Lab5

Yue Zhang

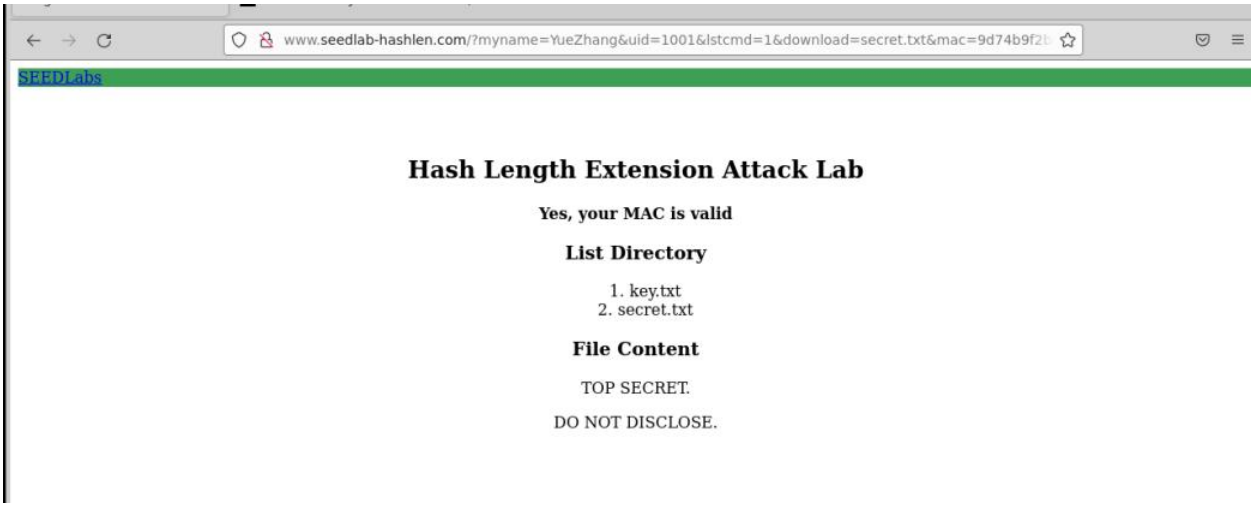
Task1

```
seed@ip-172-31-30-228:/home/ubuntu/lab5/Labsetup$  echo -n "123456:myname=YueZhang&uid=1001
&lstcmd=1&download=secret.txt" | sha256sum
9d74b9f2bcc221e35e3bde1c7c946150ea0320b62db1c5b427385eb7a45b2044  -
```

I calculated the MAC.

Then I sent this request to the server program.

http://www.seedlab-hashlen.com/?myname=YueZhang&uid=1001&lstcmd=1&download=secret.txt&mac=9d74b9f2bcc221e35e3bde1c7c946150ea0320b62db1c5b427385eb7a45b2044



This is result.

Task2

```
seed@ip-172-31-30-228:/home/ubuntu/lab5/Labsetup$ ls -l task2.txt
-rw-rw-r-- 1 seed seed 41 Oct 26 19:58 task2.txt
```

This is a message to padding:”123456:myname=YueZhang&uid=1001&lstcmd=1”

It’s 41 bytes

64 – 41 = 23 bytes

length of message in bits is 8 * 41 = 328, in hex \x01 \x48

Padding is

"\x80"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"

"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
"\x01\x48"

%80%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%01%48

Task3

```
seed@ip-172-31-30-228:/home/ubuntu/lab5/Labsetup$  echo -n "123456:myname=YueZhang&uid=1001
&lstcmd=1" | sha256sum
fce503bbb2ca87fe766527dfc0f1496e962c5493e73ca0da96cb232dc0345d41  -
```

“fce503bbb2ca87fe766527dfc0f1496e962c5493e73ca0da96cb232dc0345d41” is a valid MAC.

```c
#include <openssl/sha.h>
int main(int argc, const char *argv[])
{
int i;
unsigned char buffer[SHA256_DIGEST_LENGTH];
SHA256_CTX c;
SHA256_Init(&c);
for(i=0; i<64; i++)
SHA256_Update(&c, "*", 1);
// MAC of the original message M (padded)
c.h[0] = htole32(0xfce503bb);
c.h[1] = htole32(0xb2ca87fe);
c.h[2] = htole32(0x766527df);
c.h[3] = htole32(0xc0f1496e);
c.h[4] = htole32(0x962c5493);
c.h[5] = htole32(0xe73ca0da);
c.h[6] = htole32(0x96cb232d);
c.h[7] = htole32(0xc0345d41);
// Append additional message
SHA256_Update(&c, "&download=secret.txt", 20);
SHA256_Final(buffer, &c);
for(i = 0; i < 32; i++) {
printf("%02x", buffer[i]);
}
printf("\n");
return 0;
}
```

In this way, we can generate a new MAC

```
9d74b9f2bcc221e35e3bde1c7c946150ea0320b62db1c5b427385eb7a45b2044
```

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.

Then we sent the request to the server program, we can read secret.txt file.

Task4

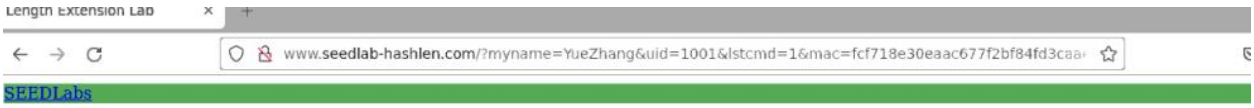
```
#!/bin/env python3
import hmac
import hashlib
key='123456'
message='123456:myname=YueZhang&uid=1001&lstcmd=1'
mac = hmac.new(bytearray(key.encode('utf-8')),
msg=message.encode('utf-8', 'surrogateescape'),
digestmod=hashlib.sha256).hexdigest()
print(mac)
~
```

```
seed@ip-172-31-30-228:/home/ubuntu/lab5/Labsetup$ python3 mac.py
fcf718e30eaac677f2bf84fd3caae3afadcf1b75b273d68ab43cef6b5c760e56
```

I using HMAC to calculate the MAC.

Then I send following request to the server program.

http://www.seedlab-hashlen.com/?myname=YueZhang&uid=1001&lstcmd=1&mac=fcf718e30eaac677f2bf84fd3caae3afadcf1b75b273d68ab43cef6b5c760e56



Hash Length Extension Attack Lab

Sorry, your MAC is not valid

Malicious request failed.

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 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 extension attack.