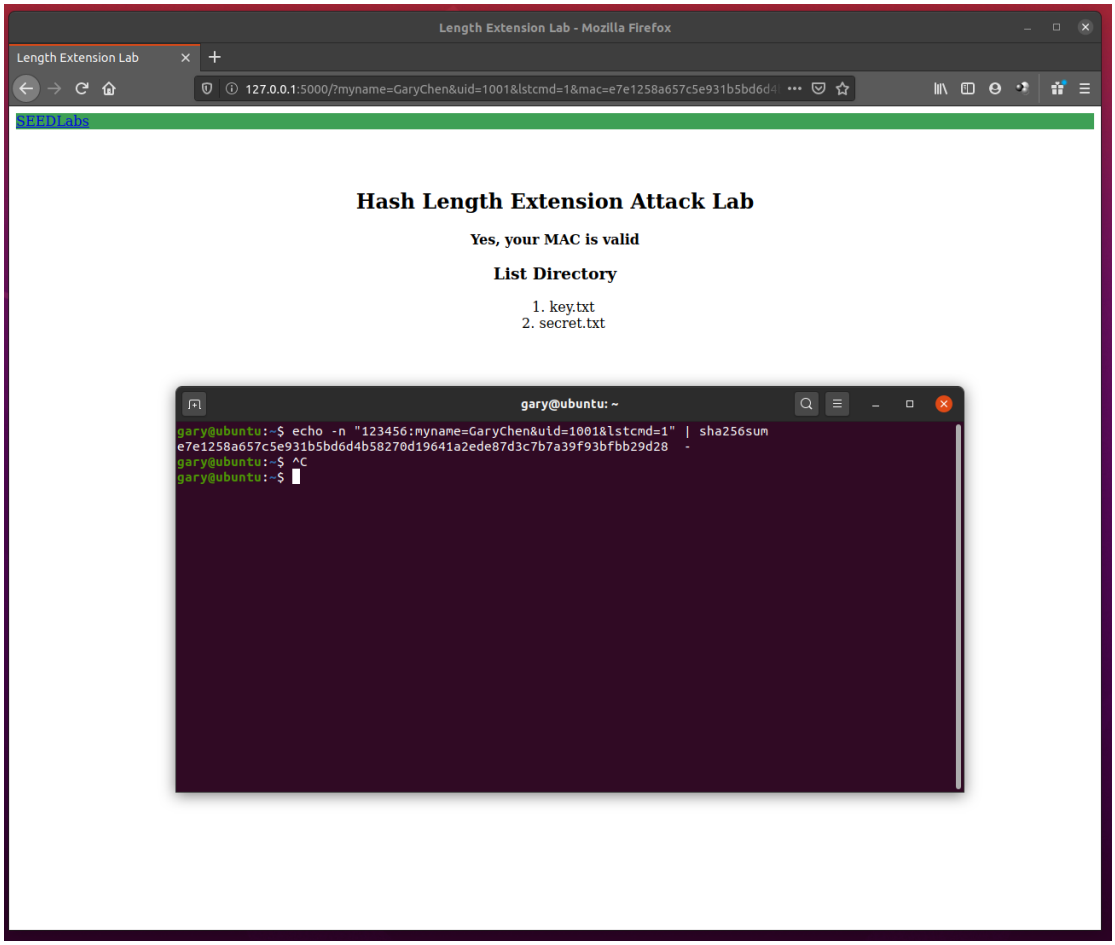
資訊安全 LAB

40647027S 陳冠穎

Task1:



Task2:

