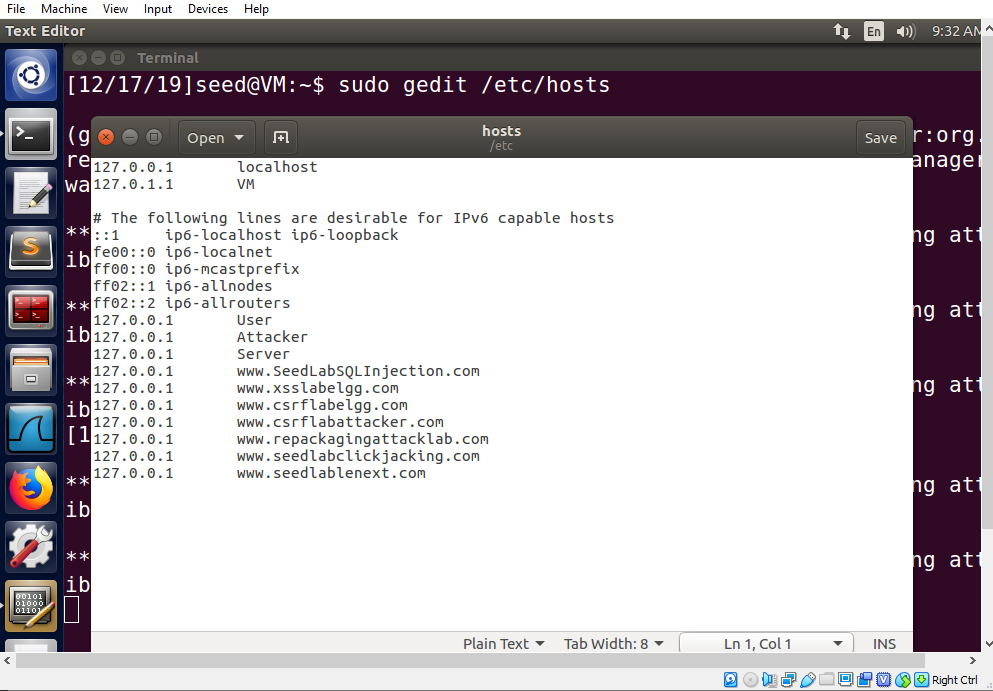
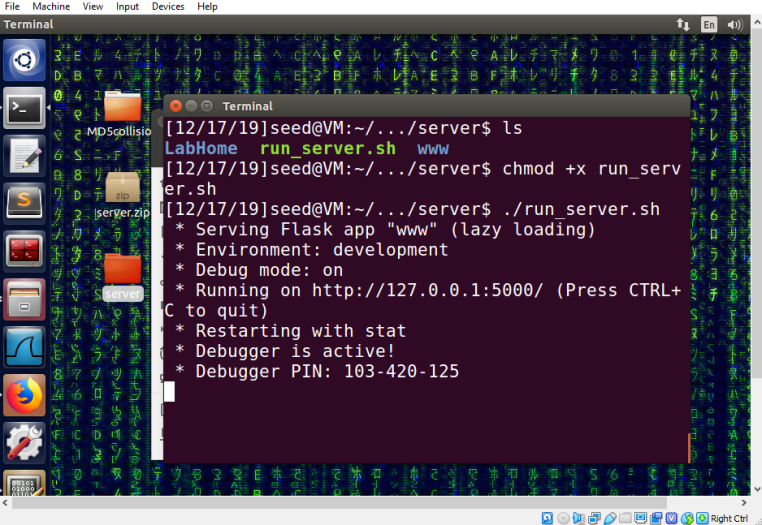
**Hash Length Extension Attack Lab**

Liangyu W

Add seedlablenext to the /etc/hosts file:



