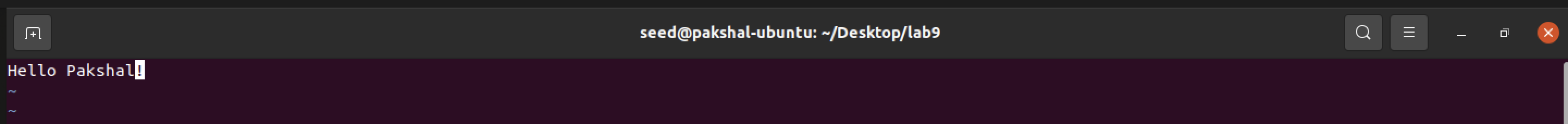
**LAB-9**

**Pakshal Bhandari**

**Task – 1**

In this task, we will generate two different files with the same MD5 hash values. The beginning parts of these two files need to be the same, i.e., they share the same prefix. We can achieve this using the md5collgen program, which allows us to provide a prefix file with any arbitrary content.

The prefix.txt I have created looks like this.



Using this prefix file(prefix.txt) shown above, we run the following md5collgen command.

A black background with white text

Description automatically generated

The above command creates two files out1.bin and out.2.bin

We run diff on both the files.



We can see that the out1.bin and out2.bin files are different. Now let us run md5sum on both the files.

A screenshot of a computer

Description automatically generated

We can see that even though both the files were distinct but still we got the same MD5 hash for both the files out1.bin and out2.bin

Now we run bless on both out1.bin and out2.bin to view the contents of binary file.

A screenshot of a computer

Description automatically generated

Running bless on out1.bin

A screenshot of a computer

Description automatically generated

Running bless on out2.bin

We can see few offset locations are different when comparing both files for example, the offset locations of 0273, 0155 etc. are different, even when they have the same MD5 hash value.

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

md5collgen adds in padding to make the length a multiple of 64.

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

We create the prefix file with exactly 64 bytes as shown below.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

We can see that prefix\_64.txt size is 64 bytes. Now we run md5collgen to generate two newout1.bin and newout2.bin files as shown below.

A screenshot of a computer program

Description automatically generated

Now we run diff to see if the two files are distinct.

A screenshot of a computer

Description automatically generated

We can see that both the files are distinct. Now we find the md5 hash to see if the hash is different for both or same.

A screenshot of a computer

Description automatically generated

We can see that files are distinct but the md5 hash both the files are same.

We now use bless to view the contents of the binary file i.e. newout1.bin and newout2.bin

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

We can see few offset locations are different when comparing both files for example, the offset locations of 0265, 0256 etc. are different, even when they have the same MD5 hash value.

**Task 2 Understanding MD5’s Property**

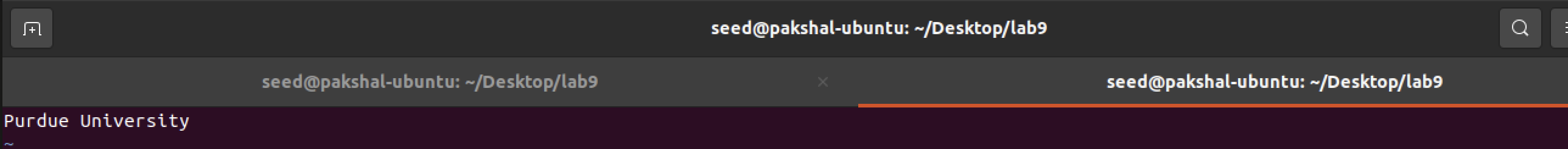
In this task we are supposed to generate two output files for a prefix and add a common suffix to both to show the property that by adding the same suffix to the two input files will give two output files with identical hash values.

I am using the same output binary files from Task 1.

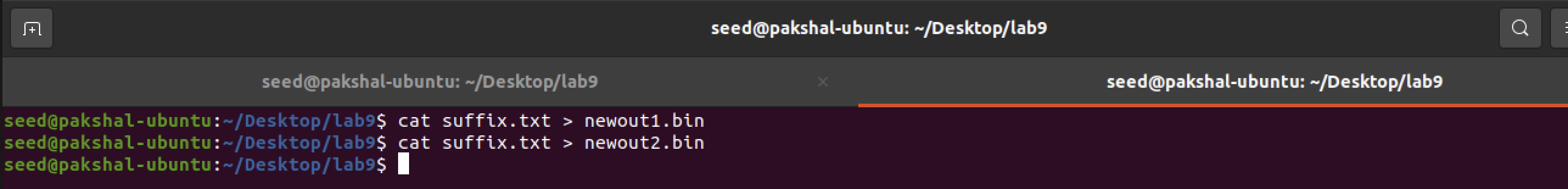
**A screenshot of a computer

Description automatically generated**

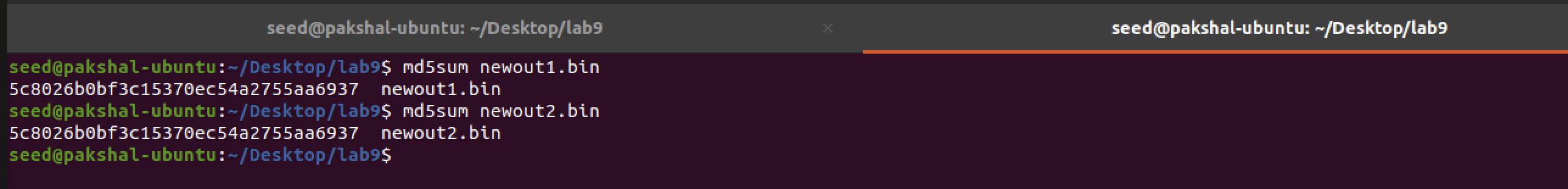
Now I have created the suffix file(suffix.txt) with the following contents.



