Introduction to Information Security - CS 458

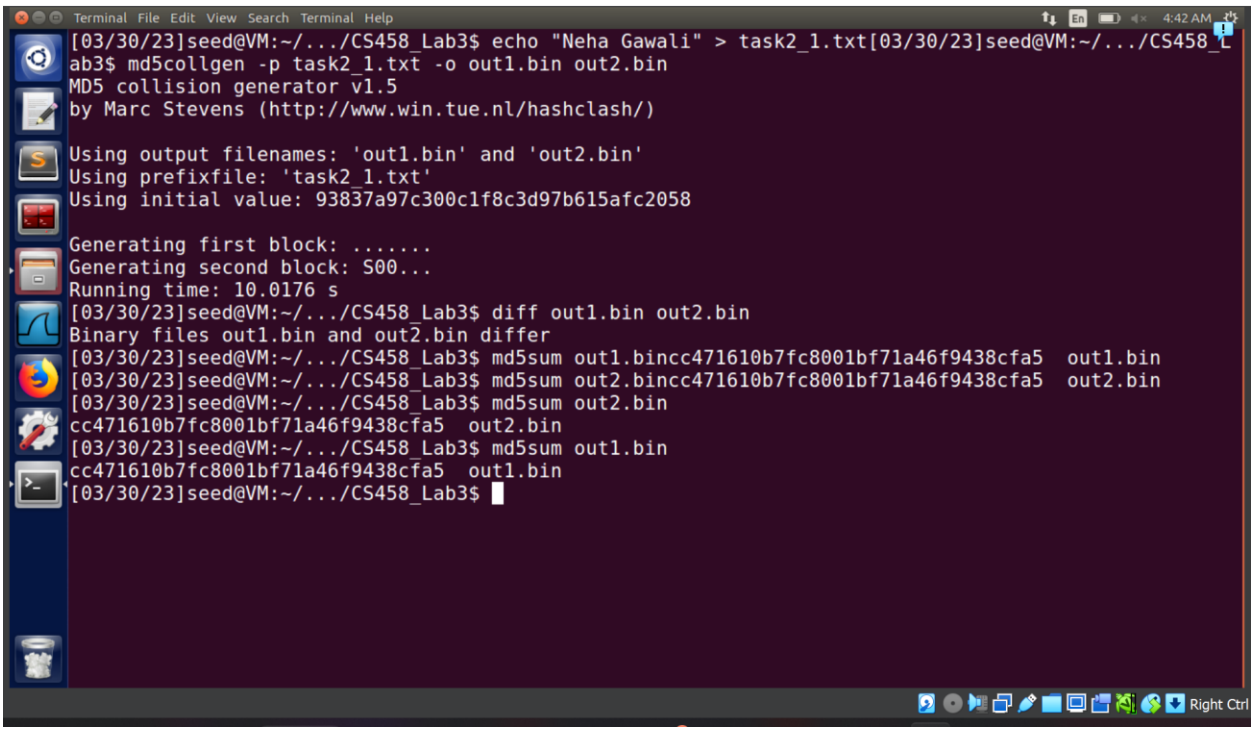
Neha Ramesh Gawali A20523722

Lab 3 - MD5 Collision Attack Lab

2.1 Task 1: Generating Two Different Files with the Same MD5 Hash

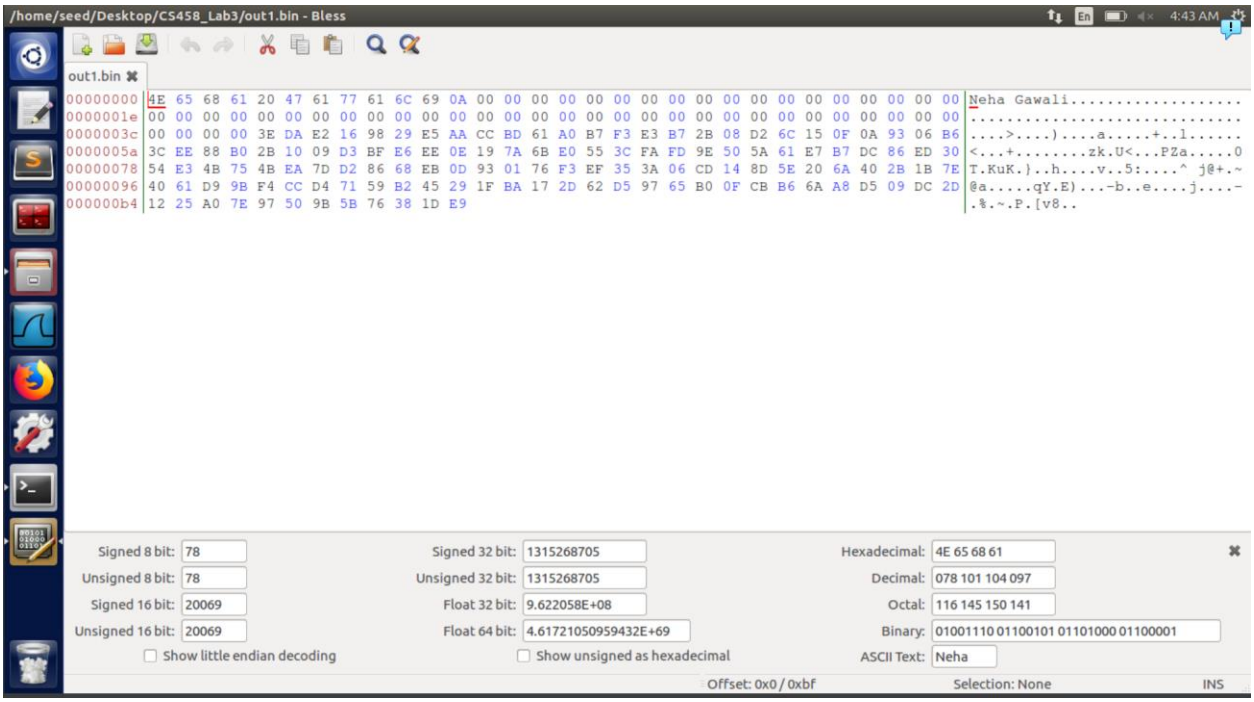
When I opened the two output files, out1.bin and out2.bin, in the hex editor, I could see that the first part (i.e., prefix) of both files is identical. However, the remaining parts of the two files are different. Even though the files are different, they share the same MD5 hash value.

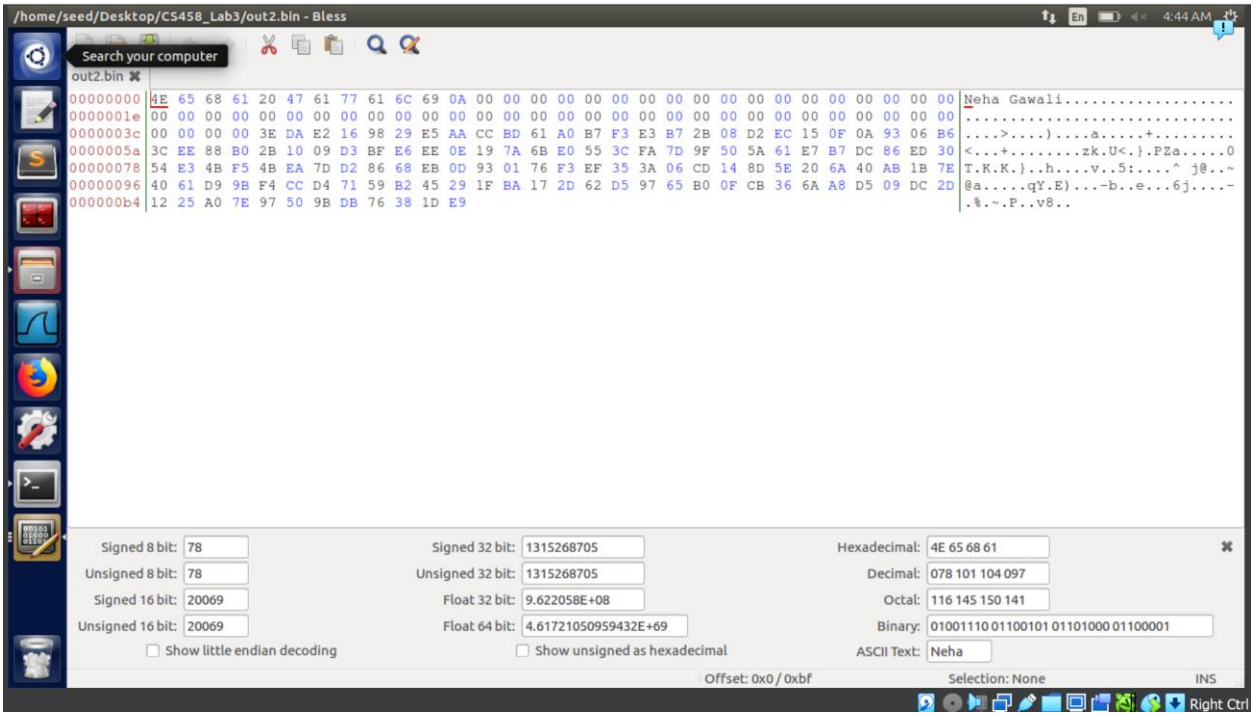
Question 1. If the length of your prefix file is not multiple of 64, what is going to happen?



