HW 10

Mike Hanling

09APR18

Questions

1. (5 points) Why must there be a de-escalation of privilege when the login program executes the shell for an authenticated user?

   The getty program runs as root, and not all users have the sam permissions as root. Therefore, permissions need to be scaled back to just those fo the authenticated user.

2. (10 points) Consider the following program with the following permission strings below, if you (as your username) were to run these programs, what capabilities (group and user permissions) would the executing program have?

   (a) -rwxr-x–x 1 aviv scs 8622 Mar 30 10:40 a.out

   User: execute
   Group: none

   (b) -rwsr-x–x 1 aviv scs 8622 Mar 30 10:40 a.out

   User: read/write/execute
   Group: none

   (c) -rwxr-s–x 1 aviv scs 8622 Mar 30 10:40 a.out

   User: execute
   Group: read/execute

   (d) -rwsr-s–x 1 aviv scs 8622 Mar 30 10:40 a.out

   User:read/write/execute
   Group: read/execute

3. (5 points) What is the difference between the real and effective user and group id of a running process?

   The real user id is the actual identifier of the current user.
   The effective user id is the identifier that the prrogram will run under.

4. (15 points) Provide a short, plain-English, description of each of the system calls below:

   (a) getuid()

Get the real user id.

(b) getgid()

Get the real group id.

(c) geteuid()

Get the effective user id.

(d) getegid()

Get the effective group id.

(e) setuid(uid)

Set the real user id to uid.

(f) setgid(gid)

Set the real group id to gid.

(g) setreuid(uid,euid)

Sets the real user id to uid and the effective user id to euid.
Using -1 for either leaves it unchanged.

(h) setregid(gid,egid)

Sets the real group id to uid and the effective group id to euid.
Using -1 for either leaves it unchanged.

5. (10 points) Consider the following chmod statements, provide the permission string, that is the permission string rwxrwxrwx represents 777. Be careful about setbits.

(a) chmod 6750 a.out

rwsr-s—

(b) chmod 4750 a.out

rwsr-x—

(c) chmod 2750 a.out

rwx-r-s—

6. (5 points) Suppose you are writing a setuid program, and you want downgrade the effective permission of the program back to the user who is executing the program. Provide one line of C that can do that.

seteuid(getuid());

7. (5 points) What does the library call system() do?

   It does the fork-exec-wait cycle for us. Place any command to run at the command in quotes as the argument, and system() executes it.

8. (5 points) Explain how the environment variable PATH is used to select which program to execute when using execvp() or system() or in a shell?

   The actual executable for the command is searched in all the directories (in order) in the PATH variable. Any time a command is run, the PATH variable is used to find the executable.

9. (10 points) The following program has a (multiple!) security flaw, describe how to exploit it. And, provide at least one way to change the program to protect it from the attack you described?

```
#include <stdio.h>
#include <stdlib.h>
int main(){
  system("cat sample.db | cut -d ',' -f 3 | sort  | uniq");
}
```

   If the PATH variable is corrupted, then different executables can be run if they have the same name, and are found earlier while searching through PATH. One way to fix that is to use execv (not execvp) in the classic fork-exec-wait cycle and specifically give the executable to be run.

10. (10 points) The following program has two security flaws, describe them and how to exploit them.

```
int main(){
  char cmd[1024];
  char fname[40];
  printf("Enter file name:\n");
  scanf("%s",fname);
  snprtinf(cmd,1024,"/bin/cat %s",fname);
  system(cmd);
}
```

    Overflow on fname and not checking for bash code. Understanding how much buffer is allotted to fname and then presenting more would then overwrite into other variable, such as cmd. Also, if bash code is used as input properly (add a ; and write some bad stuff), then the program has been taken over. In addition, the PATH variable is not set with setenv(), so a PATH attack could also take place.

11. (10 points) Describe a solution to each of the security flaws you identified in the previous question.

    When reading in fname, use bound checking the same way it was done for cmd. Check for any bash symbols in the input that could be hazardous and turn them to null. Use setenv() in the beginning of the main to ensure that proper directories are being checked for the executables.

12. Consider yourself as a software developer designing a tool for your organization that takes advantage of different UNIX system tools. As such, you wish to make use of the system() and popen() calls to inter operate with your tool and the standard UNIX tools. While the tool need high privilege levels (e.g., to log users in, access different information), individual users may need varying lesser privilege levels, but not necessarily equal across users.
    (5 points) Describe three potential ethical and legal impacts on your organization (with respect to actions attackers could take) if your software was designed insecurely.

Assuming I leave vulnerabilities open on purpose or find out about them after the fact:
1. I could easily curl some executables, chmod them, and then PATH attack any computer.
2. I could raise my permissions when executing and then rm -rf /, getting rid of all that person's files.
3. I could raise permissions and use that to snoop on people/collect info about them that I was not authorized to get.

(5 points) Describe a three coding practices you can employ to reduce the ethical and legal impacts of insecurity in your software.

1. I can ensure to use setenv() in my programs to make sure PATH attacks will not happen.
2. I can always bounds check any input to suppress buffer overflow attacks.
3. I can scrub all input of any bash-like characters/commands in order to not allow for bash to be inadvertently run.