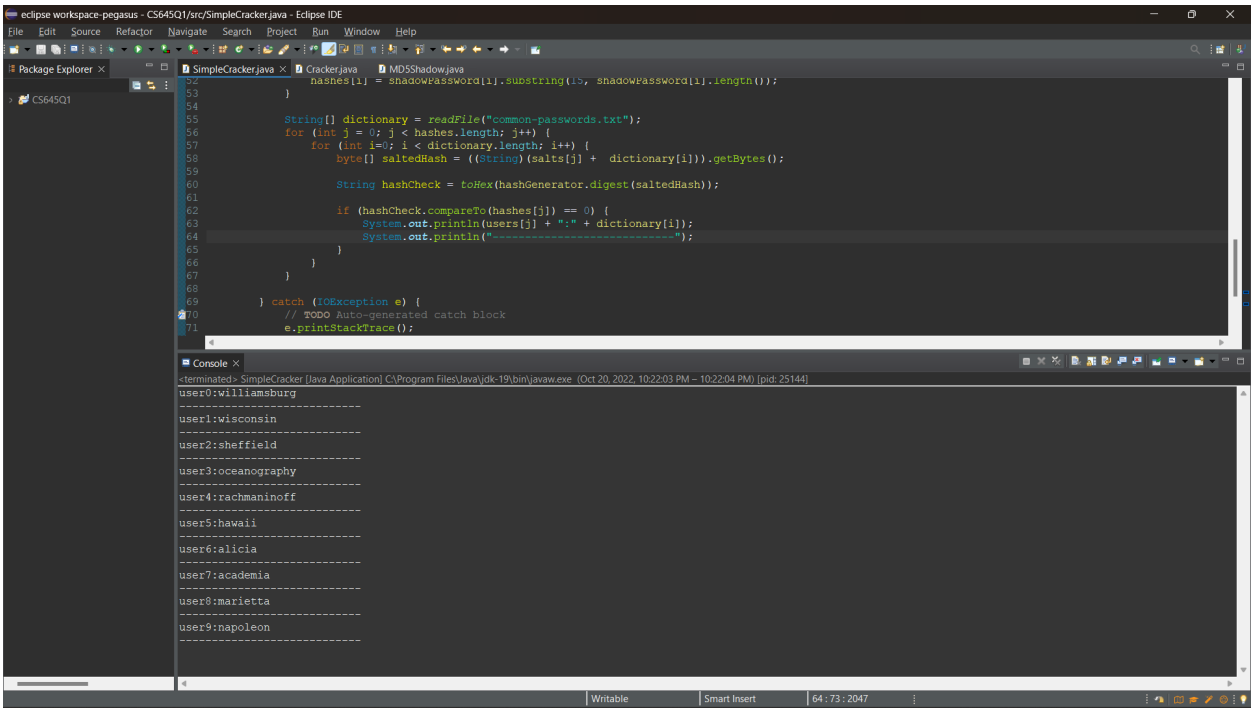
CS645: Security & Privacy in Computer System | Fall 2022 | Project 1

