**ACS 545 Cryptography and Network Security**

**Lab 9: MD5 Collision Attack Lab**

**Name:** Vijay Anand Pandian

**Task 1 - Generating Two Different Files with the Same MD5 Hash**

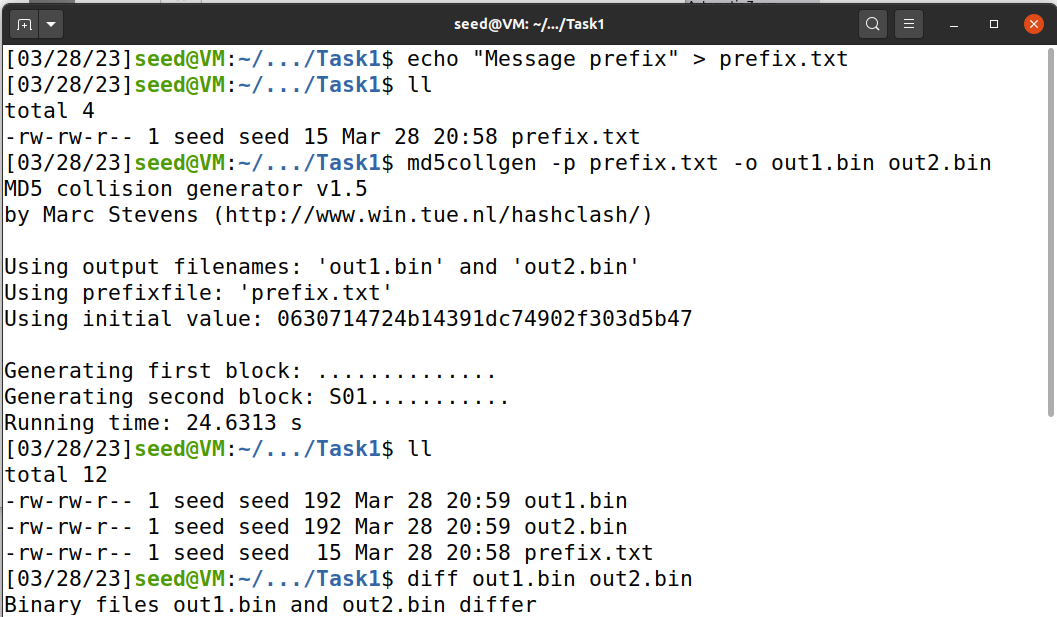
**Implementation and Output:**

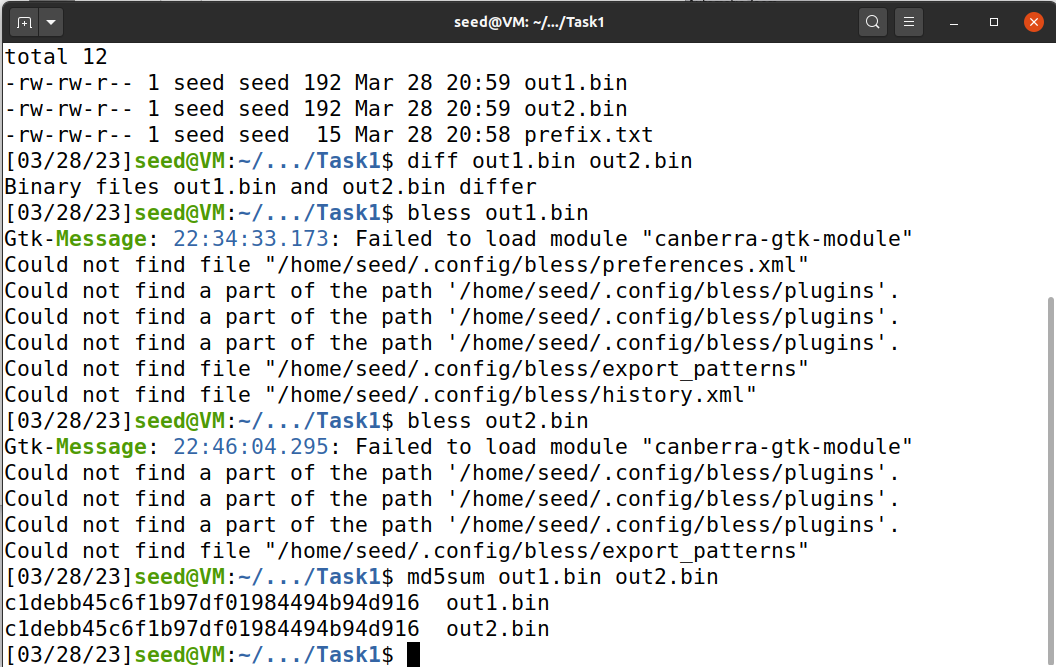
**Graphical user interface, text

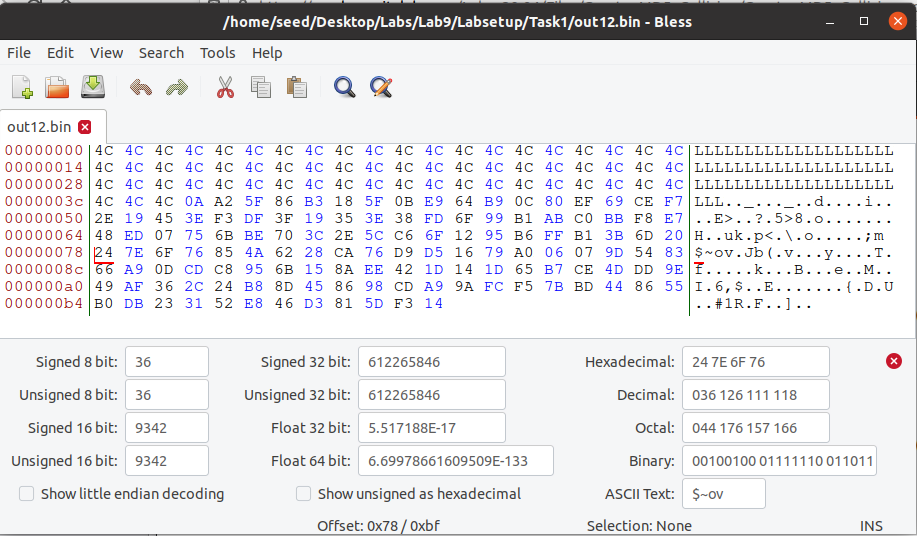