Now we use the cat command to concatenate suffix.txt to both newout1.bin and newout2.bin.



We now generate the md5 hash values for both the above files i.e. newout1.bin and newout2.bin



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

In this task we are given a C program, and we to create two different versions of this program, such that the contents of their xyz arrays are different, but the hash values of the executables are the same.

A screenshot of a computer program

Description automatically generated

Compiling the C program using gcc into a.out file and opening it with bless to view the contents.

We now compile print\_array.c into a.out using gcc and then open the file using bless.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

We take the offset value to be 0x3040 which in decimal is 12352, as last two digits are divisible by 4.

We run the head command using the offset value to generate prefix.

A screenshot of a computer program

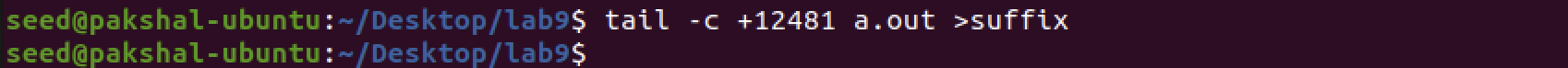
Description automatically generated

The binary files out1.bin and out2.bin have a size of 12480 bytes. The prefix has a size of 12352 bytes, so we start the suffix from 12481.

A screenshot of a computer program

Description automatically generated

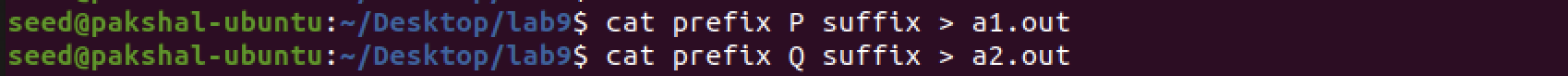
Now we generate the suffix by using the offset value of 12481.



Generating P and Q values.



Here we concatenate the prefix and suffix to both P and Q.



We use diff to see if both files are distinct or not.

A screenshot of a computer program

Description automatically generated

From the above picture we can see that both the binary files are different but there md5 hash is the same.

A screenshot of a computer

Description automatically generated

We can see that we have 2 different versions of the C program as show above where the contents of the array is different but the hash values of the executables are the same.

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

In this task, we must generate two versions of the same program array with the same hash values. Also, one program will always execute benign instructions, while the other program will execute malicious instructions.

A screenshot of a computer program

Description automatically generated

**A screen shot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

From above picture we can see that the binary files out2.bin and out1.bin have a size of 12480 bytes and prefix has 12352 bytes.

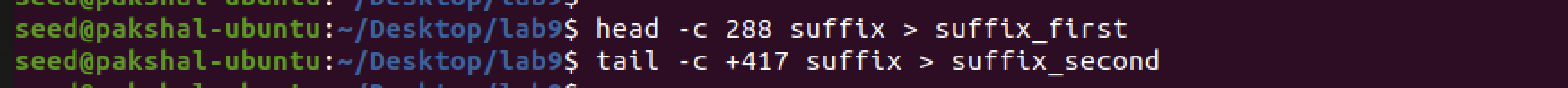
So, the suffix would start from 12481.



We need to overwrite the next 128 bytes of the array with P in out1.bin and Q in out2.bin as shown below.



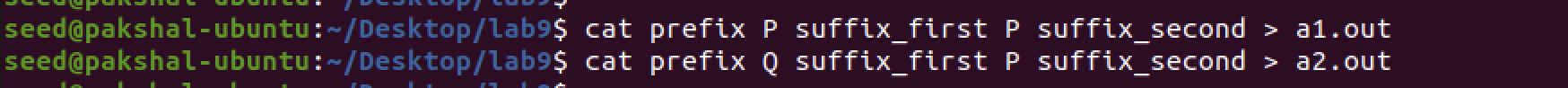
Now we write first 288 bytes with suffix\_first and everything but the first 417 bytes with suffix\_second because the suffix array has suffix\_first, P or Q and suffix\_second.



Now we concatenate the files we have found into the 2 output files.

First file: **prefix + P + suffix\_first + P + suffix\_second**

Second file: **prefix + Q + suffix\_first + P + suffix\_second**

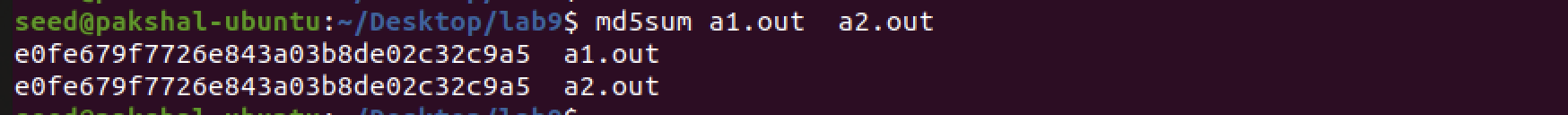
****

Now we execute and see the program to see if one program executes benign and another malicious.

**A purple background with white text

Description automatically generated**

We can see that the one is executing benign and other one malicious. Now we compute the md5 hash for both a1.out and a2.out.



We can see that both a1.out and a2.out have the same hash value. Hence we have successfully demonstrated the attack.