# Task1

Graphical user interface, text

Description automatically generated

First, I follow the instruction to build to two files who have the same MD5 hash value and save them into out1.bin and out2.bin. Using “diff out1.bin out2.bin” command, the output shows that they have the different content. I use binary editor to view them, and it can be noticed that the output is different. At the same time, the output of md5sum is same, which means they have the same hash value.

Graphical user interface, text

Description automatically generated with medium confidenceGraphical user interface

Description automatically generated with medium confidence

## Q1:

Graphical user interface

Description automatically generatedGraphical user interface, application

Description automatically generated

If the length of your prefix file is not multiple of 64, the space will be padded by 0 to make sure the prefix file size are the multiple of 64. It can be observed at the pictures as above.

## Q2:

Text

Description automatically generated

Build two files with 64 bytes prefix file. And the outputs shown as follows. It can be noticed that prefix file is not padded because the file size is 64 bytes.

Graphical user interface, application

Description automatically generatedA picture containing graphical user interface

Description automatically generated

## Q3:

Text

Description automatically generated

It is not complete different, use diff command to check the different parts. The different can be noticed as above.

# Task 2

Graphical user interface, text

Description automatically generated

I built out1.bin and out2.bin in task 1, and MD5(out1.bin) == MD5(out2.bin). I build a new file and cat it with out1.bin and out2. bin. Then I observed that the MD5 value also same. It can be proofed that the hash value will also be same after two files have the same hash value concatenate same content.

# Task 3

#include <stdio.h>

unsigned char xyz[200] = {

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"};

int main(){

int i;

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

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

}

printf("\n");

}

I modify the C code as above, and gcc it gets T3. Open T3 file and can be find the array are saving at 0x3020 that the decimal is 12320.

A screenshot of a computer

Description automatically generated

We need the size of prefix file is multiply 64 bytes, and 12352 can be divided by 64. Therefore, I split the first 12352 bytes to prefix, and split from the 12352+128 bytes to the end of the file to suffix. I use prefix to build two files “prefix1” and “prefix2” that have the same MD5 hash value. Then cat prefix1 and suffix to P, cat prefix2 and suffix to Q. Finally, verify P and Q. The hash value of P and Q are same, and P and Q can be executable. However, their outputs are not same.

Text

Description automatically generated

# T4

#include <stdio.h>

unsigned char X[200] = {

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

};

unsigned char Y[200] ={

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

};

int main(){

int t =1;

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

if (X[i] != Y[i]){

t = 0;

}

}

if(t==0){

printf("run malicious\n");

}else{

printf("run benign\n");

}

return 0;

}

Text

Description automatically generated

This is the whole process of Task4.

Compile T4.c and open T4. Find Array X and Array Y. It can be noticed that Array X starts at 12320 Bytes, Array Y starts at 12544 Bytes.

A screenshot of a computer

Description automatically generated

First, I split the prefix first, I choose the same size in Task 3, which include 32 bytes “A”.

Graphical user interface, application

Description automatically generated

To make sure X==Y in the first version, array Y should be split into first 32 bytes and last (200-128-32 = 40) Bytes.

Therefore, I first split T4 at (12544+32 = 12576) bytes and save content in t1, and split t1 at (12352+128+1 = 12481) bytes and save the content into presuf.

Graphical user interface

Description automatically generated

Then, I split T4 at (12576+128+1=12705)bytes and save the content in postsuf.

Graphical user interface, application

Description automatically generated

The next step is finding P and Q. I use prefix to get two new file “prefix1” and “prefix2” who have the same hash value. Based on the Task1, P and Q should be the last 128 bytes of “prefix1” and “prefix2”, because the prefix is the multiply of 64 bytes. Split last 128 bytes into “prefix1” and “prefix2” and get P and Q.

Then, I cat presuf, P, postsuf to get suffix.

Graphical user interface, application

Description automatically generated

Then, cat prefix1, suffix to get t4out1; cat prefix2, suffix to get t4out2.

Finally, verify t4out1, t4out2. The hash value of t4out1 and t4out2 are same, and t4out1 and t4out2 can be executable. However, their outputs are not same. T4out1 output that X and Y are same, and t4out2 output that X and Y are different.