**MD5 Collision Attack**

Chawin Metah

Department of Computer Science, Purdue University Fort Wayne

ACS 54500: Cryptography and Network Security

Lab 8

Dr. Zesheng Chen

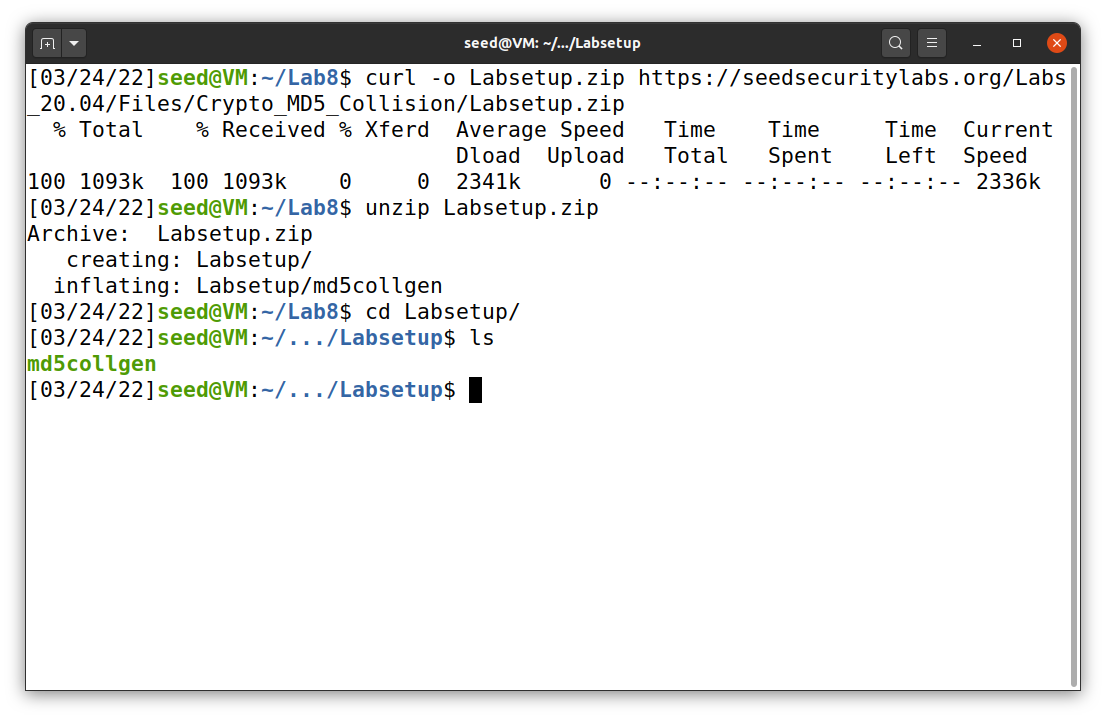
April 13, 2022

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

Similar to the previous labs, I downloaded and unzip Labsetup.zip, which contains md5collgen (Figure 1). It can be downloaded with the following command: curl -o Labsetup.zip https://seedsecuritylabs.org/Labs\_20.04/Files/‌Crypto\_MD5\_Collision/Labsetup.zip. I used the unzip command to extract the file.

