Applied Cryptography

Lab 6: Hash Length Extension Attack Lab

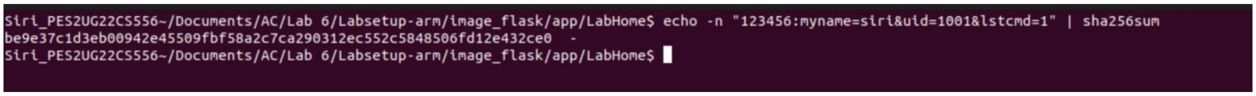
Name: Siri N Shetty SRN: PES2UG22CS556

Problem 1: Send Request to List Files

Step 1 : Finding UID.
Step 2 : Calculating mac command
Step 3 : Sending the request
Step 4 : Cmd with download secret.txt
Step 5 : Sending request for download secret.txt

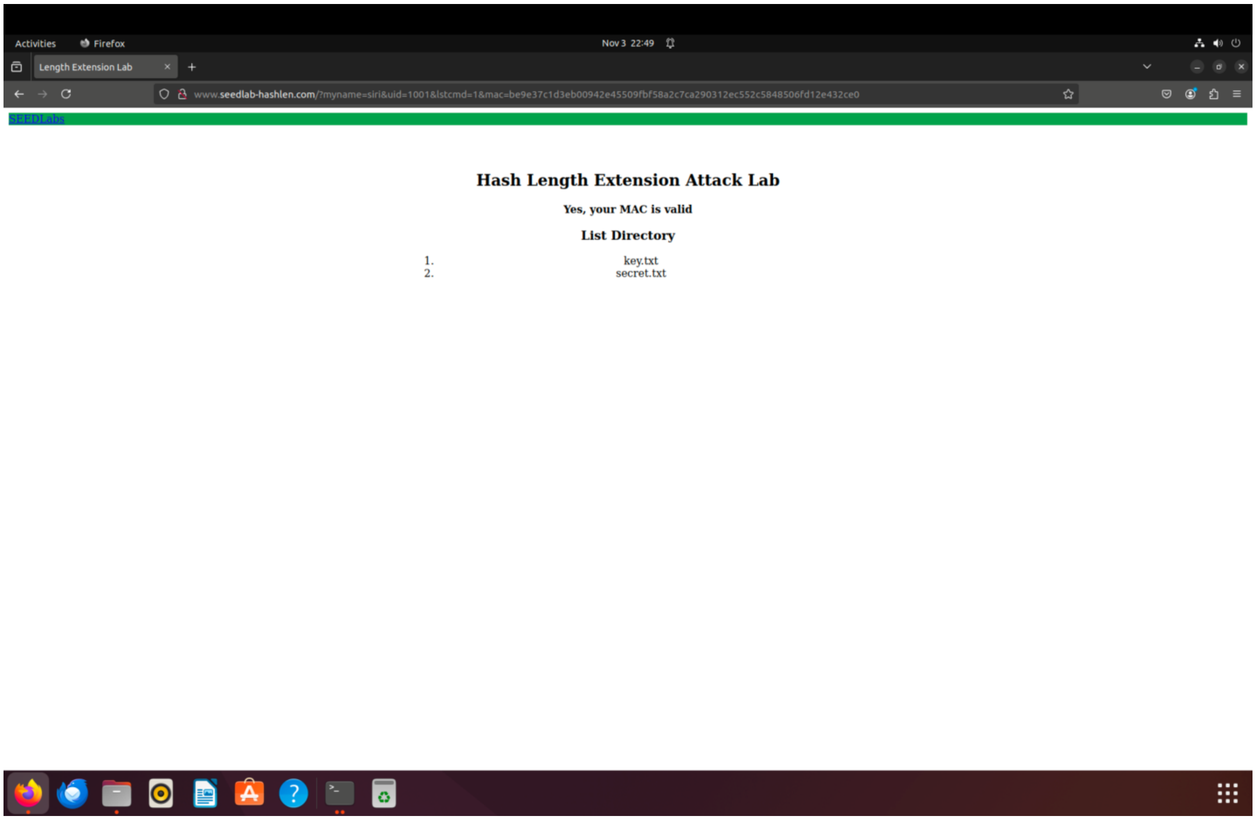
Expected Deliverables -

i) Code Output Screenshot for step 2



```
Siri_PES2UG22CS556~/Documents/AC/Lab 6/Labsetup-arm/image_flask/app/LabHome$ echo -n "123456:myname=siri&uid=1001&lstcmd=1" | sha256sum
be9e37c1d3eb00942e45509fbf58a2c7ca290312ec552c5848506fd12e432ce0  -
Siri_PES2UG22CS556~/Documents/AC/Lab 6/Labsetup-arm/image_flask/app/LabHome$ 
```

ii) Code Output Screenshot for step 3



www.seedlab-hashlen.com/?myname=siri&uid=1001&lstcmd=1&mac=be9e37c1d3eb00942e45509fbf58a2c7ca290312ec552c5848506fd12e432ce0