Description automatically generated**

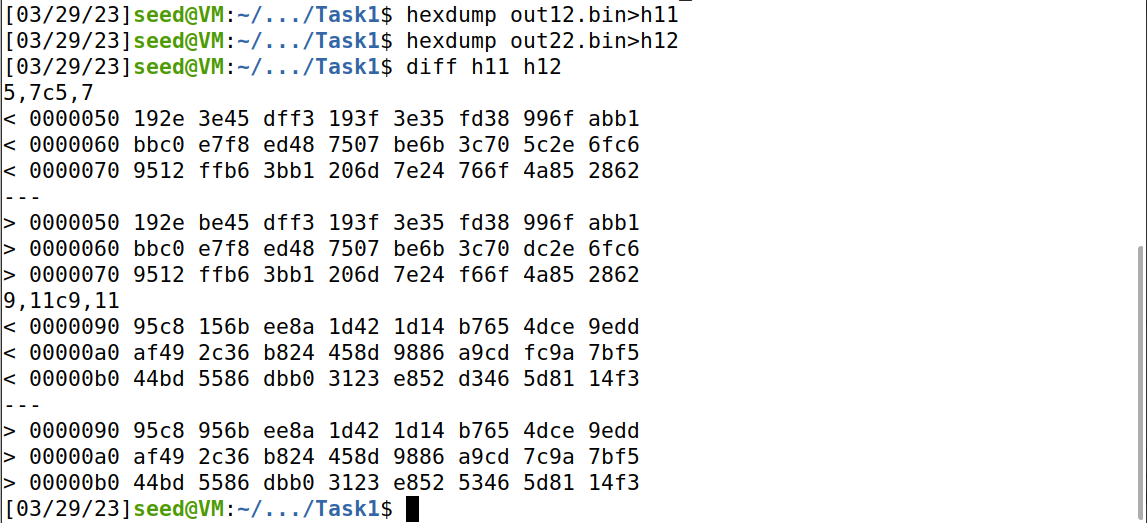
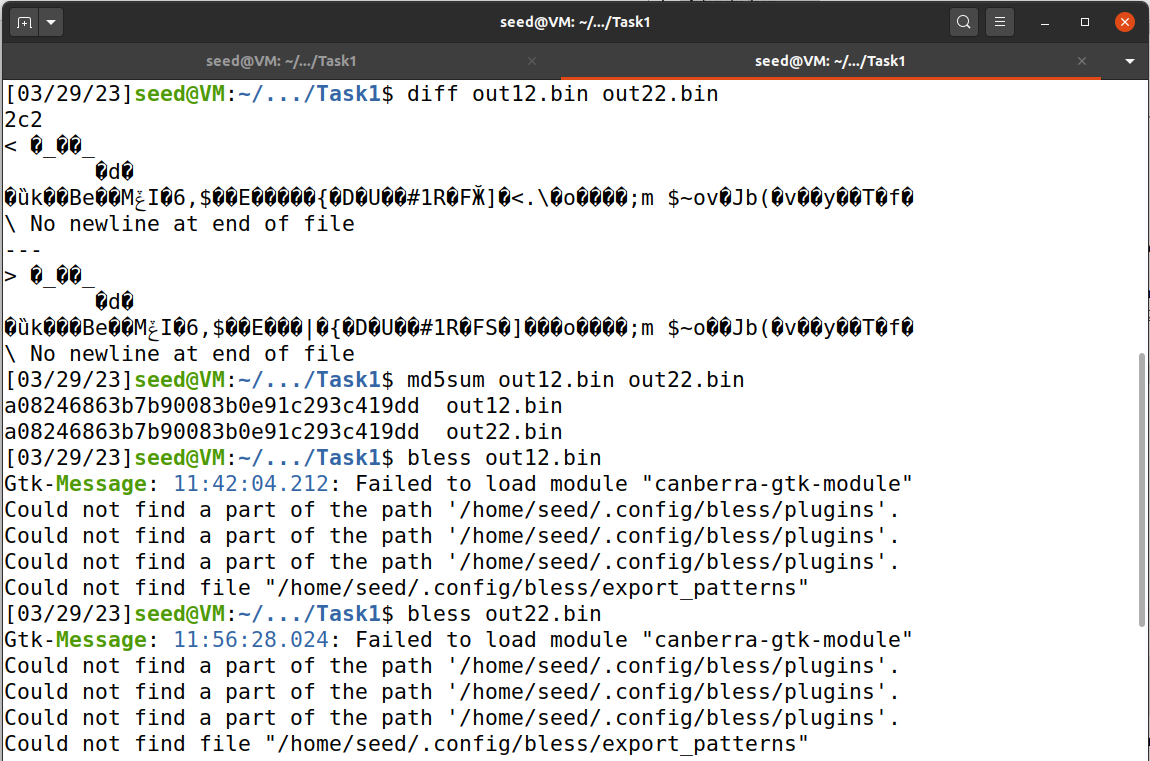
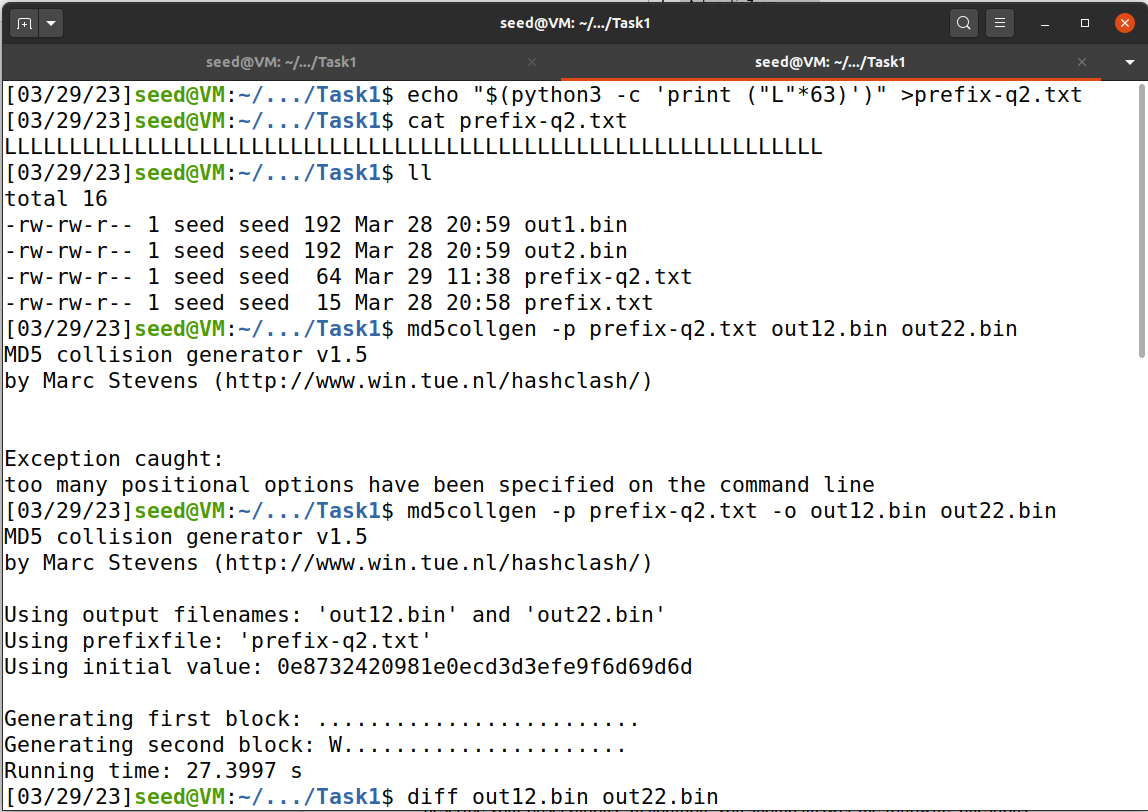
**Graphical user interface, text, application