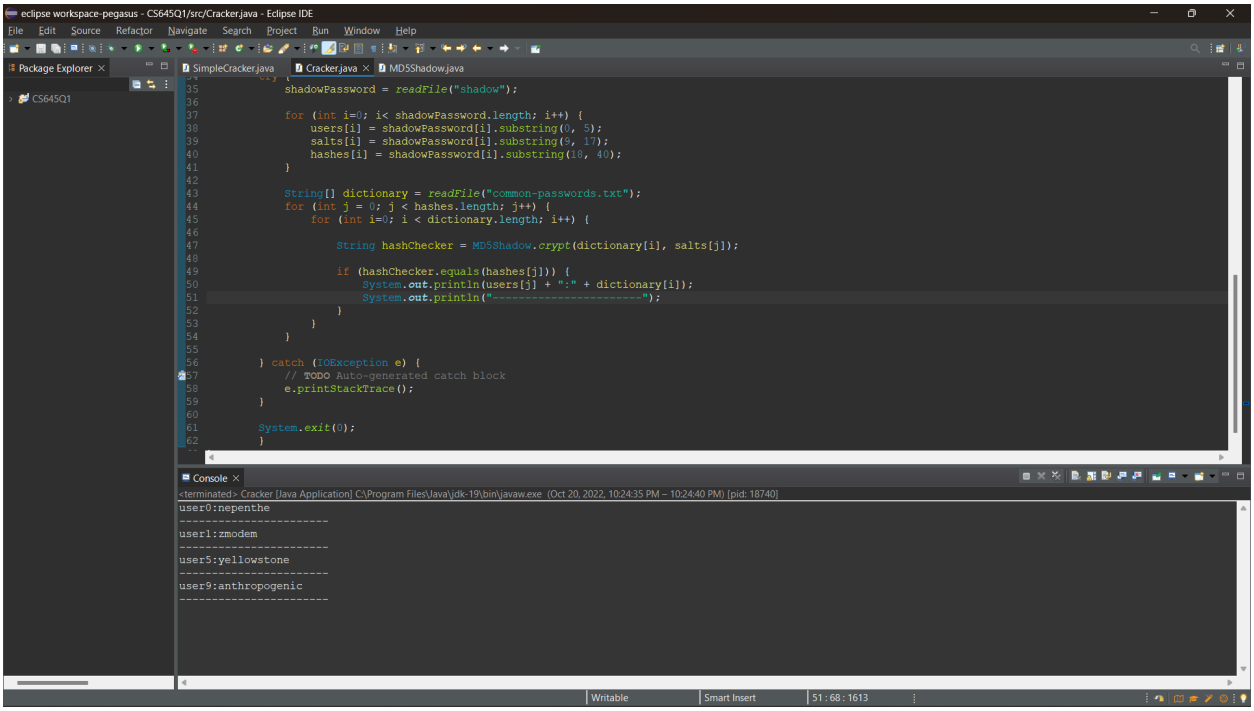
Submitted by:
Rudranil Maity (ucid- rm964), Kush Borikar (ucid- kb97) , Anshuman Singh (ucid- as4372)

Q1.
Part 1: Ans:



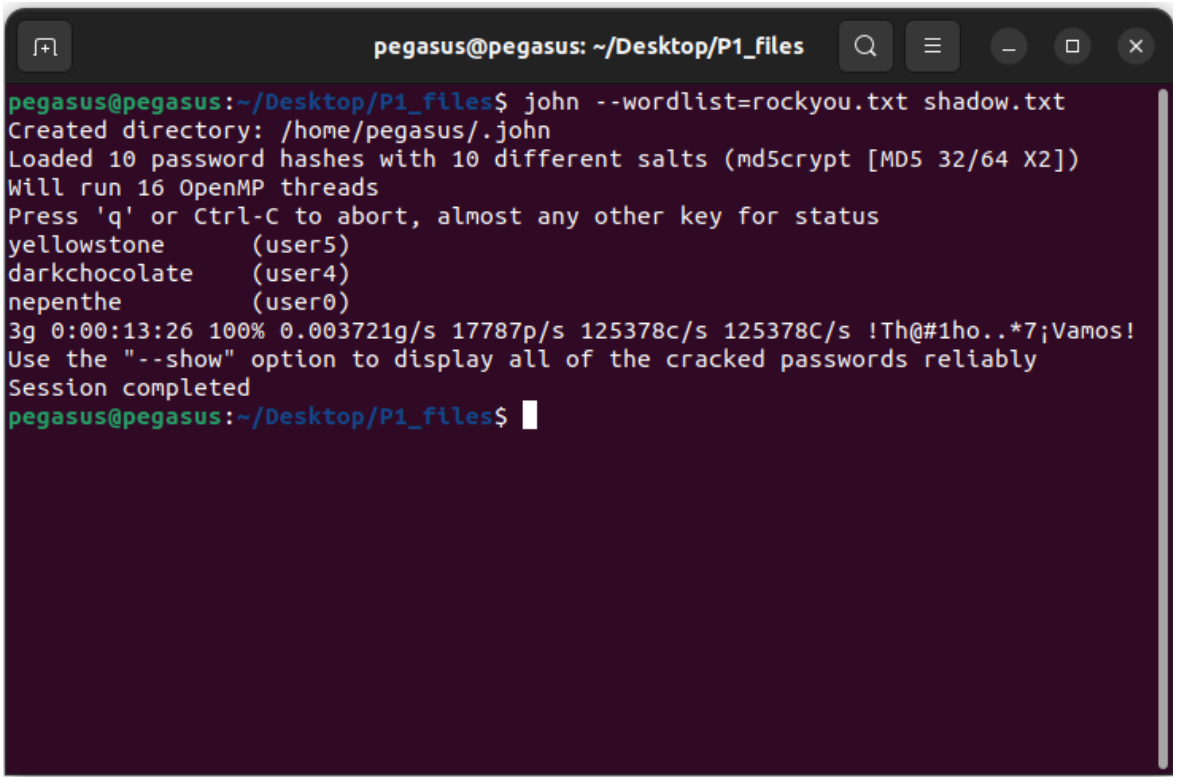
As we can see from the screenshot above of our running code “SimpleCraacker.java”, we’re getting the common passwords used by each user as the output by taking the “simple-shadow” and “common-passwords” files as the input.. The source code is attached with the assignment.

