

COS214 Tutorial 11

Roberto Togneri, 2000

SOLUTIONS

1. (a) A computer science department has a large collection of UNIX machines on its local network. Users on any machine can issue a command of the form:
- machine4 who**
- and have the *who* command executed on *machine4*, without having the user log in on the remote machine. This feature is implemented by having the user's kernel send both the command and the *uid* to the remote machine. Currently the department's workstations are on the local network. It is proposed to allow students to also connect their PC to the network. Why is this a dangerous move?
- (b) One of the computer support people proposes to solve the security problem by restricting such commands to a group of trusted hosts. Is this solution fool-proof?

Answer

- (a) The student PC can send a message to a departmental workstation asking it to carry out some command on behalf of, say, the super-user (*uid*=0). The machine receiving the message has no way of telling if the command really did originate with the super-user, or with a student.
- (b) No! A student can set up a PC to masquerade as one of the trusted hosts! To avoid detection of multiply named hosts, the student can simply unplug the targetted workstation or force it to crash.
2. (a) After getting your degree, you apply for a job as director of a large university computer center that has just put its ancient operating system out to pasture and switched over to UNIX. You get the job. Fifteen minutes after starting work, your assistant bursts into your office screaming: "Some students have discovered the algorithm we use for encrypting passwords and posted it on the bulletin board." What should you do?
- (b) When a file is removed, its blocks are generally put back on the free list, but they are not erased. Do you think it would be a good idea to have the operating system erase each block before releasing it?

Answer

- (a) Nothing! The password encryption algorithm is public since you require a secret key to perform the decryption. Furthermore, passwords are encrypted by the login program as soon as they are typed in, and the encrypted password is compared to the entry in the password file. So what you need to know is the actual password.
- (b) From a security point of view it would be ideal since confidential information is not left lying around in the filesystem, but from a performance point of view it would generate a large number of disk writes and degrade performance. The compromise solution is for the system to only erase files which have a "security" attribute.

3. The Unix file system has *rxw* permissions for the user (i.e. owner), the group, and others. Write, in pseudo code, a function, *check()*, that is given the user id (*uid*) and group id (*gid*) of the calling process, an operation (*op* = r, w, or x), the owner and group id of the file (*fuid*, *fgid*) and the set of 9 bits for the permissions (*perm*). It returns a boolean to indicate whether the operation is legal.

Answer

```
check(int uid, int gid, char op, int fuid, int fgid, int perm)
{
    int op_perm;

    switch(op) {
        case 'r': op_perm= 4;    /* binary 100 */
                break;
        case 'w': op_perm= 2;    /* binary 010 */
                break;
        case 'x': op_perm= 1;    /* binary 001 */
                break;
        default: error( );
    }

    if (uid==fuid)
        op_perm=op_perm<<6; /* owner so check owner bits */
    else if(gid==fgid)
        op_perm=op_perm<<3; /* same group so check group bits */

    if (perm & op_perm)
        return(TRUE);
    else return(FALSE);
}
```

4. (a) Explain what the UNIX *rxw* bits mean for:
- (i) a regular file
 - (ii) a directory
- (b) Derive the UNIX file type and permissions, uid and gid:
- (i) to allow only the owner read/write access to a file?
 - (ii) to allow only a selected group of users to create/delete/list/modify files in a directory?
 - (iii) to allow only a selected group of users to list/modify files in a directory, but a single maintainer to create/delete files?
- (c) What can you say about the following UNIX file permissions:
- (i) drwx-wx-x john student
 - (ii) -rw-rw-rw root sys
 - (iii) -rwxr-x--x root wheel

Answer

- (a)(i) *r* = can read/copy the contents of a file
w = can modify the contents of a file
x = can execute the file (i.e. file contains executable object code)
- (ii) *r* = can list the contents of the directory, but with errors if the *x* is not set
w = can create/delete the contents of a directory, but only if *x* is also set
x = can access the contents of a directory and perform the *r* or *w* actions

- (b) (i) -rw----- owner group
(ii) drwxrwx--- owner group
(iii) drwxr-x--- owner group
- (c)(i) owner john can create/delete/list files in a directory, other student users can create/delete but not list the directory contents, all other users can access the contents of the directory but cannot list the contents.
(ii) all users can read/write the file.
(iii) owner root can copy/modify/execute the program file, users in group wheel can copy and execute the file, all other users can only execute the file.

5. Consider a system that supports 5000 users. Suppose that you want to allow 4990 of these users to be able to access one file. How would you specify this protection scheme in UNIX?

Answer
Put the 10 users you want to deny access to in one group (e.g. *noperm*) and set the file perm access to, say, 0404 (i.e. -r----r-- owner *noperm* file)

6. For each of the following protection problems, indicate which mechanisms (ACL, C-lists or UNIX *rx*) can be used most effectively:
- (a) Ken wants his files readable by everyone except his office mate
(b) Mitch and Steve want to share some secret files
(c) Linda wants some of her files to be public.
(d) George wants his files read/writable by a close group of friends, readable by another group of friends, and inaccessible by everybody else.

Answer

(a) ACL attributes for files → (*Ken*, rw-) (*office-mate*, ---) (*, r-)
UNIX perm → -rw----r-- *Ken* noperm files
(*office mate* belongs to noperm group)
C-list not easy without a default attribute of (*, r-)

(b) ACL attributes for secret files → (*Mitch*, rw-) (*Steve*, rw-)
UNIX perm → -rw-rw---- *Mitch* secret files
(*Mitch* owns files, *Mitch* and *Steve* belong to secret group)
C-list attributes for *Mitch* and *Steve* domains → (files, rw-)

(c) ACL attribute for public files → (*, r-)
UNIX perm → --rw-r--r-- *Linda* student files
C-list not easy without a default attribute of (*, r-)

(d) ACL attribute for files → (*close_group*, rw-) (*group*, r-) (*, ---)
C-list attribute for *close_group* domain → (files, rw-)
group domain → (files, r-)
UNIX perm does not allow this type of access.