Start the server program

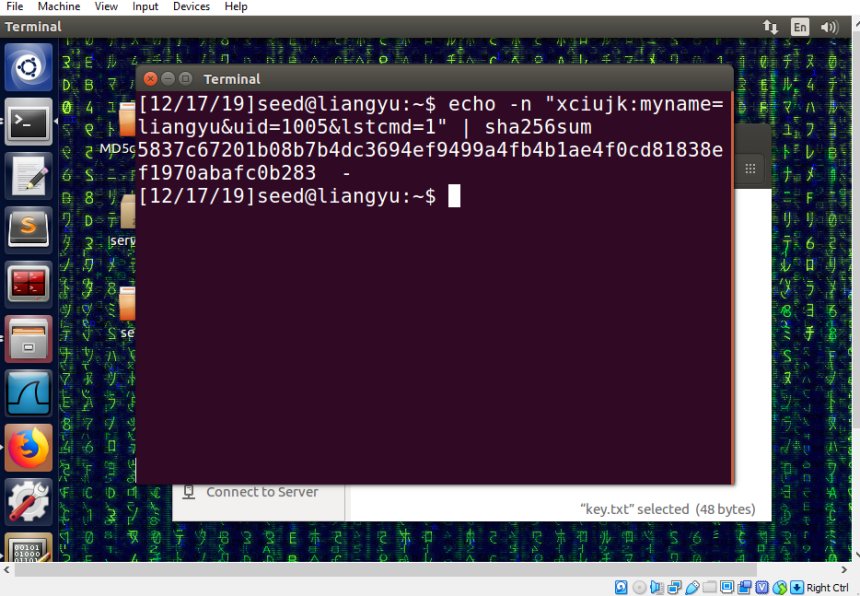


**Task 1: Send Request to List Files**

We choose uid 1005 and its corresponding key value xciujk from the LabHome directory.

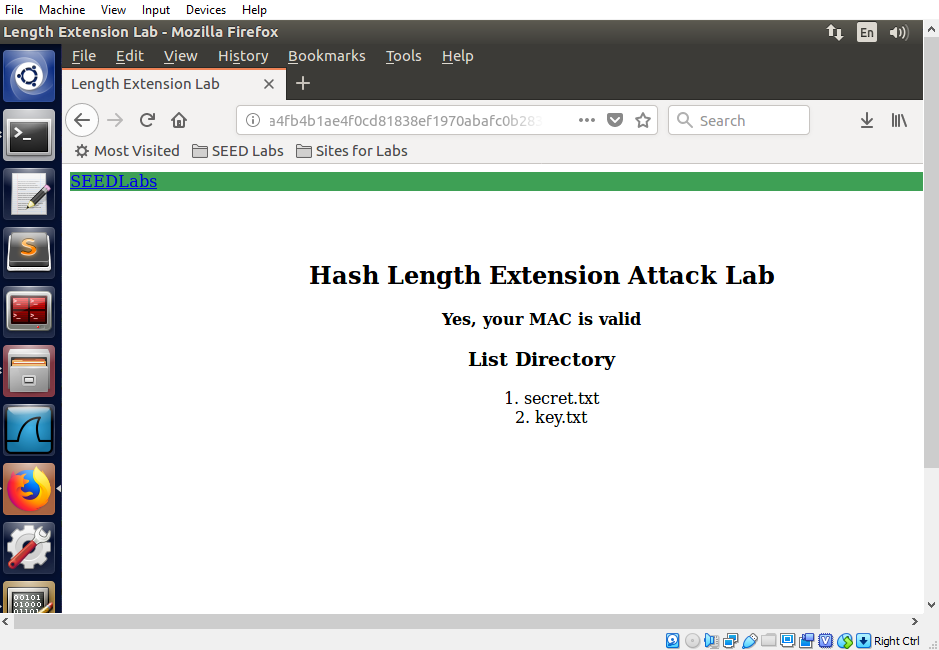
Generate a MAC using the command:

echo -n "xciujk:myname=liangyu&uid=1005&lstcmd=1" | sha256sum



We send the following request to the server:

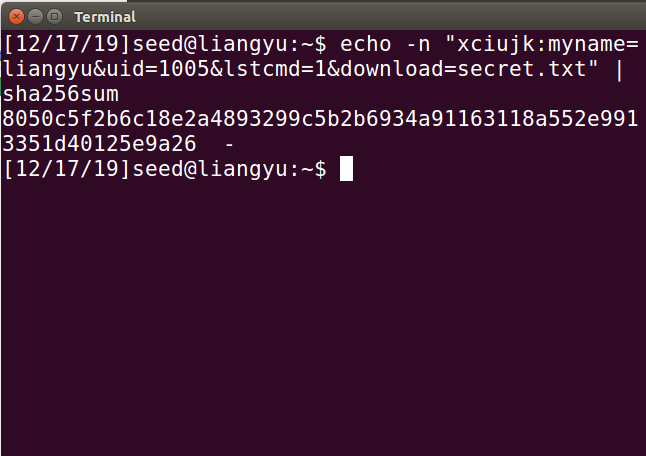
http://www.seedlablenext.com:5000/?myname=liangyu&uid=1005&lstcmd=1&mac=5837c67201b08b7b4dc3694ef9499a4fb4b1ae4f0cd81838ef1970abafc0b283

