APPLIED CRYPTOGRAPHY

**Lab 7: MD5 Collision Attack Lab**

Name: Siri N Shetty SRN: PES2UG22CS556

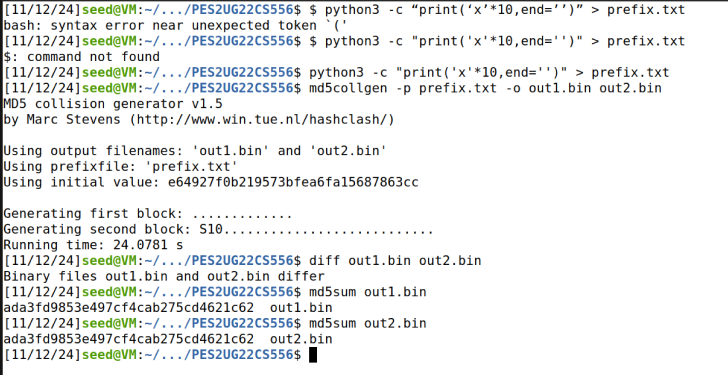
**Problem 1: Generating Two Different Files with the Same MD5 Hash**

Step 1 : Run the 5 commands for length 10.

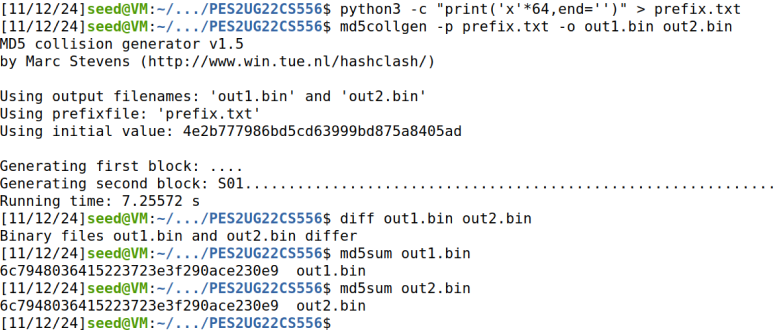
Step 2 : Run the 5 commands for length greater than 64.

Step 3 : Run the 5 commands for length 64.

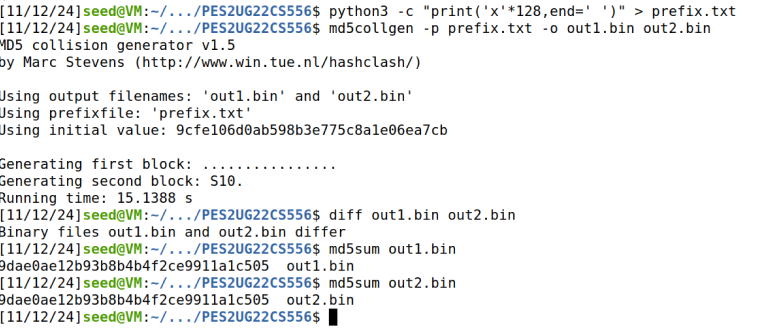
Expected Deliverables -   
i) Output Screenshot (Terminal should have SRN visible) for step 1



ii) Output Screenshot (Terminal should have SRN visible) for step 2



iii) Output Screenshot (Terminal should have SRN visible) for step 3



iv) Are the data (128 bytes) generated by md5collgen completely different for the two output files? Please identify all the bytes that are different. {answer for all of above i), ii), iii)}

Prefix 1 has 10 bytes: This file is not a multiple of 64 bytes. The MD5 hashing algorithm works on 64-byte blocks, so when the length of prefix is less than 64 bytes, the padding will be added to ensure that the input is a multiple of 64 bytes.

Prefix 2 has 64 bytes: Since this is exactly 64 bytes, the MD5 algorithm can directly process this block without padding.

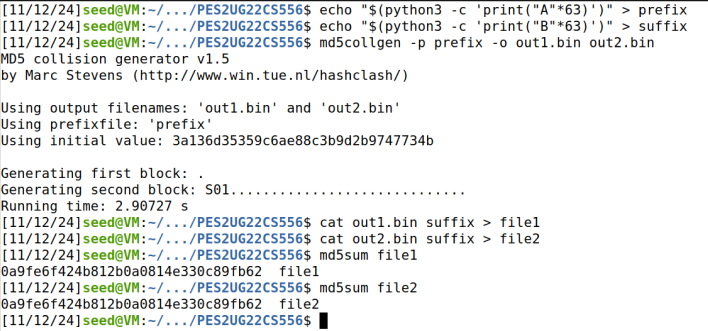
Prefix 3 has 70 bytes: This file exceeds 64 bytes, meaning it spills into the second block. The MD5 algorithm will need to process the first 64-byte block and then the remaining 6 bytes with padding to complete the second block.

No, the 128 bytes generated by md5collgen are not completely different. The tool introduces a controlled change that allows two distinct files to share the same MD5 hash. The differences in the files will likely be localized to certain bytes that md5collgen manipulates to achieve the hash collision.

**Problem 2 : Understanding MD5’s Property**

Step : Run the 7 commands

Expected Deliverables -   
i) Output Screenshot (Terminal should have SRN visible) for step 1



ii) Are the hashes same? Explain your observation.

The MD5 collision was created in out1.bin and out2.bin to intentionally generate the same hash, regardless of their internal differences.

