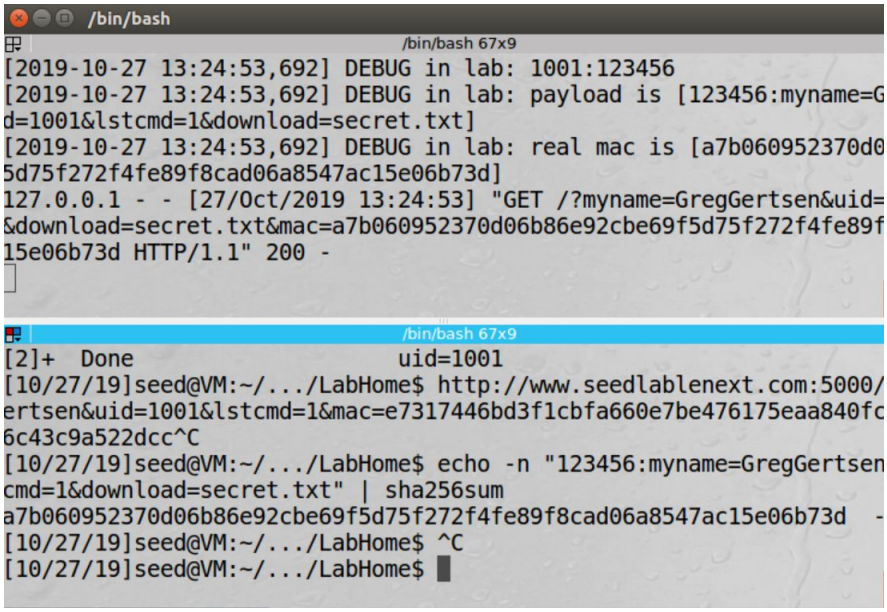
Greg Gertsen
Nazim Zerrouki
TCSS 481
Lab02