SEEDLabs

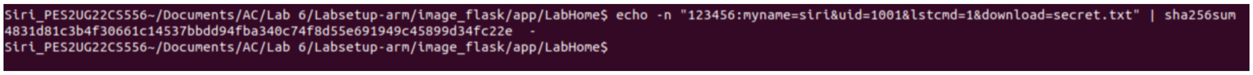
Hash Length Extension Attack Lab

Yes, your MAC is valid

List Directory

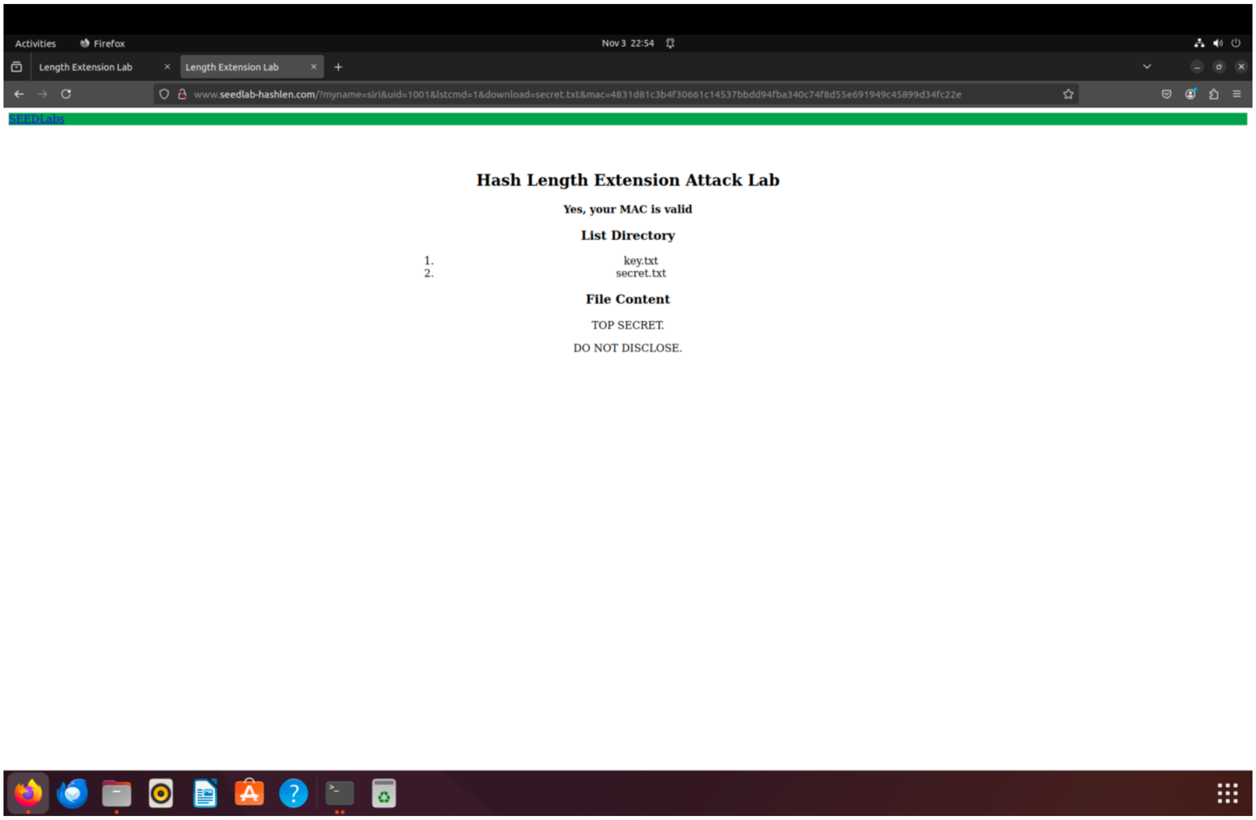
1. key.txt
2. secret.txt

iii) Code Output Screenshot for step 4



```
Siri_PES2UG22CS556~/Documents/AC/Lab 6/Labsetup-arm/image_flask/app/LabHome$ echo -n "123456:myname=siri&uid=1001&lstcmd=1&download=secret.txt" | sha256sum
4831d81c3b4f30661c14537bbdd94fba340c74f8d55e691949c45899d34fc22e  -
Siri_PES2UG22CS556~/Documents/AC/Lab 6/Labsetup-arm/image_flask/app/LabHome$
```

iv) Code Output Screenshot for step 5



www.seedlab-hashlen.com/?myname=siri&uid=1001&lstcmd=1&download=secret.txt&mac=4831d81c3b4f30661c14537bbdd94fba340c74f8d55e691949c45899d34fc22e

Hash Length Extension Attack Lab

Yes, your MAC is valid

List Directory

1. key.txt
2. secret.txt

File Content

TOP SECRET.

