Applied Cryptography (UE20CS314)

Hash Length Extension

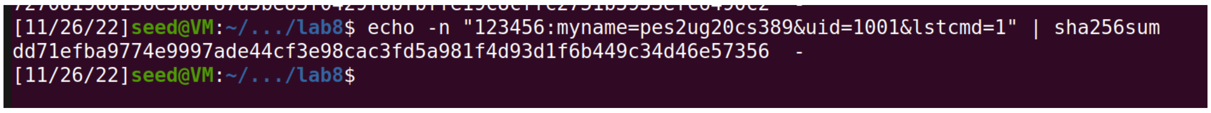
Name: Vishwa Mehul Mehta

SRN: PES2UG20CS389

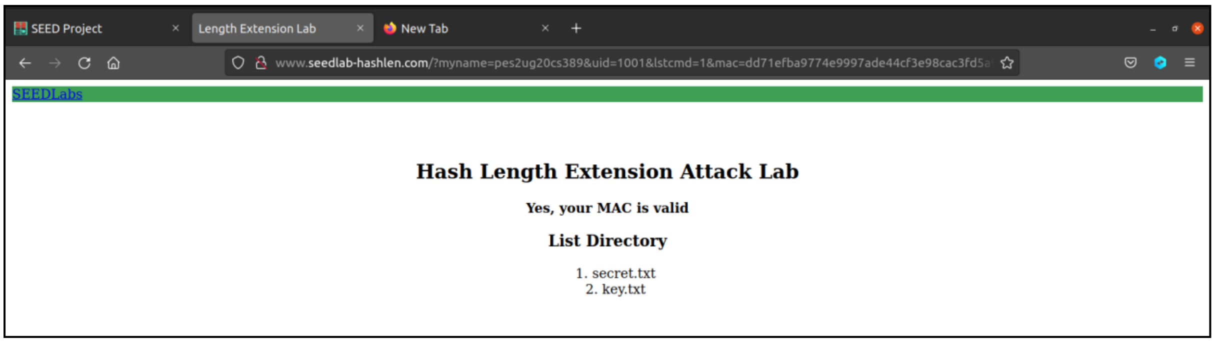
Section: F

Task 1:

Screenshot:



```
[11/26/22]seed@VM:~/.../lab8$ echo -n "123456:myname=pes2ug20cs389&uid=1001&lstcmd=1" | sha256sum
dd71efba9774e9997ade44cf3e98cac3fd5a981f4d93d1f6b449c34d46e57356  -
[11/26/22]seed@VM:~/.../lab8$
```



www.seedlab-hashlen.com/?myname=pes2ug20cs389&uid=1001&lstcmd=1&mac=dd71efba9774e9997ade44cf3e98cac3fd5a