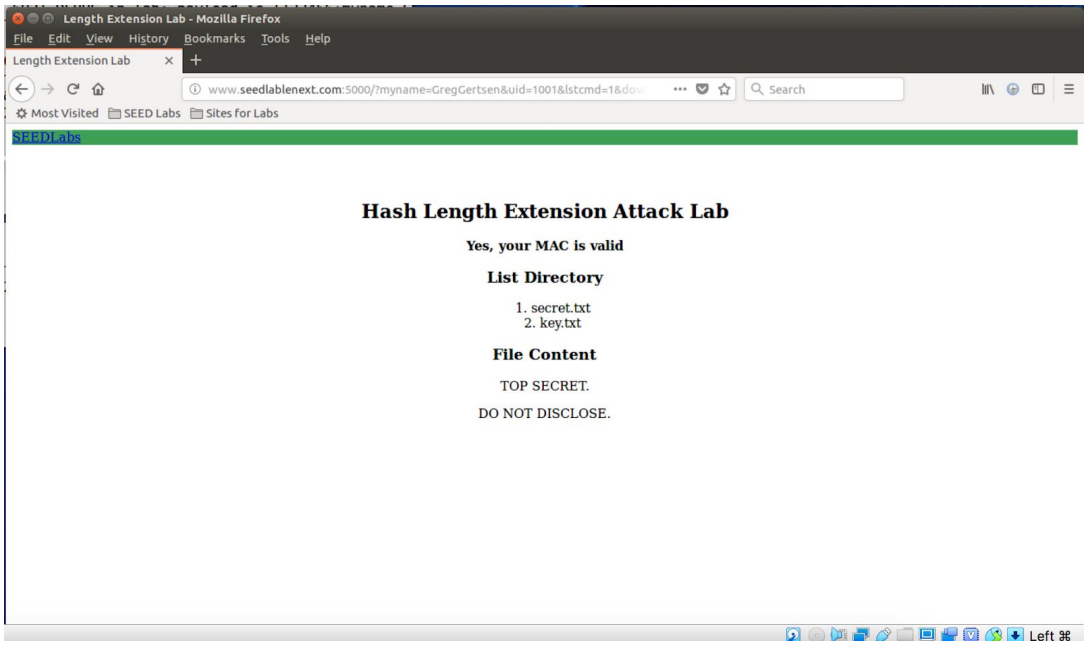
3. Tasks

3.1 Task 1: Send Request to List Files

Making the MAC



Download command and results



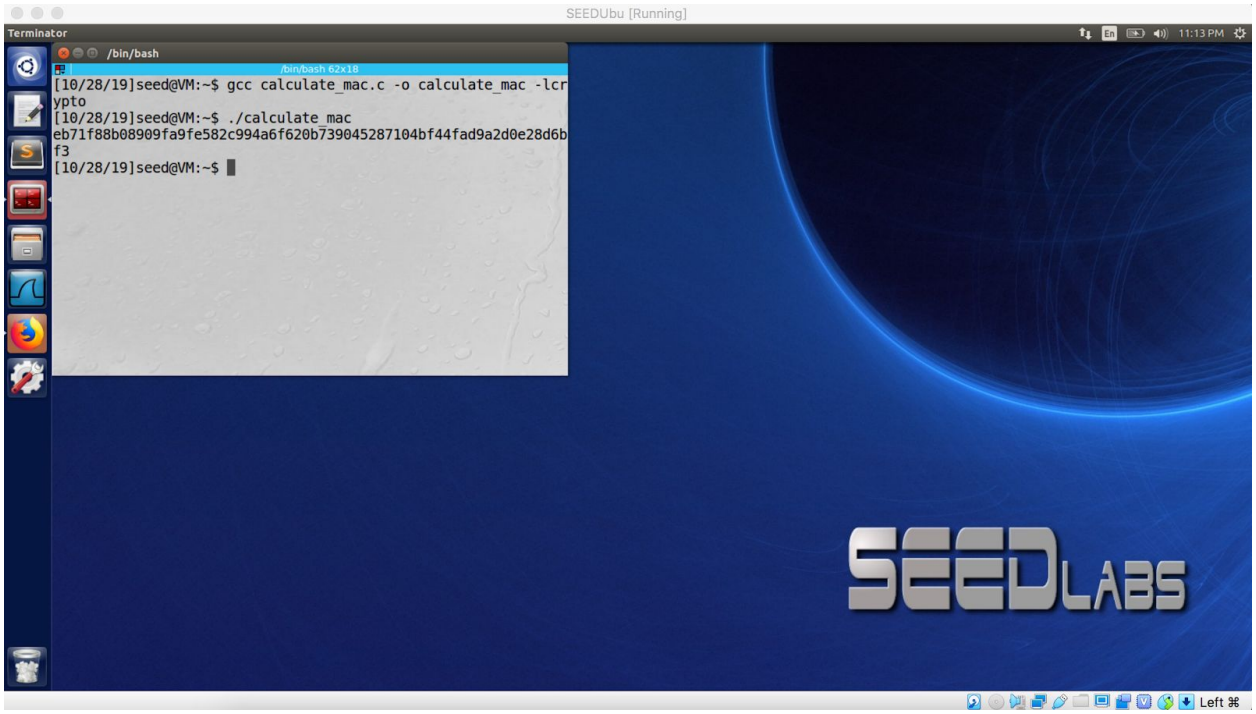
3.2 Task 2: Create Padding

Message to pad “123456:myname=GregGertsen&uid=1001&lstcmd=1”
Message is 43 bytes. 64-43 = 21 bytes. Length of message in bits = 43*8 = 344 bits. In hex = \x01 \x58. Padding is:

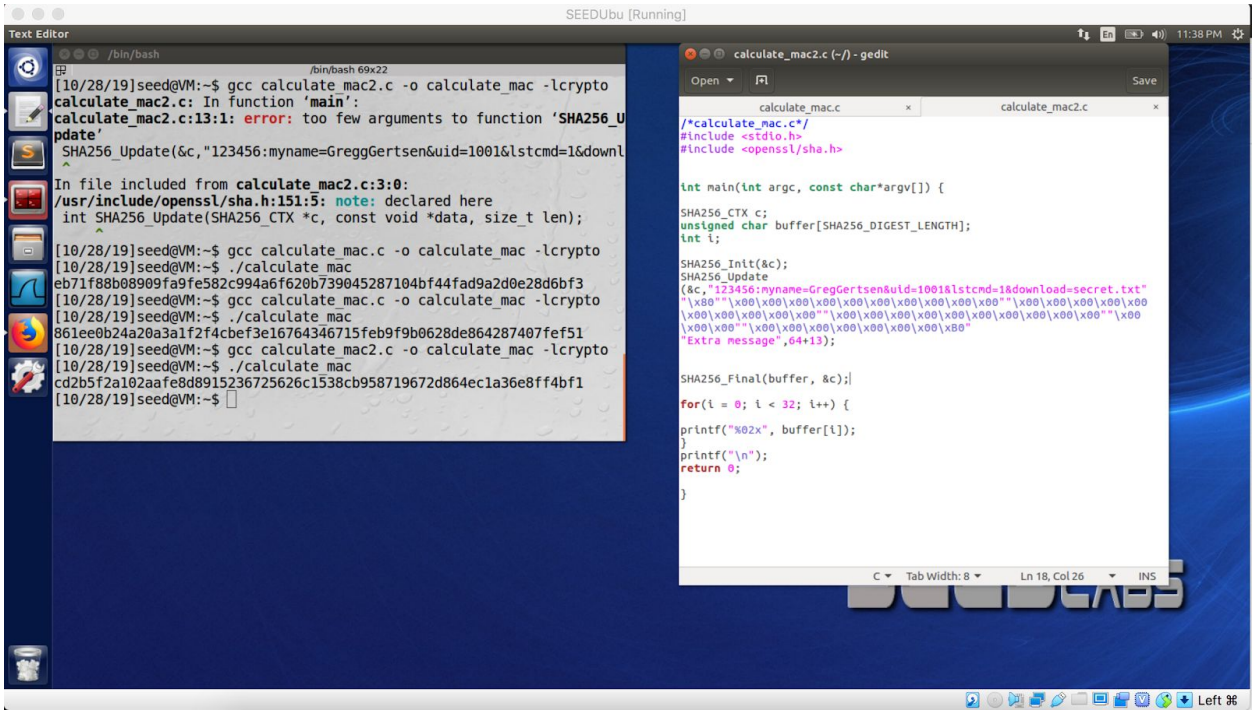
```
"\x80"
"\x00\x00\x00\x00\x00\x00\x00"
"\x00\x00\x00\x00\x00\x00\x00\x00"
"\x00\x00\x00\x01\x58"
```