**Figure 1**

*Labsetup.zip*

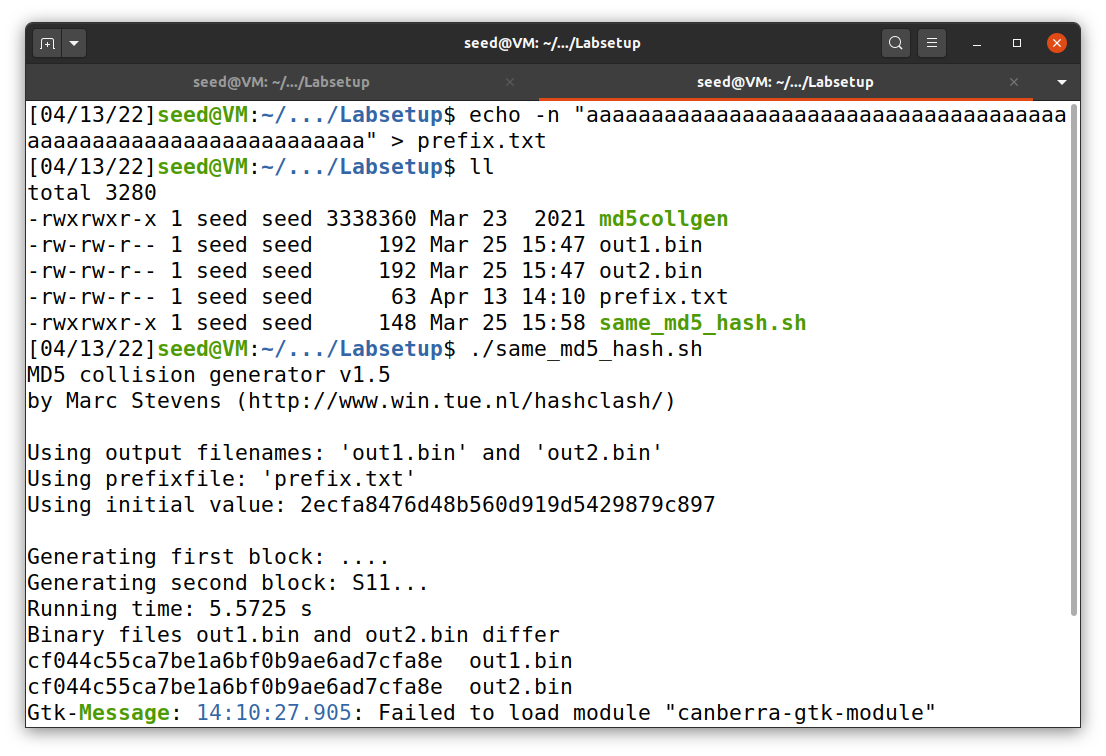


After the file is setup, I create a shell file to run the commands for this task. The file and commands are not shown here but uploaded to my GitHub repository as same\_md5\_hash.sh.

First, I created a file name prefix, as shown in Figure 2. I used “ll” command to show that it has a size of 63 bytes. Next, I ran same\_md5\_hash.sh and I gave the output of out1.bin and out2.bin. When those files are being compared using diff command, I stated that both files are different. However, when using md5sum, it appears to be the same. To confirm this, bless command also being used and both output files are shown in Figure 3. The output files have the same prefix (which comes from the prefix.txt) but different suffixes.

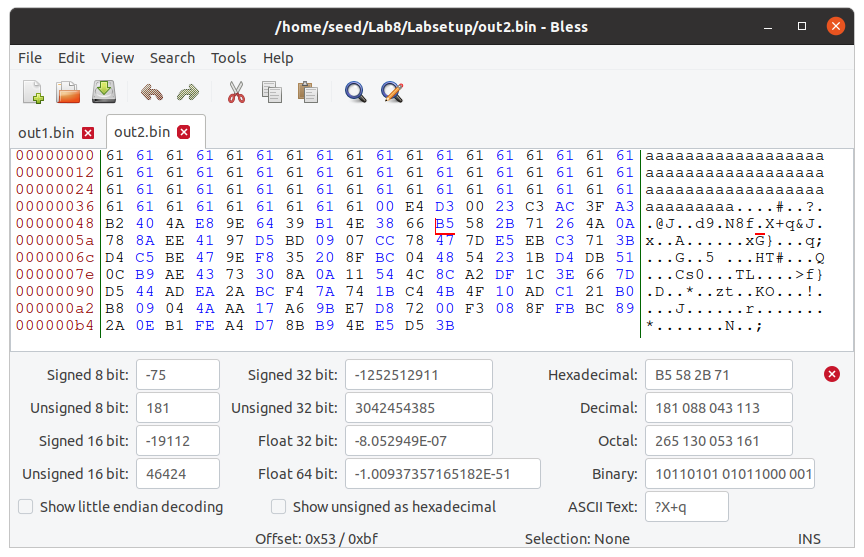
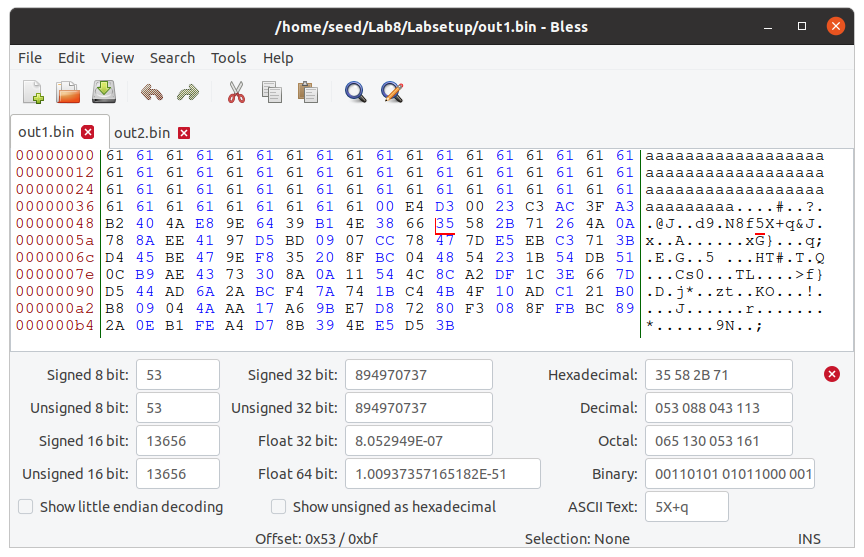
**Figure 2**