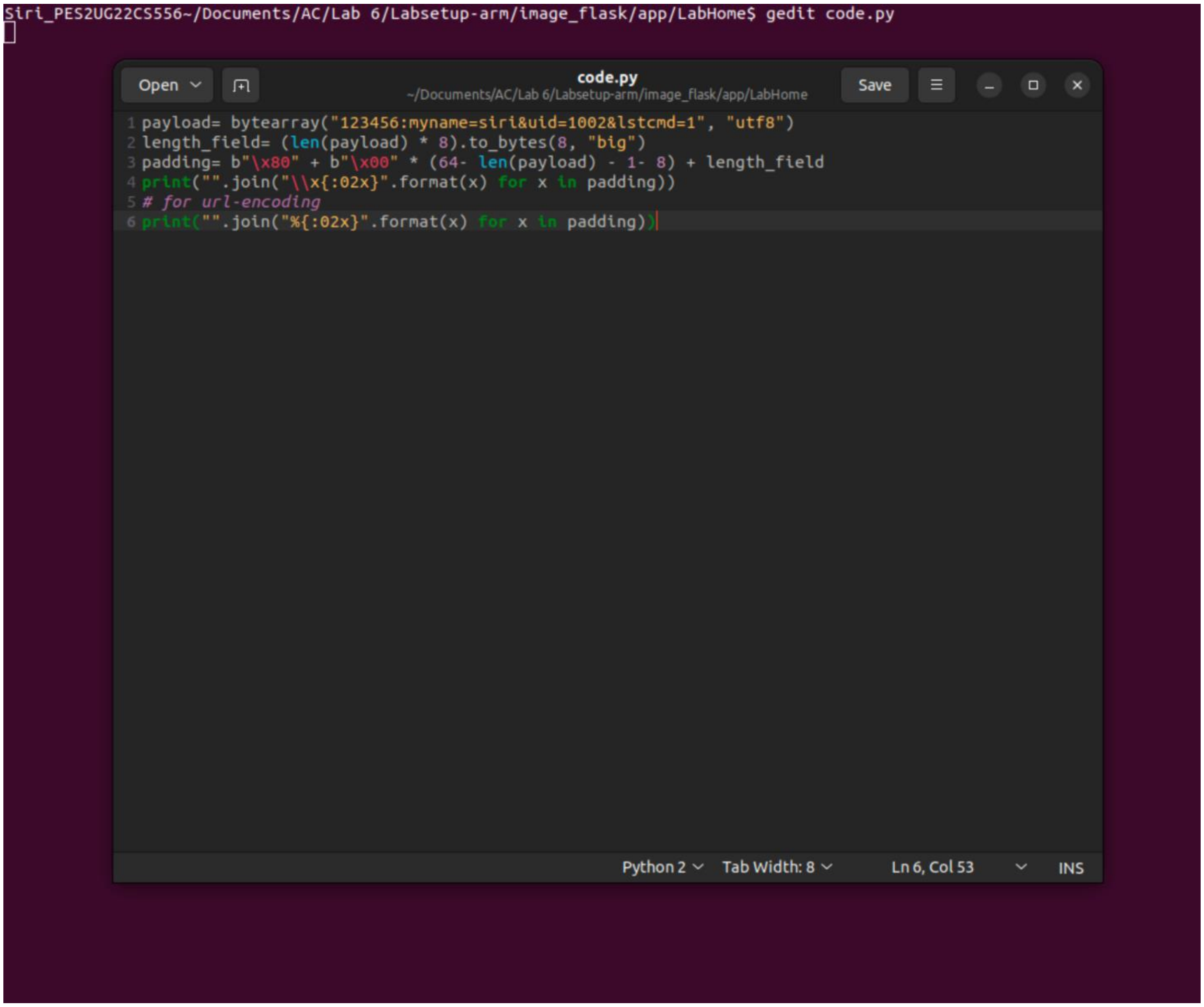
DO NOT DISCLOSE.

Problem 2 : Create Padding:

Step 1 : Creation of code file.
Step 2 : Run the code file.

Expected Deliverables -

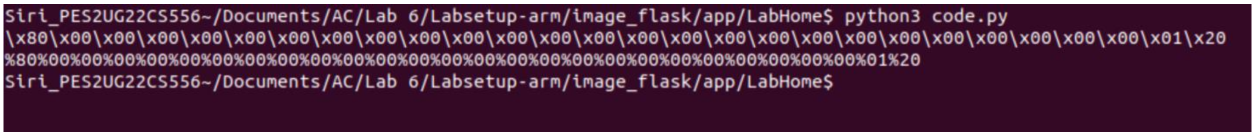
i) Output Screenshot for step 1



```
Siri_PES2UG22CS556~/Documents/AC/Lab 6/Labsetup-arm/image_flask/app/LabHome$ gedit code.py
```

```python
payload= bytearray("123456:myname=siri&uid=1002&lstcmd=1", "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))
# for url-encoding
print("".join("%{:02x}".format(x) for x in padding))
```

ii) Code Output Screenshot for step 2



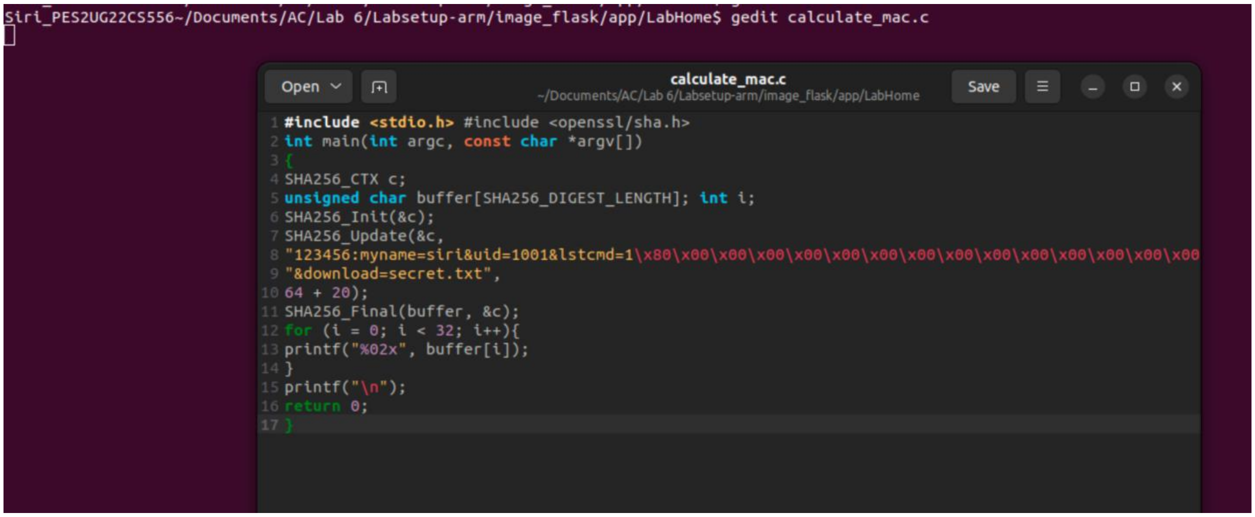
```
Siri_PES2UG22CS556~/Documents/AC/Lab 6/Labsetup-arm/image_flask/app/LabHome$ python3 code.py
\x80\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01\x20
%80%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%01%20
Siri_PES2UG22CS556~/Documents/AC/Lab 6/Labsetup-arm/image_flask/app/LabHome$
```