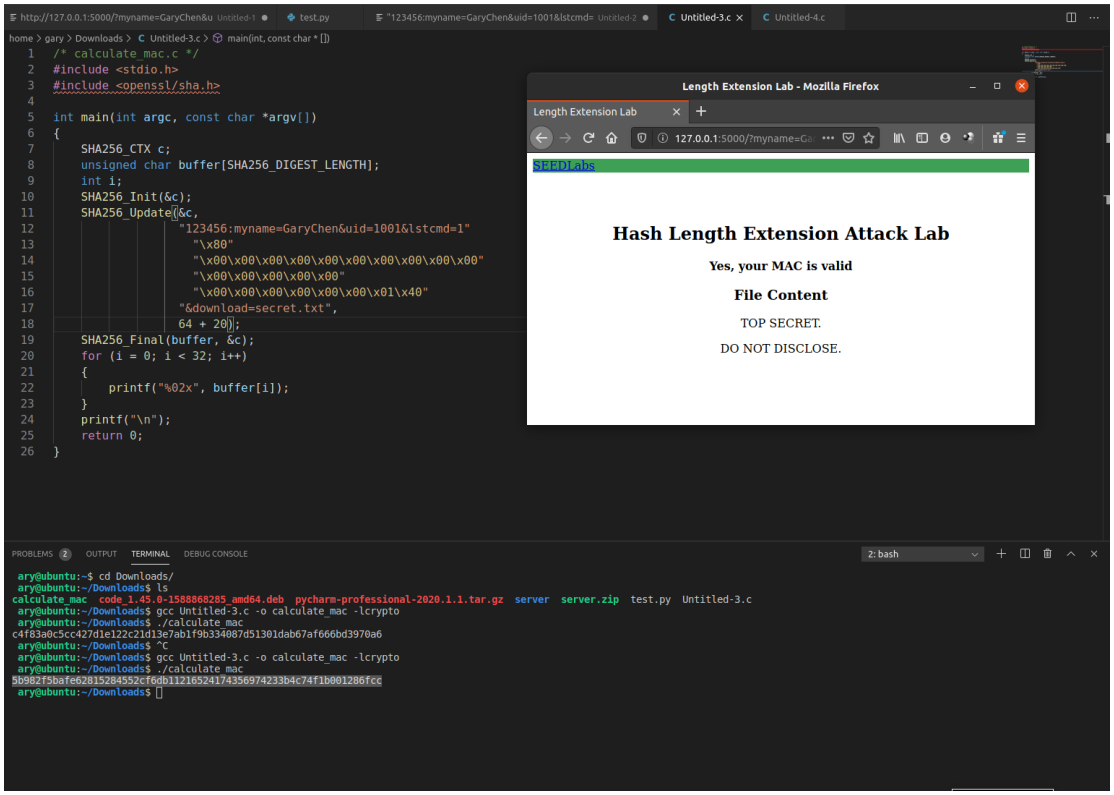
123456:myname=GaryChen&uid=1001&lstcmd=1: 40 Bytes

40 * 8 = 320 bits = 0x140 bits

```
"123456:myname=GaryChen&uid=1001&lstcmd=1"
"\x80"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
"\x00\x00\x00\x00\x00"
"\x00\x00\x00\x00\x00\x00\x01\x40"
```

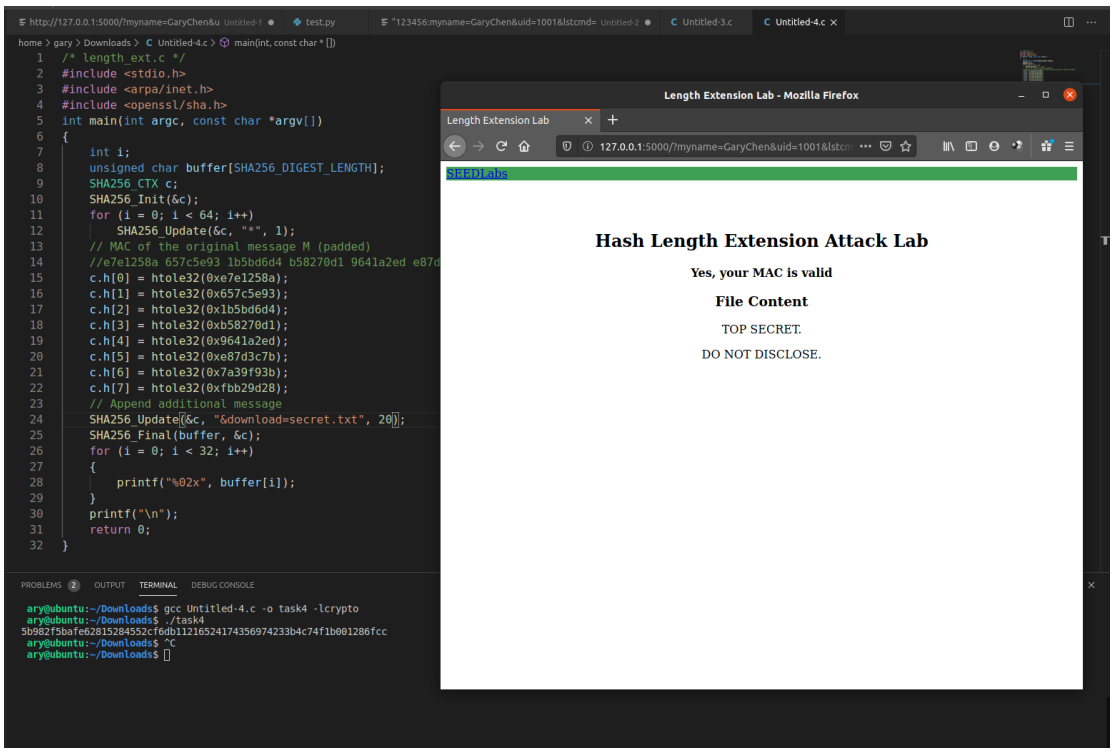
Task3:

MAC: 5b982f5bafe62815284552cf6db1121652417435697 4233b4c74f1b001286fcc
Request:
http://127.0.0.1:5000/?myname=GaryChen&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%01%40&download=secret.txt&mac=5b982f5bafe62815284552cf6db1121652417435697 4233b4c74f1b001286fcc