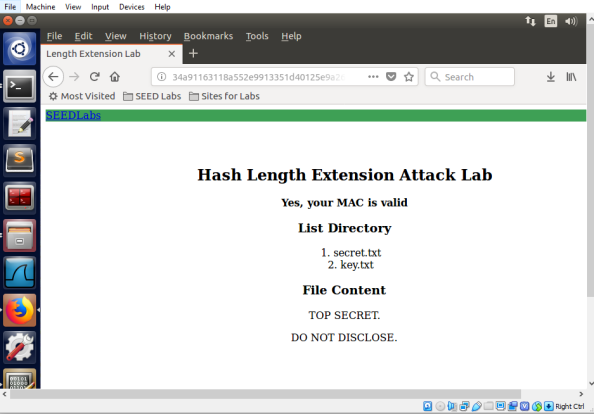


The server verifies the MAC and lists the files in its directory.

Similarly, we use the following request to download file from the server:

http://www.seedlablenext.com:5000/?myname=liangyu&uid=1005&lstcmd=1&download=secret.txt&mac=8050c5f2b6c18e2a4893299c5b2b6934a91163118a552e9913351d40125e9a26





**Task 2: Create Padding**

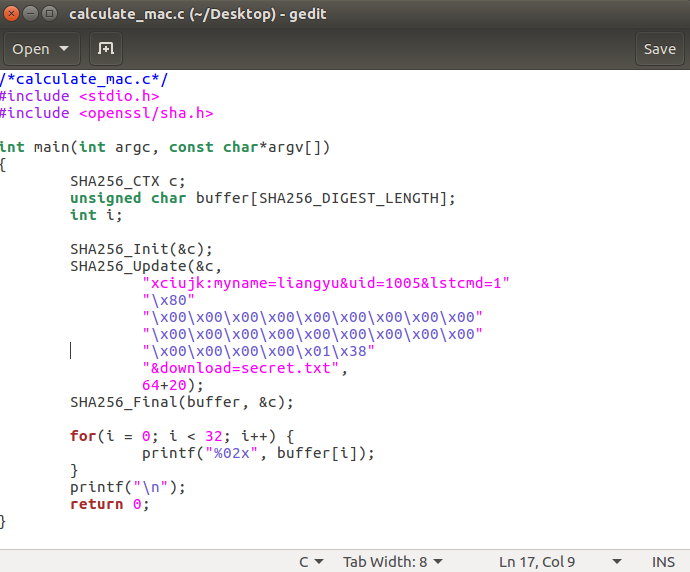
The message “xciujk:myname=liangyu&uid=1005&lstcmd=1” has a length of 39. So 25 bytes need to be padded.

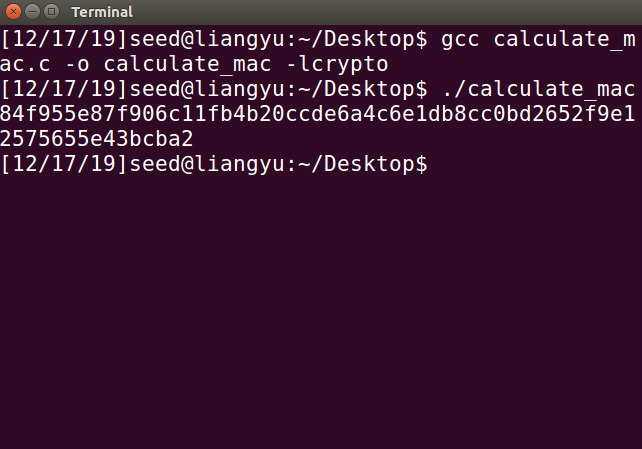
Thus for “xciujk:myname=liangyu&uid=1005&lstcmd=1”, the padding is:

“\x80\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x27”

**Task 3: Compute MAC using Secret Key**

We create the following C program:

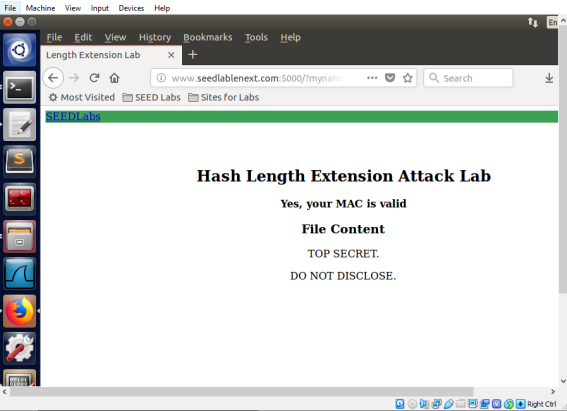




Compile and running the program generates our MAC.