*Task 1 Question 1*



**Figure 3**

*Task 1 Question 1 output files*



Out2.bin

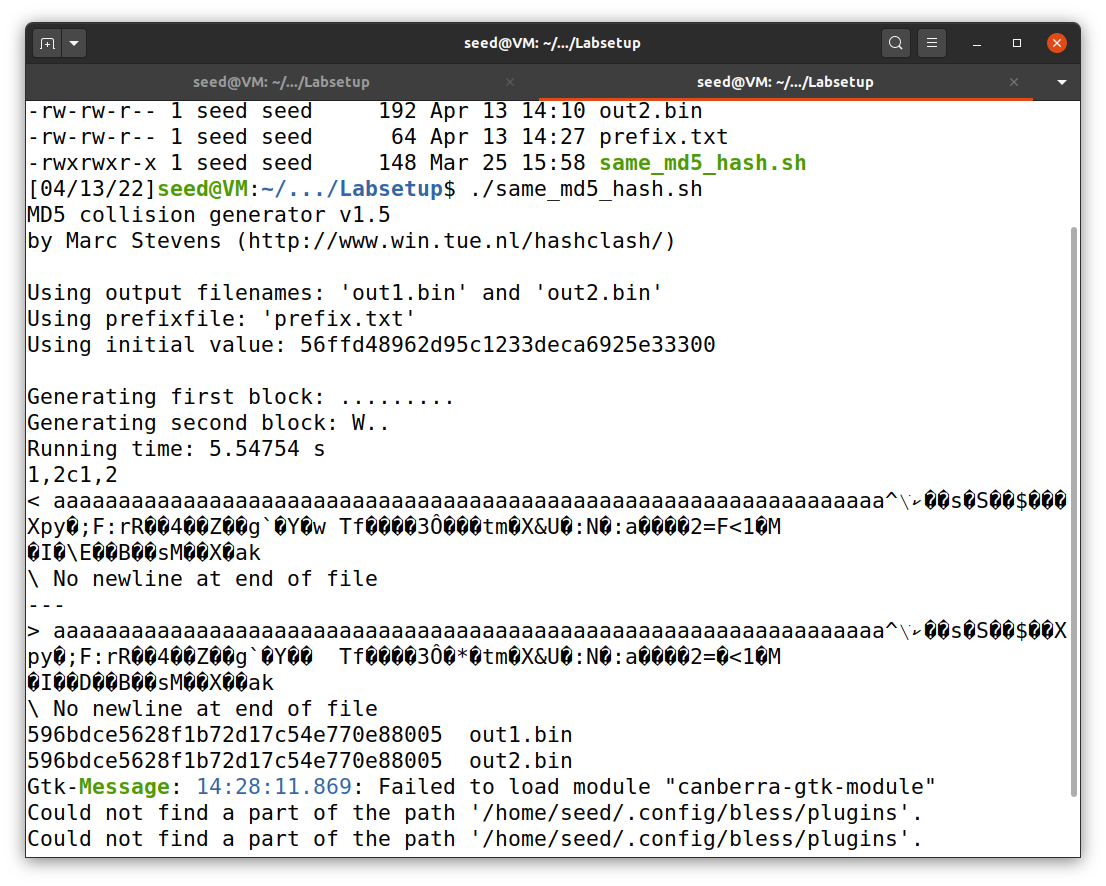
Out1.bin

Now, I run it again but with prefix size of 64 bytes. The results are shown in Figure 4 and 5. It seems like diff command cannot evaluate the result and md5sum still gives exact same output. When using bless, however, it shows the difference. What