Graduate Program in Information Science and Telecommunications and Networking
School of Information Sciences
University of Pittsburgh

TEL2821/IS2150: INTRODUCTION TO SECURITY
Lab: Operating Systems and Access Control

Version 1.0, Last Edited 09/20/2005

Name of Students: ______________________________

____

Date of Experiment: ______________________________

Graduate Program in Information Science and Telecommunications and Networking
School of Information Sciences
University of Pittsburgh

Part I: Objective

The objective of the exercises presented here is to familiarize the students with the access control features available in the Microsoft Windows, UNIX-based and Solaris systems, and to induce the student to analyze the similarities and differences in the access control in different operating systems.

Part II: Equipment/Software

- A PC with Microsoft Windows 2000 installed on it.
- A PC with Linux installed on it.
- A PC with Solaris 8 (or above) installed on it.

1. Microsoft Windows

Access control refers to the ability of a user to access a particular object and possibly modify it. In terms of operating systems, access control refers to the ability of a user to read, write or execute a certain file or folder. In this laboratory, we shall study the access control framework for Microsoft Windows and UNIX-based platforms, by taking Microsoft Windows 2000 and Linux as respective examples.

The Microsoft Windows 2000/XP/2003 series of OSs introduced access control for files, directories and devices. Before we introduce access control for these operating systems, let us take a look at how objects are arranged in these systems.

The Active Directory service was introduced in the NT family of operating systems as a means of arranging all users, devices and objects at a centralized location and allowing these networked entities to find each other through this service. Entities are known as objects and they are arranged into a hierarchical structure known as the logical structure by the administrators. A collection of objects that share the same security policies is known as a domain (a container object) and multiple domains can be arranged hierarchically into a tree. A forest is a complete instance of the Active Directory that consists of a set of domains that trust each other through a two-way transitive trust. This arrangement of objects into logical structures enables easy management of the objects and allows for more flexible access control. The place Active Directory has in the network is shown in Figure 1.

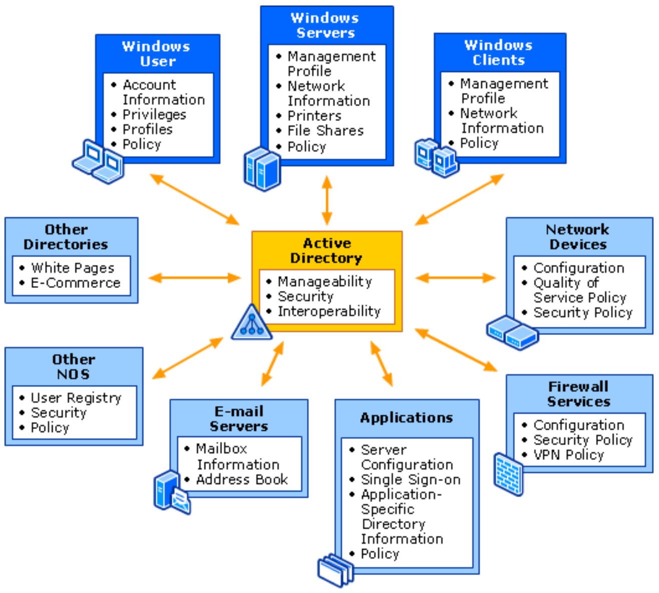

Figure 1: Active Directory

Every entity - users, groups, domains, processes - has a Security Identifier (*SID*) uniquely associated with it. The SID is very similar to the UID in UNIX. Objects that have some operations associated with them and to which access must be controlled are called *securable objects*. Securable objects have *security descriptors* associated with them that consist of DACLs that describe which users or groups have what access rights over them, SACLs that describe how auditing is done and the SID of the owner of the object. Every time an object is created, a security descriptor can be assigned to it, but if it is not assigned, it will inherit it from its parent object. A *security context* is associated with every process (or user) which describes which groups it belongs to, what privileges it has and what accounts are associated with it. The security context is maintained in an *access token*. ACLs for an object contain the SID of the intended *trustee* and an access mask for the various access rights. When access is requested, the access token of the accessing object, is checked with the security descriptor of the accessed object, to see if the access should be permitted or not. An example is given in figure 2.