Introduction to Information Security - CS 458

Neha Ramesh Gawali A20523722





Ans: Files are padded with zeros.

1. Here I created one 11-bit file with content “Neha Gawali” and named it as task2_1.txt
2. Using command: $ md5collgen -p prefix.txt -o out1.bin out2.bin
3. Generates two output files, out1.bin and out2.bin, for a given a prefix file task2_1.txt

Introduction to Information Security - CS 458

Neha Ramesh Gawali A20523722

4. To check whether the output files are distinct or not Used the diff command:

   md5sum command to check the MD5 hash of each output file. See the following commands.

   $ diff out1.bin out2.bin

   $ md5sum out1.bin

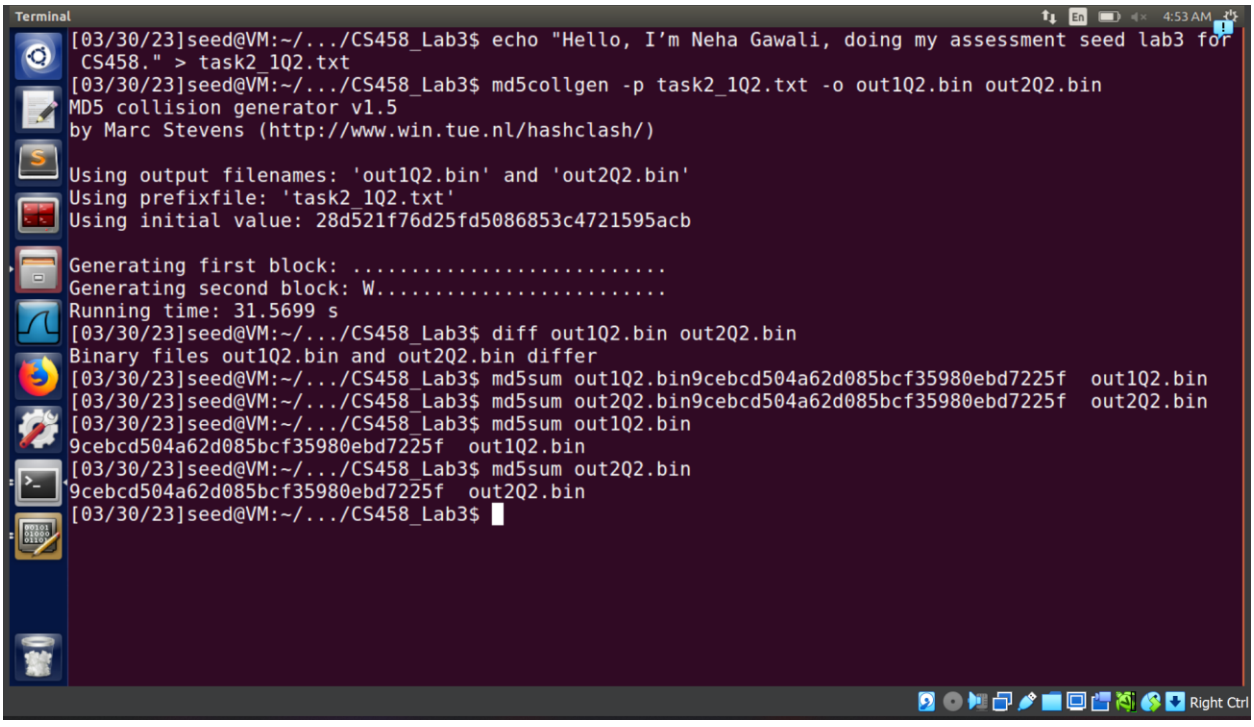
   $ md5sum out2.bin

   Diff command showed out1.bin and out2.bin are different.

5. Opened both out1.bin and out2.bin in hex bless, I observed both files contain same values.

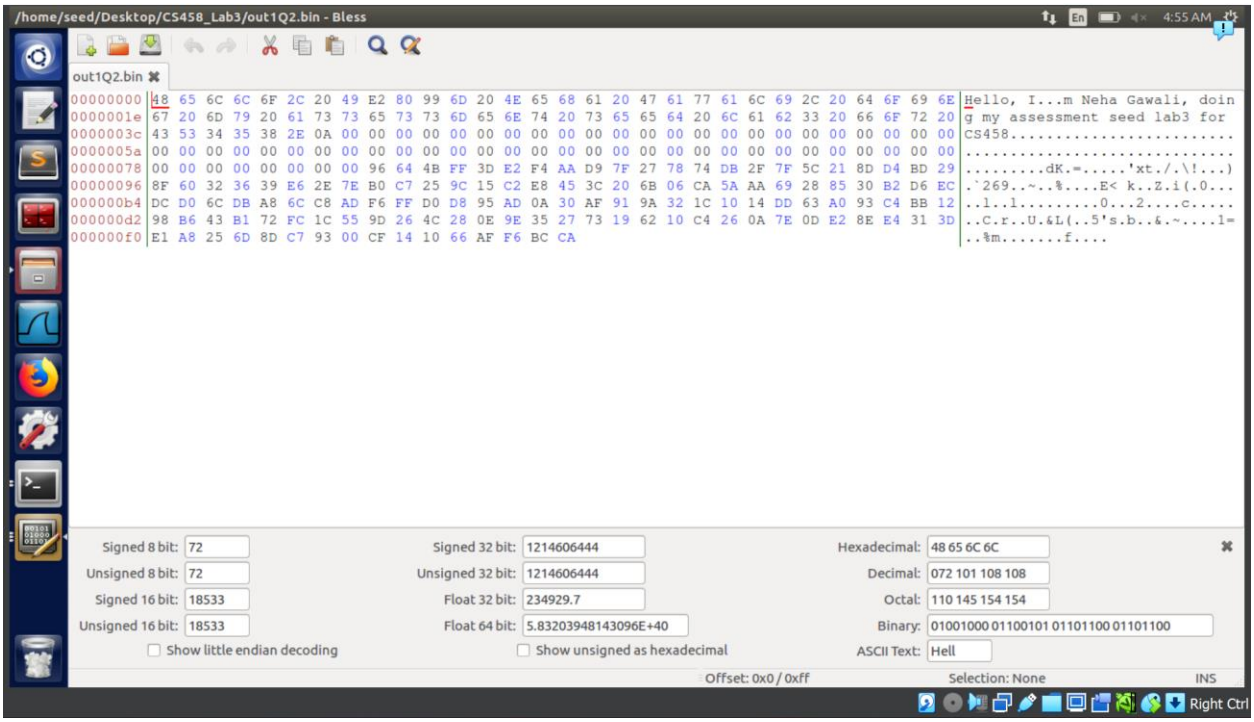
Observation:
If the length of the prefix file is not a multiple of 64, then md5collgen will pad the prefix with zeros to make it a multiple of 64. This is because MD5 operates on blocks of 64 bytes. Therefore, if the length of the input is not a multiple of 64, the algorithm pads the input with zeros until its length is a multiple of 64. This padding may change the MD5 hash of the file.

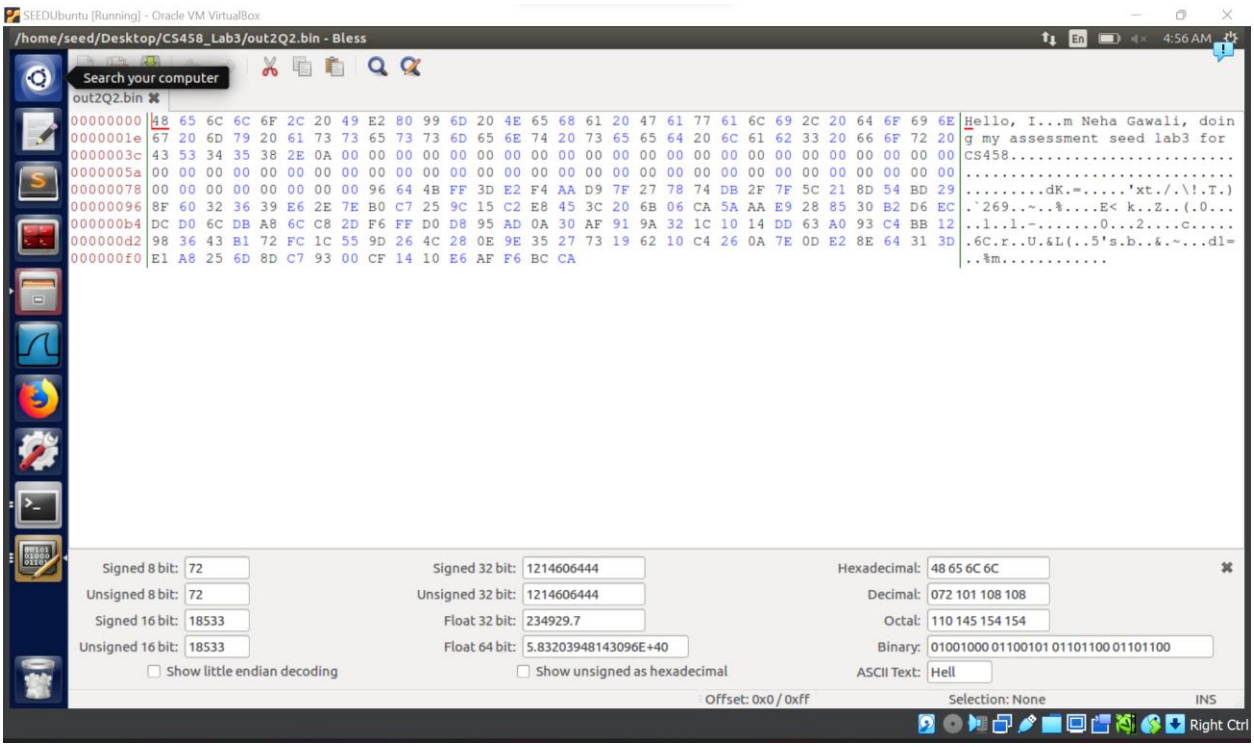
Question 2. Create a prefix file with exactly 64 bits file, and run the collision tool again, and see what happens :



