MD5 Collision Attack

Klas Mannberg
klaman-8@student.ltu.se

10 September 2020

Contents

1 Lab Tasks

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

Question 1:
The output will have our prefix and P/Q, but no space inbetween. See image 1.
Question 2:
Having a prefix of exactly 64 bytes seems to add an empty region between our input and P/Q. See image 2. Since it is already past 64 bytes when it adds the length, it is forced to add additional 64 space.
Question 3:
The last bytes of the output, P/Q, have a few differences. All other data is the same.

1.2 Task 2: Understanding MD5's Property

If we know that the MD5 hash is the same with inputs M and N, then adding a suffix T to both will make the new files share the same md5 hash. To show this I run the collision generator on a prefix of 64 bytes, which will generate $MD5(M) = MD5(N)$. In theory adding the same data T to both with *cat* will gives us $MD5(M||T) = MD5(N||T)$. This is what we do in image 3. I add the file "suffix" to the end of both output files. The suffix file just contains the string $BBBBBBBBBB$ in hex. The files having the same MD5 hash confirms the property.

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

The prefix ranges need to be in multiple of 64. Using bless I can find the array location which allows me to calculate the byte range from the start of the file $prefix$, or the bottom $suffix$. The closest prefix range which allowed the prefix to be divisible by 64, and be within the array, was 4224. We know the prefix is 4224 and that P/Q will add 128 bytes, so the suffix must be the only remaining bytes of the file $\implies tail - c + 4289\, a.out > suffix$.

After by running md5collgen on the prefix to get P/Q added, we appended the suffix to both output files. The result is two executables with same md5 hash and different array values, see image 5.

1.4 Task 4: Making the Two Programs Behave Differently

The goal here is to create a program with code that only triggers when a difference in data is detected. From Task 3 we know that we can make a program have different data, with the same MD5 hash. Using arrays we can leave a

section of code that only triggers when it detects a difference between them. See image 6 for the code.

Since the prefix is 4160 (bytes to the start of the array) we do $head -c\,4160\,a.out > prefix$ and the suffix is the bytes after the 128 byte region $\implies tail -c +4289\,a.out > suffix$. After cutting out the 128 bytes for P and Q from the output of $md5collgen$ we could now have two programs with different arrays. But it's a bit more compilcated since if we only change one of the arrays then the condition in our code will trigger for both programs (since P is different from the A's in the array). The solution to this is to cut another 288 bytes, the distance to start of the second array, and insert P there aswell. The only remaining part we need to cut out is the last bytes of the suffix after P in the second array, which is *Suffix3* in the example below.

Patching all this together we get:

1. done1 = Prefix + P + Suffix2 + P + Suffix3
2. done2 = Prefix + Q + Suffix2 + P + Suffix3

Suffix2 is the space between the end of 128 byte P or Q in one array and the start of P in the second
Suffix3 is the last bytes of suffix after second P

So now our hidden code only triggers on the program "done2" while both has the same MD5 hash. See image 7 for the result of $md5sum$.

Done1 will send you a nice message while done2 will curse your directory with 10 useless files! How terribly evil, I know.