Task4:
算出來的 MAC 會跟 Task3 知道 Key 之下算出來的一樣。
MAC: 5b982f5bafe62815284552cf6db1121652417435697 4233b4c74f1b001286fcc
Request:
http://127.0.0.1:5000/?myname=GaryChen&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%01%40&download=secret.txt&mac=5b982f5bafe62815284552cf6db1121652417435697 4233b4c74f1b001286fcc

也就是說我只要知道原本合法 request 與 MAC，再把 padding 算出來，之後再後面加別的指令也可以在不知道 key 的情況下算出合法的 MAC。



Task5:

將 verify_mac 更改如下

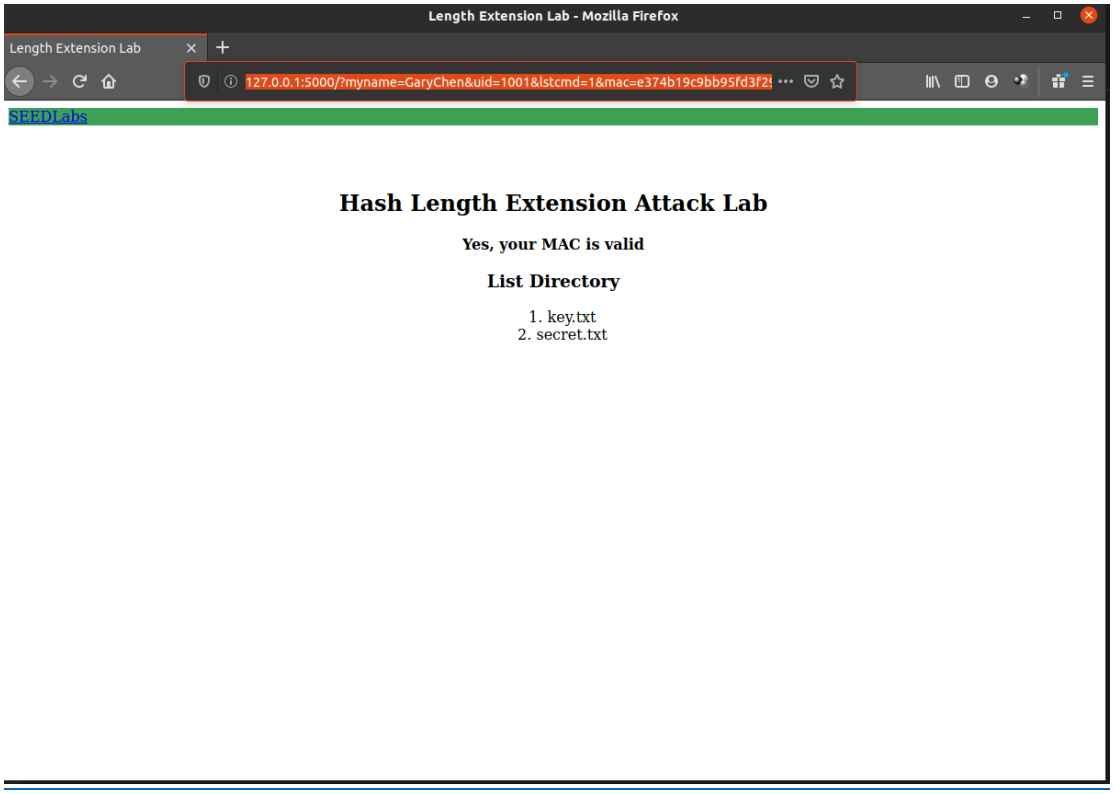
```
def verify_mac(key, my_name, uid, cmd, download, mac):
    download_message = '' if not download else '&download=' + download
    message = ''
    '''
    if my_name:
        message = 'myname={}&'.format(my_name)
    message += 'uid={}&lstcmd='.format(uid) + cmd + download_message
    '''
    message = 'lstcmd={}'.format(cmd) + download_message

    payload = key + ':' + message
    app.logger.debug('payload is [{}]'.format(payload))
    #real_mac = hashlib.sha256(payload.encode('utf-8', 'surrogateescape')).hexdigest()
    real_mac = hmac.new(bytearray(key.encode('utf-8')), msg=message.encode('utf-8','surrogateescape'), digestmod=hashlib.sha256).hexdigest
    app.logger.debug('real mac is [{}]'.format(real_mac))
    if mac == real_mac:
        return True
    return False
```

使用新的方法算出 MAC

Request:

http://127.0.0.1:5000/?myname=GaryChen&uid=1001&lstcmd=1&mac=e374b19c9bb95fd3f29007cdf1c8e2edd3a16e769801a1c4417608c47c350d66



成功。

因為 HMAC 的作用方式會先將 key+message 做一次 hash 後，再做第二次的 hash，因此攻擊者無法再用長度擴展攻擊任意更改 message。