Introduction to Information Security - CS 458

Neha Ramesh Gawali A20523722





Ans:

Here,

Introduction to Information Security - CS 458

Neha Ramesh Gawali A20523722

1. Created a file with exact 64 bits. Content: Hello, I'm Neha Gawali, doing my assessment seed lab3 for CS458. Saved it as task2_1Q2.txt.
2. Using command: $ md5collgen -p task2_1Q2.txt -o out1Q2.bin and out2Q2.bin
3. Generates two output files, out1Q1.bin and out2Q2.bin , for a given a prefix file task2_1Q2.txt
4. To check whether the output files are distinct or not Used the diff command:

md5sum command to check the MD5 hash of each output file. See the following commands.

$ diff out1Q2.bin out2Q2.bin

$ md5sum out1Q2.bin

$ md5sum out2Q2.bin

Diff command showed out1Q2.bin and out2Q2.bin are different.

5. Opened both out1Q2.bin and out2Q2.bin in hex bless, I observed both files contain same values.

After observing both the output files, shows that there were no zero-padding added.

Observation:
If the length of the prefix file is not a multiple of 64, then md5collgen will pad the prefix with zeros to make it a multiple of 64. This is because MD5 operates on blocks of 64 bytes. Therefore, if the length of the input is not a multiple of 64, the algorithm pads the input with zeros until its length is a multiple of 64. This padding may change the MD5 hash of the file.

Question 3: Question 3. Are the data (128 bytes) generated by md5collgen completely different for the two output files? Please identify all the bytes that are different.

1. Created a file with 128 bits. Content: "Hello there, this is Neha Ramesh Gawali reporting live from IIT. I'm planning to complete CS458 (infosec) this spring semester." Saved it as task2_1Q3.txt.
2. Using command: $ md5collgen -p task2_1Q3.txt -o out1Q3.bin and out2Q3.bin
3. Generates two output files, out1Q3.bin and out2Q3.bin , for a given a prefix file task2_1Q3.txt
4. To check whether the output files are distinct or not Used the diff command:

md5sum command to check the MD5 hash of each output file. See the following commands.

$ diff out1Q3.bin out2Q3.bin

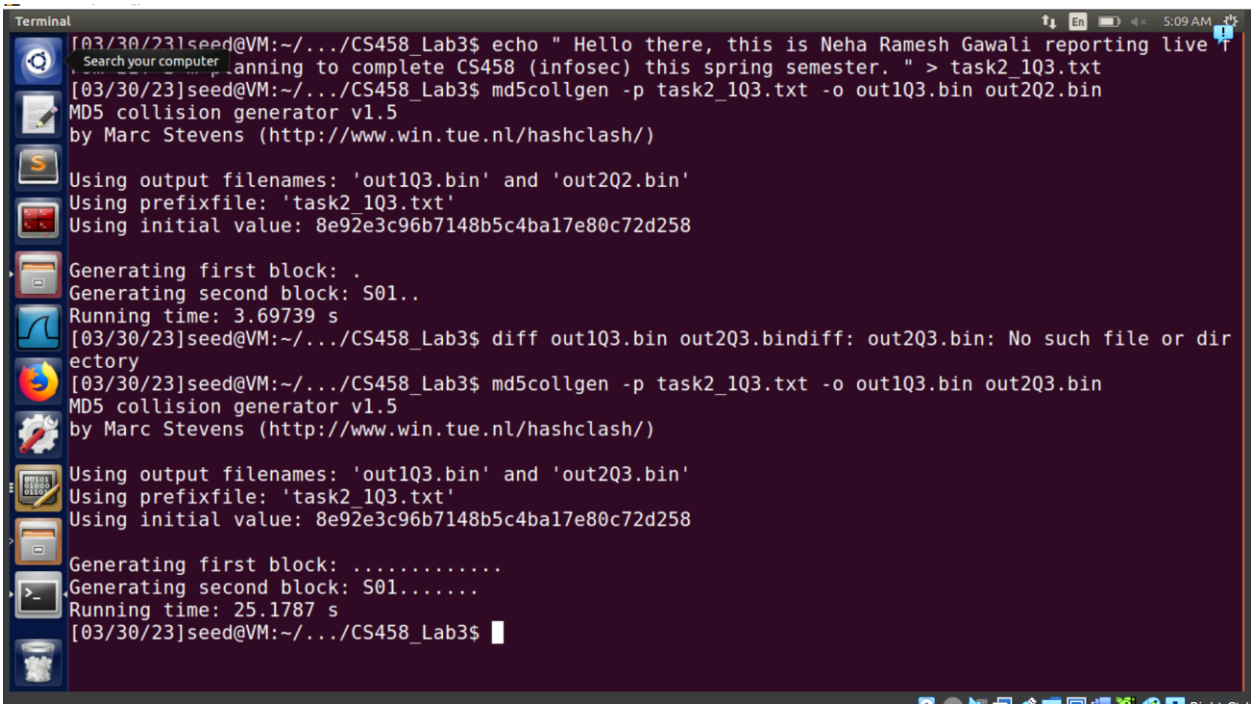
Introduction to Information Security - CS 458