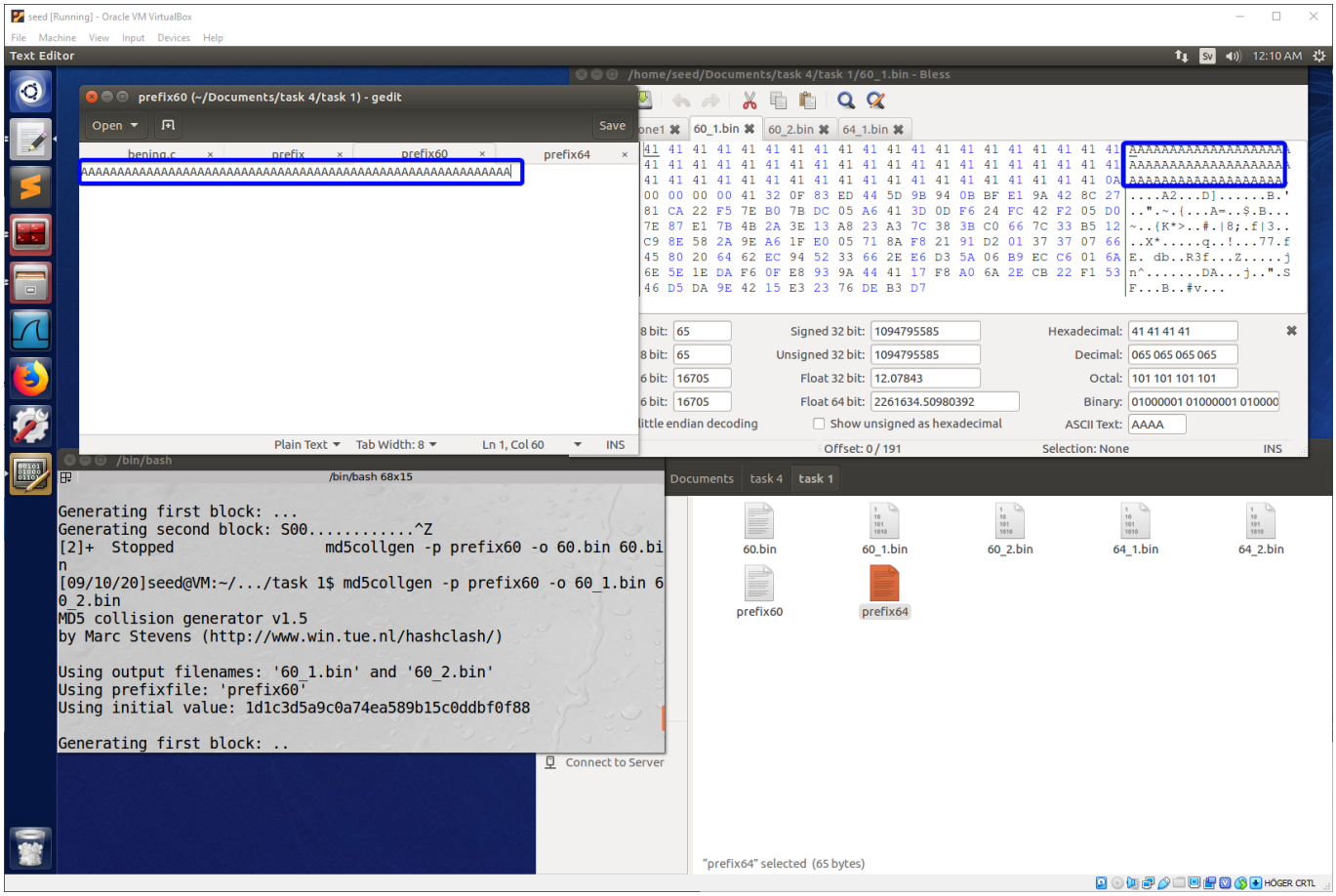

Figure 1: Task 1, Question 1: Prefix and P/Q

2 Images

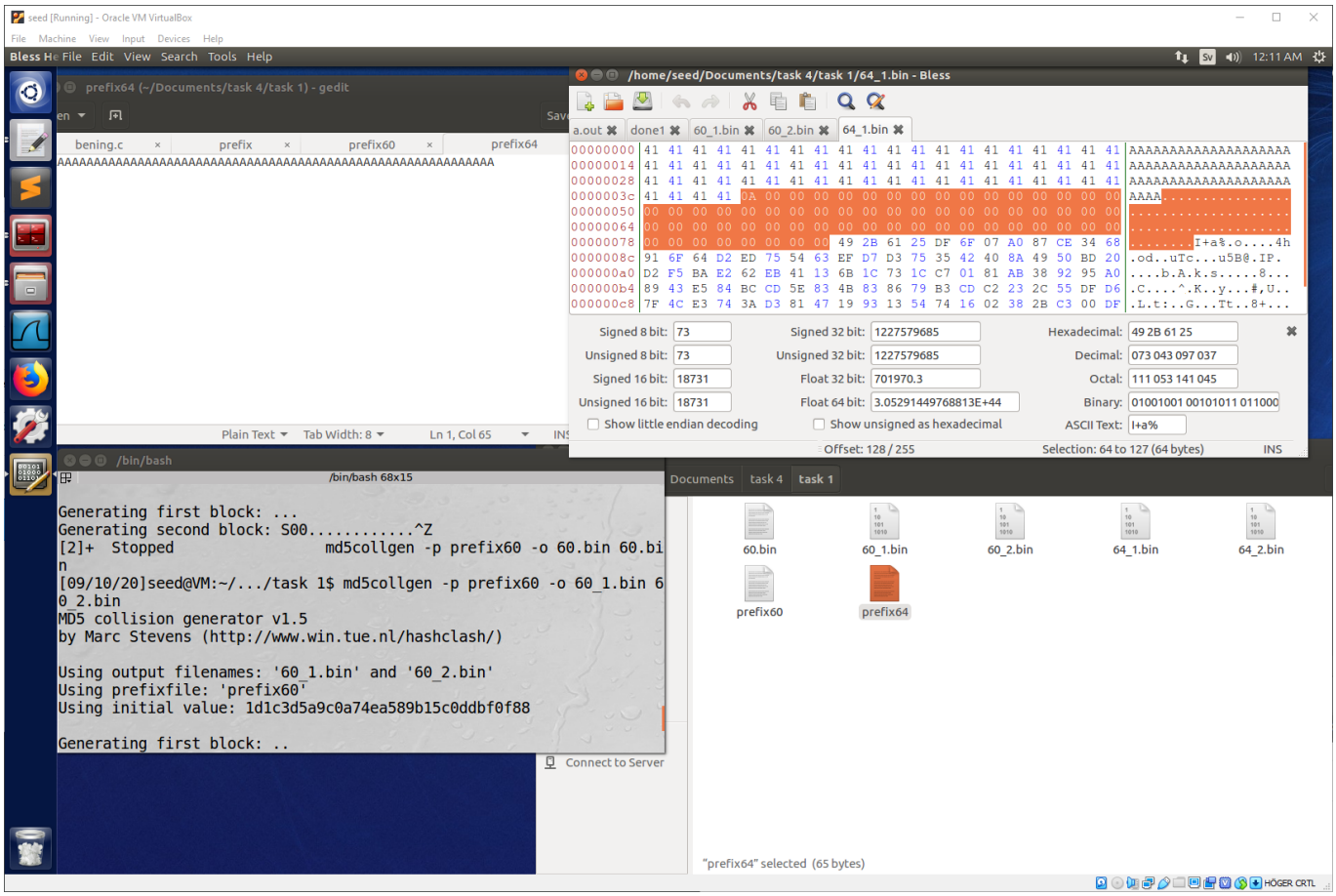