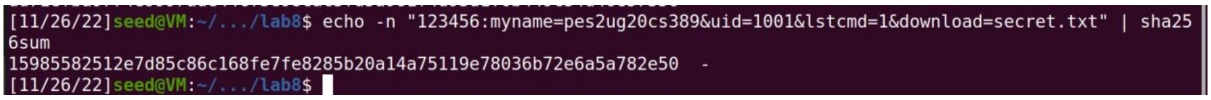
SEEDLabs

Hash Length Extension Attack Lab

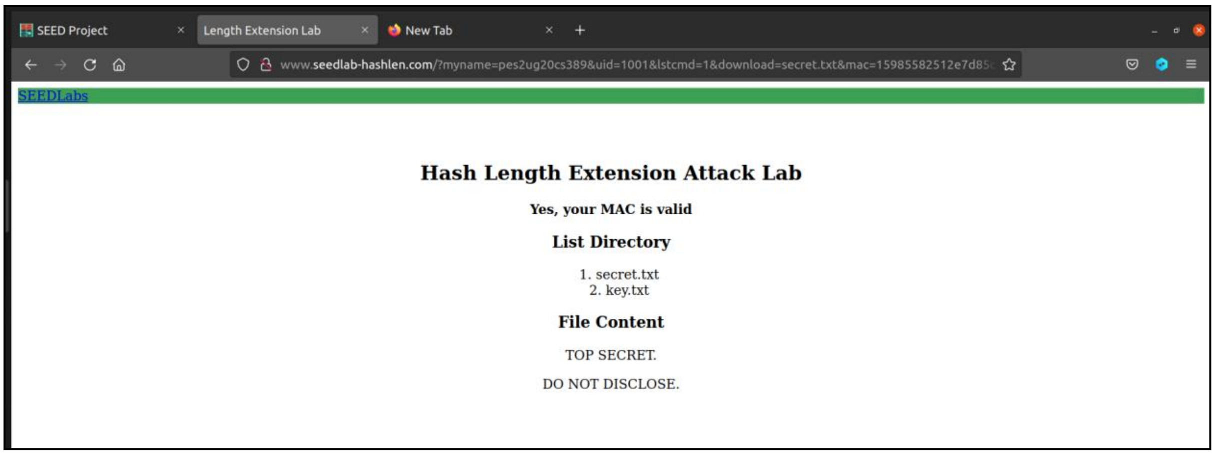
Yes, your MAC is valid

List Directory

1. secret.txt
2. key.txt



```
[11/26/22]seed@VM:~/.../lab8$ echo -n "123456:myname=pes2ug20cs389&uid=1001&lstcmd=1&download=secret.txt" | sha25
6sum
15985582512e7d85c86c168fe7fe8285b20a14a75119e78036b72e6a5a782e50  -
[11/26/22]seed@VM:~/.../lab8$
```



www.seedlab-hashlen.com/?myname=pes2ug20cs389&uid=1001&lstcmd=1&download=secret.txt&mac=15985582512e7d85

SEEDLabs

Hash Length Extension Attack Lab

Yes, your MAC is valid

List Directory

1. secret.txt
2. key.txt

File Content

TOP SECRET.

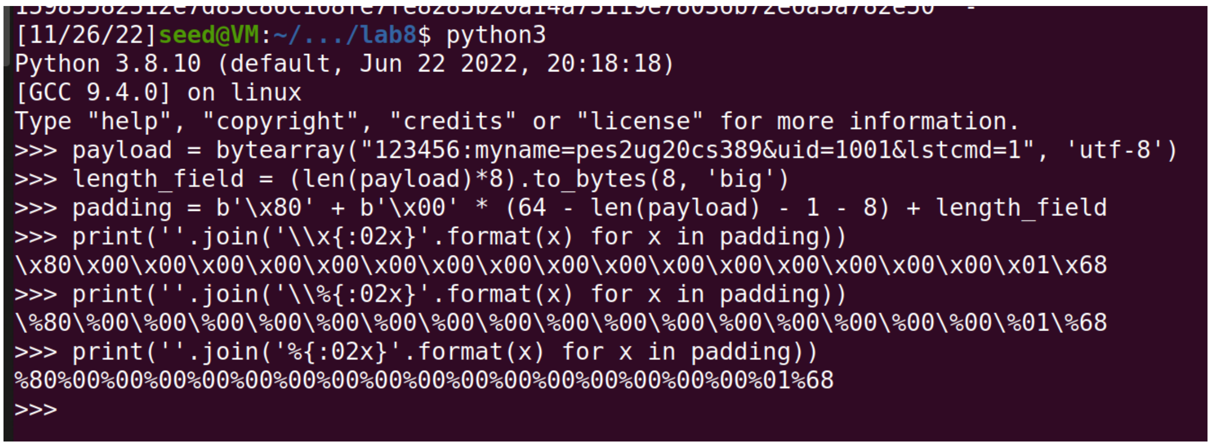
DO NOT DISCLOSE.

Observation:

We get the contents of the file secret.txt along with its contents and also list the directory with key.txt and secret.txt.

