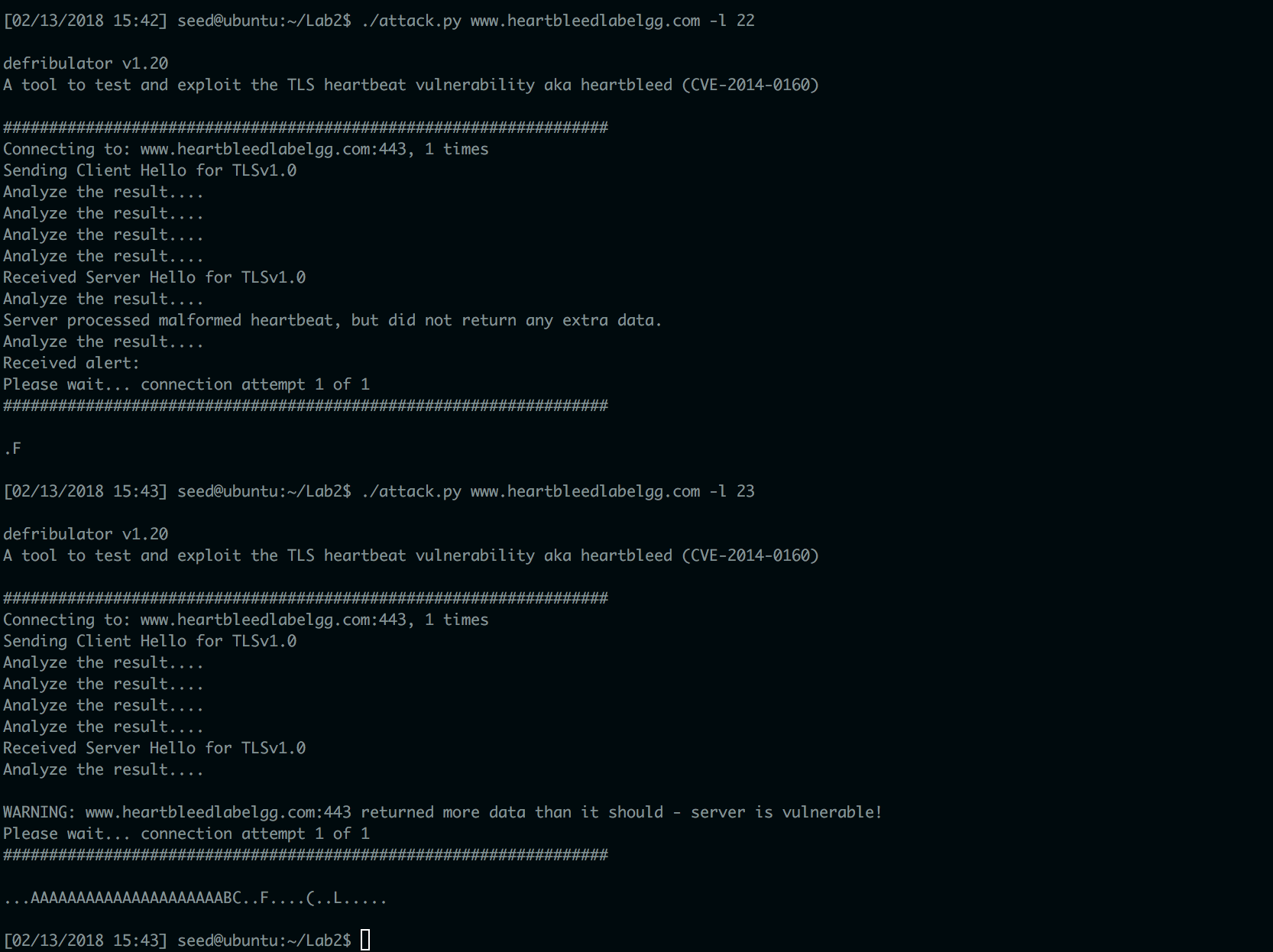
* Task 1:  
  Here are some screenshots documenting me logging into the website, adding Boby as a friend, and sending him a private message.  
  Logging in as admin:  
    
  Adding Boby as a friend:  
  