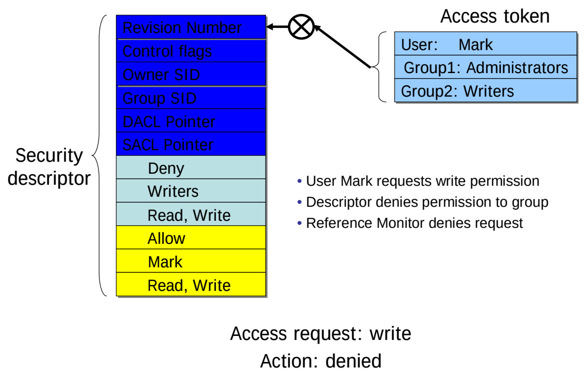

Figure 2: Example of Access Request

For added security, to protect sensitive data, the Encrypted File System (EFS) was introduced in the NT families. The EFS allows users to encrypt objects created by them so that no other object can access them. The encryption is done using an EFS certificate that the user gets and multiple users can be added to allow access, with the help of their EFS certificates.

Lab Procedures

Exercise 1.1: Adding users to the system

1. Go to **Start** -> **Control Panel** -> **Users and Passwords**.
2. If a user `telcom2810` already exists, remove it.
3. Click on **Add** to add a new user. Enter the username as `telcom2810` and password as `introtosecurity`. Select **Restricted User** as the group for its group membership.
4. Click **OK**.

Exercise 1.2: Studying the effects of using the **Read-Only** and **Hidden** attributes of a file.

1. Create a TXT document in your *My Documents* folder.
2. Right-click on the icon, and select *Properties*.
3. Check the **Read-Only** checkbox and click on **OK**.
4. Then open the document, add some text and try to save the changes.
    a. What happens? Explain why it happens.

____
____
____
____
____
____

5. Copy the file and place it in the *C:/Documents and Settings/All Users/Documents* folder.
6. Then logoff and logon as `telcom2810`.
7. Open the *C:/Documents and Settings/All Users/Documents* folder and try modifying the contents of the file. Are the results the same as in Step 4?
8. Repeat Step 2 and uncheck the **Read-Only** box.
   a. What happens? Explain why it happens.

____
____
____
____
____
____

9. Logoff and log back as `Administrator`.
10. Right-click on the icon and select *Properties*.
11. Check the **Hidden** checkbox and click on Ok.
    a. Do you see the icon now? ___________
12. Go to the *Tools* menu and select *Folder Options*.
13. Under the *View* tab, check the radio button that says "**Show Hidden Files and Folders**". Click **OK**.
    a. What do you see? ____

Exercise 1.3: Demonstrate the use of encryption of files.

1. Log in as `Administrator`.
2. In the *C:/Documents and Settings/All Users/Documents*, create a TXT document.
3. Right-click the document icon and select *Properties*. Click on the *Advanced* button next to the **Hidden** checkbox.
4. Check the checkbox that causes encryption of the file's contents and select it only for that file.
5. Then log off and log on as `telcom2810`.
6. Go to *C:/Documents and Settings/All Users/Documents* and try to access the previously created document.
7. What is the result?

____
____
____

Exercise 1.4: To explicitly assign permissions to different users for a given file.

1. Log in as `Administrator`.
2. In the *C:/Documents and Settings/All Users/Documents*, create a TXT document.
3. Right-click the document icon and select *Properties*. Under the *Security* tab, click **Add** and select `telecom2810` and click **Add**.
4. What are the default permissions given to this user? Go back and repeat the process, this time adding the group `Users`. Are the default permissions any different?

   ____
   ____
   ____
   ____

5. Click the *Advanced* button and under the *Permissions* tab, select *telcom2810* and click *View/Edit*.
6. Notice the number of permissions that can be added for this user. Also notice the option of explicitly allowing and denying permissions.
7. Check the **Create Files/Write Data** and **Create Folders/Append Data** checkboxes, under the *Allow* column.
8. Check the **Write Attributes** checkbox under the *Deny* column. After clicking Ok once and again at the *Advanced* window, what do you see? What does that mean?

Questions

1. Repeat the steps in Exercise 1.4, but create the file in the directory *C:/Documents and Settings/All Users/Documents*. Then logoff and logon as `telcom2810`. Is it possible to view the hidden file?