Problem 3 : The Length Extension Attack

Step 1a : Creation of code file.
Step 1b: Compile and run the code
Step 2 : Loading the URL
Step 3a : Generate New MAC
Step 3b: Generate a new padding
Step 4 : Visit the generated URL

Expected Deliverables -

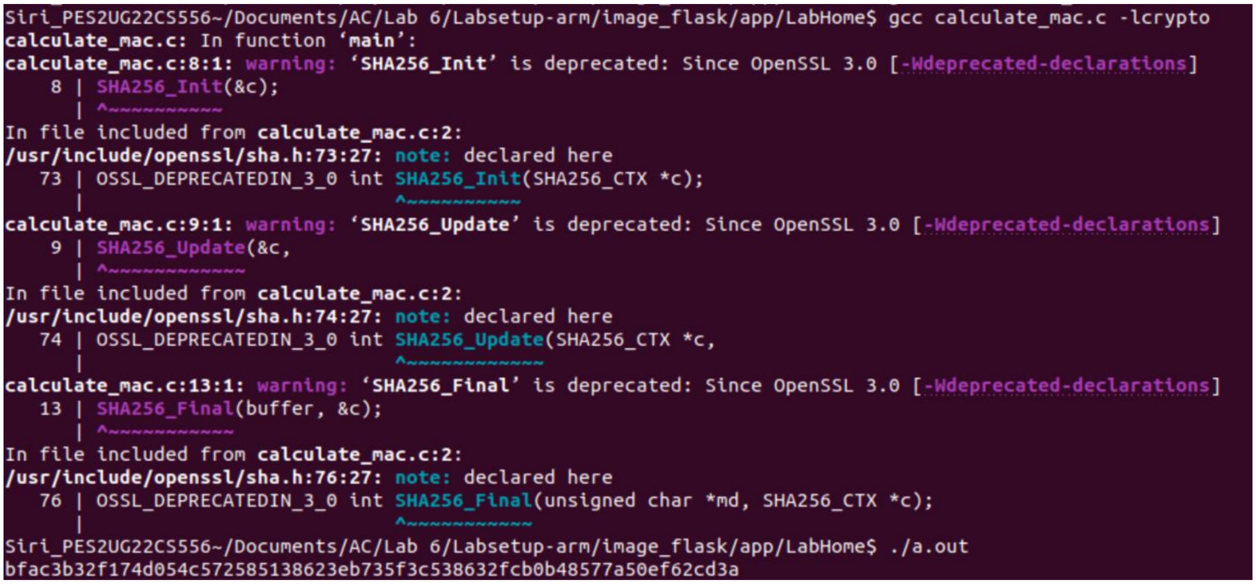
i) Code Output Screenshot for step 1a



```
Siri_PES2UG22CS556~/Documents/AC/Lab 6/Labsetup-arm/image_flask/app/LabHome$ gedit calculate_mac.c
```