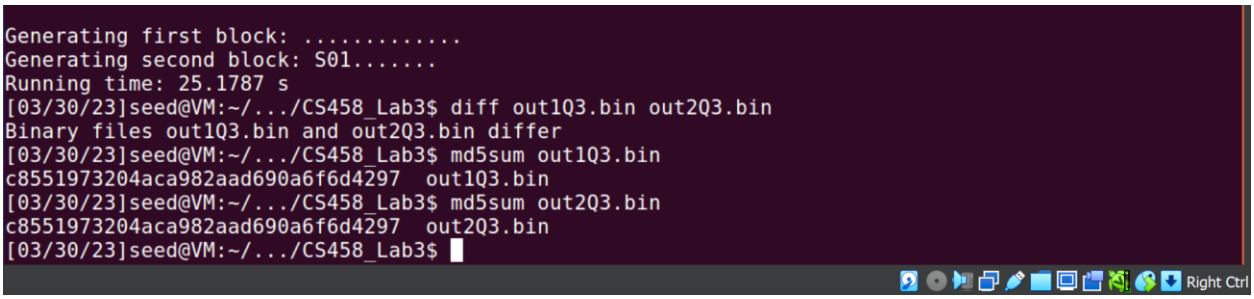
Neha Ramesh Gawali A20523722

$ md5sum out1Q3.bin

$ md5sum out2Q3.bin

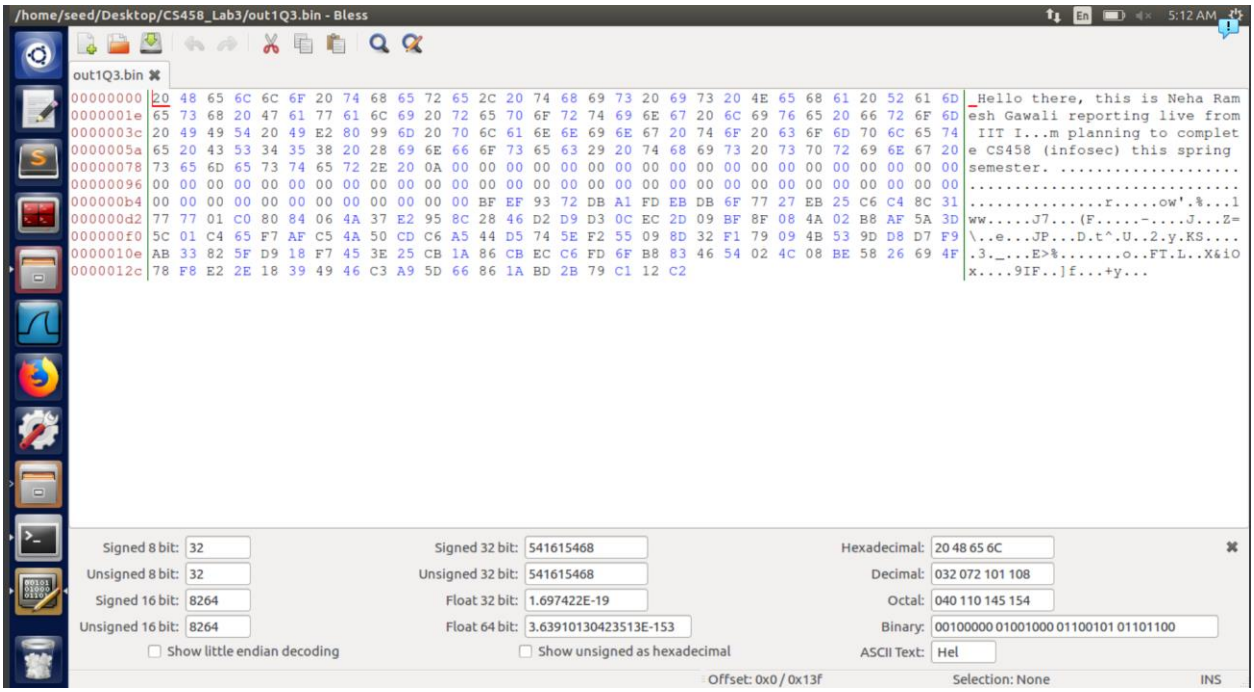
Diff command showed out1Q3.bin and out2Q3.bin are different.

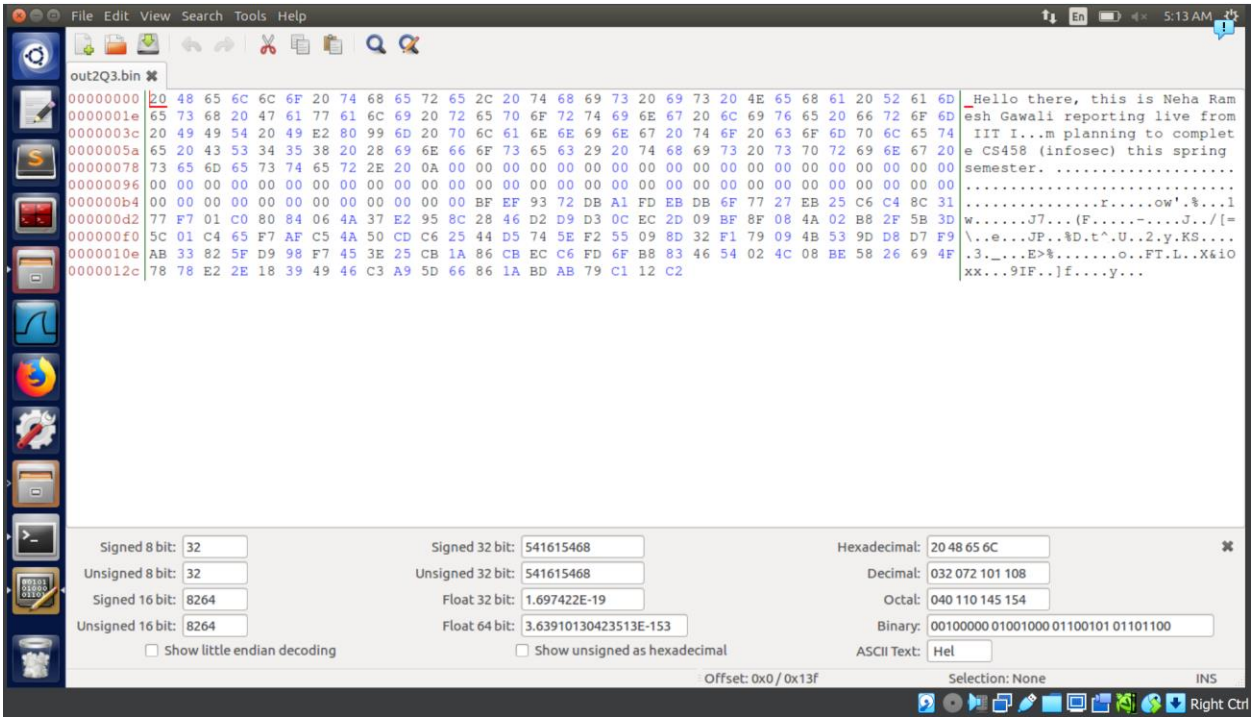




Introduction to Information Security - CS 458

Neha Ramesh Gawali A20523722





Observation:

It shows that there are some differences, just few bytes difference. After multiple trials, it is noted that these differences are not constant.The data generated by md5collgen is not completely different for the two output files. Only the last 64 bytes of each file are different, and the first 64 bytes (which come from the prefix file) are identical. Therefore, the bytes that are different are the last 64 bytes of out1.bin and out2.bin.

Introduction to Information Security - CS 458

Neha Ramesh Gawali A20523722

2.2 Task 2: Understanding MD5's Property

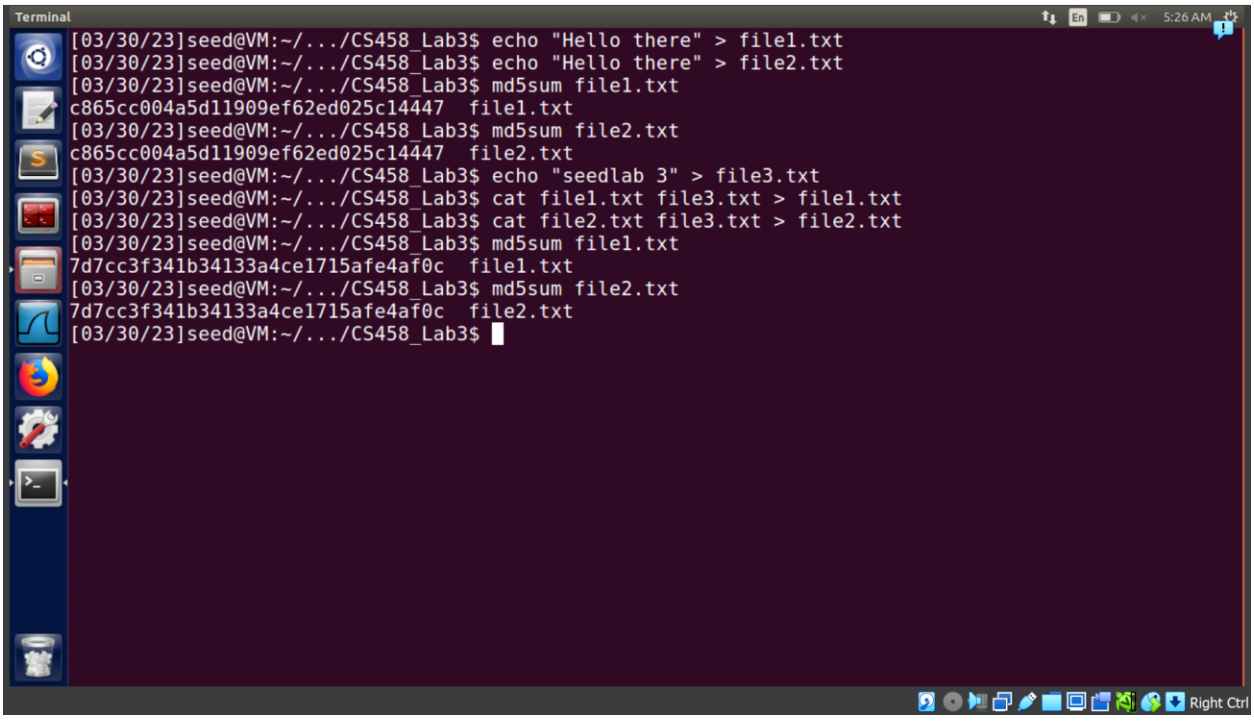
To conduct this experiment, I first generated two outputs using the md5sum command on two different files, let's call them file1 and file2. I then checked the MD5 hash of both files using the md5sum command to confirm that they were different.

Next, I concatenated a random string to the end of each file using the cat command, creating a new file as file3. I then checked the MD5 hash of new file using the md5sum command.

The results showed that even after adding the random string to each file, had the same MD5 hash value. This supports the property of the MD5 algorithm that states that if two inputs have the same hash, then adding the same suffix to both inputs will result in two outputs that have the same hash value.

The reason for this is that the MD5 algorithm processes the message in fixed blocks of 64 bytes and keeps an internal state by compressing the current blocks and state. Since the MD5 hashes of both file1 and file2 were the same, it can be inferred that the internal state of both files is also the same after algorithm execution. Therefore, when a random string is appended to each file, there will be a point at which the compression function for both files will run on the same block and state, resulting in both files having the same MD5 hash value.

In conclusion, this experiment demonstrates the property of the MD5 algorithm that if two inputs have the same hash, then adding the same suffix to both inputs will result in two outputs that have the same hash value.