interesting is that for both questions (with prefix size of 63 and 64 bytes), the output files start to differ at the bit 0x53.

In addition, for question 3, md5collgen produces different files that cannot view using md5sum. Both files appear to have the same size of 192 bytes.

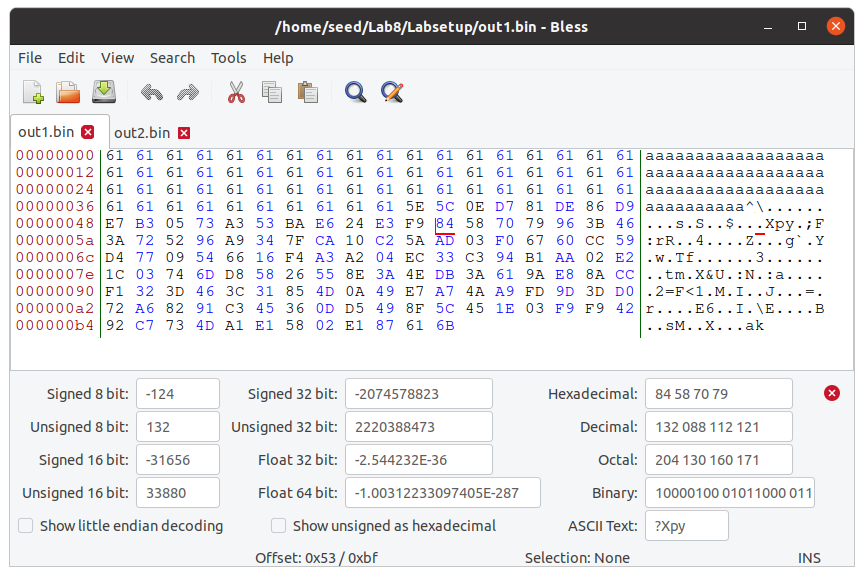
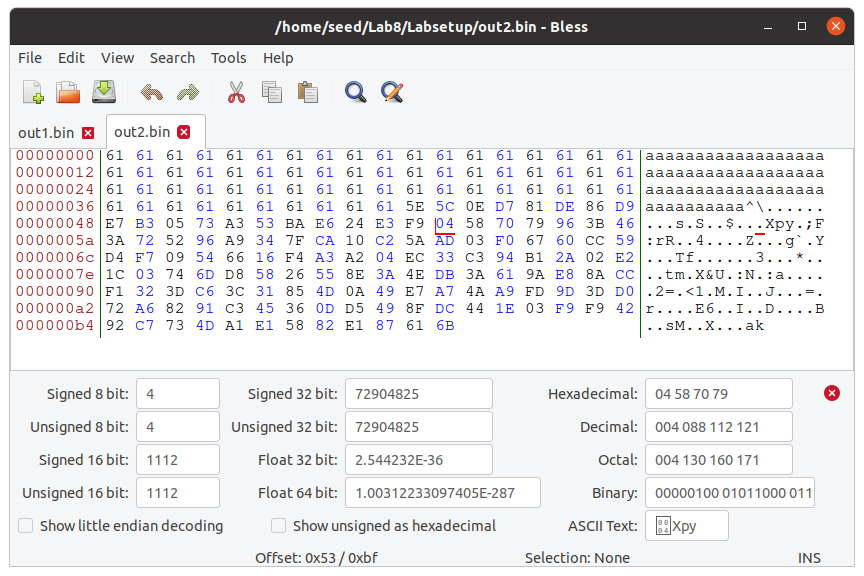
**Figure 4**