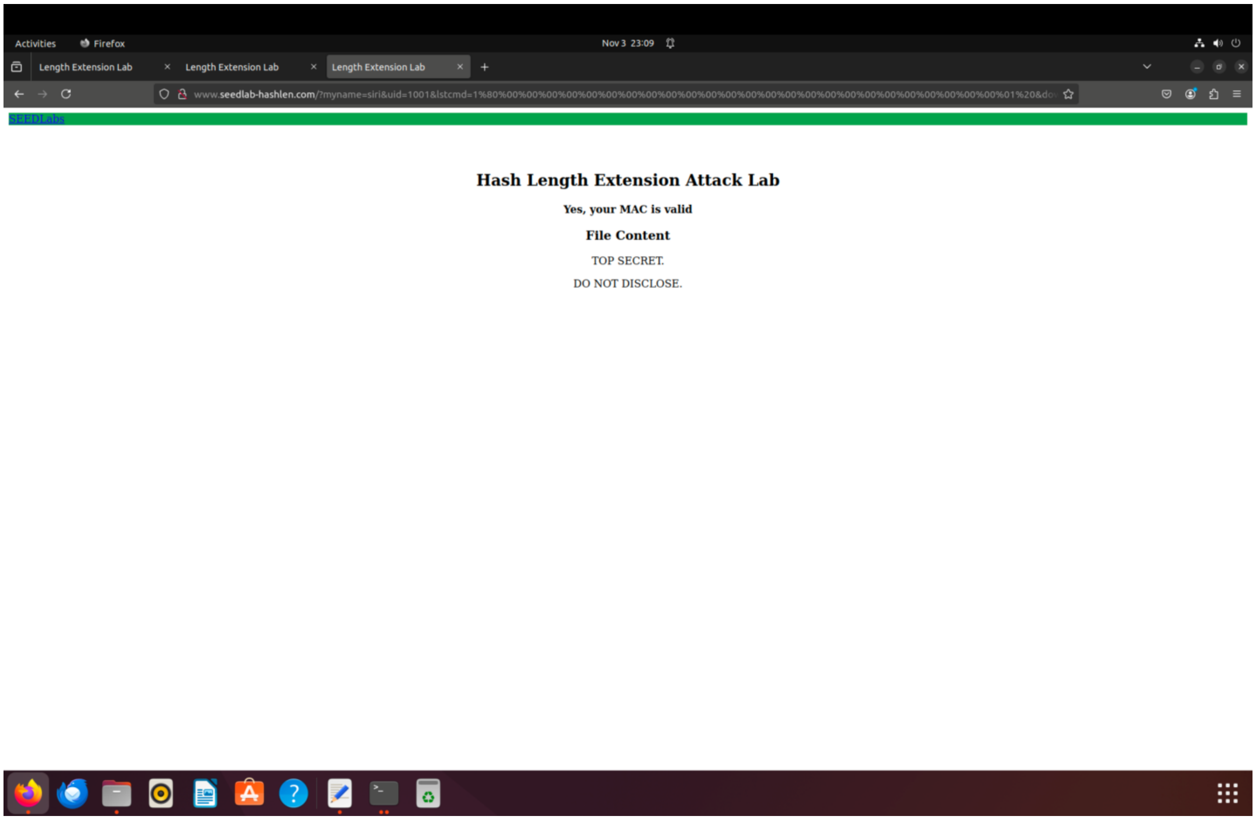
```c
#include <stdio.h> #include <openssl/sha.h>
int main(int argc, const char *argv[])
{
SHA256_CTX c;
unsigned char buffer[SHA256_DIGEST_LENGTH]; int i;
SHA256_Init(&c);
SHA256_Update(&c,
"123456:myname=siri&uid=1001&lstcmd=1\x80\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
"&download=secret.txt",
64 + 20);
SHA256_Final(buffer, &c);
for (i = 0; i < 32; i++){
printf("%02x", buffer[i]);
}
printf("\n");
return 0;
}
```

ii) Code Output Screenshot for step 1b



```
Siri_PES2UG22CS556~/Documents/AC/Lab 6/Labsetup-arm/image_flask/app/LabHome$ gcc calculate_mac.c -lcrypto
calculate_mac.c: In function ‘main’:
calculate_mac.c:8:1: warning: ‘SHA256_Init’ is deprecated: Since OpenSSL 3.0 [-Wdeprecated-declarations]
    8 | SHA256_Init(&c);
      | ^~~~~~~~~~~
In file included from calculate_mac.c:2:
/usr/include/openssl/sha.h:73:27: note: declared here
   73 | OSSL_DEPRECATEDIN_3_0 int SHA256_Init(SHA256_CTX *c);
      |                           ^~~~~~~~~~~
calculate_mac.c:9:1: warning: ‘SHA256_Update’ is deprecated: Since OpenSSL 3.0 [-Wdeprecated-declarations]
    9 | SHA256_Update(&c,
      | ^~~~~~~~~~~~~
In file included from calculate_mac.c:2:
/usr/include/openssl/sha.h:74:27: note: declared here
   74 | OSSL_DEPRECATEDIN_3_0 int SHA256_Update(SHA256_CTX *c,
      |                           ^~~~~~~~~~~~~
calculate_mac.c:13:1: warning: ‘SHA256_Final’ is deprecated: Since OpenSSL 3.0 [-Wdeprecated-declarations]
   13 | SHA256_Final(buffer, &c);
      | ^~~~~~~~~~~~
In file included from calculate_mac.c:2:
/usr/include/openssl/sha.h:76:27: note: declared here
   76 | OSSL_DEPRECATEDIN_3_0 int SHA256_Final(unsigned char *md, SHA256_CTX *c);
      |                           ^~~~~~~~~~~~
Siri_PES2UG22CS556~/Documents/AC/Lab 6/Labsetup-arm/image_flask/app/LabHome$ ./a.out
bfac3b32f174d054c572585138623eb735f3c538632fcb0b48577a50ef62cd3a
```

iii) Code Output Screenshot for step 2


Activities
Firefox
Nov 3 23:09
Length Extension Lab
Length Extension Lab
Length Extension Lab
www.seedlab-hashlen.com/?myname=siri&uid=1001&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%00%00%00%00%01%20&do
SEEDLabs

