CS 645 - Project 1

Ida Jebakirubai Stephen Joseph

Susmita Biswas

Included answers for Problem 1 (part3), Problem 2 and Problem 3

Problem 1(Part 3)

*1.*Searched with "dictionary attack password list download" on the web and found the link https://wiki.skullsecurity.org/Passwords . Downloaded the file rockyou.txt.bz2 and ran the program Cracker.java that we have developed in Part 2. It decrypted password – 'darkchocolate' of user4 which was not present in the given password file – common-passwords.txt.

Hence after adding the password - 'darkchocolate' to common-passwords.txt file, the program Cracker.java output the below result:

user0:nepenthe

user1:zmodem

user4:darkchocolate(extra output)

user5:yellowstone

User9:anthropogenic

Problem 2

(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.

Answer1

The passwd command is belonging to the root as set-UID. As learnt in the class, if the setuid is set on an executable file, then any users able to execute the file will automatically execute the file with the privileges of the files owner. This allows the system designer to permit trusted programs to be run which a user would otherwise not be allowed to execute.

```
-rwsr-xr-x  1 root root       53128 Mar 29  2016 passwd
```

After coping the passwd to the own home directory, we can see that the passwd program has lost its setuid bit, and the access is just -rwxr-xr-x 1; This prevents access are not getting transferred down.

```
-rwxr-xr-x  1 root root 53128 Oct 18 17:24 passwd
```

(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.

Answer2

When we copy the zsh file as logged as root and grant the set-UID, the access prevailed when we log as a normal user. We can see that the file zsh under /tmp/ folder has the set-UID enabled, -rwsr-xr-x.

This is potentially a loophole where zsh is not protected to be prevent the access leak and attack by the attacker in the server side by creating a symbolic link which is extremely dangerous.

Commands used:

```
[10/11/20]seed@VM:~$ sudo su
root@VM:/home/seed# cp /bin/zsh /tmp
root@VM:/tmp# chmod 4755 zsh
root@VM:/tmp# exit
[10/11/20]seed@VM:~$ cd /tmp
[10/11/20]seed@VM:/tmp$ ./zsh
[10/11/20]seed@VM:/tmp$ ls -l /tmp/zsh
-rwsr-xr-x 1 root root 756476 Oct 11 22:35 /tmp/zsh
```

(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.

Answer3

No, we did not get the root privilege. While copying the /bin/bash and giving the set-uid permission, the access is not granted like zsh.
Bash seems to have been protected. /bin/bash has certain built-in protection that prevent the abuse of the Set-UID mechanism.This would prevent the hackers to not get a hold of the accounts and resources in the system and the way to create symbolic link is also restricted through this.

```
[10/11/20]seed@VM:/tmp$ sudo su
root@VM:/tmp# cp /bin/bash /tmp/
root@VM:/tmp# chmod u+s bash
root@VM:/tmp# exit
exit
[10/11/20]seed@VM:/tmp$ ls -al bash
-rwsr-xr-x 1 root root 1109564 Oct 11 23:24 bash

[10/11/20]seed@VM:/tmp$ ./bash
```

```
bash-4.3$ id
uid=1000(seed) gid=1000(seed)
groups=1000(seed),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),113(lpadmin),128(samb
ashare)
bash-4.3$
```

(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.

Answer4:

No, it is not a good practice to allow regular user to execute the program /tmp1/ bad_ls instead of /bin/ls.
We can observe that by copying the sh command, we are able to grant the root access to the program.
This is potential risk as the hacker can get hold of the environment variable to run the unauthorized program by means of getting hold of root password.
Buy this way they are able to access the /etc/shadow file. The /etc/passwd file contains basic information about each user account on the system, including the root user which has full administrative rights, system service accounts, and actual users.