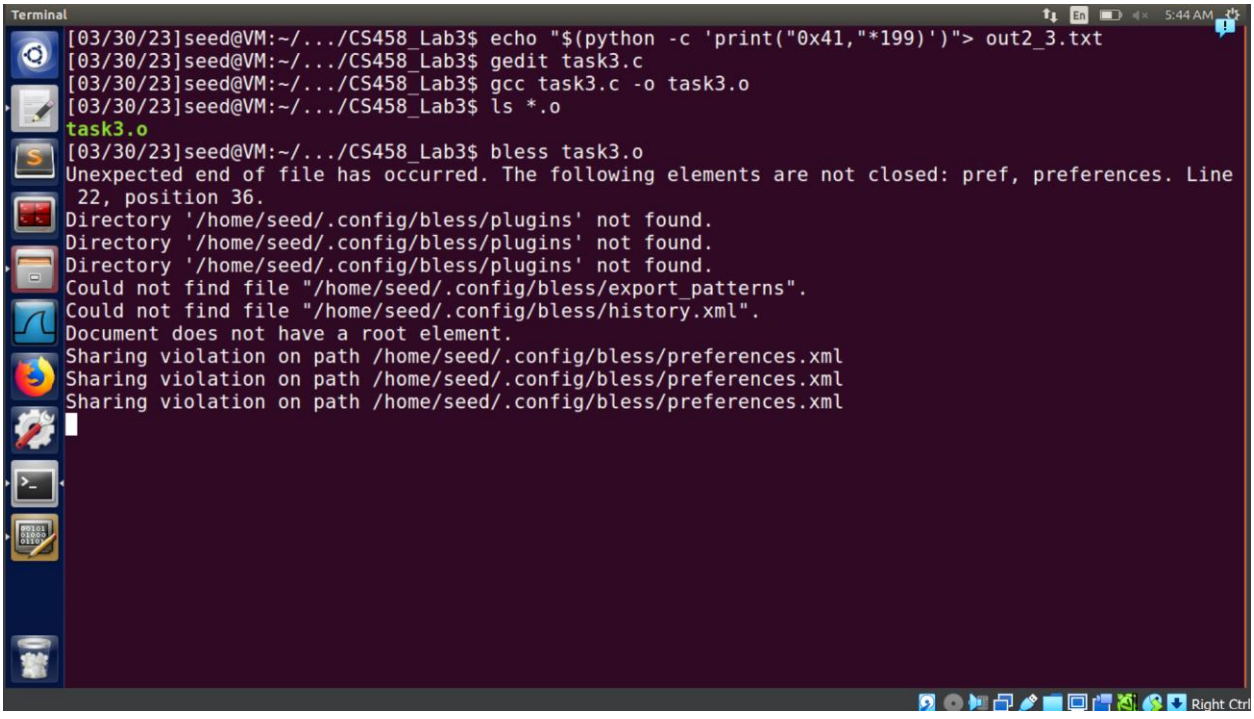
Introduction to Information Security - CS 458

Neha Ramesh Gawali A20523722

2.3 Task 3: Generating Two Executable Files with the Same MD5 Hash

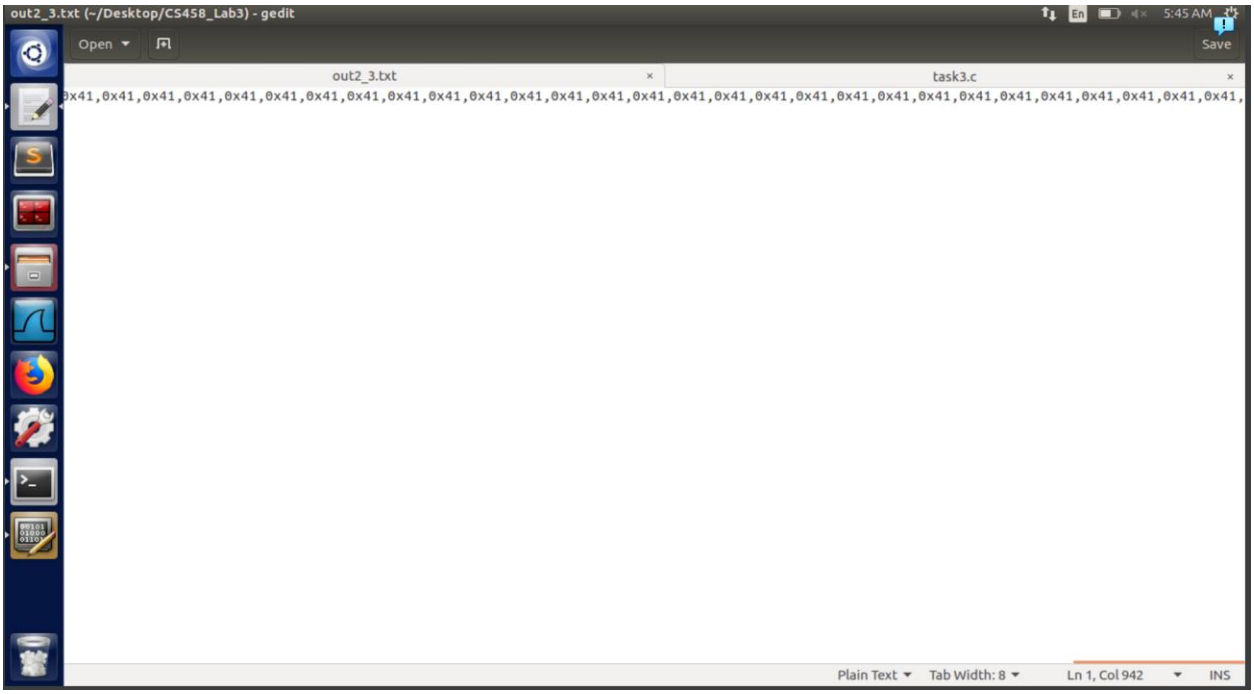
Based on the given procedure, the objective was to create two different binary files with the same MD5 hash value. The process involved creating a prefix and suffix for the binary files. The prefix was determined by dividing the total number of bytes (4160) by the prefix size (64), which resulted in 65. The prefix was then added to the total number of bytes, which resulted in 4224. This was then used as the prefix (task3p) for the binary files. Similarly, the suffix (task3s) was created by adding 128 to the total number of bytes, which resulted in 4353.

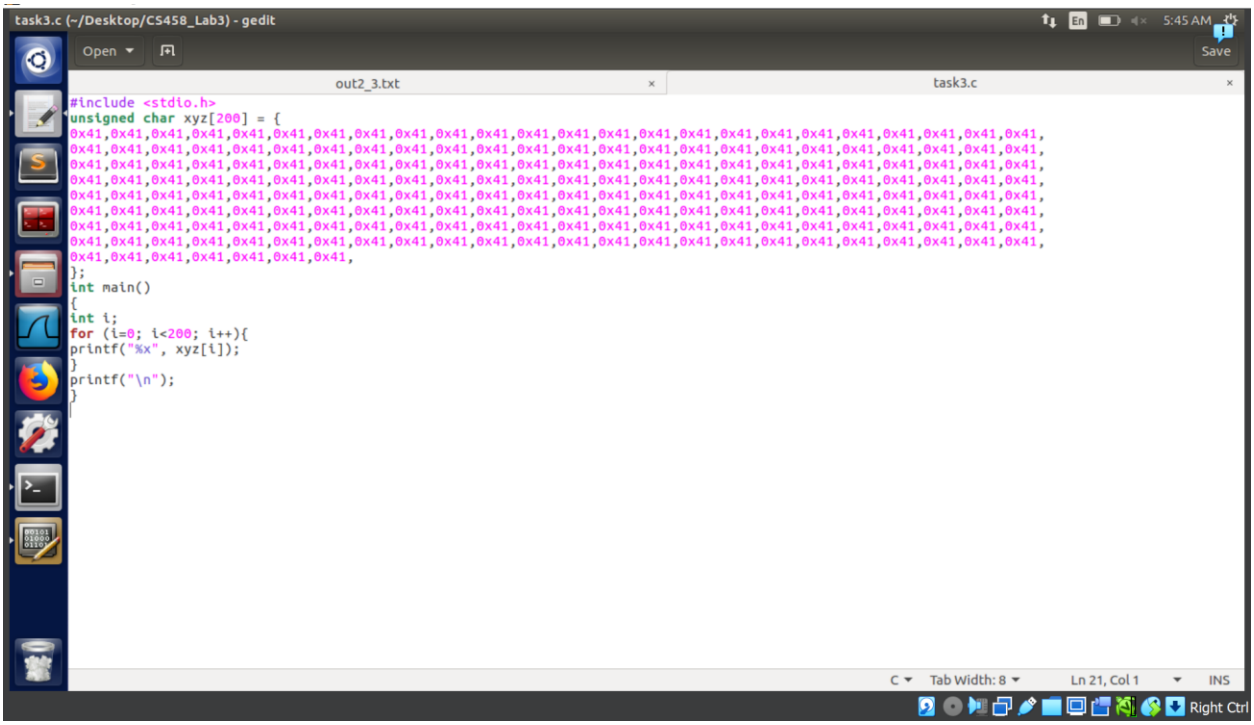
After creating the prefix and suffix, the p q values were taken from the bins file, and both the prefix and suffix were concatenated. This resulted in two binary files with the same MD5 hash value. These files were then checked using the bless command. It was observed that both the files had the same MD5 hash value but different suffixes. To validate the experiment, executable permission was added to both files and they were run. It was noted that the output was different, which proves that the experiment was successful in creating two different binaries from a single binary, both producing different output but having the same md5 hash.



