CSCD 303 Security
Lab 2 – User Access Control

Introduction

Access Control Concepts in Unix-Like Systems
Users and Groups

Like most other operating systems, Unix-like (e.g. Solaris [1], OpenBSD [2], GNU/Linux [3] etc.) systems are designed for multiple users and groups. A group on a Unix-like system is simply a set of users. A single user may be member of multiple groups.

Read, Write and Execute Flags

Files and directories in Unix-like systems can have zero to three of the following flags:
- Read - The user or group may read the contents of a file
- Write - The user or group may change the contents
- Execute - The user or group may execute the file

Furthermore, access to files and directories is divided into three different levels:
- User (Owner)
- Group
- Everybody (others)

A file or directory always belongs to one group and one user. Unlike on Windows, files cannot be owned by multiple groups or users.

Flags in Detail

The flags are actually translated into a bitwise representation as follows:
- Bit 0 Execute flag
- Bit 1 Write flag
- Bit 2 Read flag

These bits can be interpreted as a octal number. The flags for user, group and others can be concatenated into a 3-digit representation. E.g. the number 754 means that the owner has the rights to read, write and execute, the group has permission to execute and read, and everyone else may only read the file.

Because the octal representation of access flags may be difficult to handle for many users, it has become common to use the letters r, w and x as representation for access rights.

If more than one user has to access a file, users can be assigned to groups. That means if the two users U 0 and U 1 need to write to the file example.txt, one has to create a new group for these two users and set the group permissions for the file accordingly.

Utilities
The programs chmod, chgrp and chown can be used to modify access rights on most

Unix-like systems. chmod changes the access flags of a file or directory, chgrp is used to change the group and chown changes the owner of a file.

Examples
• chown user1 file.txt - Change the ownership of file.txt to the user user1.
• chmod 600 file.txt - Make the file file.txt read- and writeable exclusively to its owner (the digitwise binary representation of 600 is 110 000 000).
• chmod 640 file.txt - Owner can read and write; members of the group can read the file file.txt and allow its owner to read and write (The digitwise binary representation of 640 is 110 100 000).
• chgrp students file.txt - Change the group of file.txt to the group students.

The setuid and setgid bits are special bits to change the user or group of an executable during runtime. When set, they allows other users to execute a program with the rights of its owner and or group.

Practice Access Control with User Permissions

1. Start Kali Linux
For this lab, we will be using the Kali Linux distribution in the Virtual Box VM.
Reboot the Lab Machine and choose Debian instead of Windows.
Log on to Debian with your EWU user id. Choose Virtual Box from the menu, start the Kali OS

2. Once Kali Linux has booted up, log in with user name: root password: root

3. Start a Terminal logged in as root.

4. You will be creating two users, Alice and Bob and setting up permissions between them. The default in many Linux systems is to allow users to at least see each others directories.

Create the Users Alice and Bob

Type the following commands at the terminal

```
# useradd -m alice             // Creates a user., alice with home directory /home/alice, group alice
# passwd:  alice                // Set password to alice

# useradd -m bob              // Creates a user., bob with home directory /home/bob, group bob
# passwd:  bob                // Set password to bob
```

Can check some things about the users you created, by using the id and group commands

```
# id alice    and   # id bob   //Brings up information about these users including their groups
# groups alice  and # groups bob  // Brings up the groups they belong to
```

5. Create Files for Alice and Bob

Become the user alice and create some files for her

```
# su – alice
  passwd: alice
```

To check that you are alice and are in alice's home directory, type
 # whoami   and   #pwd
You should be alice and be in the /home/alice directory

 Now create a file or two for alice

 # echo “This is a short file” > alicefile1   // creates a file called, alicefile1
 # echo “Another short file” > alicefile2   // creates a file called, alicefile2

 Check the default permissions on these files, and the home directory fo alice
 # ls -l   and   # ls -ld /home/alice

 Now, become user, bob and repeat creating some tiles for bob
 First, become root again, type: #exit   // Exits the user shell for Alice

 # su – bob
   passwd: bob

To check that you are bob and are in bob's home directory, type
 # whoami   and   #pwd
You should be bob and be in the /home/bob directory

 Now create a file or two for bob

 # echo “This is a short file” > bobfile1   // creates a file called, bobfile1
 # echo “Another short file” > bobfile2   // creates a file called, bobfile2

 Check the default permissions on these files, and the home directory fo bob
 # ls -l   and   # ls -ld /home/bob

 Question 1: With these default permissions, can alice and bob read each others files? Yes or No.

