

CSC 490 Computer and Network Security Programming Assignment

Due 6:00PM, Monday, March 3, 2008

(100 Points)

Traditional UNIX security model allows users to set different levels of access to a file based on basic permissions modes and three classes of users (owner/group/other). You can use this model to grant any combination of the permissions to any class of users. To complete this homework, you will have to implement a file system permission simulator (program).

Your simulator will read a series of commands from standard input, and the commands will perform a variety of file system operations. The file system tree in your simulator has to be represented by an in-memory data structure. In the simulator, each command has to be checked for permission. If the user does not have the appropriate permission, the command must be rejected.

To make your simulator work, you need to let the simulator read two other files containing a list of valid user names and group names. Note that the list of valid users and groups will not change during the course of the execution of your program.

The format of the input commands is:

user.group command pathname
where "user" and "group" are valid names per your configuration files, and "*" indicates every user or every group.

The input commands include

mkdir	create directories;
rmdir	delete directories;
create	create files.
delete	delete files.
read	read files
write	write files
acl	change the access control list on a file.
dump	print the file system and its state
chown	change the ownership of a file

It is assumed that each command is input by that user. The user and group files are system files.

The "mkdir", "create", and "acl" commands may all be followed in the input stream by a multi-line ACL. Each line of the ACL starts with one or more white space characters, per the usual Unix definition. The format of the lines is:

```
user.group permstring
```

Again, "user" and "group" are valid names per your configuration files or "*". "permstring" may be either "-" or some sequence of the letters "r", "w", and "x". Note that "r", "w", and "x" denote read, write, and execute, respectively. The sequence of ACL entries for that file or directory are terminated by the occurrence of the next command, i.e., something that's not preceded by white space.

Pathnames are Unix-style, i.e., elements are delimited by /.

The chown command takes a mandatory, single-line ACL as input. The permissions and group are ignored; the user is used for the new owner and group of the file.

At the start of execution, the file system consists of a single node / owned by some user, with a null ACL.

Your input data must create the file /etc/passwd that is world-readable but writable only by the system administrator. There must also be a directory /tmp that is world-writable.

Please note that the specified semantics may be ambiguities and, therefore, you have to make your own assumptions and decisions before documenting them.

Submission Guidelines:

Please use tar to archive your source program, a Makefile, one or more sets of test data, and README files together.

Each set of test data includes a user file, a group file, and a set of commands and ACLs. The README file should document a way of using your program and any assumptions made about the semantics.

The archive should be called '<your name>.prog.tar'.

No hardcopy submission is required for this assignment. E-mail your archive to xqin@auburn.edu with the subject line of "[CSC 490] Programming Assignment" before the beginning of class on the due-date 03/03/2008. **In the body of your e-mail, please include your name and student-id number.**

In what follows, we describe how to use tar:

To combine multiple files and/or directories into a single file, use the following command:

```
tar -cvf file.tar inputfile1 inputfile2
```

Replace `inputfile1` and `inputfile2` with the files and/or directories you want to combine. You can use any name in place of `file.tar`, though you should keep the `.tar` extension. If you don't use the `f` option, `tar` assumes you really do want to create a tape archive instead of joining up a number of files. The `v` option tells `tar` to be verbose, which reports all files as they are added.

To separate an archive created by `tar` into separate files, at the shell prompt, enter:

```
tar -xvf file.tar
```

Grading Criteria:

You are to work individually, and all work should be your own.

A correct implementation of the simulator is worth 70 points. Note that efficiency is not a grading criterion in this case. **Ideally, you have to make your program as robust as possible by checking the correctness of users' input. Other data items also need to be checked in the same manner. Alternatively, you are allowed to assume that users never input incorrect data. However, this impractical assumption may lead to 5% deduction on the coding part.**

A useful and descriptive README file is worth 15 points.

Readability of your source code and adhering to the coding style is worth 15 points.

There is a 15% deduction per day late. No credit is given after three days.