Task 2:

Screenshot:



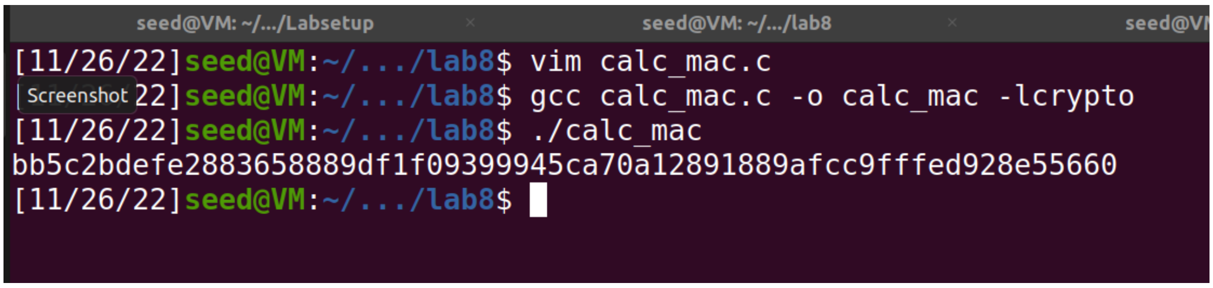
```
[11/26/22]seed@VM:~/.../lab8$ python3
Python 3.8.10 (default, Jun 22 2022, 20:18:18)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> payload = bytearray("123456:myname=pes2ug20cs389&uid=1001&lstcmd=1", 'utf-8')
>>> length_field = (len(payload)*8).to_bytes(8, 'big')
>>> padding = b'\x80' + b'\x00' * (64 - len(payload) - 1 - 8) + length_field
>>> print(''.join('\\x{:02x}'.format(x) for x in padding))
\x80\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01\x68
>>> print(''.join('\\%{:02x}'.format(x) for x in padding))
\%80\%00\%00\%00\%00\%00\%00\%00\%00\%00\%00\%00\%00\%00\%00\%00\%00\%01\%68
>>> print(''.join('%{:02x}'.format(x) for x in padding))
%80%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%01%68
>>>
```

Observation:

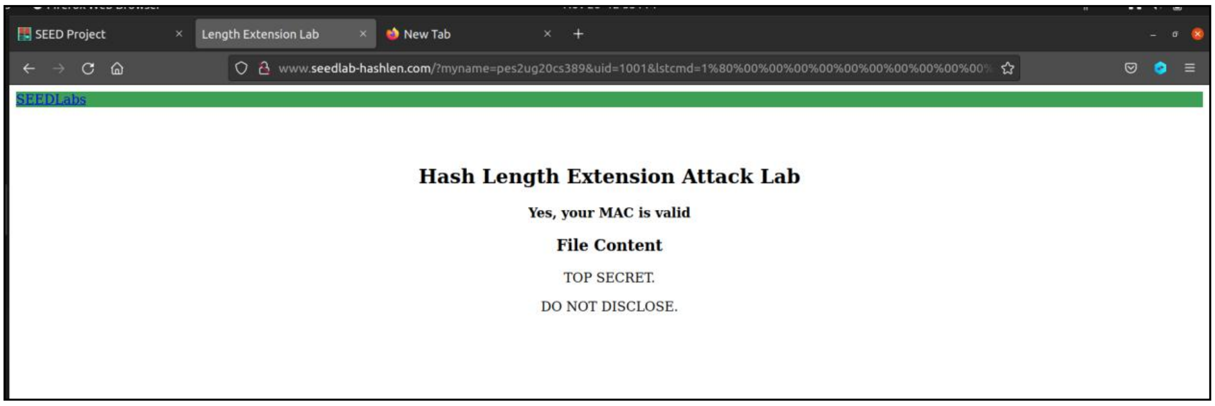
We generate the url and hash padding.