3.3 Task 3: Compute MAC using Secret Key

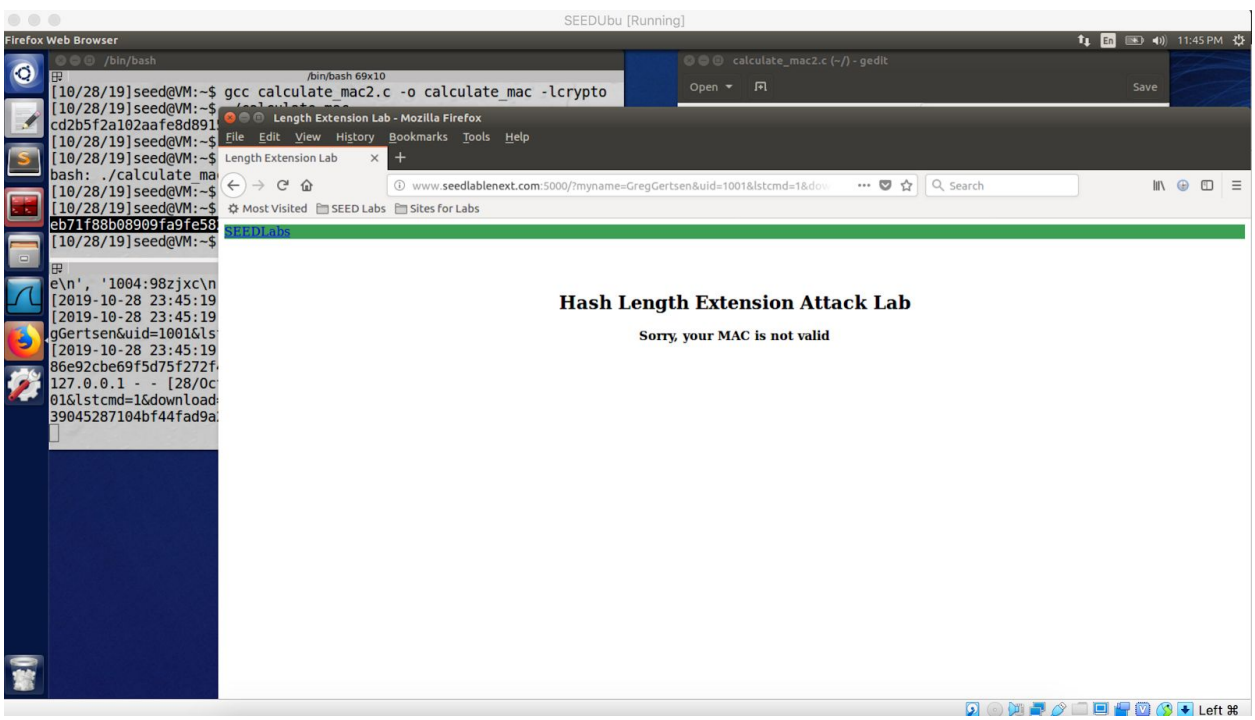
Compiled and ran calculate_mac.c



Changing the message and compiling

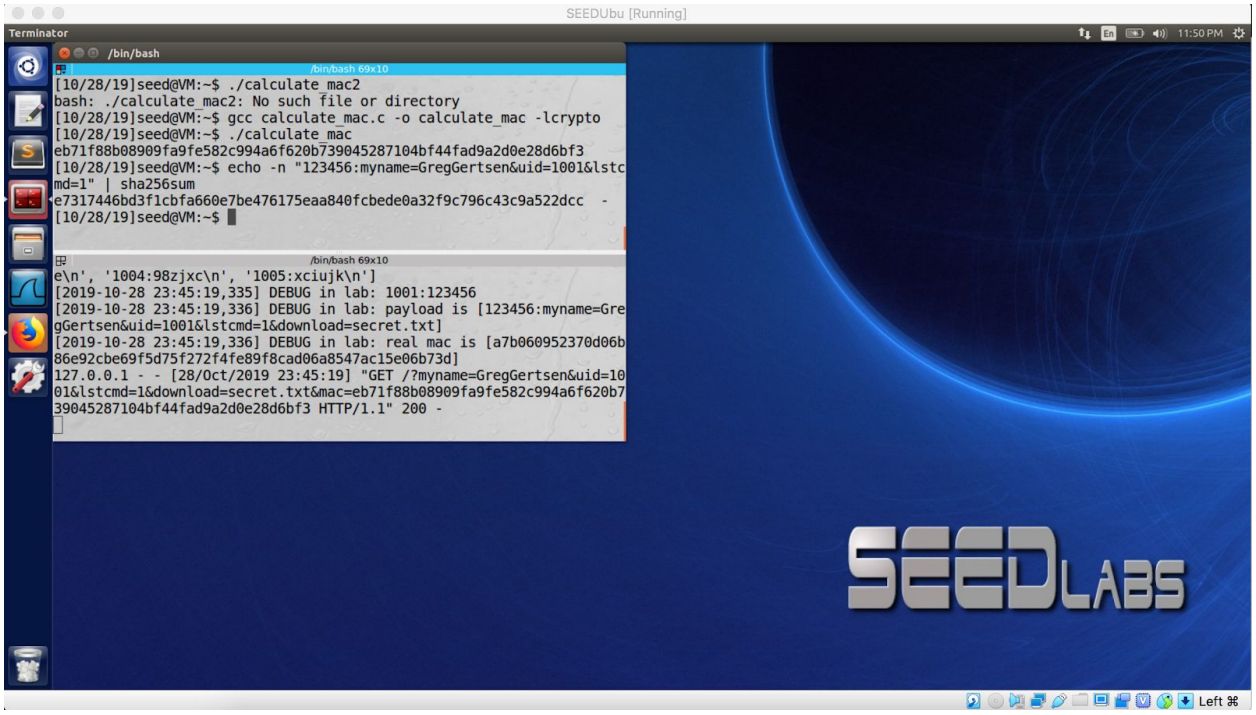


