Secure System Design: Threats and Countermeasures
CS392

Date: 17th Jan 2019 Assignment 1
Submission Filename: firstname.pdf Due Date: 22nd Jan 2019
Full Marks 50

Description

Set-UID is an important security mechanism in Unix like 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, but unfortunately, it is also the culprit of many bad things. Therefore, the objective of this assignment is two-fold: (1) Appreciate its good side: understand why Set-UID is needed and how it is implemented. (2) Be aware of its bad side: understand its potential security problems.

Tasks

This is an exploration assignment. Your main task is to "play" with the Set-UID mechanism in linux, and write a report to describe your discoveries. You are required to accomplish the following tasks in linux:

1. Figure out why *"passwd"*, *"chsh"*, *"su"*, and *"sudo"* commands need to be Set-UID programs. What will happen if they are not? If you are not familiar with these programs, you should first learn what they can do by reading their manuals. Please copy these commands to your own directory; the copies will not be setuid programs. Run the copied programs, and observe the differences (if there is any) with the corresponding setuid programs. $4 \times 5 = 20$

2. (a) Login as root, copy /bin/zsh to /tmp, and make it a set-root-uid program with permission 4755. Then login as a normal user, and run /tmp/zsh. Will you get root privilege? Please describe your observation. If you cannot find /bin/zsh in your operating system, please use the following command to install it:
   - Note: in our pre-built *Ubuntu* VM image, *zsh* is already installed.
   - For *Fedora*
     ```
     $ su
       Password: (enter root password)
     # yum install zsh
     ```
   - For Ubuntu
     ```
     $ su
       Password: (enter root password)
     # apt-get install zsh
     ```

   (b) Instead of copying /bin/zsh, this time, copy /bin/bash to /tmp, make it a set-root-uid program. Run /tmp/bash as a normal user. will you get root privilege? Please describe and explain your observation.
   $2 \times 5 = 10$

3. Different shells may have different built-in protection mechanisms to prevent the abuse of the setuid mechanism. This experiment will help you to explore whether there is any differences in two shells /bin/bash and /bin/zsh.

   In some linux distributions (such as Fedora and Ubuntu), /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*.

   ```
   $ su
     Password: (enter root password)
   # cd /bin
   # rm sh
   # ln -s zsh sh
   ```

   The difference between system() and execve(). *Before you work on this task, please make sure that* /bin/sh *is pointed to* /bin/zsh.

Background: Bob works for an auditing agency, and he needs to investigate a company for a suspected fraud. For the investigation purpose, Bob needs to be able to read all the files in the company's *unix* system; on the other hand, to protect the integrity of the system, Bob should not be able to modify any file. To achieve this goal, Vince, the superuser of the system, wrote a special set-root-uid program (see below), and then gave the executable permission to Bob. This program requires Bob to type a file name at the command line, and then it will run `/bin/cat` to display the specified file. Since the program is running as a root, it can display any file Bob specifies. However, since the program has no write operations, Vince is very sure that Bob cannot use this special program to modify any file.

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
  char *v[3];

  if(argc < 2) {
    printf("Please type a file name.\n");
    return 1;
  }

  v[0] = "/bin/cat"; v[1] = argv[1]; v[2] = 0;

  /* Set q = 0 for Question a, and q = 1 for Question b */
  int q = 0;
  if (q == 0){
    char *command = malloc(strlen(v[0]) + strlen(v[1]) + 2);
    sprintf(command, "%s %s", v[0], v[1]);
    system(command);
  }
  else execve(v[0], v, 0);

  return 0 ;
}
```

(a) Set $q = 0$ in the program. This way, the program will use `system()` to invoke the command. Is this program safe? If you were Bob, can you compromise the integrity of the system? For example, can you remove any file that is not writable to you? (Hint: remember that `system()` actually invokes `/bin/sh`, and then runs the command within the shell environment. We have tried the environment variable in the previous task; here let us try a different attack. Please pay attention to the special characters used in a normal shell environment).

(b) Set $q = 1$ in the program. This way, the program will use `execve()` to invoke the command. Do your attacks in task (a) still work? Please describe and explain your observations. $5 + 5 = 10$

4. Now reset `/bin/sh` to `/bin/bash` and carry out the previous experiment. State your observations in terms of protection that `/bin/bash` provides over `/bin/zsh`. $5 + 5 = 10$

Submission

You need to submit a detailed report to describe what you have done and what you have observed; you also need to provide explanation to the observations that are *interesting* or *surprising*. Add necessary snapshots of your experiment wherever applicable in support of your observation. Upload your file using following link only.

https://my.pcloud.com/#page=puplink&code=IJvZjp3Tnkdog98L8hSpYogyDSoUERMk