Introduction to Information Security - CS 458

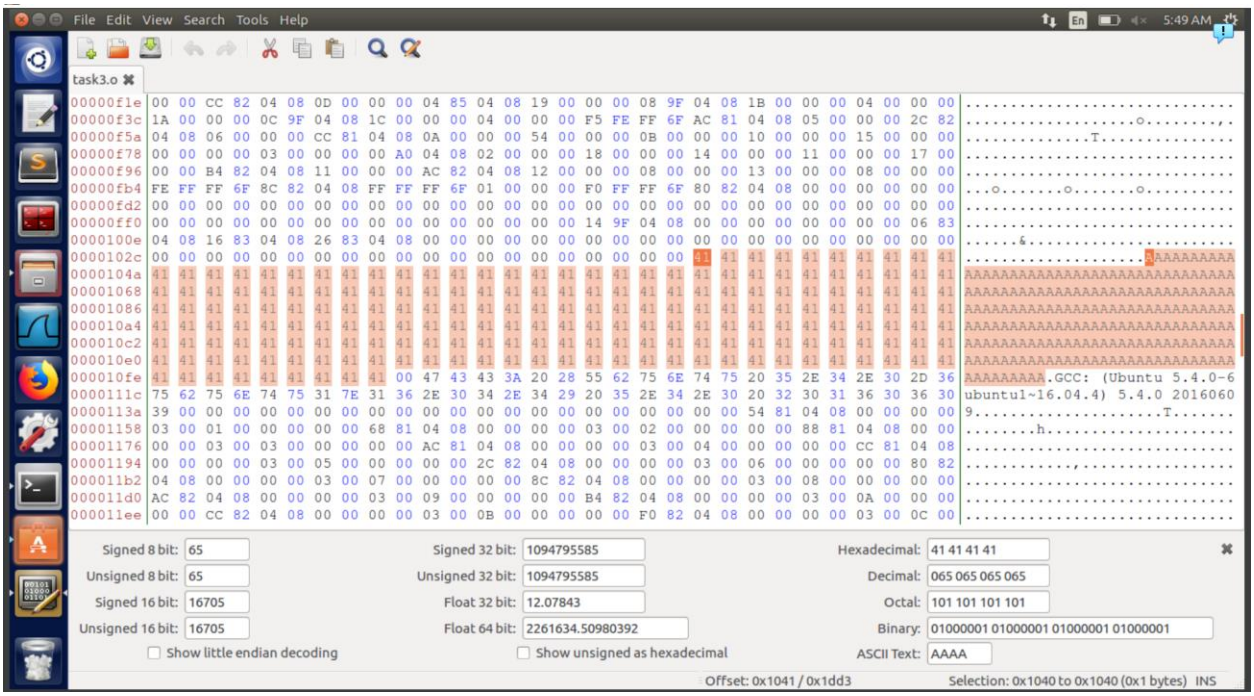
Neha Ramesh Gawali A20523722



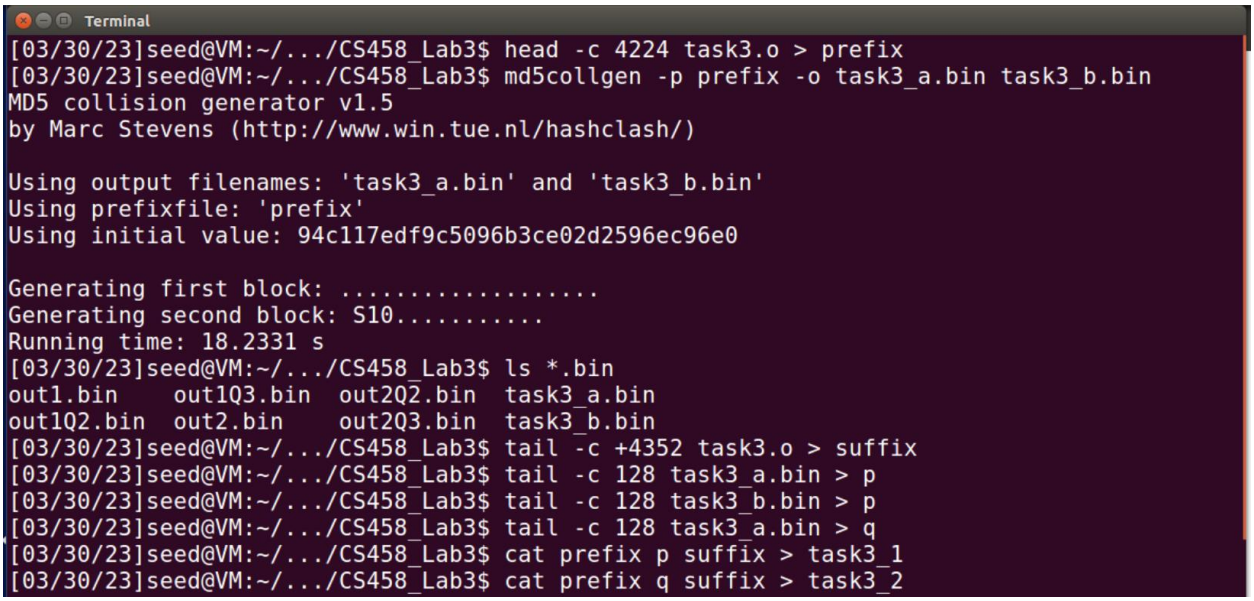


Introduction to Information Security - CS 458

Neha Ramesh Gawali A20523722



We can run md5collgen on the prefix to generate two outputs that have the same MD5 hash value .



Neha Ramesh Gawali A20523722

```
[03/30/23]seed@VM:~/.../CS458_Lab3$ md5sum task3_1
9b445ad46046ce2e7ee18edbf6f4adda  task3_1
[03/30/23]seed@VM:~/.../CS458_Lab3$ md5sum task3_2
9b445ad46046ce2e7ee18edbf6f4adda  task3_2
[03/30/23]seed@VM:~/.../CS458_Lab3$ chmod +x task3_1
[03/30/23]seed@VM:~/.../CS458_Lab3$ ./task3_1
41414141414141414141414141414141414141414141414141414141414141414141414141414141414141414
14141414141414141414141414141414141414113b8ecfc7aee138b66e5d4ac21908bd9f1607470d4b862e827
ae135a61fabd4c48c6ee891d6ceb217361e8c5c1711c440ca2631bf80757c9d167b4e30ed817dee7bd378e851
31f13f04b19721cad2f847545da2d04db5105f1a60a8b6d76bc58a746ae1376fb835ac277ec1274a99cc5f74b
0ad5d29cd76a82ea304141414141414141
[03/30/23]seed@VM:~/.../CS458_Lab3$
```

```
[03/30/23]seed@VM:~/.../CS458_Lab3$ chmod +x task3_2
[03/30/23]seed@VM:~/.../CS458_Lab3$ ./task3_2
41414141414141414141414141414141414141414141414141414141414141414141414141414141414141414
14141414141414141414141414141414141414113b8ecfc7aee138b66e5d4ac21908bd9f160f470d4b862e827
ae135a61fabd4c48c6ee891d6ceb217361e845c1711c440ca2631bf80757c9d967b4e30ed817dee7bd378e851
31f13f04b19721cad2f8487545da2d04db5105f1a60a8b6d76bc58a746ae1376fb835ac2f7ec1274a99cc5f74
b0ad5d29cdf6a82ea304141414141414141
[03/30/23]seed@VM:~/.../CS458_Lab3$
```

In conclusion, the experiment was successful in creating two different binary files with the same MD5 hash value. This could have significant implications in various fields such as cybersecurity and digital forensics.

Overall, this task demonstrates the potential vulnerabilities in relying solely on the MD5 hash for verifying the integrity of executable files, as it is possible to generate different files with the same hash.