I was not able to connect, got the following message



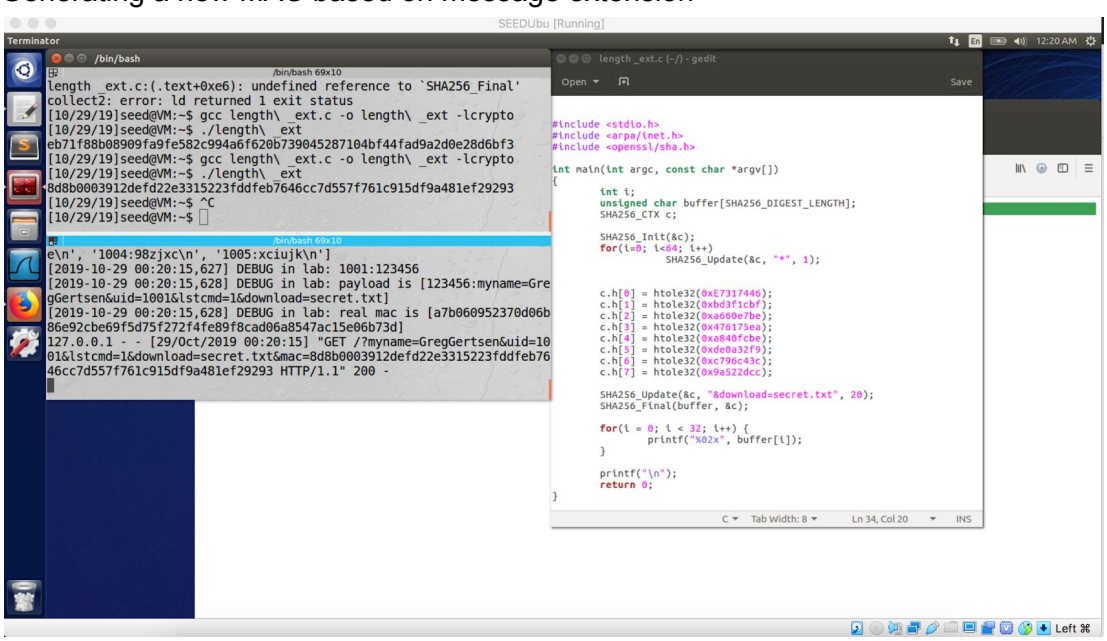
3.4 Task 4: The Length Extension Attack

Generating a valid MAC



Mac generated:
E7317446bd3f1cbfa660e7be476175eaa840fcbede0a32f9c796c43c9a522dcc