Figure 2: Task 1, Question 2: Empty region

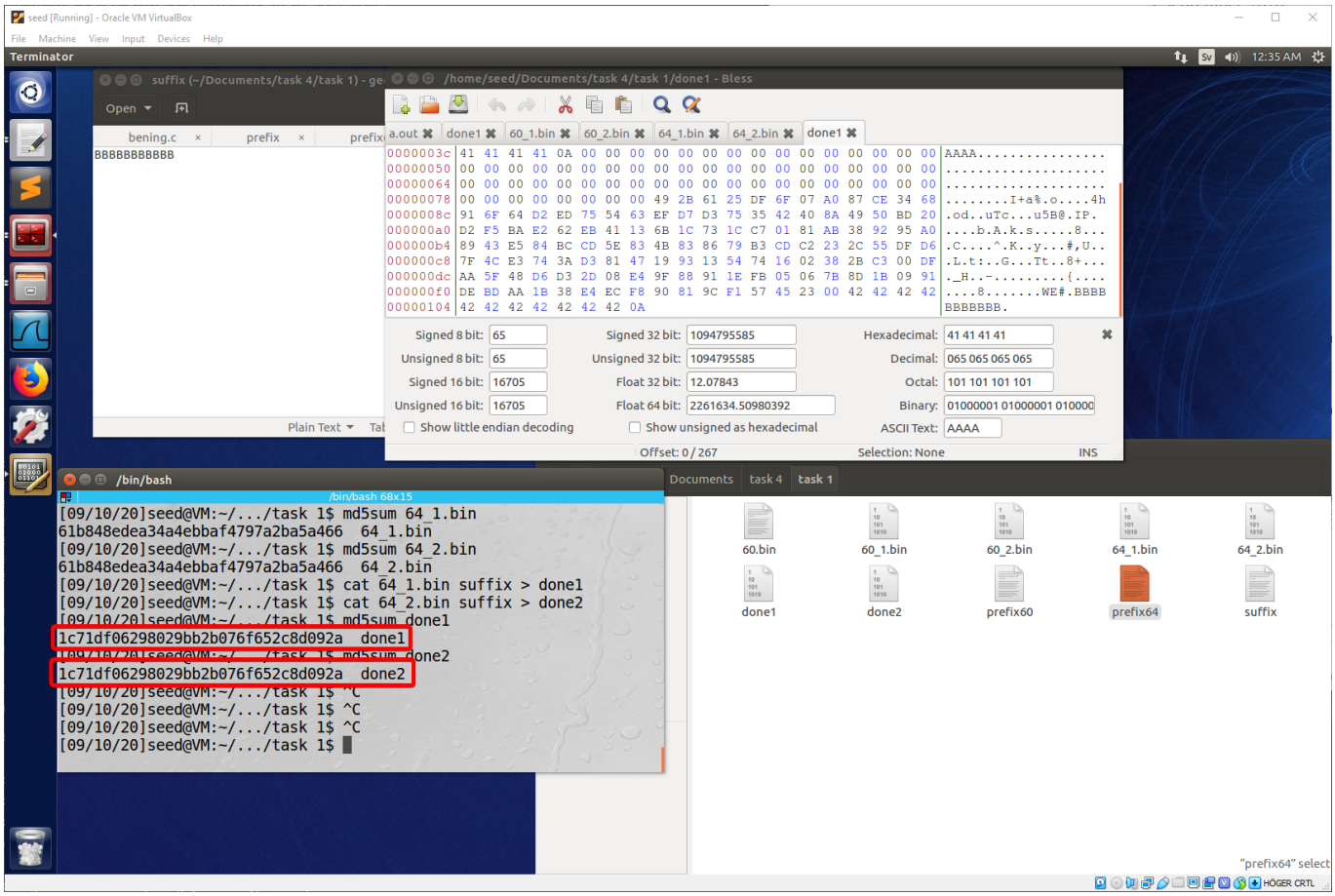

Figure 3: Adding the suffix, T "BBBBBBBBBBB" to the outputs $64_1.bin$ and $64_2.bin$ of the collision generator. Results have the same MD5 Hash