*Task 1 Question 2*

**

**Figure 5**

*Task 1 Question 2 output files*

**

Out2.bin

Out1.bin

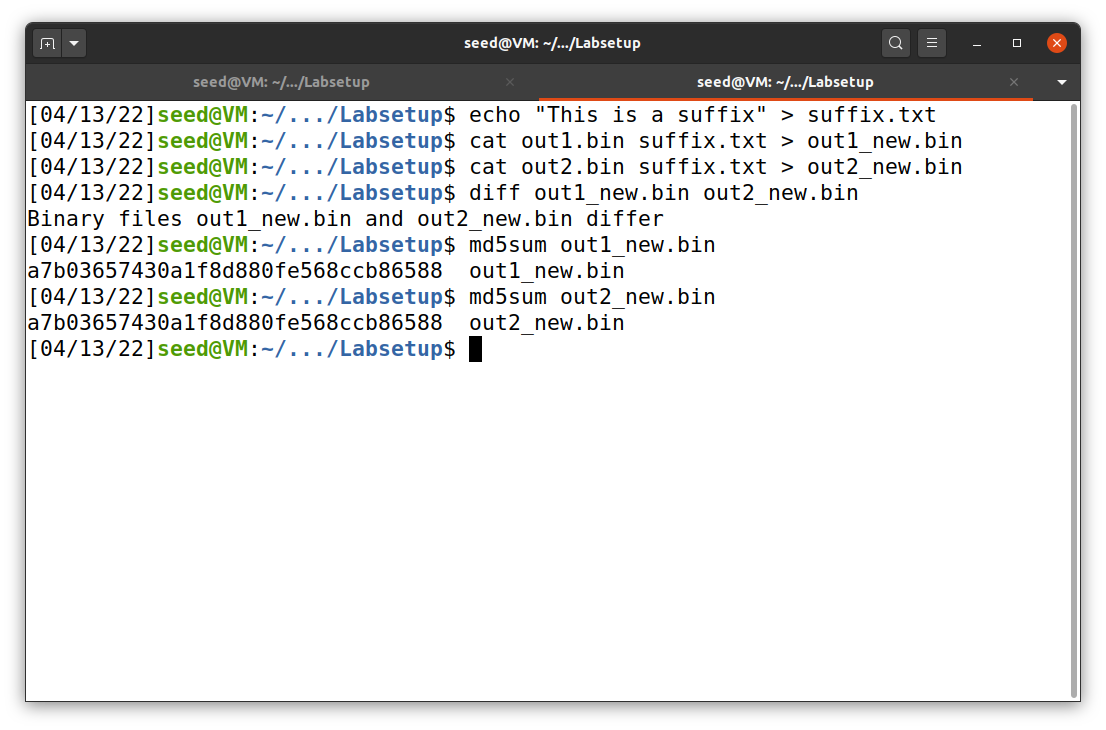
**Task 2: Understanding MD5’s Property**

In this task, I created a new text file called suffix.txt. I used the output files that is similar to the one in task1 question 1 (out1.bin and out2.bin). Then, I concatenated them with suffix.txt and ran the commands below to check the result. The result (in Figure 6) shows that the property of MD5 is that, when they have the same md5sum and concatenated with the same file, it produce the same intermediate hash value (IHV).