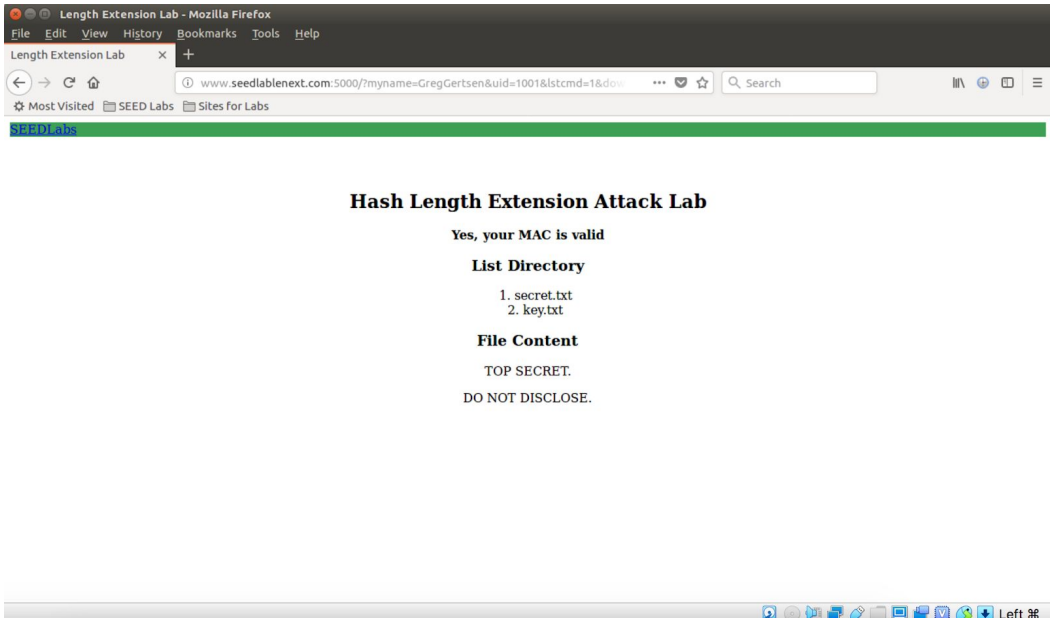
Generating a new MAC based on message extension



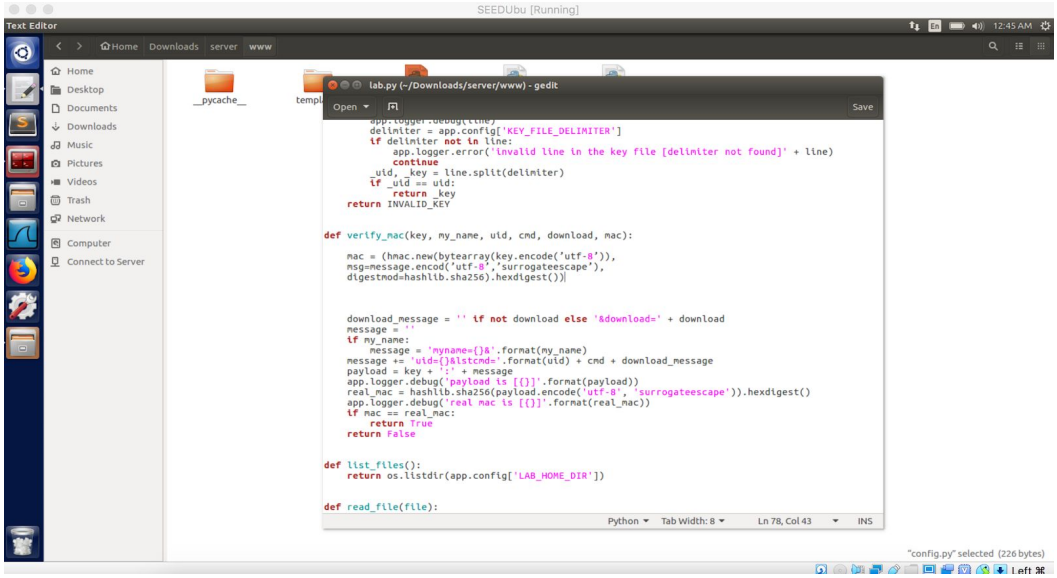
New MAC generated with addition of download command:

8d8b0003912defd22e3315223fddfeb7646cc7d557f761c915df9a481ef29293

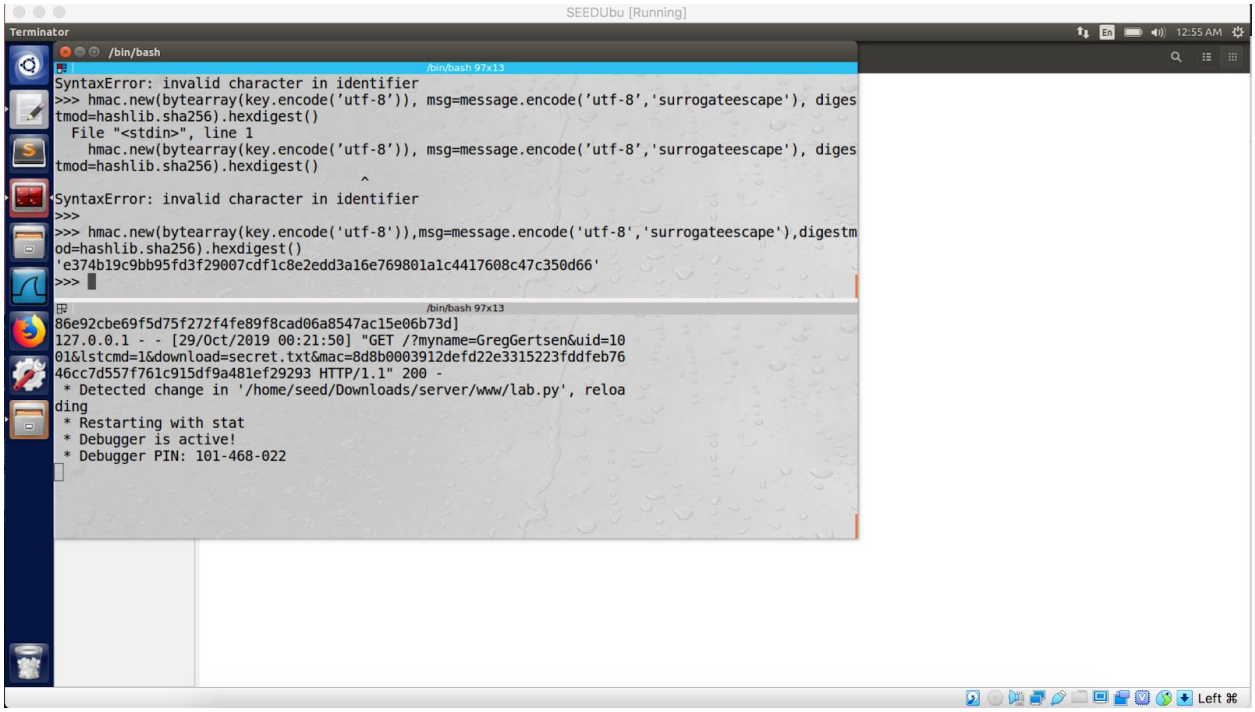
Sent to the server and was able to read secret.txt



3.5 Task 5: Attack Mitigation using HMAC
Modifying server program's verify_mac() function



Was able to generate using python HMAC



HMAC is not prone to length extension attack. It computes the hash twice. A secret key is used initially to make two other keys which are referred to as inner and outer. The first time through, the algorithm will make an internal hash which comes from the message and the first key. The second time through the algorithm, the HMAC is made from inner hash result and the outer key. This makes it immune to length extension attacls.