When you append the same suffix to both files, it does not introduce enough change to break the hash collision, so both final files, file1 and file2 still have the same hash.

This result highlights the weakness of MD5 in collision resistance: **two different files can still produce the same hash, even after modifications**.

**Problem 3 : Generating Two Executable Files with the Same MD5 Hash**

Step 1 : Create the code file [Let us initialise the array with 200 A’s so that it is easy locate 200 A’s in the binary. ]

#include unsigned char xyz[200] = { "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"};

int main()

{

int i;

for(i = 0; i < 200; i++)

{

printf("%x", xyz[i]);

}

printf("\n");

}

Step 2: Compile the code and open the executable in a hex editor

1. gcc task3.c -o task3
2. bless task3

Step 3 : Truncate the prefix and suffix

$ head -c 12352 task3 > prefix

$ tail -c +12480 task3 > suffix

Step 4 : Generate two files with the prefix and append the suffix to the m to make normal programs. Make them executable.

$ md5collgen -p prefix -o P Q

$ cat P suffix > arr1.out

$ cat Q suffix > arr2.out

$ sudo chmod +x arr1.out arr2.out

Step 5: Verify the success of the task

$ ./arr1.out > f1

$ ./arr2.out > f2

# Compare the md5 sums

$ md5sum arr1.out

$ md5sum arr2.out

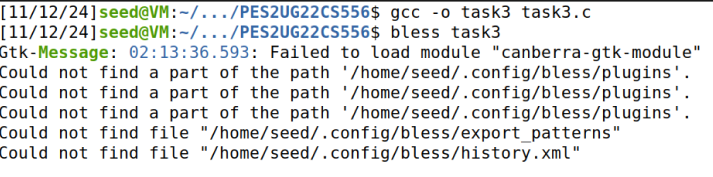
# Compare the output of the programs

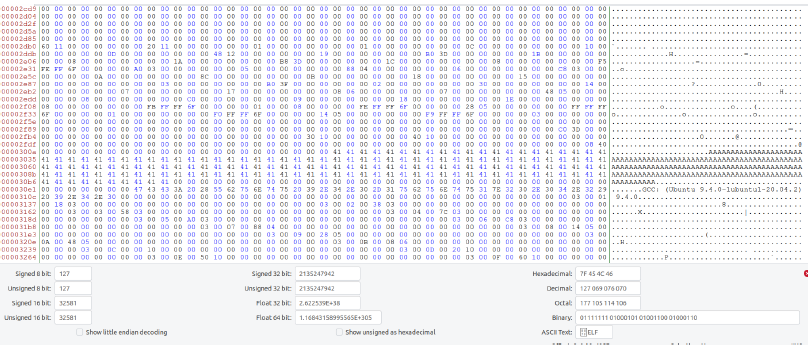
$ diff f1 f2

Expected Deliverables -   
i) Output Screenshot (Terminal should have SRN visible) for step 1



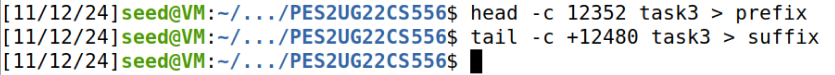
ii) Output Screenshot (Terminal should have SRN visible) for step 2.



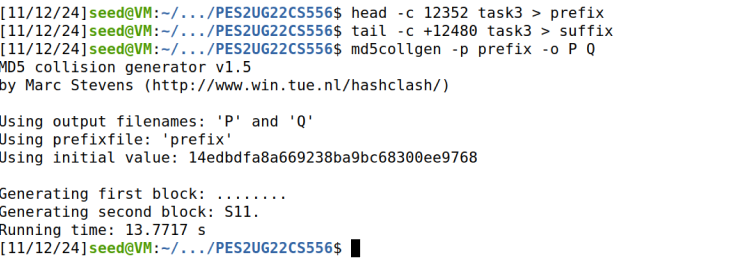


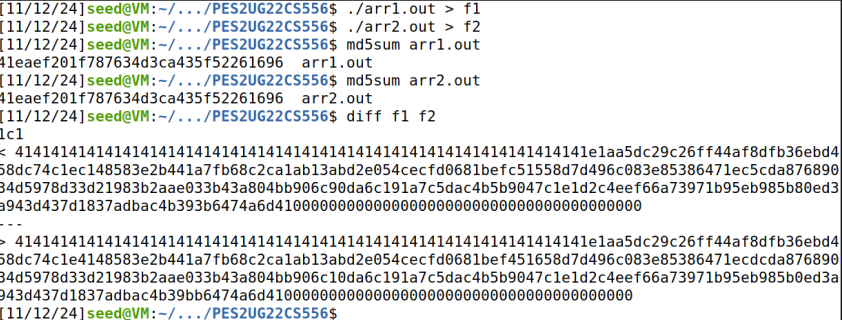
Important: Highlight the starting end ending hexadecimal offsets

iii) Output Screenshot (Terminal should have SRN visible) for step 3



iv) Output Screenshot (Terminal should have SRN visible) for step 4





v) Explain observations

* I observed that file1 and file2 produced the same MD5 hash even though they had different content.
* This happened because the MD5 algorithm is vulnerable to collision attacks, which allow two different inputs to generate the same hash.
* The md5collgen tool exploits this by manipulating a specific part (128-byte region) of the files, causing them to differ while maintaining the same hash.
* This shows why MD5 is no longer secure for verifying file integrity or for cryptographic purposes.