Task 3:

Screenshot:



```
[11/26/22]seed@VM:~/.../lab8$ vim calc_mac.c
[11/26/22]seed@VM:~/.../lab8$ gcc calc_mac.c -o calc_mac -lcrypto
[11/26/22]seed@VM:~/.../lab8$ ./calc_mac
bb5c2bdefe2883658889df1f09399945ca70a12891889afcc9fffed928e55660
[11/26/22]seed@VM:~/.../lab8$
```



www.seedlab-hashlen.com/?myname=pes2ug20cs389&uid=1001&lstcmd=1%80%00%00%00%00%00%00%00%00%00%00

SEEDLabs

Hash Length Extension Attack Lab

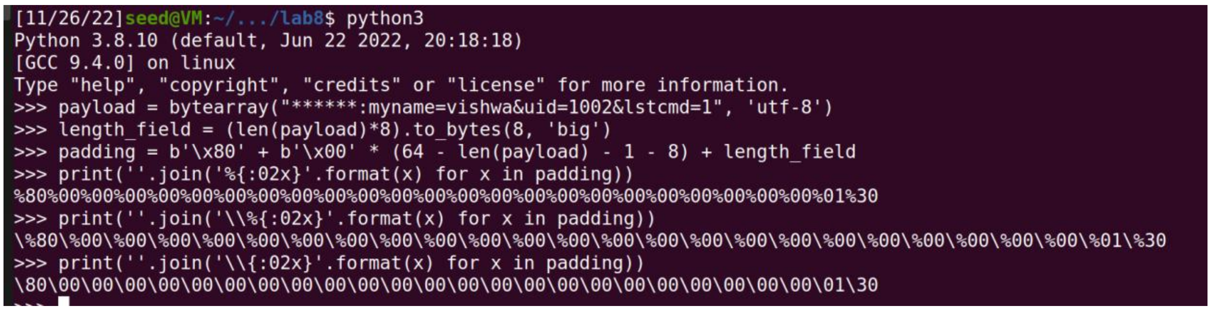
Yes, your MAC is valid

File Content

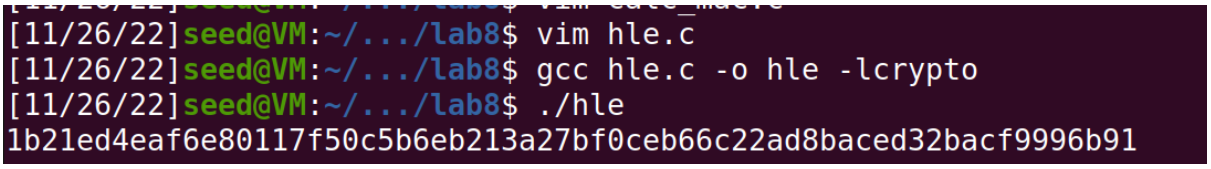
TOP SECRET.

DO NOT DISCLOSE.