Description automatically generated**

****

****

**Graphical user interface, text

Description automatically generated**

**Observation and Explanation:**We can see that the two output files share the same prefix and have different suffixes. The two files are different but their MD5 hash values are same.

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

If the prefix is not a multiple of 64, the remaining data will be padded with zeroes to make it a multiple of 64.

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

When we create a file with exactly 64 bytes there will be no padding and the entire 64bytes will be the data which we give, if the data is less than 64bytes it will be padded with 0.

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.

In the last screenshot above we can see that not all the bytes are different and many bytes are similar and some are different. I have highlighted the difference in the above screenshot.

**Task 2 - Understanding MD5’s Property**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Observation and Explanation:**

Where there are two files with the same hash function, even after adding a suffix we can see from the above screenshot highlighted that the MD5 values are the same.

**Task 3 - Generating Two Executable Files with the Same MD5 Hash**

**Code:**

#include <stdio.h>

unsigned char xyz[200] = {

0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,

0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,

0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,

0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,

0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,

0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,

0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,

0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,

0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,

0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,

0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,

0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,

0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,

0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,

0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,

0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,

0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,

0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,

0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,

0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41

};

int main()

{

int i;

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

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

}

printf("\n");

}

**Implementation and Output:**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated with medium confidenceA picture containing text

Description automatically generated**

**Explanation and Observation:**

We can see that the MD5 values are the same. We have achieved the target of two new executable files which output with different content. However, the MD5 values are the same.

**Task4 - Making the Two Programs Behave Differently**

**Code:**

#include <stdio.h>

#define LENGTH 400

unsigned char X[LENGTH]= {

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

};

unsigned char Y[LENGTH]= {

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

};

int main()

{

int i = 0;

for (i =0; i< LENGTH; i++){

if (X[i] != Y[i]) break;

}

if (i==LENGTH){

printf("%s\n", "Executing benign code... ");

}

else {

printf("%s\n", "Executing malicious code... ");

}

return 0;

}

**Implementation and Output:**

**Graphical user interface, text

Description automatically generatedGraphical user interface, text

Description automatically generatedGraphical user interface, text, application

Description automatically generated**

**Explanation and Observation:**

When we run the above the first version will execute benign instructions and the second version will run malicious instructions. The task emphasizes that creating two completely different programs that produce the same MD5 hash value is hard. The two hash-colliding programs produced by md5collgen need to share the same prefix, and if we need to add some meaningful suffix to the outputs produced by md5collgen, the suffix added to both programs also needs to be the same. Therefore, the task provides an approach to generating two different programs that can pass certification and have the same MD5 hash.