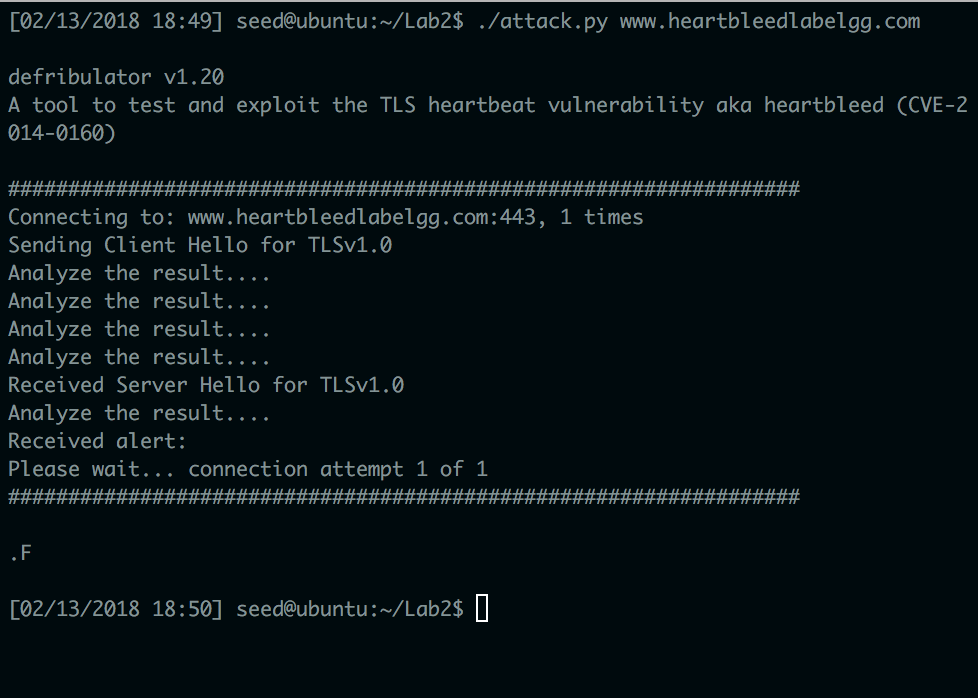
Sending Boby a private message:  
  
Now here are some screenshots documenting the use of attack.py to reveal the (supposedly) private info I gave to the server.  
This run of attack.py revealed the admin login information:  


This run of attack.py revealed the private message I sent Boby:  
  
  
Although I was able to reveal the login information and the private message information with attack.py, I could not reveal clearly that I had added Boby as a friend or any other activities that I had done as a the admin. Each heartbleed attack was carried out simply by running attack.py from the command line with no flags, and with www.heartbleedlabelgg.com as the target website. My observations are that the heartbleed attack is useful as long as it reveals data that is easily understandable outside of the context of the server (e.g. database information, message content, username/password pairs, etc).

* Task 2:  
  The following screenshots document me experimenting with different length values for the heartbleed attack.

Heartbleed attack with length 0x015B:  
Heartbleed attack with length 0x005B:  
  
Notice that there is less data output by the attack.  
  
Heartbleed attack with length 83:  
  
Now there is even less data that is output by attack.py

Heartbleed attacks with lengths 22 and 23:  
  
Question 2.1: As the length variable decreases, the amount of data leaked by the server lessens.  
Question 2.2: As indicated by the last screenshot above, the boundary appears to be 22.

* Task 3:
  + Task 3.1:  
    A screenshot of a run of attack.py after the OpenSSL library is updated to the latest version:  
    
  + Task 3.2:  
    In order to fix the code in Listing 1, the length of the payload buffer must be compared to the given payload int to ensure that they are equal before the buffer is copied and returned.  
      
    Regarding Alice, Bob, and Eva, Alice is slightly correct, Bob is correct, and Eva is wrong. Alice is slightly correct because while it is true that missing the boundary checking during the buffer copy is a cause of protected data being leaked, it is not the only fundamental reason why the heartbleed vulnerability is present in the code. Bob is completely correct in asserting the the user input needs to be verified, because the fundamental cause of the heartbleed vulnerability is trusting the data given by the users. Eva is wrong because length is an important piece of metadata that should be given with every request where data is uploaded.