We send the following request to the server:

http://www.seedlablenext.com:5000/?myname=liangyu&uid=1005&lstcmd=1%80%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%01%38&download=secret.txt&mac=84f955e87f906c11fb4b20ccde6a4c6e1db8cc0bd2652f9e12575655e43bcba2



**Task 4: The Length Extension Attack**

We previously generated the valid MAC

5837c67201b08b7b4dc3694ef9499a4fb4b1ae4f0cd81838ef1970abafc0b283

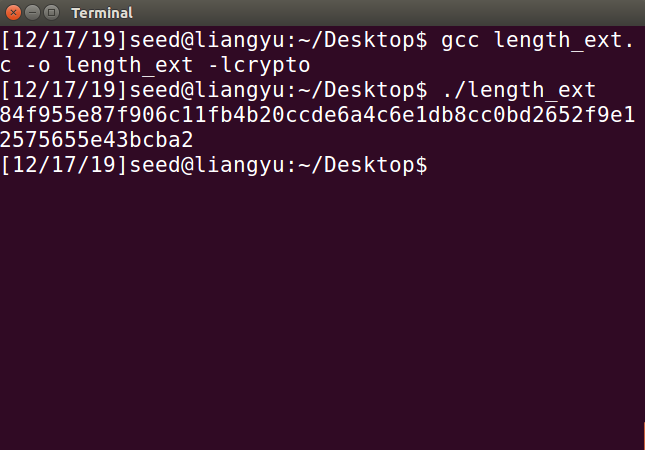
For the request http://www.seedlablenext.com:5000/?myname=liangyu&uid=1005&lstcmd=1

We create the following length\_ext.c program, it computes a new MAC based on the previously generated MAC and the added message:



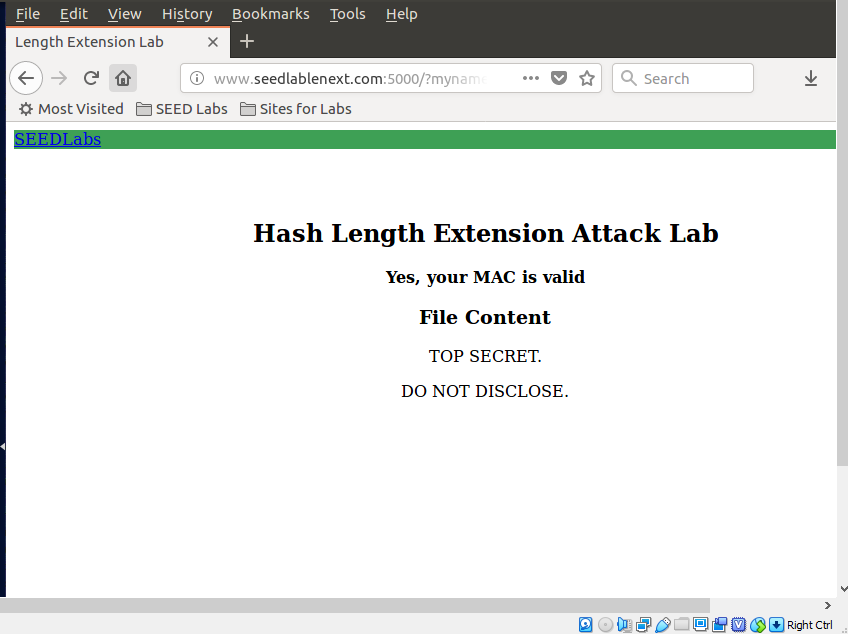
Compiling and running the program gives us the new MAC:

84f955e87f906c11fb4b20ccde6a4c6e1db8cc0bd2652f9e12575655e43bcba2



Using the new MAC we construct the following request:

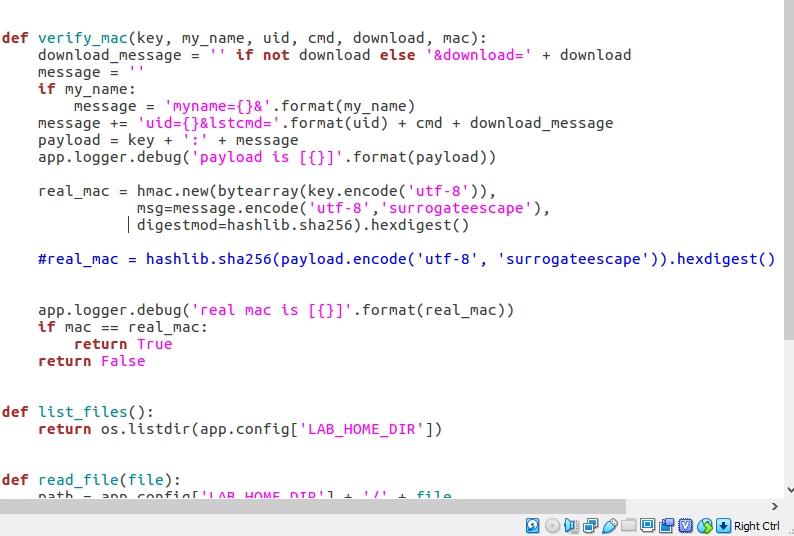
http://www.seedlablenext.com:5000/?myname=liangyu&uid=1005&lstcmd=1%80%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%01%38&download=secret.txt&mac=84f955e87f906c11fb4b20ccde6a4c6e1db8cc0bd2652f9e12575655e43bcba2



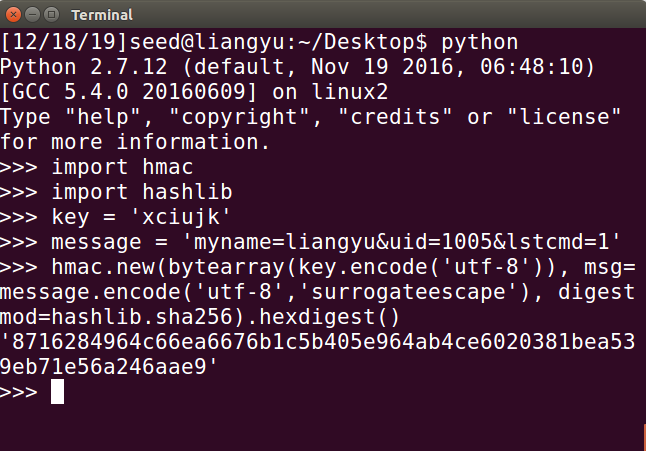
This allows us to successfully download the content of the file without knowing the secret key. But in order to construct the padding, we had to know the length of the secret key.

**Task 5: Attack Mitigation using HMAC**

We modify the server program’s verify\_mac() function as follows:

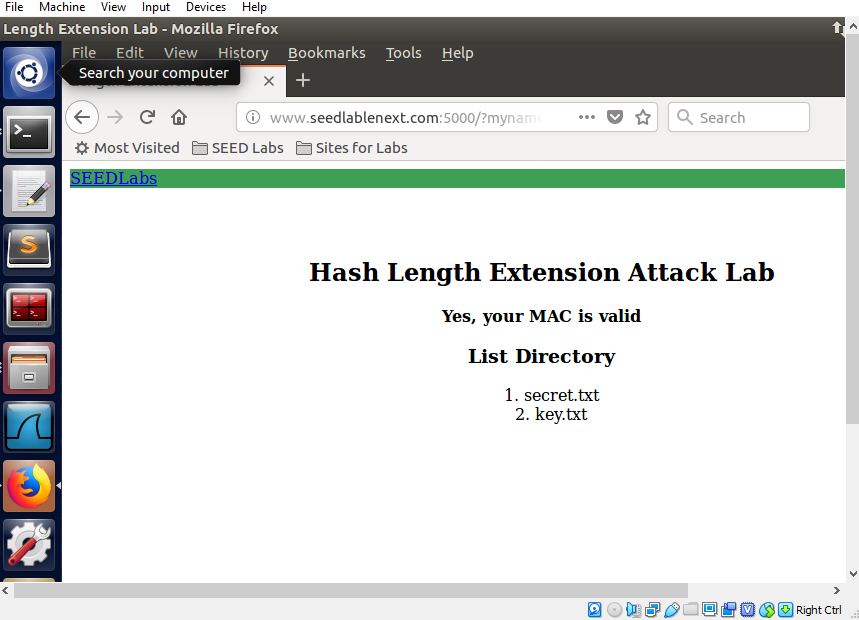


The hash calculated using HMAC is 8716284964c66ea6676b1c5b405e964ab4ce6020381bea539eb71e56a246aae9



Construct the new request using HMAC as follows:

http://www.seedlablenext.com:5000/?myname=liangyu&uid=1005&lstcmd=1&mac=8716284964c66ea6676b1c5b405e964ab4ce6020381bea539eb71e56a246aae9



Server successfully verifies the new HMAC.

The original MAC is calculated by hashing the key concatenated to the message:

MAC = hash(key + message), HMAC is calculated using two rounds of hashing:

HMAC = hash(key + hash(key + message)). This prevents the length extension attack demonstrated above.