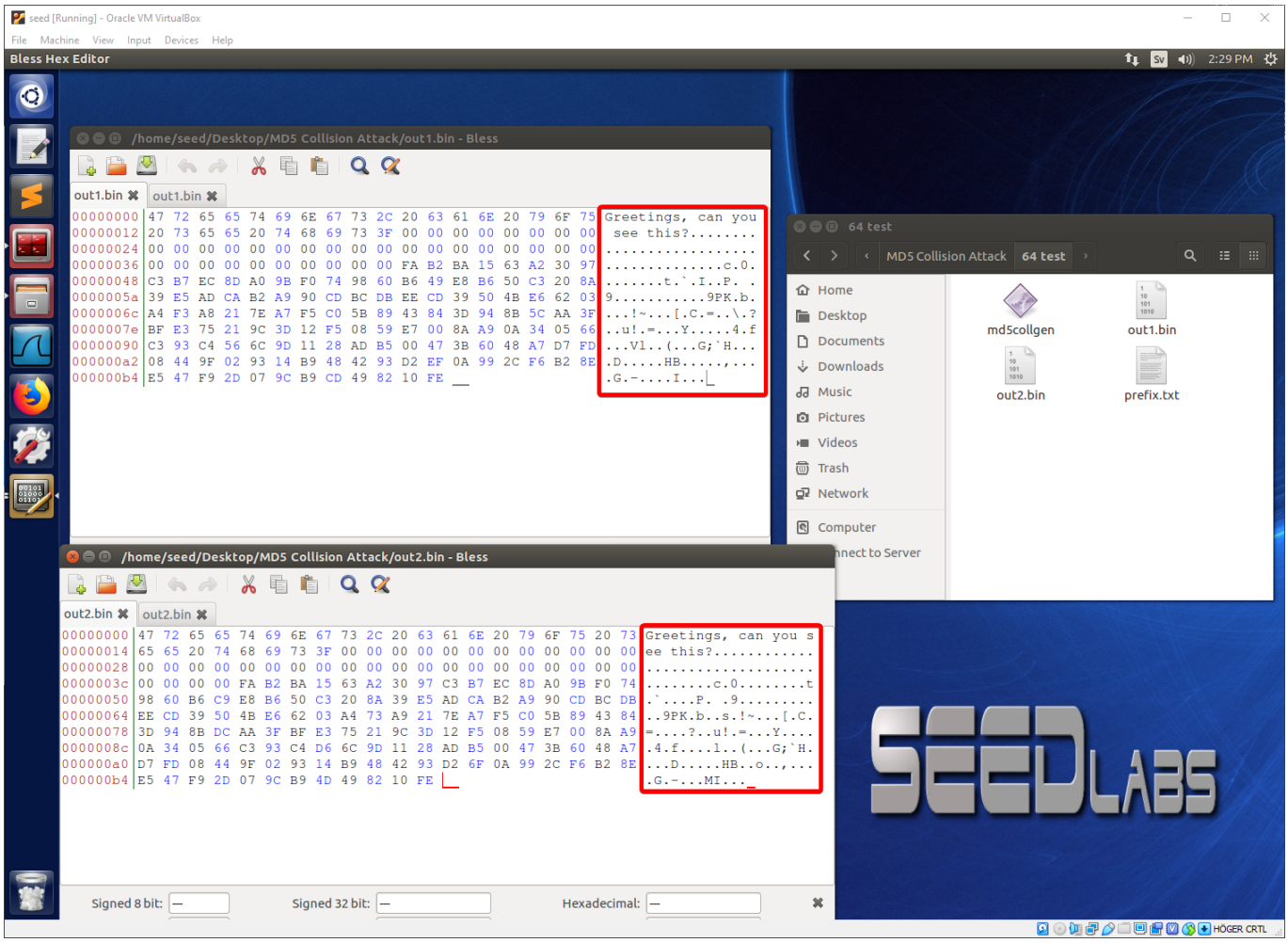

Figure 4: The prefix+padding+generated contents of the bin files

