Massachusetts Institute of Technology
6.915: Network and Computer Security
Professor Ronald L. Rivest

Handout 34
November 28, 1995

Problem Set 6 Solutions

Problem 6-1. User–mountable file systems

The `mount` command may be used to abrogate system security. For example if a floppy disk is mounted over the true `/etc` directory the password file may be modified by the person mounting the floppy.

However if you look at the man page of the `mount` command you will see that it requires that you own that directory you are mounting on. In this case the above attack would not work.

Problem 6-2. Write permission to terminal

Find out what terminal is used by the system superuser and reprogram on of its function keys to give you superuser privileges. Notice that you may not need write permission to the terminal in order to do this. For example you can obtain write permission by opening a `talk` channel with the superuser. Or you could access the superuser terminal when he/she is logged off and use your login to reprogram the keys. If the superuser does not reprogram the keyboard every time he/she logs in, the attack will succeed.

Problem 6-3. Non–atomic updates

Here is a possible attack:

1. Look for a writable directory in the same file system of the file you wish ot own (for example `/etc/passwd`.)
2. Next a program attempts to run the `mkdir` command to create a directory in `/tmp`. Let's call it `foo`. Make the `mkdir` command run in the background with the `nice` option to make it low priority. This will increase the opportunity window.
3. Meanwhile in hte foreground, running at normal priority, is a program watching for the creation of the directory `/tmp/foo`.
4. As soon as you foreground program spots the `tmp/foo` directory, it removes it and creates a link from `/etc/passwd` to `/tmp/foo`.
5. If the timing is successful, the `mkdir` command will come along and change the ownership of `/tmp/foo` (a.k.a. `/etc passwd`) to you.

Problem 6-4. The comand `chroot`

First of all I would like to clarify the functioning of `chroot`. From the man page of UNIX:
`/usr/etc/chroot newroot command`

> The given `command` is executed relative to the new root. The meaning of any initial slashes in path names is changed for a command and any of its children to `newroot`. Furthermore, the initial working directory is `newroot`.

This means that running `chroot newroot command` will run a single execution of the command `command` where `newroot` is going to be the root directory. Conversely to things that I said during office hours `chroot` does not change the root for future executions of `command`.

If `chroot` was widely available to every user then an user could mount the following attack.

- Prepare a fake `/etc/passwd` password file in his/her directory.
- Run `login` while chrooted to his/her own directory. Since the login will run with his/her directory as root it will use the fake password file.
- Log in as root using the new `/etc/passwd` in your directory. This is not enough since all the process the user will start are children of the `login` process, so they are confined to the new root directory (i.e. the user directory).
- Create a "setuid to root" script file in his/her directory.
- Exit the chrooted process, and now use the script file to become root again.

Problem 6-5. Viruses

This one was hard. There were many suggestions given by the students. The most common was to have the virus utilize a scanner–like technique (look for "scanner's signatures") in order to infect scanners first.