2.4 Task 4: Making the Two Programs Behave Differently

```
Terminal
[03/30/23]seed@VM:~/.../CS458_Lab3$ gedit task2_4.c
```

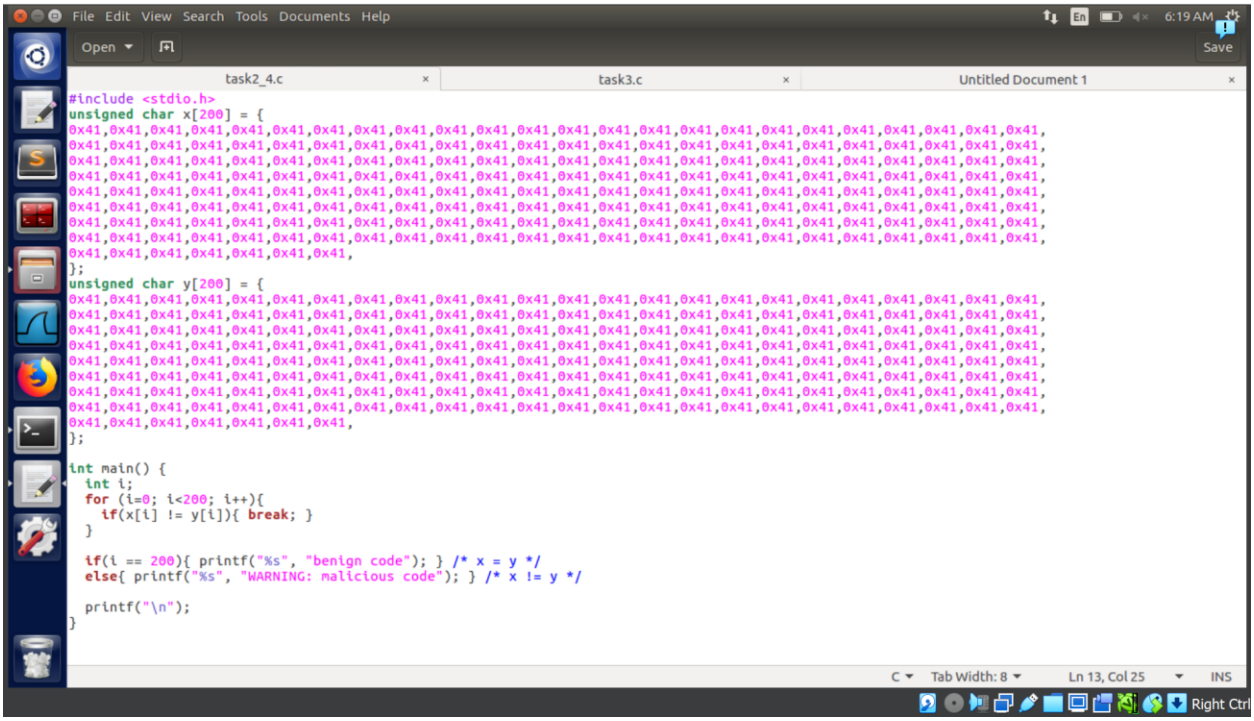
This experiment highlights the potential vulnerabilities in relying solely on the MD5 hash for verifying the integrity of executable files. It demonstrates how it is possible to create two programs with the same MD5 hash but different behaviour, and how one can use a benign program's MD5 hash to obtain certification for a malicious program.

To achieve this, the program was modified by incorporating two arrays, one containing benign code and the other containing malicious code. The prefix and suffix were identified, and the p and q values were obtained by generating two outputs using the md5 command. The benign code

was obtained by combining the prefix, p values, and suffix, while the malicious code was obtained by replacing the first p value with multiple qs and rearranging the layout.

The process of obtaining the desired output took around three hours of experimentation. This shows that with some effort, it is possible to create two different programs with the same MD5 hash, and that relying solely on the hash to verify program integrity can be risky.

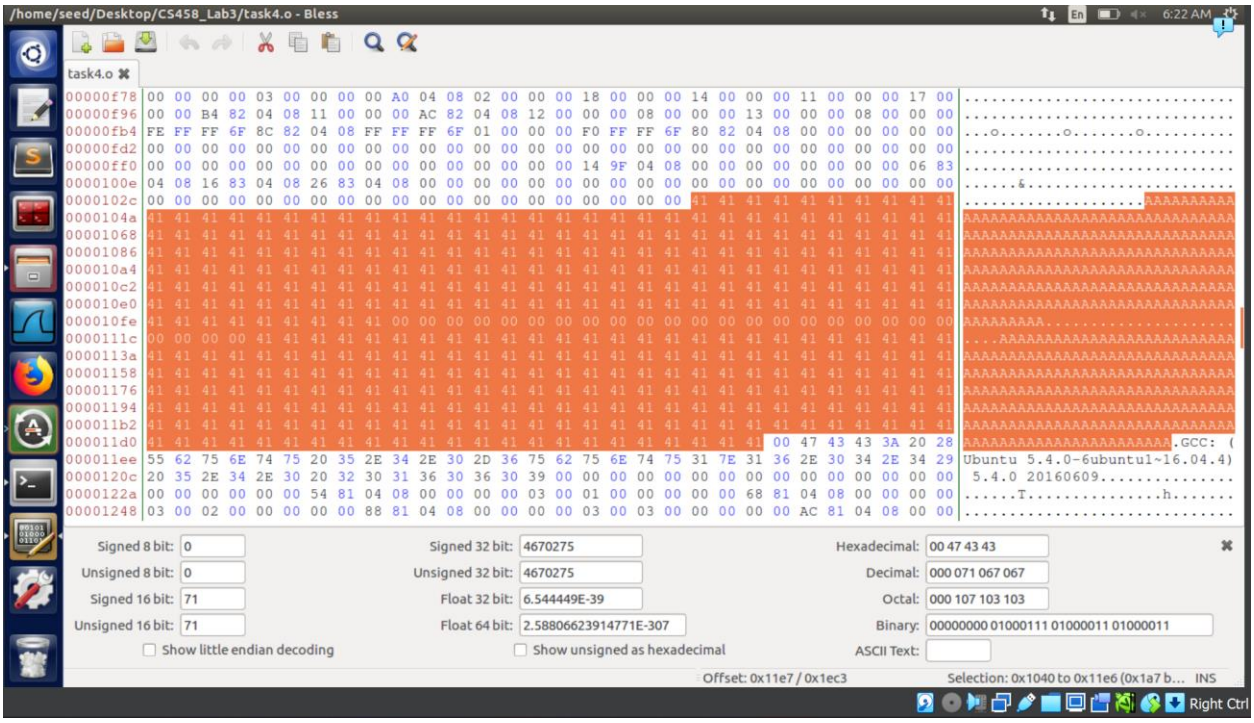
Overall, this experiment demonstrates the need for additional security measures beyond MD5 hash verification, such as digital signatures, code signing certificates, and other cryptographic methods to ensure that the program being executed is the intended one.



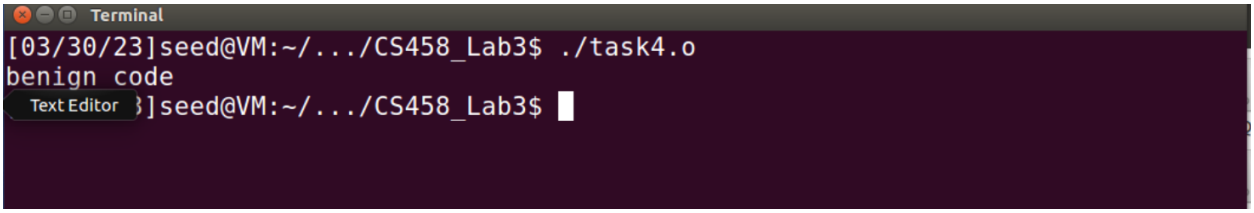

```
[03/30/23]seed@VM:~/.../CS458_Lab3$ gcc task2_4.c -o task4.o
[03/30/23]seed@VM:~/.../CS458_Lab3$ bless task4.o
```

