Jaleel Rogers

Professor Mukhopadhyay

CIS 4204.01

02 February 2024

```
ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 1 MB

Dictionary cache hit:
* Filename..: rockyou.txt
* Passwords.: 14344384
* Bytes.....: 139921497
* Keyspace..: 14344384

5f4dcc3b5aa765d61d8327deb882cf99:password
482c811da5d5b4bc6d497ffa98491e38:password123
7ecc19e1a0be36ba2c6f05d06b5d3058:weak
```

Figure 1 - Executing Hashcat

For the entirety of the assignment, I used the Kali OS. The objective of the first part of the assignment is to create hash collisions by using a dictionary executed on hashcat. For the dictionary of my choice I used a total of three; rockyou, crackstation, and md5encrypt. For figure 1, I inserted the text file hash.txt containing the hashes 7ecc19e1a0be36ba2c6f05d06b5d3058, 5f4dcc3b5aa765d61d8327deb882cf99, 482c811da5d5b4bc6d497ffa98491e38, and f7bbdf9e9e4d3112c852f142cd6ddc7a using the rockyou dictionary. The command used was hashcat -a0 -m0 hashes.txt rockyou.txt. The reason I used three dictionaries is that not all hashes had a corresponding plaintext value in rockyou dictionary. Specifically, the hash f7bbdf9e9e4d3112c852f142cd6ddc7a in both rockyou and crackstation dictionaries.

Jaleel Rogers

Professor Mukhopadhyay

CIS 4204.01

02 February 2024

Figure 2 - Executing md5encrypt

In Figure 2, I used the third dictionary, md5 encrypt, which is a browser-based dictionary in order to get the plaintext. The plaintext for 7ecc19e1a0be36ba2c6f05d06b5d3058 is weak, 5f4dcc3b5aa765d61d8327deb882cf99 is password, 482c811da5d5b4bc6d497ffa98491e38 is password123, and f7bbdf9e9e4d3112c852f142cd6ddc7a is Smith@1998.

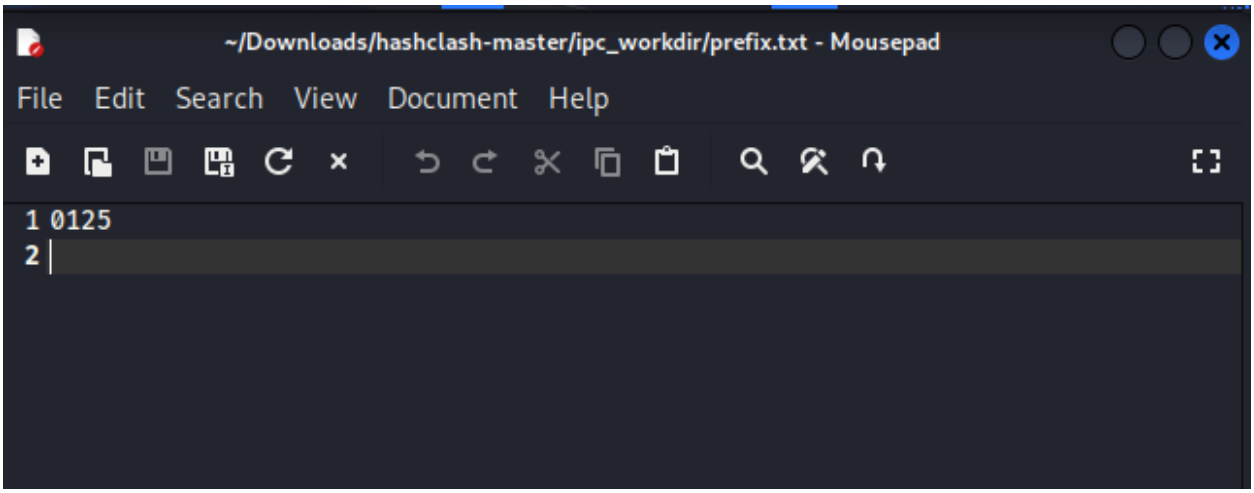

Figure 3 - prefix.txt file

Jaleel Rogers

Professor Mukhopadhyay

CIS 4204.01

02 February 2024

In the second part of the assignment the objective is to get two outputs in the form of binary files from a string input then hash those binary values to see if they result in the same hash. Figure 3, is the prefix.txt containing the string 0125.

