



System Security Week 4

Cryptography: HASHING and integrity check

Learning Outcomes

In this lab, you will:

- Calculate checksums of different files
- Compare the checksums for identifying corrupted/ modified contents
- Use one-way hash function and Message Authentication Code (MAC)
- Learn how to flip bits of input file using a binary/hex editor

Introduction

The objective of this lab is to familiarise the students with one-way hash functions and Message Authentication Code (MAC). Moreover, how we can use hash functions for integrity checking of contents downloaded from the internet. After finishing the lab, in addition to gaining a deeper understanding of the concepts, students should be able to use OpenSSL to generate one-way hash value and MAC for a file and can compare the checksums of the file to identify unwanted or corrupted files.

Integrity checking

It is essential to verify the authenticity of a file to avoid incidences of downloading suspicious data and infecting your computer with viruses, malware and other harmful malicious components. Checking the integrity of a file aims at checking that either a file is genuine or verifying that a file has not been modified by untrusted parties.

Now, how to verify the integrity of a file? or in other words how to check that a file contents has not been changed/modified.

1. As most file hosting sites put the hash values along with the file to be downloaded. For example go to OpenSSL website by clicking on this link:
<https://www.openssl.org/source/old/1.0.2/>

KBytes	Date	File	Checksum
5229	2019-Dec-20 13:25:43	openssl-1.0.2u.tar.gz	SHA256 (PGP sign) (SHA1)
5223	2019-Sep-10 13:56:31	openssl-1.0.2s.tar.gz	SHA256 (PGP sign) (SHA1)
5229	2019-Sep-10 13:53:14	openssl-1.0.2t.tar.gz	SHA256 (PGP sign) (SHA1)
5223	2019-Feb-26 14:34:59	openssl-1.0.2r.tar.gz	SHA256 (PGP sign) (SHA1)
5220	2018-Nov-20 14:07:08	openssl-1.0.2q.tar.gz	SHA256 (PGP sign) (SHA1)
5213	2018-Aug-14 13:08:43	openssl-1.0.2p.tar.gz	SHA256 (PGP sign) (SHA1)
5204	2018-Mar-27 14:03:37	openssl-1.0.2o.tar.gz	SHA256 (PGP sign) (SHA1)
5249	2017-Dec-07 13:47:59	openssl-1.0.2n.tar.gz	SHA256 (PGP sign) (SHA1)
5247	2017-Nov-02 14:51:59	openssl-1.0.2m.tar.gz	SHA256 (PGP sign) (SHA1)
5239	2017-May-25 13:09:51	openssl-1.0.2l.tar.gz	SHA256 (PGP sign) (SHA1)
5184	2017-Jan-26 13:45:54	openssl-1.0.2k.tar.gz	SHA256 (PGP sign) (SHA1)
5183	2016-Sep-26 10:04:14	openssl-1.0.2i.tar.gz	SHA256 (PGP sign) (SHA1)
5183	2016-Sep-22 10:35:26	openssl-1.0.2j.tar.gz	SHA256 (PGP sign) (SHA1)
5150	2016-May-03 13:57:13	openssl-1.0.2h.tar.gz	SHA256 (PGP sign) (SHA1)
5142	2016-Mar-01 13:54:09	openssl-1.0.2g.tar.gz	SHA256 (PGP sign) (SHA1)
5135	2016-Jan-28 14:58:54	openssl-1.0.2f.tar.gz	SHA256 (PGP sign) (SHA1)
5133	2015-Dec-03 17:48:55	openssl-1.0.2e.tar.gz	SHA256 (PGP sign) (SHA1)

Figure 1 OpenSSL integrity checking

The link will take you to the download page of OpenSSL as can be seen in Figure 4. Here you will find different information like the first column indicates the size of the file, 2nd column shows the release date, 3rd is the File, while 4th is the checksum.