diff out1\_new.bin out2\_new.bin md5sum out1\_new.bin

md5sum out2\_new.bin

**Figure 6**

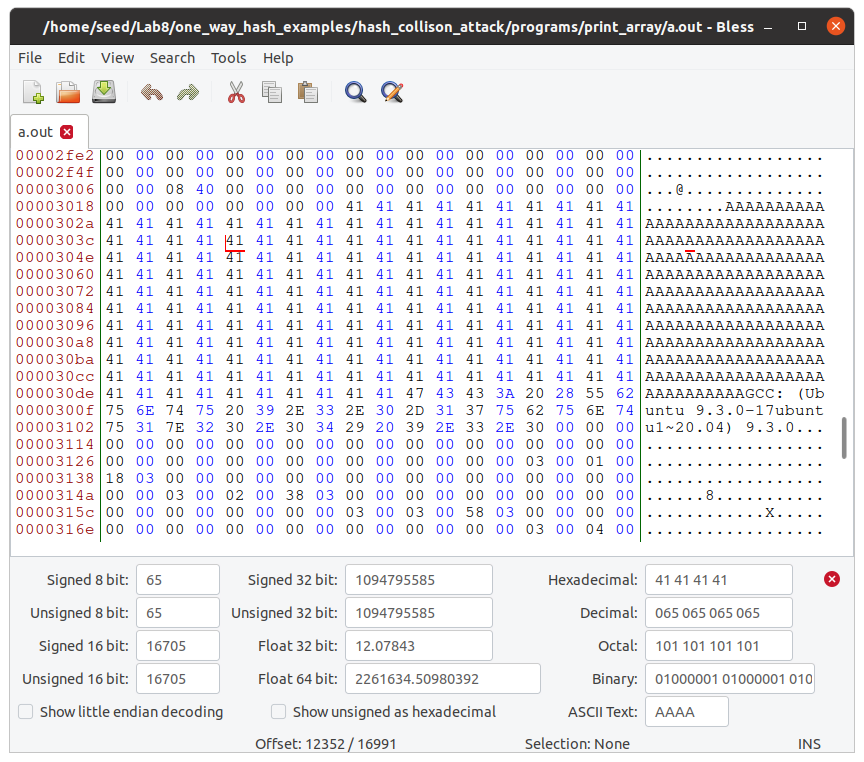
*MD5’s Property (Length Extension) *

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

I used the code print\_array.c provided in Brightspace (and I reuploaded it to my GitHub repository). I executed the commands to produce two files that give the same md5sum value. The commands are put into the shell file md5\_2\_files.sh (available on my GitHub repository). The results are shown in Figure 7 and 8.