 As bob, try to read the alice home directory,
 # ls -l /home/alice   Question 2:   Can you list the alice directory?
 Now list a file in alice's directory
 # cat /home/alice/alicefile1
 Question 3:   Did that work to display the file?
 Now, create a file as bob and try to write it to alice's directory
 # echo “This is a test of a short file” > /home/alice/bobsfile   // creates a file called, bobsfile
 Question 4:   Did that work? Why or why not?

Exit from bob and become root, then become alice.

```
#exit
#su – alice
passwd: alice
```

Do the same thing and see if alice can read bob's files.

```
# ls -l   /home/bob
```

Question 5: Can you list the bob directory?

Now list a file in bob's directory

```
# cat /home/bob/bobfile1
```

Question 6: Did that work to display the file?

Now, create a file as alice and try to write it to bob's directory

```
# echo "This is a test of a short file" > /home/bob/alicefile   // creates a file called, alicefile
```

Question 7: Did that work?

What would bob need to do to for alice to create a file in /home/bob?

6. Preserving Privacy

Say, bob and alice no longer want to allow each other to read files in their directories We can use the chmod command to alter these directory permissions or you can use chmod to alter directory permissions. Lets alter directory permissions.

As alice from the last exercise,

```
# chmod  go-rwx  /home/alice
# ls -ld    //Should show that only alice has permission to access her directory
```

Exit Alice, # exit,   Become bob, # su – bob   //Comment as root, no password needed

```
# chmod go-rwx /home/bob
# ls -l   // shows user only access to bob's directory
```

Now, try to list alice's home directory while you are still bob

```
# ls -l /home/alice
```

Question 8: Were you able to read the directory? Why or why not?

Now, become alice again and do the same thing for bob.

```
# su – alice   # passwd: alice
#ls -l /home/bob
```

Question 9: Can you read bob's directory? Why or why not?

Question 10: Instead of locking out access for the whole directory, what could you do if you wanted to prohibit access to just one file? How could you do it

7. Groups

Say, alice and bob now want to belong to a group that allows them to share files as members of a group. We will create a new group, add alice and bob to this group and then create a place where they can share files with each other, but only as members of the group.

Type:

```
# exit    // To become root again
```

Create a new group, geeks.

```
# groupadd geeks
```

To make sure the group was created, you can view the /etc/group file.

```
# tail /etc/group   // Newest group, geeks will be at the end of the group file
```

Add alice and bob to this geek group.

```
#sudo usermod -a -G geeks alice  // adds the group geeks to alices other groups
#sudo usermod -a -G geeks bob   // adds geeks to bob and he keeps his other groups
```

Check the new group has been added to each userl by typing,

```
# groups alice  //Shows all groups for the user
# groups bob
```

To allow them to share files and collaborate, lets create a directory in alice's home directory
Name the directory, business.

Alice will create this directory.

```
# su – alice   // Become alice
```

Now, create the directory and set the group ownership to the group, geeks

```
# mkdir business
# chgrp geeks business
```

Allow groups to write to the business directory

```
#chmod g+w business
```

Remove other access

```
# chmod o-rwx business          # ls -l
```

Recall, Alice changed her /home/alice directory to prevent groups and others from access.
She will need to change it back so bob can access the business directory

```
#chmod 755 /home/alice
# ls -ld /home/alice     // to check the permissions and ownership
```

Now try to create a file in the business directory in alices home directory as bob

```
#exit   // change back to root

#su – bob
# echo “Writing a file owned by bob” > /home/alice/business/bobsfile
```

Question 11: Can bob write to this directory since he is in the geeks group?
Question12: How could you test that others can't access or write to the business directory?

End.

Turn in Directions

1. Put CSCD 303 – Lab 2 in the subject line.
2. Type the questions into a document or directly in an email and email it to me.