2. Now download the file by clicking on the file link (as can be seen in the yellow box in Figure 4).
3. Next download the checksum file by clicking on the SHA256 link as can be seen in red box in Figure 4.
4. Now open the checksum file in notepad, you will see the hash values of the file as can be seen below:

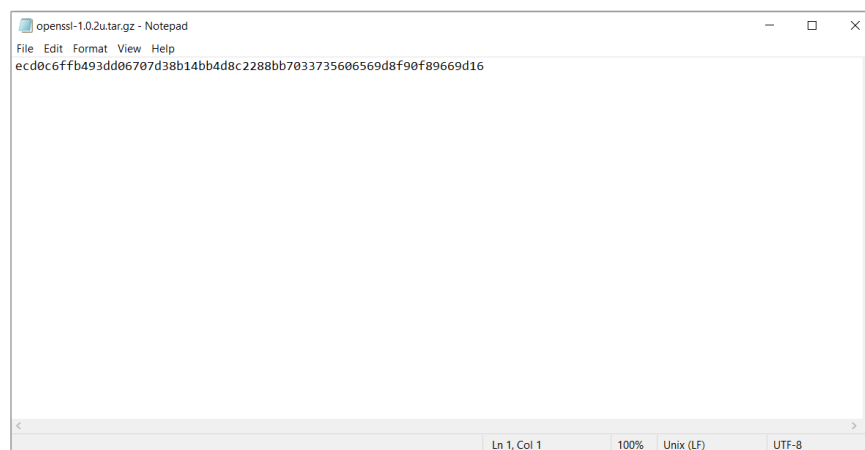
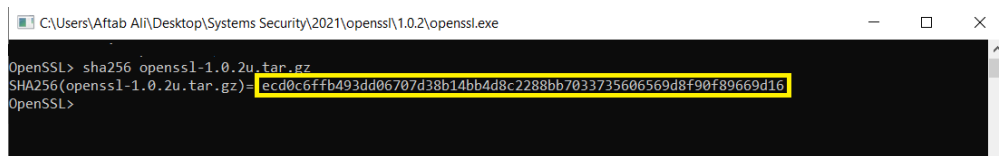


Figure 2 Downloaded checksum

5. Next calculate the sha256 hash of the downloaded file (not the checksum file, it is the OpenSSL file downloaded in step 2) by using the following command:

openssl sha256 openssl-1.0.2u.tar.gz

Note: The file *openssl-1.0.2u.tar.gz* should be in the same directory/folder



```
C:\Users\Aftab Ali\Desktop\System Security\2021\openssl\1.0.2\openssl.exe
OpenSSL> sha256 openssl-1.0.2u.tar.gz
SHA256(openssl-1.0.2u.tar.gz)=ecd0c6ffb493dd06707d38b14bb4d8c2288bb7033735606569d8f90f89669d16
OpenSSL>
```

Figure 3 Calculated checksum

Integrity checking: Tasks

1. Download another OpenSSL and its checksum file.
2. Repeat the above process and check both the checksum values are the same or not? **Yes, both are the same.**
3. What we learnt from the above process? **The file has not been changed during the download process. Authentic file.**
4. Open the *openssl-1.0.2u.tar.gz* file in 7z or WinRAR and delete any file from its contents. Then calculate the checksum again. Is it the same? **No** Why? **Because the contents have been modified now.**

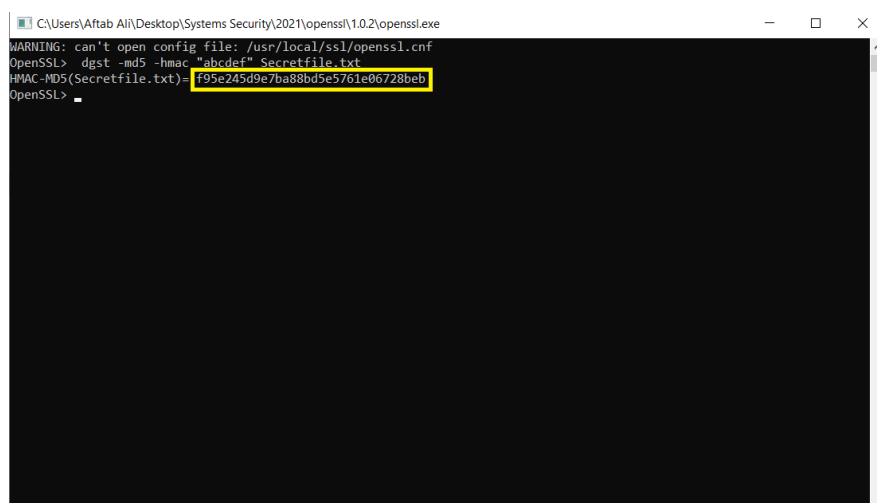
Keyed Hashing

A keyed hash message authentication code (HMAC) is a specific type of message authentication code (MAC) involving a cryptographic hash function (hence the 'H') in combination with a secret cryptographic key.