Part 2 : Answer:

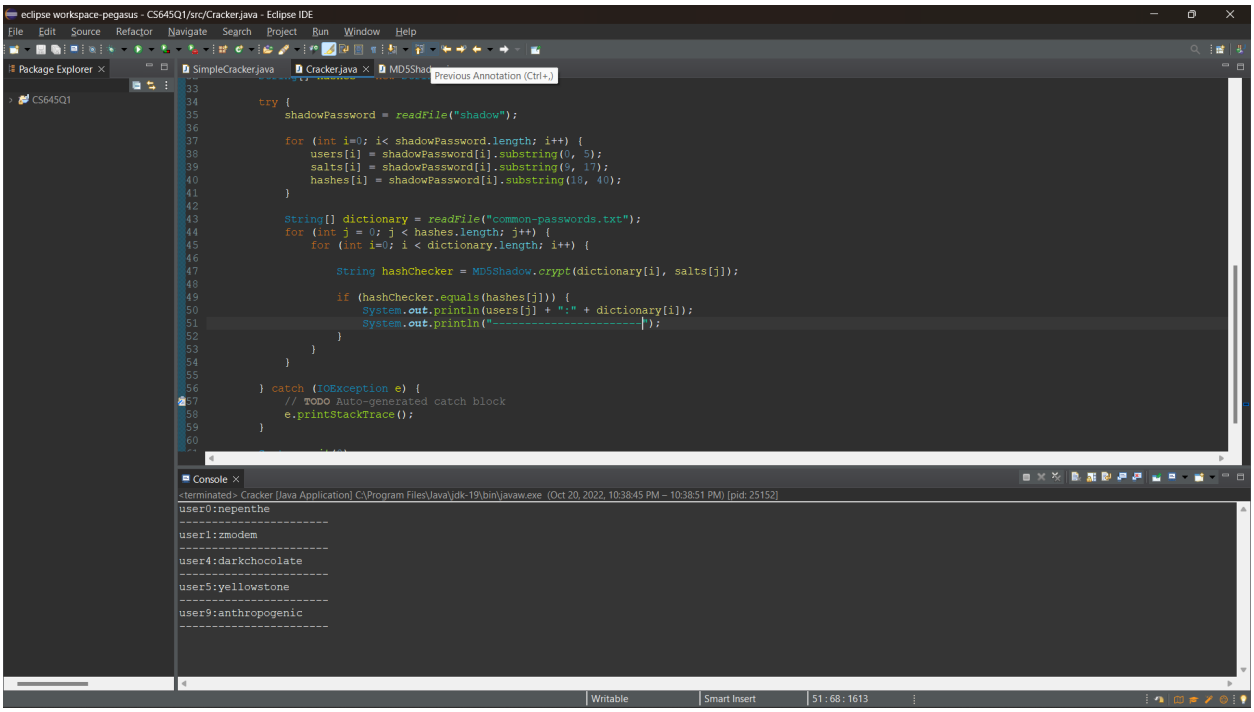


We can see from the above observation that we are getting the matched passwords of the users from the java file “Cracker.java”. The other users’ passwords didn’t match with the password file. We took “shadow” and “common-passwords” as the input. The source code is attached with the assignment.