Hash Length Extension Attack Lab

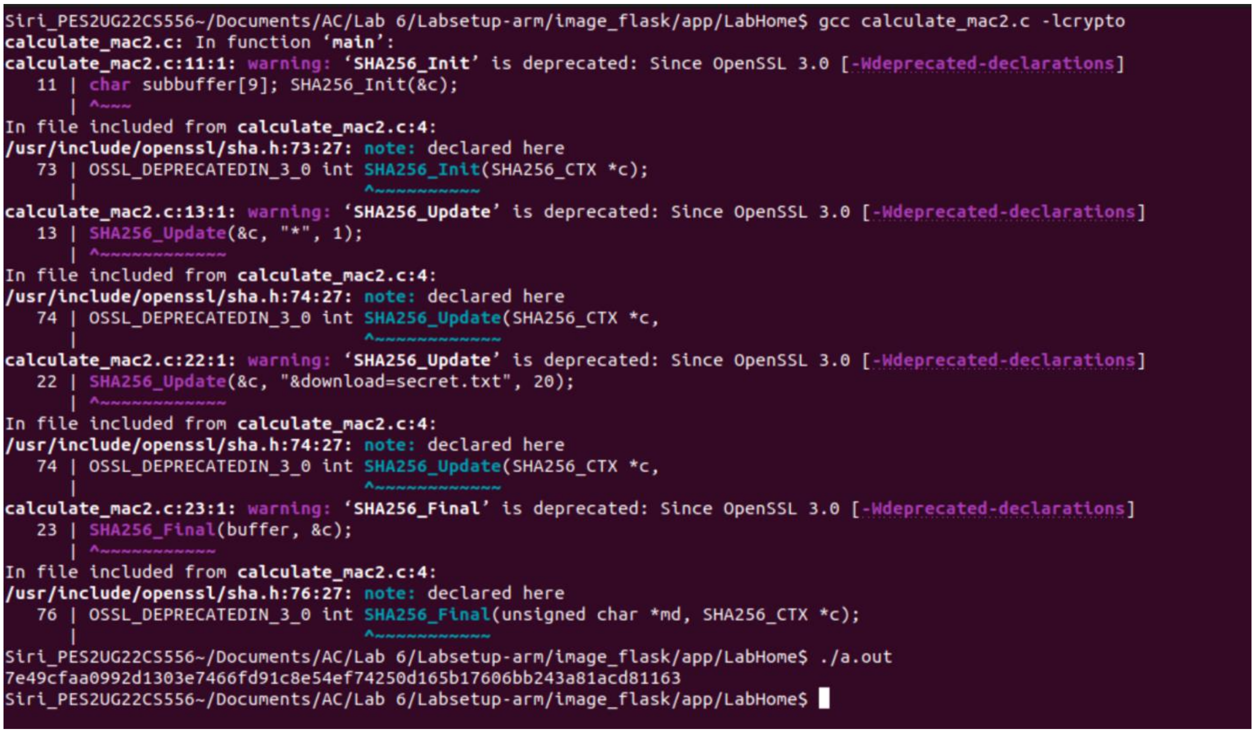
Yes, your MAC is valid

File Content

TOP SECRET.

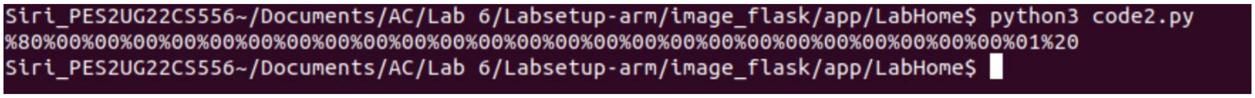
DO NOT DISCLOSE.