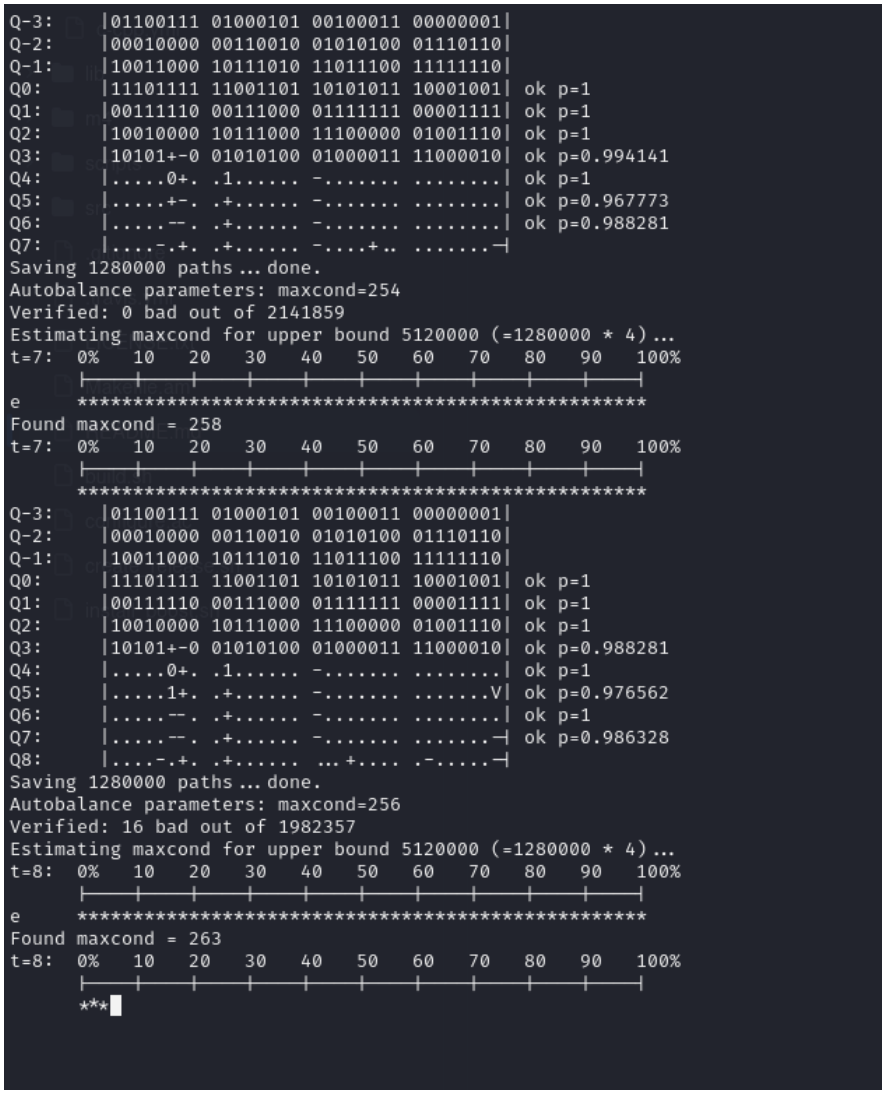

Figure 4 - Executing Hashclash

I used hashclash, a hash collision software to create the two binary files. However, in Figure 4 it kept looping for hours and would not stop, creating no output in the process.