Introduction to Information Security - CS 458

Neha Ramesh Gawali A20523722

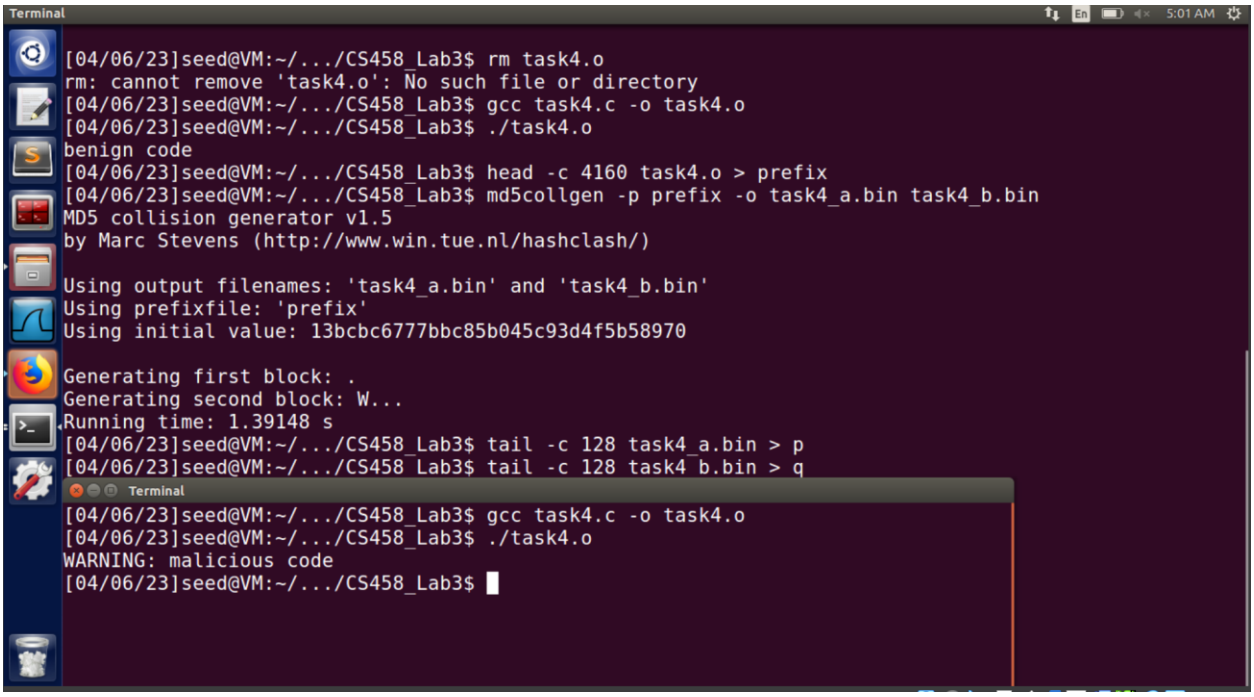


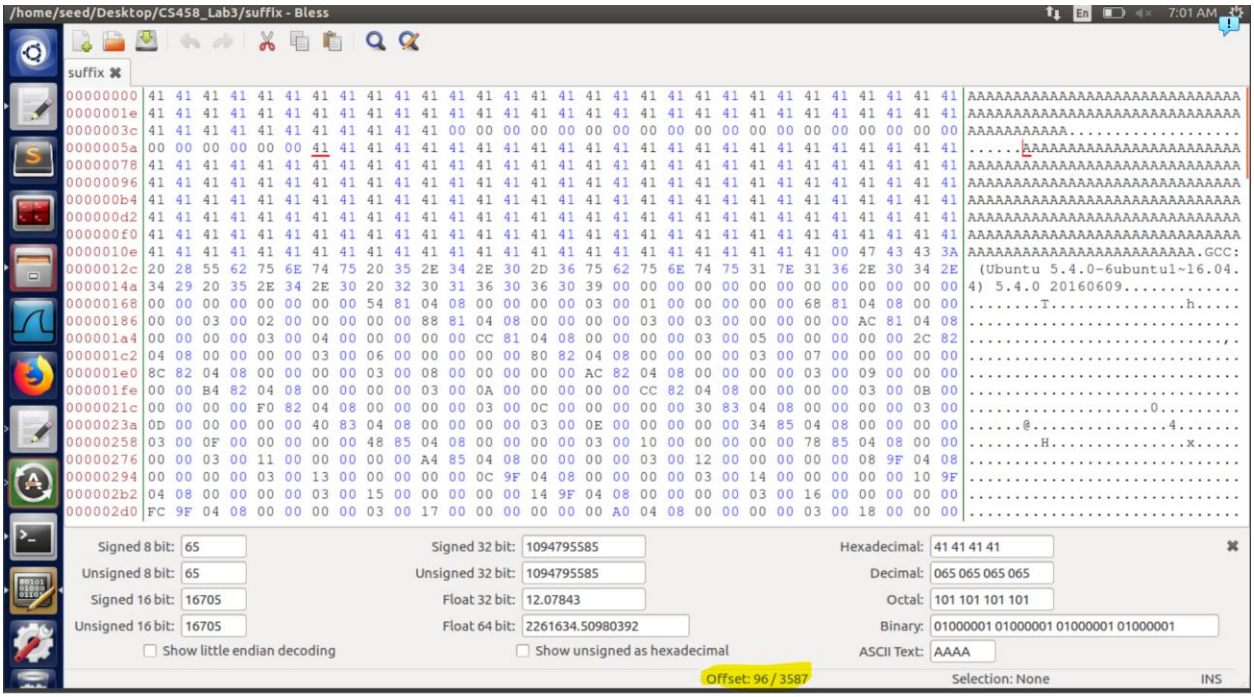

```
[03/30/23]seed@VM:~/.../CS458_Lab3$ head -c 4106 task4.o > prefix
[03/30/23]seed@VM:~/.../CS458_Lab3$ md5collgen -p prefix -o task4_a.bin task4_b.
bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'task4_a.bin' and 'task4_b.bin'
Using prefixfile: 'prefix'
Using initial value: 20f59f7e9f47c7021ccf15966ff2633c

Generating first block: .........
Generating second block: W..............
Running time: 19.6034 s
[03/30/23]seed@VM:~/.../CS458_Lab3$ tail -c 128 task4_a.bin > p
[03/30/23]seed@VM:~/.../CS458_Lab3$ tail -c 128 task4_b.bin > p
[03/30/23]seed@VM:~/.../CS458_Lab3$
```



Introduction to Information Security - CS 458

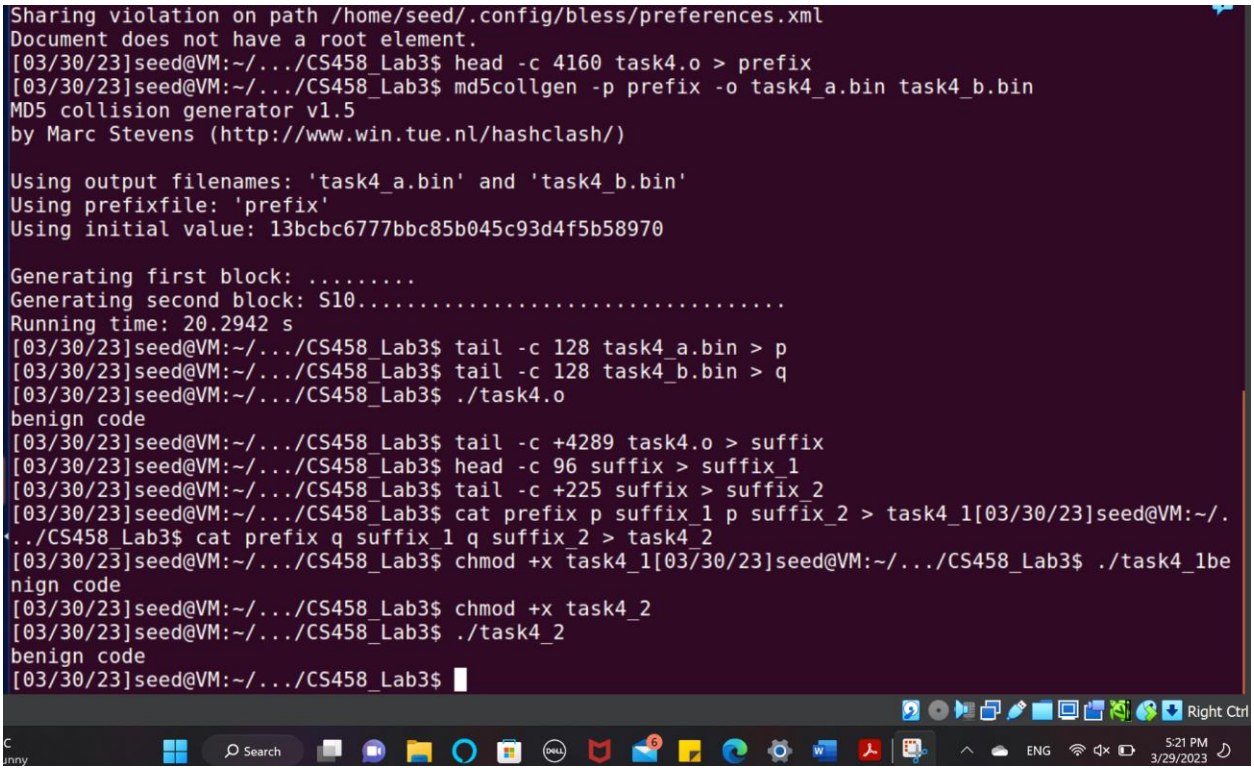
Neha Ramesh Gawali A20523722





Introduction to Information Security - CS 458

Neha Ramesh Gawali A20523722



hashes of both files will be the same even though their outputs are different. This demonstrates a potential vulnerability in relying solely on the MD5 hash to verify the integrity of executable files.

To achieve the desired result, the experiment involved creating two independent executable files that have the same prefix and suffix but different behavior. The prefix was created by dividing the total size of the binary file by 64, and the suffix was created by adding 128 bytes to the total size of the binary file. The P and Q values were taken from the binary file, and both files were concatenated with the prefix and suffix to create two executable files with the same MD5 hash.

To make one file output benign code and the other file output malicious code, the second array's contents were modified to be equivalent to one of the created arrays. The contents after the second array were added to the suffix, and the first array was transferred from out1arrcomplete to compare. This way, one file outputs benign code, and the other outputs where malicious code would be run.

Finally, both files were made executable, and their MD5 hash sums were calculated. Executing both files gives the desired results, and it is evident that the MD5 hashes of both files are the same, despite their different outputs.

Introduction to Information Security - CS 458

Neha Ramesh Gawali A20523722

Overall, this experiment highlights the potential vulnerabilities in relying solely on the MD5 hash to verify the integrity of executable files. As demonstrated in this experiment, it is possible to generate two different files with the same MD5 hash, which can lead to security risks if not properly addressed. It is important to use multiple verification methods to ensure the security and integrity of executable files.