iv) Code Output Screenshot for step 3a



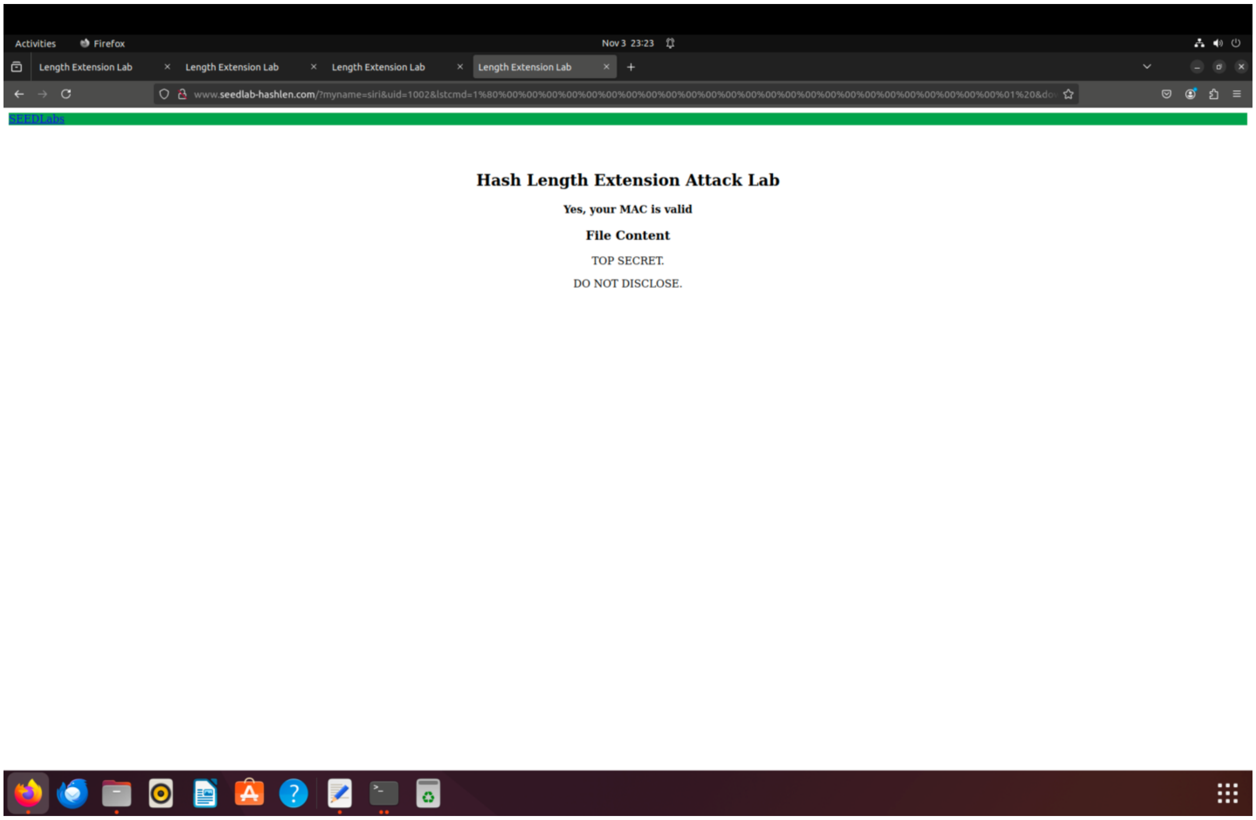
```
Siri_PES2UG22CS556~/Documents/AC/Lab 6/Labsetup-arm/image_flask/app/LabHome$ gcc calculate_mac2.c -lcrypto
calculate_mac2.c: In function ‘main’:
calculate_mac2.c:11:1: warning: ‘SHA256_Init’ is deprecated: Since OpenSSL 3.0 [-Wdeprecated-declarations]
   11 | char subbuffer[9]; SHA256_Init(&c);
      | ^~~~
In file included from calculate_mac2.c:4:
/usr/include/openssl/sha.h:73:27: note: declared here
   73 | OSSL_DEPRECATEDIN_3_0 int SHA256_Init(SHA256_CTX *c);
      |                           ^~~~~~~~~~~
calculate_mac2.c:13:1: warning: ‘SHA256_Update’ is deprecated: Since OpenSSL 3.0 [-Wdeprecated-declarations]
   13 | SHA256_Update(&c, "*", 1);
      | ^~~~~~~~~~~~~
In file included from calculate_mac2.c:4:
/usr/include/openssl/sha.h:74:27: note: declared here
   74 | OSSL_DEPRECATEDIN_3_0 int SHA256_Update(SHA256_CTX *c,
      |                           ^~~~~~~~~~~~~
calculate_mac2.c:22:1: warning: ‘SHA256_Update’ is deprecated: Since OpenSSL 3.0 [-Wdeprecated-declarations]
   22 | SHA256_Update(&c, "&download=secret.txt", 20);
      | ^~~~~~~~~~~~~
In file included from calculate_mac2.c:4:
/usr/include/openssl/sha.h:74:27: note: declared here
   74 | OSSL_DEPRECATEDIN_3_0 int SHA256_Update(SHA256_CTX *c,
      |                           ^~~~~~~~~~~~~
calculate_mac2.c:23:1: warning: ‘SHA256_Final’ is deprecated: Since OpenSSL 3.0 [-Wdeprecated-declarations]
   23 | SHA256_Final(buffer, &c);
      | ^~~~~~~~~~~~
In file included from calculate_mac2.c:4:
/usr/include/openssl/sha.h:76:27: note: declared here
   76 | OSSL_DEPRECATEDIN_3_0 int SHA256_Final(unsigned char *md, SHA256_CTX *c);
      |                           ^~~~~~~~~~~~
Siri_PES2UG22CS556~/Documents/AC/Lab 6/Labsetup-arm/image_flask/app/LabHome$ ./a.out
7e49cfaa0992d1303e7466fd91c8e54ef74250d165b17606bb243a81acd81163
Siri_PES2UG22CS556~/Documents/AC/Lab 6/Labsetup-arm/image_flask/app/LabHome$ 
```

v) Code Output Screenshot for step 3b



```
Siri_PES2UG22CS556~/Documents/AC/Lab 6/Labsetup-arm/image_flask/app/LabHome$ python3 code2.py
%80%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%01%20
Siri_PES2UG22CS556~/Documents/AC/Lab 6/Labsetup-arm/image_flask/app/LabHome$ 
```

vi) Code Output Screenshot for step 4



www.seedlab-hashlen.com/?myname=siri&uid=1002&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%00%00%00%01%20&do

SEEDLabs

Hash Length Extension Attack Lab

Yes, your MAC is valid

File Content

TOP SECRET.

DO NOT DISCLOSE.

