MD5 COLLISION LAB

Assignment 5

EDMUND LAWERH AMANOR Fundamentals of Info. Security, Fall 2024

This report is a documentation of my observation of the process while I simulated the **random number generation** lab in the SEED virtual machine. I provided screenshots, from the various stages.

I commented out the steps I took and observations made directly in the terminal window, both before and after executing commands to make is easy to follow what was happening.

Task 1: Introducing MD5 Collision

The goal of this task is to show that MD5 vulnerable is by creating the MD5 hash values for two colliding messages.

- First, I show th screenshots after copying the given colliding messages into two files and creating their binaries.

```
../md5-collision-lab$ mkdir task1 task2 task3 task4
../md5-collision-lab$ cd task1/
../task1$ touch file1.hex file2.hex
../task1$
../task1$ xxd -r -p file1.hex > file1
../task1$ xxd -r -p file2.hex > file2
../task1$
```

- Next, let's see the confirmation that MD5 hash values for colliding messages are identical whereas the SHA-256 hash values on those same messages are different.

```
[12/06/24]seed@VM:~/.../task1$
[12/06/24]seed@VM:~/.../task1$ # let's check the MD5 for file1 and file2
[12/06/24]seed@VM:~/.../task1$
[12/06/24]seed@VM:~/.../task1$ openssl dgst -md5 file1 file2
MD5(file1)= 79054025255fb1a26e4bc422aef54eb4
MD5(file2)= 79054025255fb1a26e4bc422aef54eb4
[12/06/24]seed@VM:~/.../task1$
[12/06/24]seed@VM:~/.../task1$
[12/06/24]seed@VM:~/.../task1$ # let's confirm the SHA-256 hash for the two files
[12/06/24]seed@VM:~/.../task1$
[12/06/24]seed@VM:~/.../task1$ openssl dgst -sha256 file1 file2
SHA256(file1)= 8d12236e5c4ed9f4e790db4d868fd5c399df267e18ff65c1107c328228cffc98
SHA256(file2)= b9fef2a8fc93b05e7701e97196fda6c4fbeea25ff8e64fdfee7015eca8fa617d
[12/06/24]seed@VM:~/.../task1$
```

Task 2: MD5 Collisions using Fastcoll program

In this task, the goal is to use the Fastcoll program to create our own two colliding messages whose MD5 hash values are identical. The image below shows the creation of the two collision files. And as highlighted in the snapshot below, it took **1.19093** seconds to create these two files.

```
[12/06/24]seed@VM:~/.../task2$
[12/06/24]seed@VM:~/.../task2$ echo “This is the first file” >file1
[12/06/24]seed@VM:~/.../task2$ echo “This is the second file” >file2
[12/06/24]seed@VM:~/.../task2$
[12/06/24]seed@VM:~/.../task2$ # Generating my own collisions
[12/06/24]seed@VM:~/.../task2$
[12/06/24]seed@VM:~/.../task2$ time ./fastcoll -o file1 file2
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'file1' and 'file2'
Using initial value: 0123456789abcdeffedcba9876543210

Generating first block: ..
Generating second block: S11.......
Running time: 1.19093 s

real    0m1.230s
user    0m1.179s
sys     0m0.020s
[12/06/24]seed@VM:~/.../task2$
```

- Next, let's take a look at the hex representation of these two files. As shown below, the two are quite identical.

```
[12/06/24]seed@VM:~/.../task2$
[12/06/24]seed@VM:~/.../task2$ xxd -p file1
f1e522a8b75be3b5785ab734d33eb1d57a11fe5f1712646f47267a3e6a30
b1f2f7edc0eedec301ebef5dbbd4df05d84cd34440f17962954539e5cfc7
63c78bc6280daae3c0f97b28cd87ebcd8cca15b4d3cb412f0b8c971f764f
90db4fa145c1ea48b5ea84e8a25716a067d182ef026384c450f7b497dd44
3d0c65e391d6a577
[12/06/24]seed@VM:~/.../task2$
[12/06/24]seed@VM:~/.../task2$
[12/06/24]seed@VM:~/.../task2$
[12/06/24]seed@VM:~/.../task2$ xxd -p file2
f1e522a8b75be3b5785ab734d33eb1d57a11fedf1712646f47267a3e6a30
b1f2f7edc0eedec301ebef5dbbd4df85d84cd34440f17962954539e5cf47
63c78bc6280daae3c0f97b28cd87ebcd8cca15b4d3cb41af0b8c971f764f
90db4fa145c1ea48b5ea84e8a25716a067d1826f026384c450f7b497dd44
3d0c656391d6a577
[12/06/24]seed@VM:~/.../task2$
```