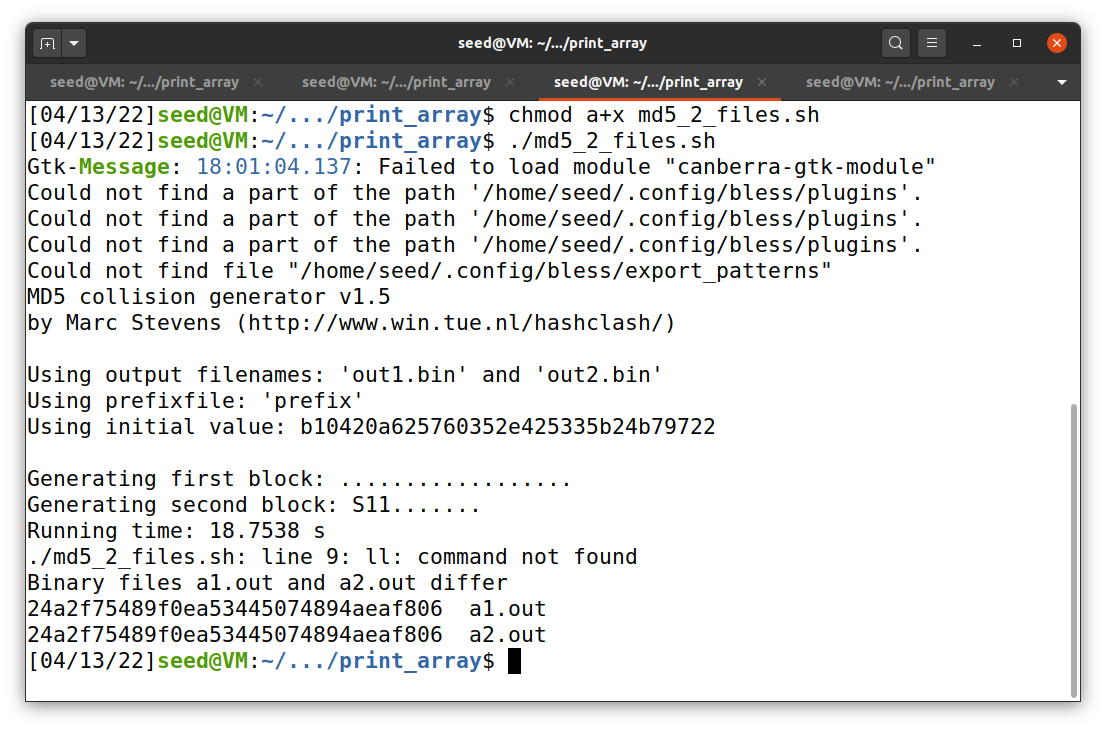
**Figure 7**

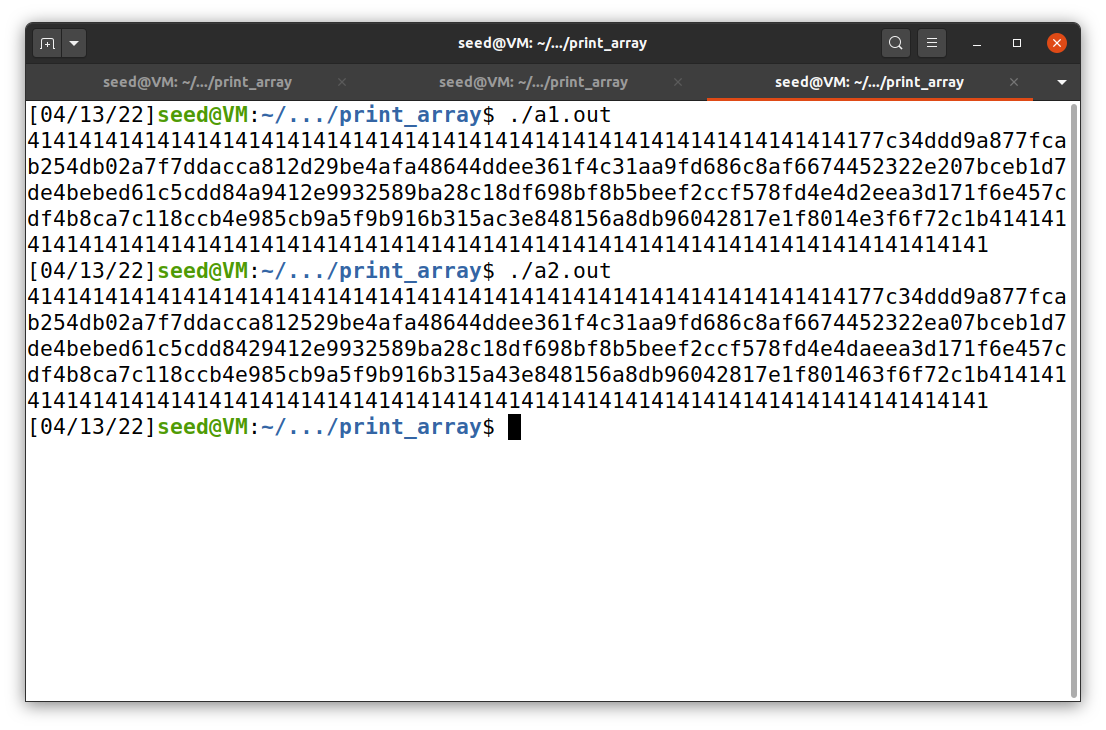
*Bless of task 3*



**Figure 8**

*The results for task 3*





The code (in shell command file) does the following:

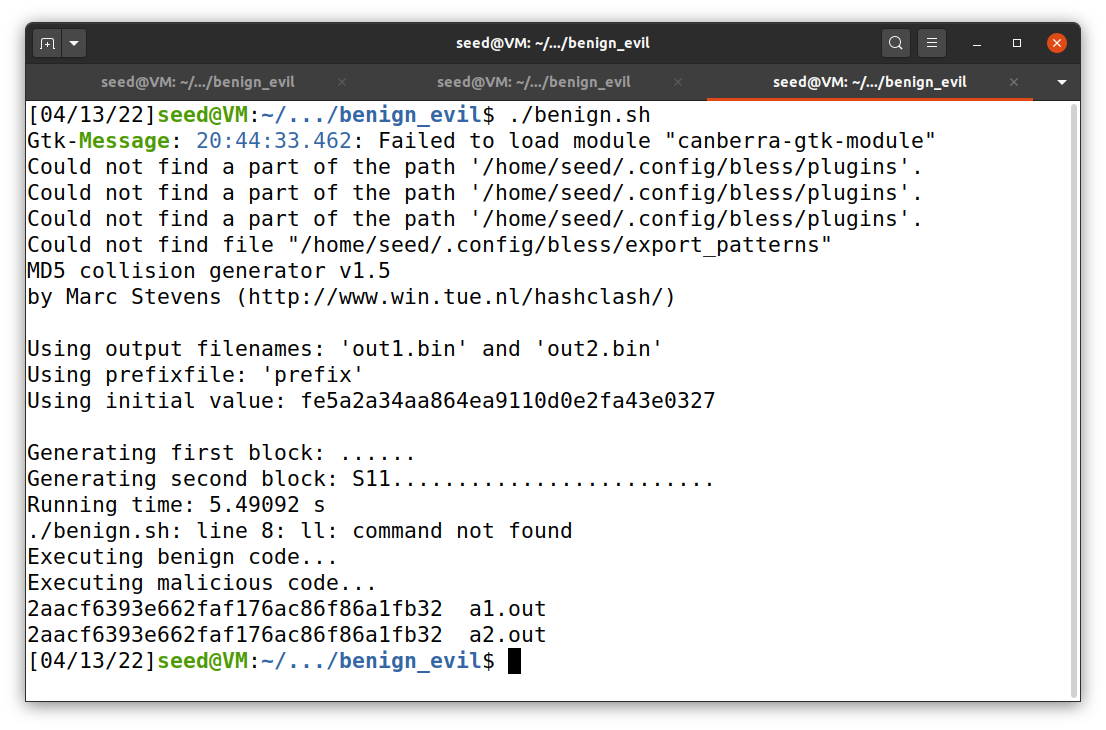
* Saves the first 12352 bytes, which can be divided by 64, of a.out as a prefix.
* Saves the data from the 12481st byte to the end of the file a.out as a suffix.
* Saves the last 128 bytes of out1.bin as P.
* Saves the last 128 bytes of out2.bin as Q.
* Concatenate them as follow:
  + Prefix || P || Suffix
  + Prefix || Q || Suffix
* Compare both output files (they produce the same value of md5sum but different value when execute them, as shown in Figure 8)

**Task 4: Making the Two Programs Behave Differently**

I used the code benign\_evil.c provided in Brightspace (and I also reuploaded it to my GitHub repository). I executed the commands to produce two files that give the same md5sum value. One is the benign file, and the other is the malicious file. The commands are put into the shell file benign.sh (available on my GitHub repository). The results are shown in Figure 9. The idea for this task is close to the one that describe in task 3.

**Figure 9**

*The results for task 4*

**

The code (in shell command file) does the following:

* Compile benign\_evil.c to a.out
* At this point, I used “bless” only to show a.out. The numbers below should be obtained before this.
* Saves the first 12352 bytes, which can be divided by 64, of a.out as a prefix.
* Generates md5collgen with the above prefix and output two files: out1.bin and out2.bin
* Saves the data from the 12481st byte to the end of the file a.out as a suffix.
* Saves the last 128 bytes of out1.bin as P.
* Saves the last 128 bytes of out2.bin as Q.
* Saves the first 288 bytes of suffix as suffix\_1.
* Saves the data from the 417th byte to the end of the file suffix as suffix\_2.
* Concatenate items as follow:
  + Prefix || P || Suffix\_1 || P || Suffix\_2 <<< This is benign code
  + Prefix || Q || Suffix\_1 || P || Suffix\_1 <<< This is malicious code
* Compare both output files (they produce the same value of md5sum, as shown in Figure 9)

**References**

Du, W. (2019). Computer & internet security: A hands-on approach (2nd ed.). Independently published.

Du, W. (n.d.). MD5 Collision Attack Lab. SeedLabs 2.0. /https://seedsecuritylabs.org/‌Labs\_20.04/Crypto/Crypto\_MD5\_Collision/