Commands used:

```
[10/13/20]seed@VM:~$ cd /bin
[10/13/20]seed@VM:/bin$ sudo su
root@VM:/bin# ls -l sh
lrwxrwxrwx 1 root root 4 Jul 25  2017 sh -> dash
root@VM:/bin# rm sh
root@VM:/bin# ln -s zsh sh
root@VM:/bin# ls -al sh
lrwxrwxrwx 1 root root 3 Oct 13 12:47 sh -> zsh
root@VM:/# mkdir /tmp1
root@VM:/# chmod --reference=tmp tmp1
root@VM:/tmp1# sudo vi bad_ls.c
#include int main() { system("ls"); return 0; }
root@VM:/tmp1# gcc -o bad_ls bad_ls.c
root@VM:/tmp1# chmod 4755 bad_ls
root@VM:/tmp1# exit
[10/15/20]seed@VM:/tmp1$ cp /bin/sh /tmp1/ls
[10/15/20]seed@VM:/tmp1$ ./bad_ls
\ a.out  bad_ls  bad_ls.c  ls
[10/15/20]seed@VM:/tmp1$ ls
\ a.out  bad_ls  bad_ls.c  ls
```

(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.

Answer 5

Now we lost the privilege of the root after setting sh to bash. As we have already saw in the earlier exercise, bash has security mechanism which prevents the inbuilt attack by any unsolicited programs.
But leveraging bash, we can prevent the leak of the content of /etc/shadow file by preventing the user confidentiality.

Commands used:

```
[10/15/20]seed@VM:/tmp1$ sudo su
root@VM:/tmp1# cd /bin
root@VM:/bin# rm sh
root@VM:/bin# ln -s bash sh
root@VM:/bin# la -al sh
lrwxrwxrwx 1 root root 4 Oct 15 00:50 sh -> bash
root@VM:/bin# exit
exit
[10/15/20]seed@VM:/tmp1$ ./bas_ls
```

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

Answer 6:

Distribution:

```
[10/18/20]seed@VM:~$ cat /etc/*-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=16.04
DISTRIB_CODENAME=xenial
DISTRIB_DESCRIPTION="Ubuntu 16.04.2 LTS"
NAME="Ubuntu"
VERSION="16.04.2 LTS (Xenial Xerus)"
```

ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 16.04.2 LTS"
VERSION_ID="16.04"
HOME_URL="http://www.ubuntu.com/"
SUPPORT_URL="http://help.ubuntu.com/"
BUG_REPORT_URL="http://bugs.launchpad.net/ubuntu/"
VERSION_CODENAME=xenial
UBUNTU_CODENAME=xenial

Kernel

[10/18/20]seed@VM:~$ uname -a
Linux VM 4.8.0-36-generic #36~16.04.1-Ubuntu SMP Sun Feb 5 09:39:41 UTC 2017 i686 i686 i686 GNU/Linux

Problem 3

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.

1. Buy a ticket online, using a prepaid credit card purchased with cash, for a fake passenger name. Make sure you do not use the name which are already on the no-fly list.
2. Check in online 24 hours before departure and print out this boarding pass.
3. Edit the HTML of the boarding pass and put your real name to make up an identical boarding pass same as the boarding pass printed in step#2. Print this out as well.
4. Go to the airport, and present the boarding pass with your real name and your real ID before entering the departure area of the airport, where passengers go through a security check and they have to present a government-issued ID and a boarding pass. As the check done is by visual inspection, they will check that they match, and then let you through the checkpoint with a minimal search.
5. Before boarding a flight, present the fake-name boarding pass to the gate-agent when you board the flight. As the check done here is to ensure the scanned information from the boarding pass matches an existing reservation in the system, so everything will look good.

Which additional security measures should be implemented in order to eliminate this vulnerability?

1. Provide the security employees scanners/computers at the gate, so that they can verify the validity of the boarding passes along with the ID proof when someone reaches at the gate.
2. Do NOT allow people to print out boarding passes online.
3. Facial ID scan along with boarding pass scan at the gate to find out any discrepancies.