- Finally, I shown in the image below that the MD5 hash for the two files are identical while the SHA256 hash for them are completely different as expected.

```
[12/06/24]seed@VM:~/.../task2$
[12/06/24]seed@VM:~/.../task2$ openssl dgst -md5 file1 file2
MD5(file1)= dd5251b635bcff382cf65009546bf51e
MD5(file2)= dd5251b635bcff382cf65009546bf51e
[12/06/24]seed@VM:~/.../task2$
[12/06/24]seed@VM:~/.../task2$
[12/06/24]seed@VM:~/.../task2$
[12/06/24]seed@VM:~/.../task2$ openssl dgst -sha256 file1 file2
SHA256(file1)= 9399cb8390baddacba21a7fc2166e5e04101a254fafcf94d15bc06d561d34aa2
SHA256(file2)= d3a50f43e34d4c3c6d545f7b9fd5388507017b0c6685e097e998e5ecc7deeec1
[12/06/24]seed@VM:~/.../task2$
```

Task 3: MD5 collision using same file prefix & suffix

First let's take a look at the preliminary steps I took to verify that the chosen prefix and suffix vulnerability of MD5 before creating my good and evil scripts. The image below shows the creation of the colliding files using the fastcoll program.

```
[12/06/24]seed@VM:~/.../task3$
[12/06/24]seed@VM:~/.../task3$ ls
fastcoll  prefix  suffix
[12/06/24]seed@VM:~/.../task3$
[12/06/24]seed@VM:~/.../task3$ ./fastcoll -p prefix -o col1 col2
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'col1' and 'col2'
Using prefixfile: 'prefix'
Using initial value: 0a0457cfbc3f8e32ee1c902d6927257b

Generating first block: ................
Generating second block: S01.................
Running time: 8.20854 s
[12/06/24]seed@VM:~/.../task3$
```

- Next, I show the concatenation of the colliding files with the suffix and verifying the md5 and sha256 hashes.

```
[12/06/24]seed@VM:~/.../task3$
[12/06/24]seed@VM:~/.../task3$
[12/06/24]seed@VM:~/.../task3$ cat col1 suffix > file1.sh; cat col2 suffix > file2.sh
[12/06/24]seed@VM:~/.../task3$
[12/06/24]seed@VM:~/.../task3$ openssl dgst -md5 file1.sh file2.sh
MD5(file1.sh)= e92a3a3f802085066d580ab085ca828b
MD5(file2.sh)= e92a3a3f802085066d580ab085ca828b
[12/06/24]seed@VM:~/.../task3$
[12/06/24]seed@VM:~/.../task3$ openssl dgst -sha256 file1.sh file2.sh
SHA256(file1.sh)= e132955e62256f67d0600533bb0a05fdccca02ea1731356037c47bcfd104e782
SHA256(file2.sh)= 8521c84ccac262ab4be8b125305648f83c73b999e460306780e44ffc54157554
[12/06/24]seed@VM:~/.../task3$
[12/06/24]seed@VM:~/.../task3$
```

- Finally, let's show the output indicating the different behaviors of the two files despite having the same MD5 hash.

```
[12/06/24]seed@VM:~/.../task3$ # now let's run the two bash scripts and see
[12/06/24]seed@VM:~/.../task3$
[12/06/24]seed@VM:~/.../task3$ ./file1.sh
bash: ./file1.sh: Permission denied
[12/06/24]seed@VM:~/.../task3$ chmod +x file1.sh
[12/06/24]seed@VM:~/.../task3$ chmod +x file2.sh
[12/06/24]seed@VM:~/.../task3$
[12/06/24]seed@VM:~/.../task3$ ./file1.sh
The sha256 digest is e676f2a94fed8de520e34f1b75e19f0e1f2055adf0b02fd12c109c74d92860d0
[12/06/24]seed@VM:~/.../task3$
[12/06/24]seed@VM:~/.../task3$
[12/06/24]seed@VM:~/.../task3$
[12/06/24]seed@VM:~/.../task3$ ./file2.sh
The sha256 digest is d12c9b5c1a05bf19a8c4f3986d369f0223f52925b025ea39247a21ee93dc2720
[12/06/24]seed@VM:~/.../task3$
```

- Now, let's observe the same process for the good and evil scripts that I made. The snapshot below shows the initial process of generating the collision files.

```
[12/09/24]seed@VM:~/.../task3$ ./fastcoll -p prefix -o col3 col4
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'col3' and 'col4'
Using prefixfile: 'prefix'
Using initial value: 0a0457cfbc3f8e32ee1c902d6927257b

Generating first block: ....
Generating second block: S11.........
Running time: 2.39565 s
```