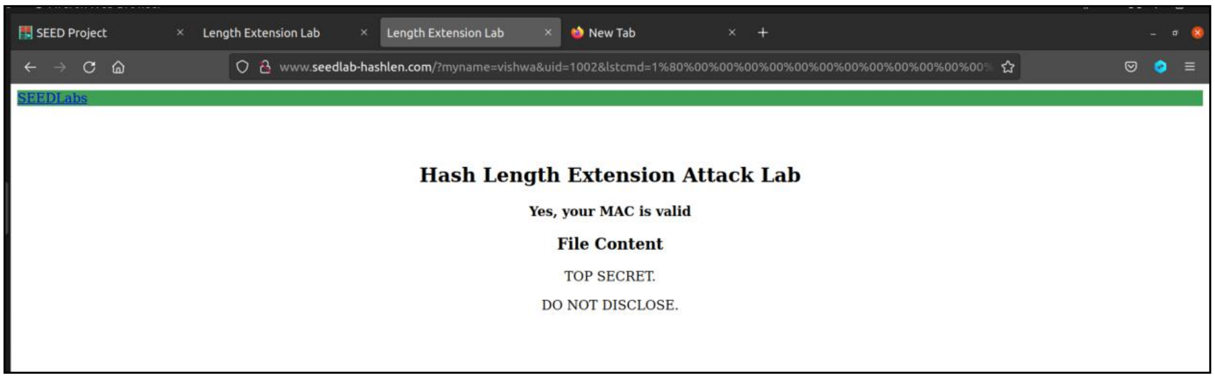
Without key:



```
[11/26/22]seed@VM:~/.../lab8$ python3
Python 3.8.10 (default, Jun 22 2022, 20:18:18)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> payload = bytearray("******:myname=vishwa&uid=1002&lstcmd=1", 'utf-8')
>>> length_field = (len(payload)*8).to_bytes(8, 'big')
>>> padding = b'\x80' + b'\x00' * (64 - len(payload) - 1 - 8) + length_field
>>> print(''.join('%{:02x}'.format(x) for x in padding))
%80%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%01%30
>>> print(''.join('\\%{:02x}'.format(x) for x in padding))
\%80\%00\%00\%00\%00\%00\%00\%00\%00\%00\%00\%00\%00\%00\%00\%00\%00\%00\%00\%00\%00\%00\%00\%00\%01\%30
>>> print(''.join('\\{:02x}'.format(x) for x in padding))
\80\00\00\00\00\00\00\00\00\00\00\00\00\00\00\00\00\00\00\00\00\00\00\00\01\30
>>> 
```



```
[11/26/22]seed@VM:~/.../lab8$ vim hle.c
[11/26/22]seed@VM:~/.../lab8$ gcc hle.c -o hle -lcrypto
[11/26/22]seed@VM:~/.../lab8$ ./hle
1b21ed4eaf6e80117f50c5b6eb213a27bf0ceb66c22ad8baced32bacf9996b91
```



SEED Project
Length Extension Lab
Length Extension Lab
New Tab
www.seedlab-hashlen.com/?myname=vishwa&uid=1002&lstcmd=1%80%00%00%00%00%00%00%00%00%00%00%00%00
SEEDLabs

Hash Length Extension Attack Lab

Yes, your MAC is valid

File Content

TOP SECRET.

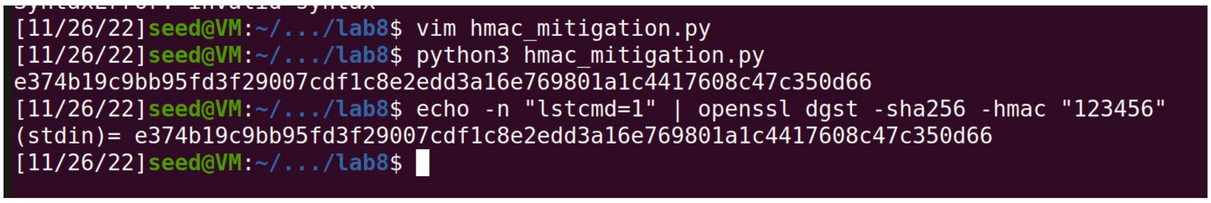
DO NOT DISCLOSE.

Observation:

The attack is successful and we can see the contents of the file secret.txt even though it is not a valid MAC address. We do both the attacks with the key and without it.

Task 4:

Screenshot:



```
[11/26/22]seed@VM:~/.../lab8$ vim hmac_mitigation.py
[11/26/22]seed@VM:~/.../lab8$ python3 hmac_mitigation.py
e374b19c9bb95fd3f29007cdf1c8e2edd3a16e769801a1c4417608c47c350d66
[11/26/22]seed@VM:~/.../lab8$ echo -n "lstcmd=1" | openssl dgst -sha256 -hmac "123456"
(stdin)= e374b19c9bb95fd3f29007cdf1c8e2edd3a16e769801a1c4417608c47c350d66
[11/26/22]seed@VM:~/.../lab8$ 
```

Observation:

We use hmac to avoid the length extension attack as the mac generated will not be same for the extended length and the original address without padding and will thus fail to authenticate the address.