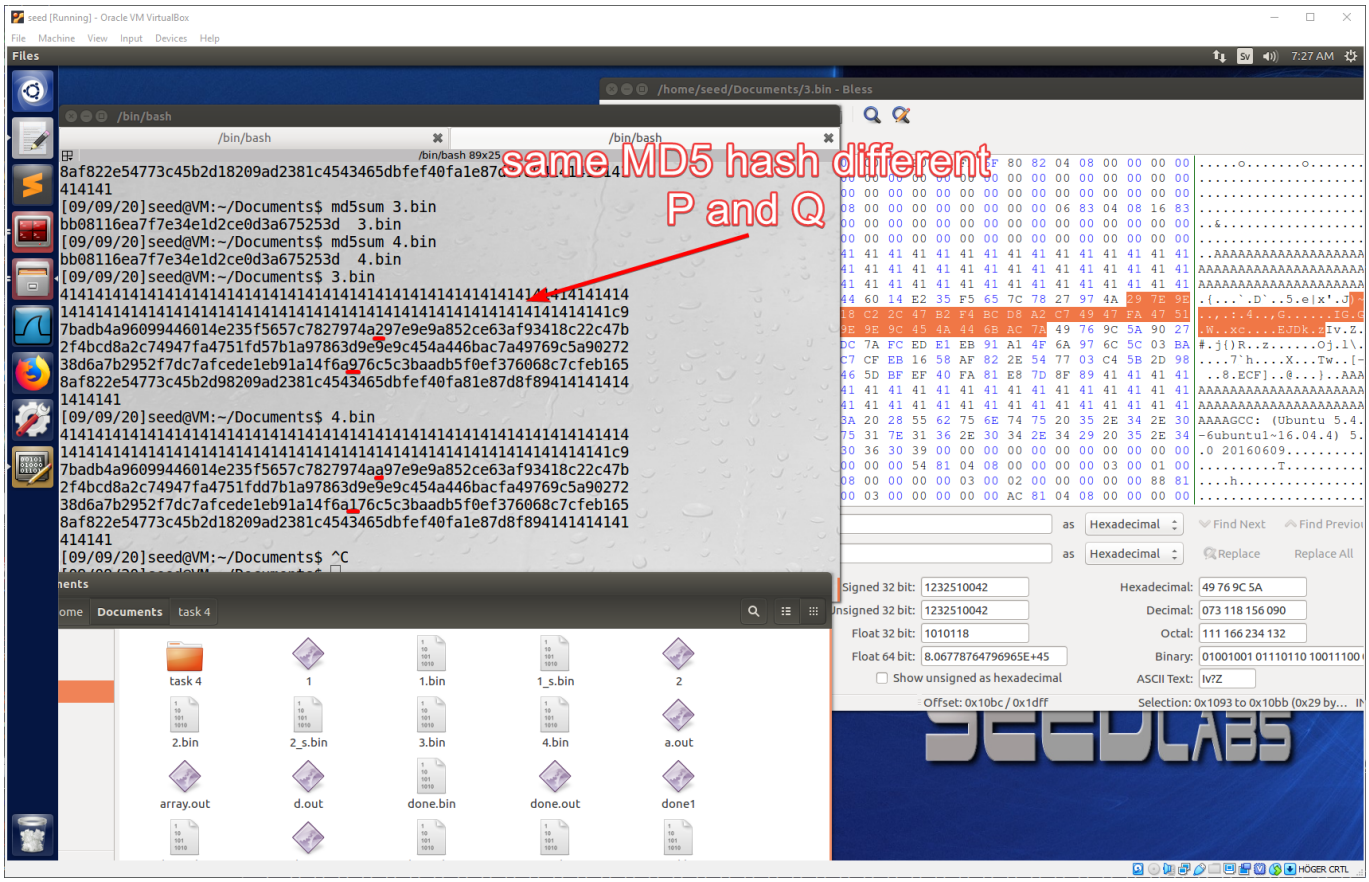

Figure 5: Same MD5 hash but modified array!