- Now I show the concatenation of the collision files with the suffix. Also, we see that MD5 hash for the two files, good.sh and evil.sh are the same as expected.

```
[12/09/24]seed@VM:~/.../task3$
[12/09/24]seed@VM:~/.../task3$ cat col3 suffix > good.sh
[12/09/24]seed@VM:~/.../task3$ cat col4 suffix > evil.sh
[12/09/24]seed@VM:~/.../task3$
[12/09/24]seed@VM:~/.../task3$ openssl dgst -md5 good.sh evil.sh
MD5(good.sh)= 5fd2acc36d09ae82eee51b9a18331f23
MD5(evil.sh)= 5fd2acc36d09ae82eee51b9a18331f23
[12/09/24]seed@VM:~/.../task3$
```

- As expected, we as next that the SHA256 hash for the same files are different.

```
[12/09/24]seed@VM:~/.../task3$
[12/09/24]seed@VM:~/.../task3$ openssl dgst -sha256 good.sh evil.sh
SHA256(good.sh)= 64c2681e9163e796c0f247fcf0d92d80a4834d7a17376e1af775a3efc8697a45
SHA256(evil.sh)= da559610d8562eef7196b11fa183475069605e9e05ac4339f3621e9cbf674676
[12/09/24]seed@VM:~/.../task3$
```

- Finally, we see the snapshot of running good.sh and evil.sh.

```
[12/09/24]seed@VM:~/.../task3$
[12/09/24]seed@VM:~/.../task3$ bash good.sh

I mean no harm.

[12/09/24]seed@VM:~/.../task3$
[12/09/24]seed@VM:~/.../task3$ bash evil.sh

Your computer is hacked!

[12/09/24]seed@VM:~/.../task3$
```

Task 4: [Extra Credit]

Here, the goal is to perform MD5 collison in other file formats. Previously, we worked with bash(.sh) files. In my case, I would use image(.png) files: mouse.png and click.png. The image below shows the execution of the program to generate the collision files for the two images mentioned above.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS

[12/09/24]seed@VM:~/.../task4$ cd scripts/
[12/09/24]seed@VM:~/.../scripts$ time ./png.py mouse.png click.png

real    0m0.273s
user    0m0.062s
sys     0m0.028s
[12/09/24]seed@VM:~/.../scripts$ md5sum collision*.png
bdc9b618d22777ee485ccb94b8e933ea  collision1.png
bdc9b618d22777ee485ccb94b8e933ea  collision2.png
beccfc9c9a9ceee94854dc1fe7f0cdc4  collision-crc1.png
beccfc9c9a9ceee94854dc1fe7f0cdc4  collision-crc2.png
6a7fb9ea68dd05c463d099e0110acf03  collision-sync1.png
6a7fb9ea68dd05c463d099e0110acf03  collision-sync2.png
[12/09/24]seed@VM:~/.../scripts$
[12/09/24]seed@VM:~/.../scripts$
[12/09/24]seed@VM:~/.../scripts$ file collision*.png
collision1.png:      data
collision2.png:      data
collision-crc1.png:  data
collision-crc2.png:  data
collision-sync1.png: data
collision-sync2.png: data
[12/09/24]seed@VM:~/.../scripts$
```

- Finally, I show the comparion of the MD5 and SHA256 hashes for the collison files generated. We clearly see that the MD5 hashes are identical while the SHA256 hashes for the same files are different as expected.

```
[12/09/24]seed@VM:~/.../scripts$
[12/09/24]seed@VM:~/.../scripts$ # Let's see the MD5 hash of the two files
[12/09/24]seed@VM:~/.../scripts$ openssl dgst -md5 collision1.png collision2.png
MD5(collision1.png)= bdc9b618d22777ee485ccb94b8e933ea
MD5(collision2.png)= bdc9b618d22777ee485ccb94b8e933ea
[12/09/24]seed@VM:~/.../scripts$
[12/09/24]seed@VM:~/.../scripts$
[12/09/24]seed@VM:~/.../scripts$ # Now let's see the hash of SHA256
[12/09/24]seed@VM:~/.../scripts$ openssl dgst -sha256 collision1.png collision2.png
SHA256(collision1.png)= 4a325ee116bea4260030819644972ab4b9240b84e4ff9a90d3f5bbf8a92fd02d
SHA256(collision2.png)= ab5e813bac2d8af443de9bf9fe775768bdd5caf0599f5432806c373e77f9f4de
[12/09/24]seed@VM:~/.../scripts$
```