Jaleel Rogers

Professor Mukhopadhyay

CIS 4204.01

02 February 2024

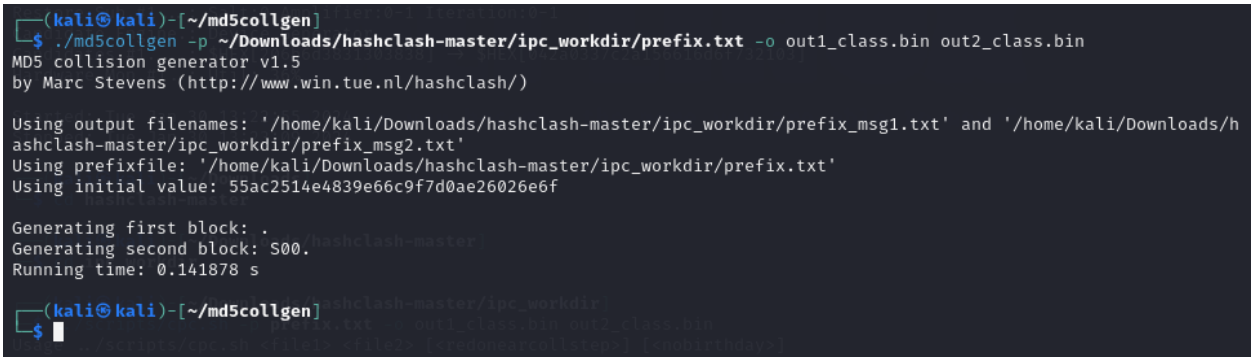

Figure 5 - Executing md5collgen

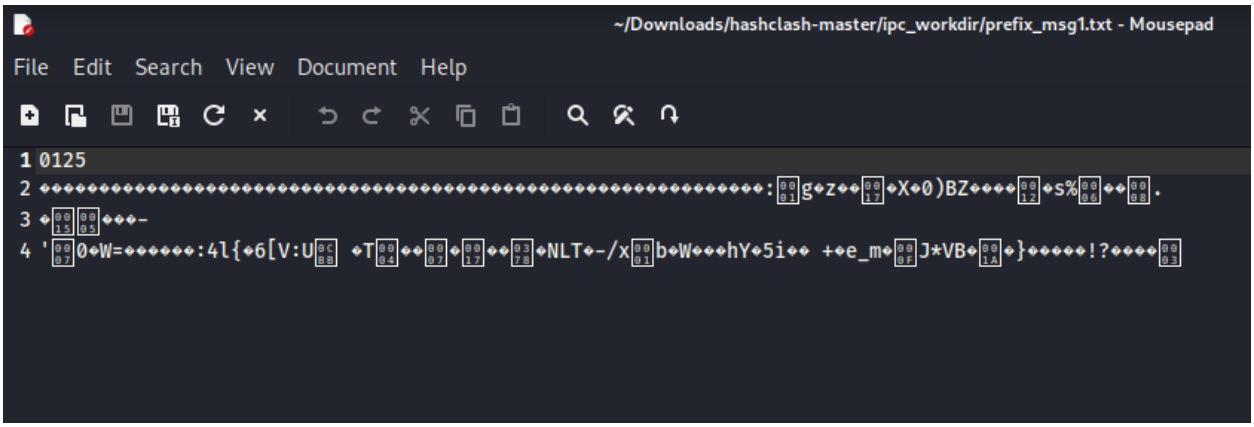

Figure 6 - msg1.txt file

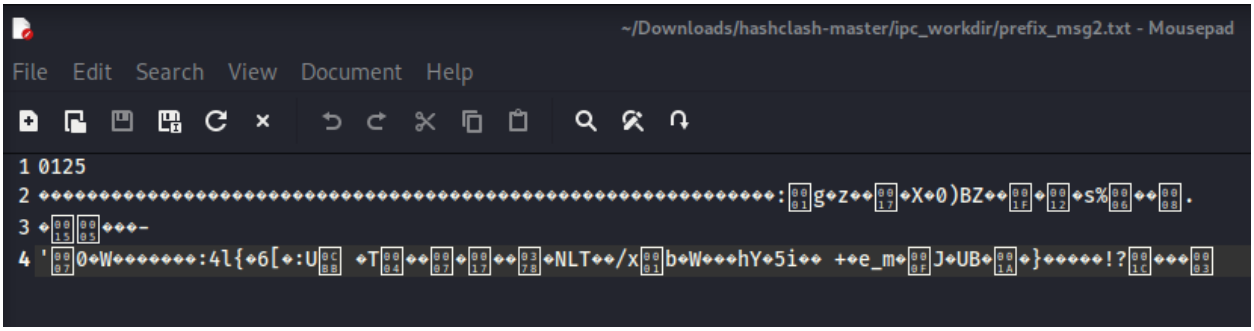

Figure 7 - msg2.txt file

Jaleel Rogers

Professor Mukhopadhyay

CIS 4204.01

02 February 2024

Figure 8 - msg1 and msg2 as binary files

I then used the md5collgen script to run the prefix txt as seen in Figure 5. Figure 6 and Figure 7 are the output of prefix.txt. In Figure 8, I used the xxd command to view the text files as binary files.

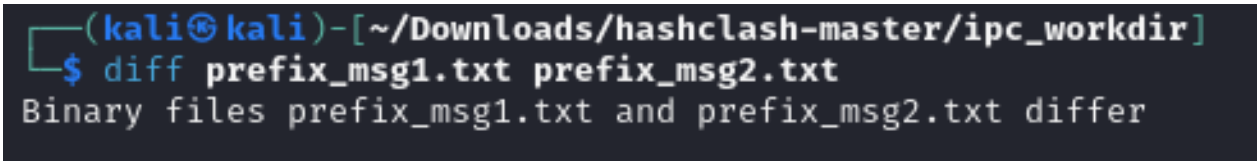

Figure 9 - Binary file comparison

In Figure 9, I then used the diff command to compare the two binary files as txt files. Both files have different values.