In this part, we would like to generate a keyed hash (i.e. MAC) of our Secretfile.txt. We can use the -hmac option as can be seen bellow:

```
openssl dgst -md5 -hmac "abcdef" Secretfile.txt
```

Running the above command will generate the hash values as can be seen in Figure 4.



```
C:\Users\Aftab Ali\Desktop\System Security\2021\openssl\1.0.2\openssl.exe
WARNING: can't open config file: /usr/local/ssl/openssl.cnf
OpenSSL> dgst -md5 -hmac "abcdef" Secretfile.txt
HMAC-MD5(Secretfile.txt)=f95e245d9e7ba88bd5e5761e06728beb
OpenSSL>
```

Figure 4 HMAC hash values

The Text inside our Secretfile.txt can be seen in Figure 5.

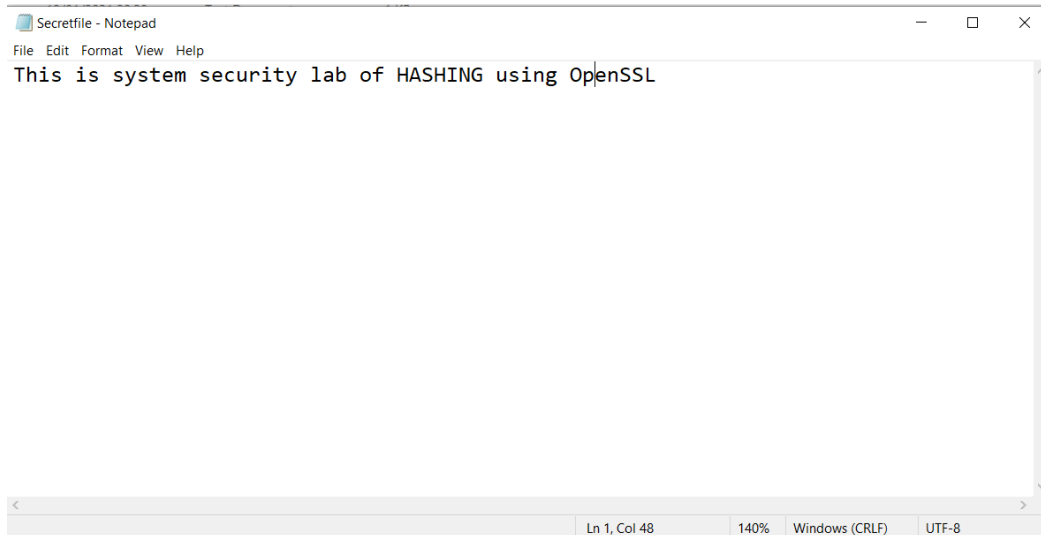


Figure 5 Our secret text

Now let's flip only one bit in our Secretfile.txt. For bit flipping, I am using 010-editor, you can download the 010 editor from the following link:

<https://www.sweetscape.com/download/010editor/>

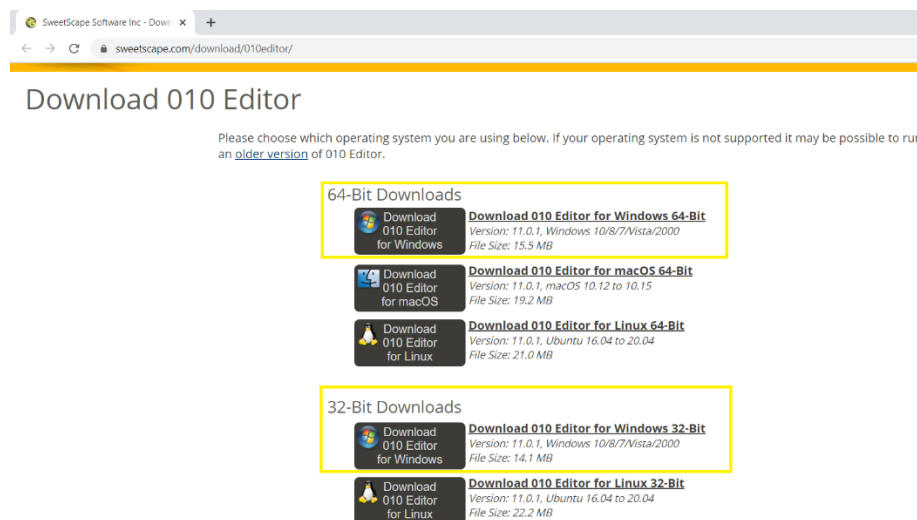


Figure 6 010 editor download page

Open the Secretfile.txt in 010-editor and select the text as can be seen in Figure 7. In the right-side window the binary value of the selected text is displayed as can be seen in the yellow box in Figure 7. You can flip any bit (In my case I am flipping the last 0 to 1). This resulted the following text:

Uhis is system security lab of HASHING using OpenSSL

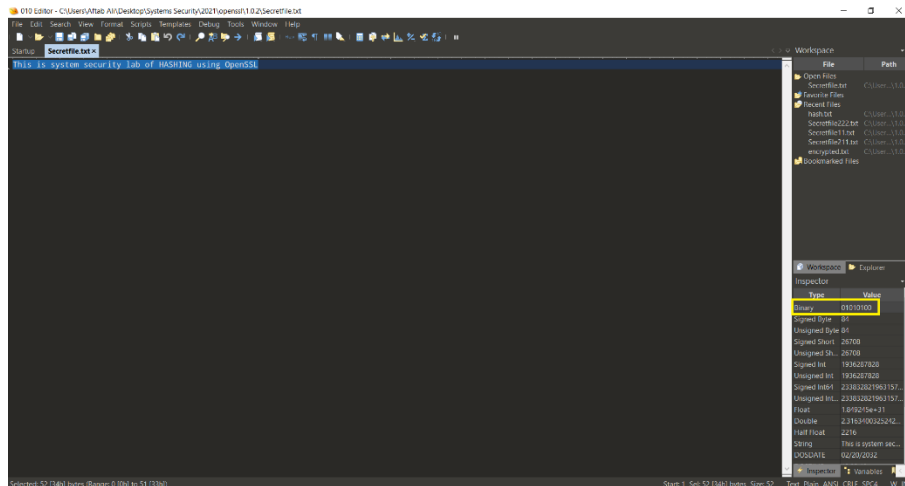


Figure 7 Bit flipping

As we flipped only a single bit in our Secretfile.txt, Now, let's see the effect in recalculating the hash values.

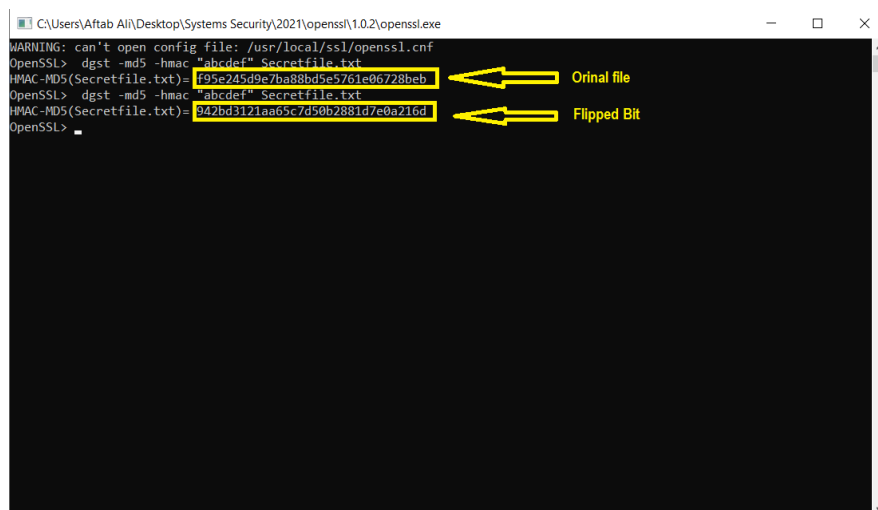


Figure 8 Original and flipped bit hash comparison

Tasks

1. Create a text file of any length
2. Generate the hash value H_1 for the file using HMAC (try using sha256 with HMAC).
3. Flip one bit of the input file.
4. Generate the hash value H_2 for the modified file.
5. Observe whether H_1 and H_2 are similar or not?