Part3: Answer:



I ran “John the ripper” password cracker to get the required password which we needed to find using the “shadow” file and a dictionary file:”rockyou.txt”. As we can see, the found password is “darkchocolate” for user 4. The “rockyou.txt” file is uploaded with the assignment. Now we added this word to the common-passwords file in the program “Cracker.java” and compiled and ran it.

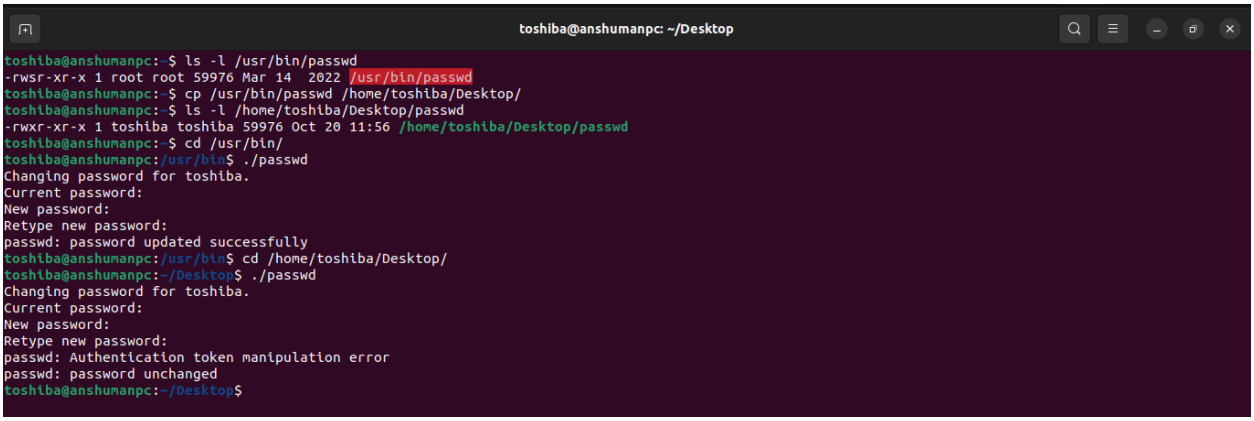


After running the updated “common-passwords” file, we can see that the updated list can catch one more matched password for user4, which is “darkchocolate”.

***FOR RUNNING THE PROGRAMS, WE COPIED THE MD5Shadow.java FILE TO THE PROJECT DIRECTORY AND INHERITED THE “crypt” METHOD.**

Q2. (a) Figure out why the “passwd” command needs to be a Root Set-UID program. What will happen if it is not? Login as a regular user and copy this command to your own home directory (usually “passwd” resides in /usr/bin); the copy will not be a Set-UID program. Run the copied program, and observe what happens. Describe your observations and provide an explanation for what you observed.

Ans:

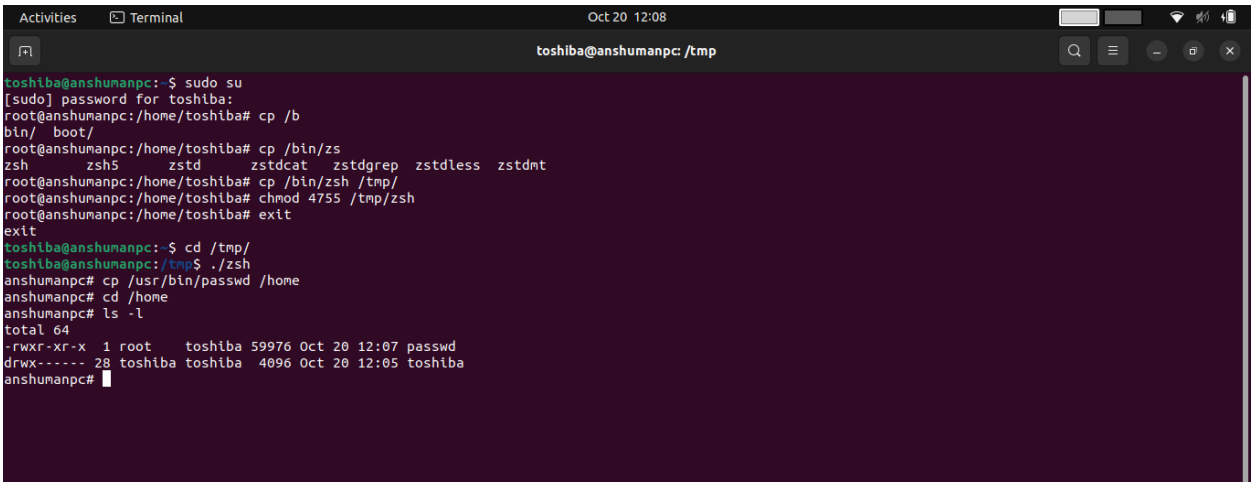


The “passwd” need to be Set-UID programs because we will need permission to change the password or even access a few files when necessary. As seen on line 3 on the above screenshot we’ve got from our observation; the “passwd” command is a Set-UID program, hence root privilege is needed for this. When we run the passwd command from the /usr/bin directory(the root directory) then it is running with the root privileges and it is able to change the password of the system. If it is not a Set-UID program then any of the users who are logging in to the system will be able to change the password of the system.

But when we’re copying it at the home directory after logging in as a regular user; it is losing its root privileges and after running it is not changing the system password.

(b1) Login as root, copy /bin/zsh to /tmp, and make it a Set-UID program with permissions 4755. Then login as a regular user, and run /tmp/zsh. Will you get root privileges?

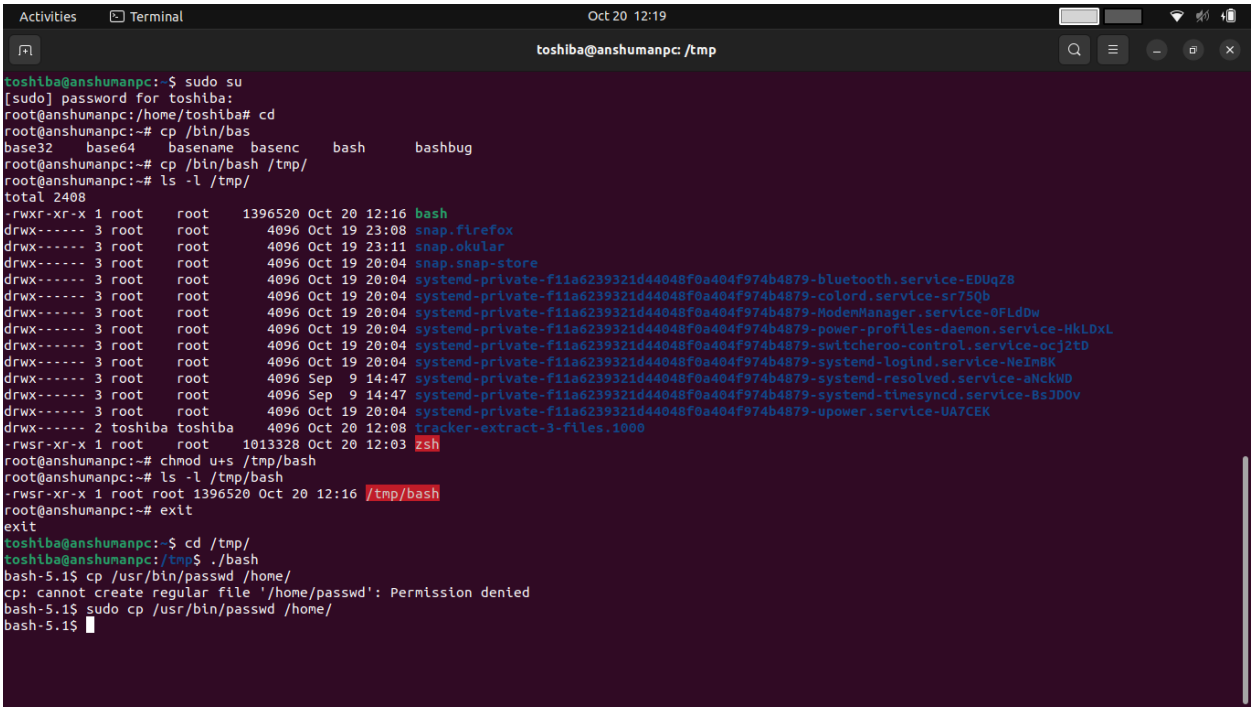
Ans:



As seen from the above observation, after logging in as a regular user, and running /tmp/zsh; root privilege is gained. First we have made it a Set-UID program with permission 4755. And then logged in as a regular user.

(b2) Login as root and instead of copying /bin/zsh, this time, copy /bin/bash to /tmp, make it a Set-UID program. Login as a regular user and run /tmp/bash. Will you get root privilege? Please describe and provide a possible explanation for your observation.

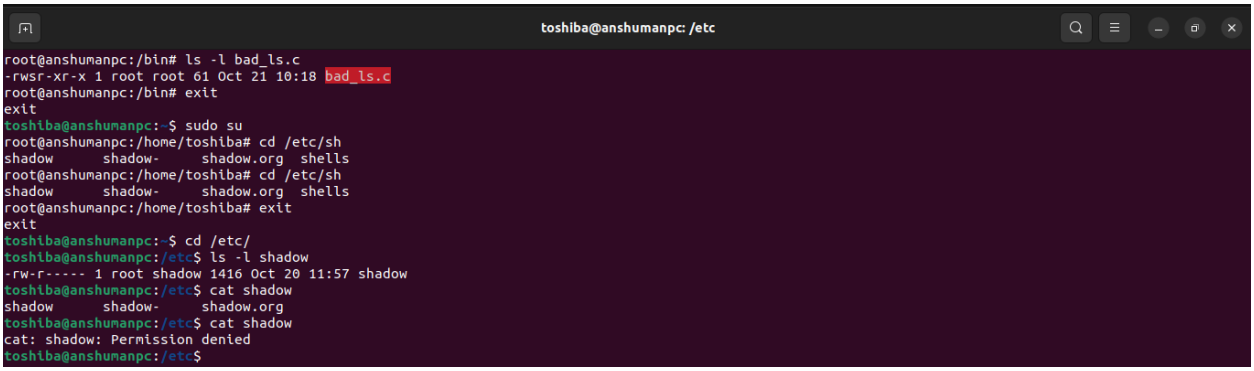
Ans:



As we can see from the above screenshot from our observation, at first the bash program was not a Set-UID program. Then we made it a Set-UID program by running *“chmod u+s /tmp/bash”* . After that we logged in as a regular user and tried to run the *“passwd”* command; which is root privileged. But it was unable to run the program. Hence, bash doesn't get root privilege.

(c1) Login as root, create a new directory /tmp1 and set it to have the same permissions as /tmp, write this program into a file named bad_ls.c, compile it (using gcc –o bad_ls bad_ls.c) and copy the executable as a Set-UID program into /tmp1 with permissions 4755. Is it a good idea to let regular users execute the /tmp1/bad_ls program (owned by root) instead of /bin/ls ? Describe an attack by which a regular user can manipulate the PATH environment variable in order to read the /etc/shadow file.

Ans:



The problem is that system("ls") would run whichever executable named ls it finds first in the user's set PATH.

This ls does not necessarily have to list the contents of a directory. Instead it could be a script like this:

#!/bin/sh

cat /etc/shadow

Let's say you place this script somewhere in a directory below your home directory, for example **/home/toshiba/bin** and add this to your PATH:

PATH="/home/toshiba/bin:$PATH"

If you now run ls, you will not get a directory listing, instead you will receive an error message:

cat: /etc/shadow: Permission denied

But if you run **bad_ls**, the ***system("ls")***-call therein will also look for an executable named *ls* in your PATH and find ***/home/toshiba/bin/ls*** instead of ***/bin/ls***. As bad_ls runs with elevated root permissions due to the Set-UID, the script named ls will also run with elevated root permissions and so will the command ***cat /etc/shadow***, which will print the contents of ***/etc/shadow***.