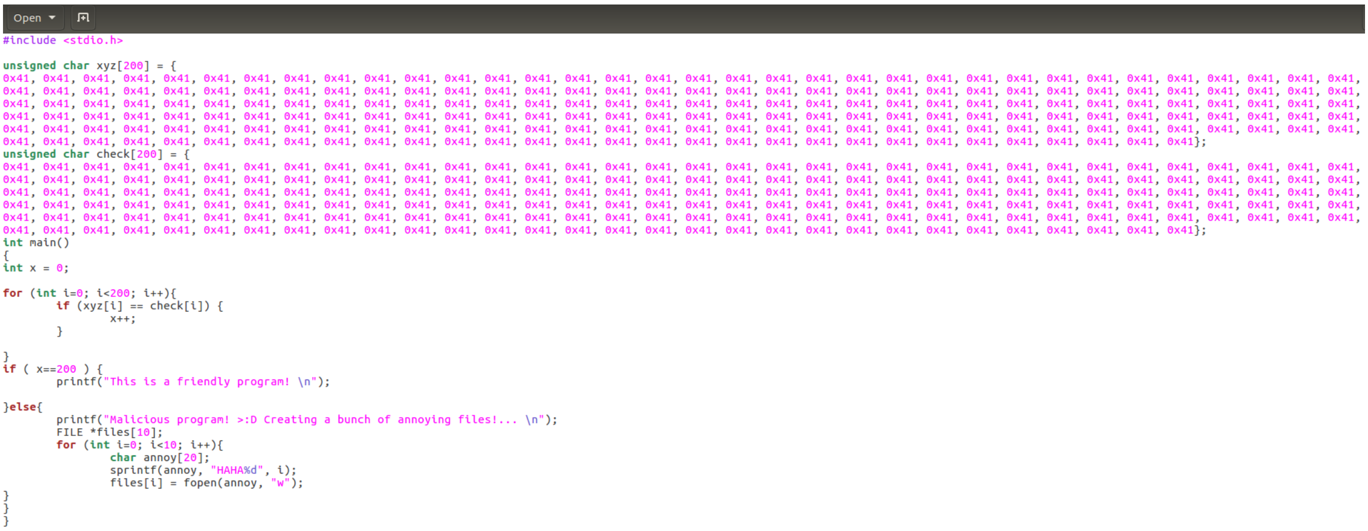

Figure 6: Source code of task 4

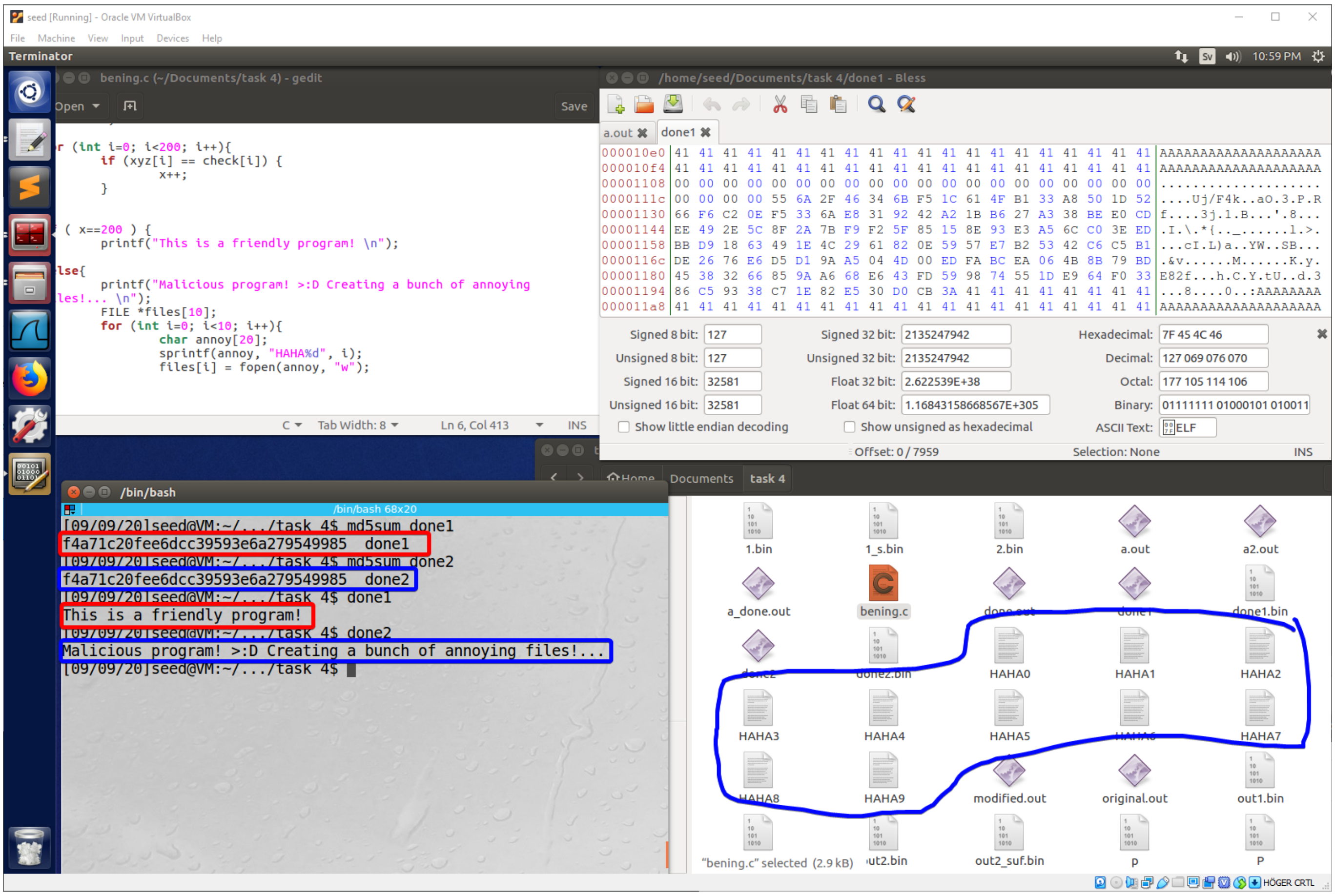

Figure 7: Task 4: Two different sections of code executing, with same MD5 hash!