Jaleel Rogers

Professor Mukhopadhyay

CIS 4204.01

02 February 2024

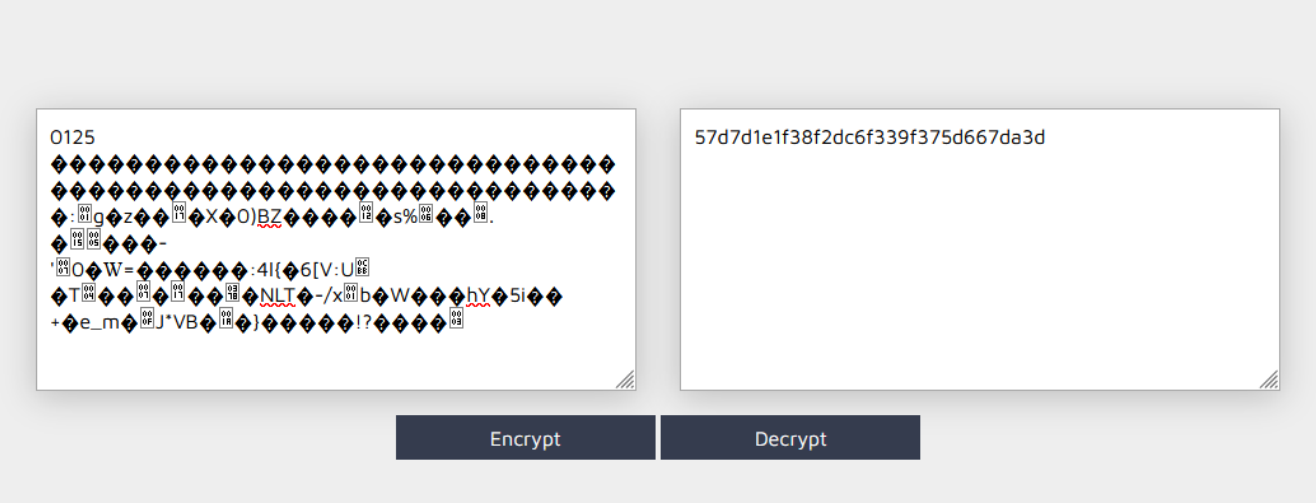

Figure 10 - msg1.txt hash

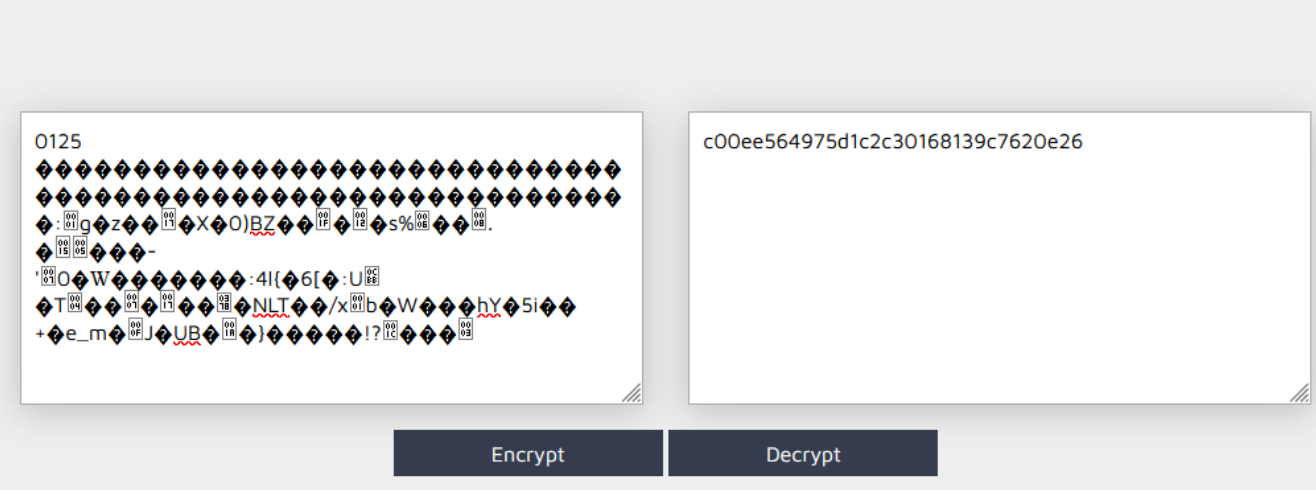

Figure 11 - msg2.txt hash

For both files they have different hash values as seen in Figures 10 and 11. I wasn't successfully able to hash the files to have the same output.