So it is a bad idea for root to let normal users run *bad_ls* as long as it has SUID privileges, because it would run any program named ls that comes first in the user's PATH.

(c2)Now, change /bin/sh so it points back to /bin/bash, and repeat the above attack. Can you still get the root privilege and list the contents of the /etc/shadow file? Describe and explain your observations.

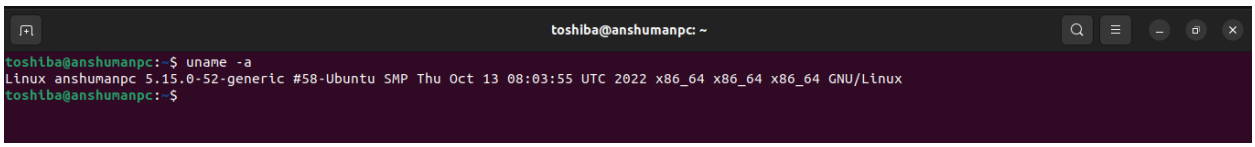
Ans: The function of a SetUID program is to allow any regular user to run the file with the privileges of the user who created the program, in this case *root*, instead of the user who executes the program, the regular user.

In (c1), we changed the default pointer of the shell to zsh which is a SUID program. Now after changing the pointer of /bin/sh from zsh back to /bin/bash, any regular user would not get the elevated privileges of the root user as the bash is not available as a SetUID program.

This means that if any regular user tries to access the /etc/shadow file, a "*Permission denied*" error is displayed because the file is being accessed by the user's privileges instead of root privileges.

(c3)Specify what Linux distribution you used for Problem 2 (distribution & kernel version). You can find this information by running the command "uname –a"

Ans:



Kernel name: Linux

Node name: anshumanpc

Kernel release: 5.15.0-52-generic

Kernel version: #58-Ubuntu SMP Thu Oct 13 08:03:55 UTC 2022

Operating system: GNU/Linux

Q3 - Consider the following security measures for airline travel. A list of names of people who are not allowed to fly is maintained by the government and given to the airlines; people whose names are on the list are not allowed to make flight reservations. Before entering the departure area of the airport, passengers go through a security check where they must present a government-issued ID and a boarding pass (the check done here is based on visual inspection: the person must match the picture on the ID and the name on the ID must match the name on the boarding pass). Before boarding a flight, passengers must present a boarding pass, which is scanned to verify the reservation (the check done here is to ensure the scanned information from the boarding pass matches an existing reservation in the system). Show how someone who is on the no-fly list can manage to fly provided that boarding passes could be generated online (as an HTML page) and then printed. Please provide a step-by-step description of the attack. Which additional security measures should be implemented to eliminate this vulnerability?

Solution -

Step 1: Using a prepaid credit/debit card purchased at a Dollar Tree or a 7/11 using cash, purchase an online ticket for a fake passenger. (Make sure not to use an extremely common name.)

Step 2: Perform a Web Check-in 24 hours prior to departure & print out the boarding pass.

Step 3: Edit the HTML of the boarding pass to make up an identical boarding pass, except that the copy lists your real name. Print this out.

Step 4: Once you reach the airport, present the boarding pass that includes your real name and along with it your real ID. The TSA will verify if both these things match and let you into the airport with minimal search. (Note: This step does not include an electronic verification of the boarding pass.)

Step 5: You use the fake-name boarding pass to board the flight at the boarding gate. Since this is the name, the ticket was purchased under, this will not ring any alarms and you can board the flight without revealing your identity.

Q - How do we fix this security issue?

- Increasing electronic security check-points – TSA employees must perform an electronic verification of the boarding pass more than once rather than one done only at the boarding gate of the flight.
- Ban Online Boarding Passes – Online boarding passes should not be allowed; they are easy to spoof. Boarding passes should be issued only at check-in counters at the airport.