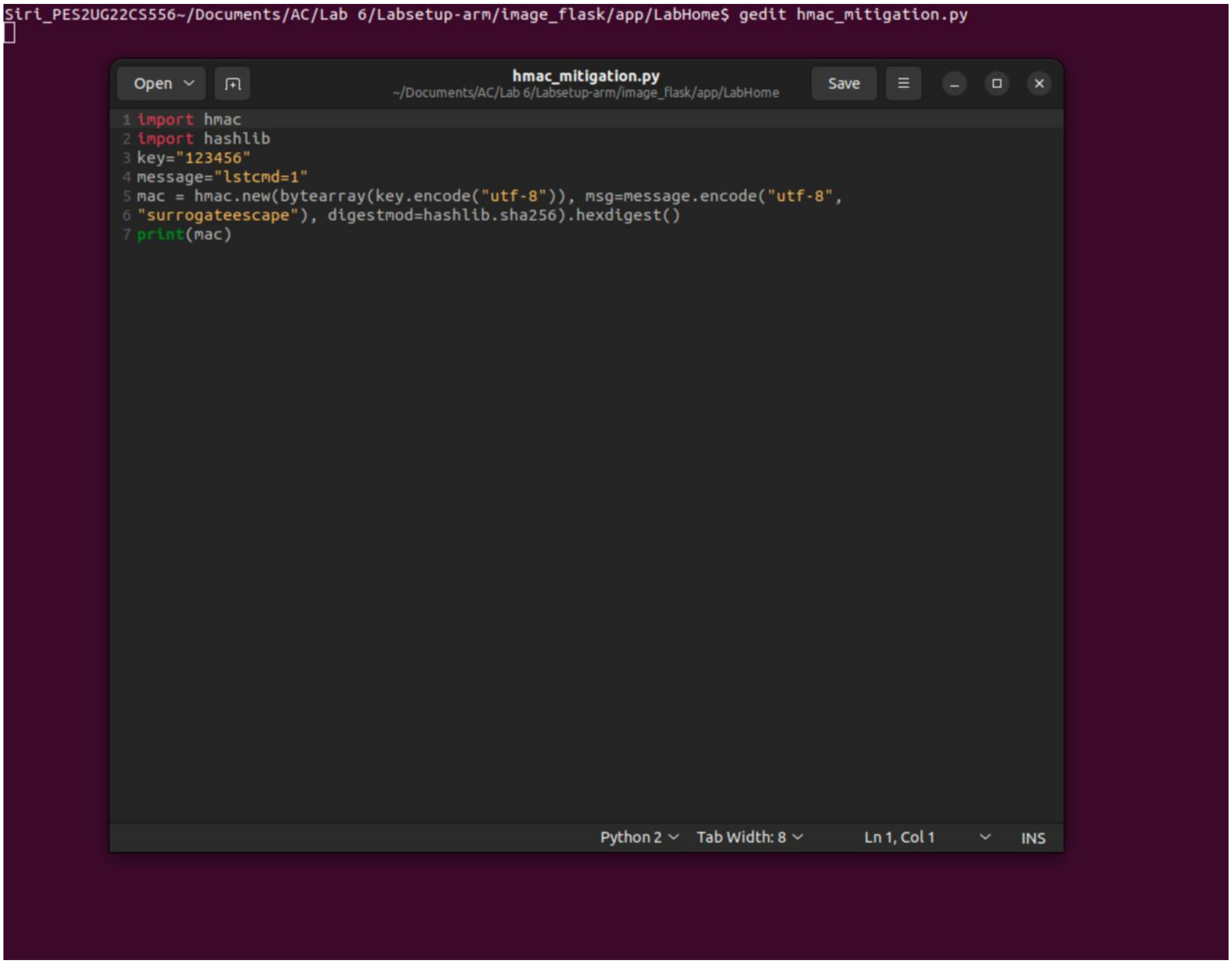
Problem 4 : Mitigation Using HMAC

Step 1 : Create the code file

Step 2 : Run the commands

Expected Deliverables –

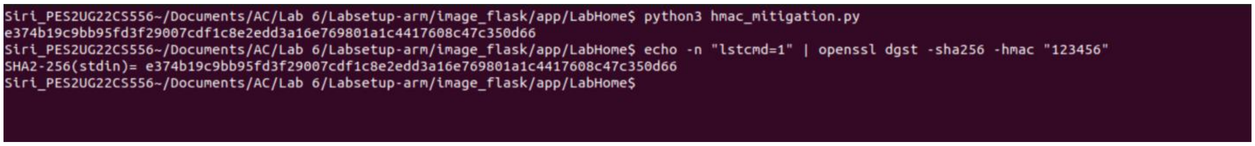
i) Code Output Screenshot for step 1



```
Siri_PES2UG22CS556~/Documents/AC/Lab 6/Labsetup-arm/image_flask/app/LabHome$ gedit hmac_mitigation.py
```

```
import hmac
import hashlib
key="123456"
message="lstcmd=1"
mac = hmac.new(bytearray(key.encode("utf-8")), msg=message.encode("utf-8",
"surrogateescape"), digestmod=hashlib.sha256).hexdigest()
print(mac)
```

ii) Code Output Screenshot for step 2



```
Siri_PES2UG22CS556~/Documents/AC/Lab 6/Labsetup-arm/image_flask/app/LabHome$ python3 hmac_mitigation.py
e374b19c9bb95fd3f29007cdf1c8e2edd3a16e769801a1c4417608c47c350d66
Siri_PES2UG22CS556~/Documents/AC/Lab 6/Labsetup-arm/image_flask/app/LabHome$ echo -n "lstcmd=1" | openssl dgst -sha256 -hmac "123456"
SHA2-256(stdin)= e374b19c9bb95fd3f29007cdf1c8e2edd3a16e769801a1c4417608c47c350d66
Siri_PES2UG22CS556~/Documents/AC/Lab 6/Labsetup-arm/image_flask/app/LabHome$
```

iii) Describe why a malicious request using length extension and extra commands will fail MAC verification when the client and server use HMAC.

HMAC involves using a secret key in both the generation and verification of the MAC. Unlike simple hash functions vulnerable to length extension attacks, HMAC adds security by incorporating the secret key, which prevents attackers from predicting or generating valid MACs for modified messages. This makes it resistant to length extension attacks because any modification to the message or commands would require knowledge of the secret key to create a valid MAC, which the attacker does not possess. Therefore, any malicious attempt to alter the message or add commands without the correct HMAC will fail verification.