2. UNIX

File Hierarchy

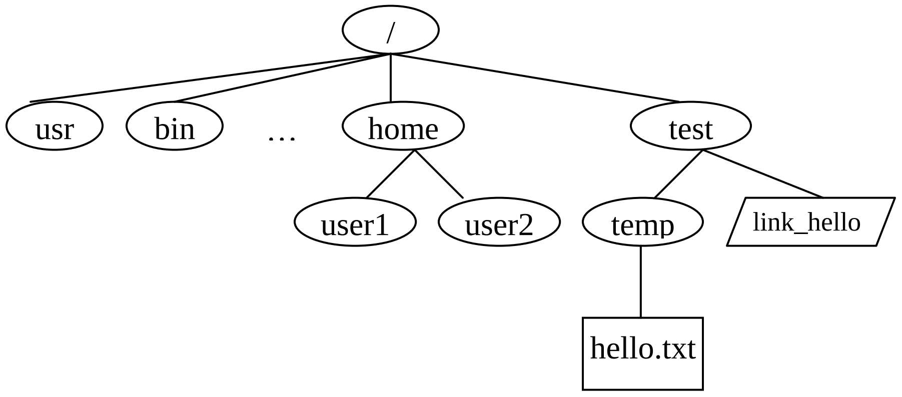

Figure 3: Example directory hierarchy

The Unix file system is organized as a hierarchy with the root (/) directory at the highest level. Each directory may contain sub-directories and files. Typically, some of the directories that may occur under the root are *usr*, *bin*, *sbin*, *home*, *var*, *boot*, *dev*, etc. In Figure 3, *user1* and *user2* are sub-directories under home. *hello.txt* is a plain-text file and *link_hello* is a *link*ing file that points to *hello.txt*. In order to access the file

/test/temp/hello.txt, the system begins its search from the root(/) folder and then to *test* and *temp* folders consecutively and then finally the file *hello.txt*.

Ownership and Permissions

Ownership of files in UNIX can be viewed in one of three ways: *owner* (creator), *group* or *others*. Using this simple notion of ownership access to files can be controlled by associating unique user ID (UID) and group ID (GID) with twelve permission bits for each file as shown below.

Permission Bits											
Extra			owner			group			others		
su	*sg*	*t*	*r*	*w*	*x*	*r*	*w*	*x*	*r*	*w*	*x*

Typically these bits are divided into three sets of three bits and three extra bits as shown in table below. *r*, *w* and *x* bits stand for read, write and execute bits for each of the owner, group and others permissions. *su*, *sg* and *t* stand for set_user_id, set_group_id and sticky bits. These 4 sets of bits are often represented in their octal digits. For example, “100 111 101 101” is represented as “4755.” When the *su* bit is set, whosoever executes the file, the UID of the process will be the owner of the file. Similarly, if *sg* is set, the GID of the process will be the owner of the file.

Exercise 2.1: Setting up File Structure and User space

The objective of this exercise is to setup the file hierarchy structure and the users that are required for the exercises in this section. The **su** command is used to switch users.

1. Login as `root` (password = "`enter 2005`")
2. Use **useradd** command to create two new users ***user1*** and ***user2*** as follows:
   a. `useradd user1 -g users -p user1`
   b. `useradd user2 -g users -p user2`

   What do `–g` and `–p` options mean?

   ____
   ____
   ____

3. Check user information with the id command. Note the uid, gid for each output.
```
a. id user1
b. id user2
d. id
```
4. Create directory structure
```
a. mkdir /test
b. mkdir /test/temp
```
5. Switch user roles as ***user1*** and then back to root using the su command
```
a. whoami
b. su user1
```
   c. `su` **OR** `su root` (password = `"enter 2005"`)
5. Create a new file as `root` user and change group ownership as well as user ownership of the file.

a. `touch /home/user2/HelloWorld`
b. `ls –l /home/user2/HelloWorld` (observe owner and group)
c. `chgrp users /home/user2/helloWorld`
d. `chown user2:users /home/users/HelloWorld`
e. `ls –l /home/user2/HelloWorld` (observe owner and group)

Explain what `chgrp` and `chown` do?

____
____
____

<u>Exercise 2.2</u>: Differences in File and Folder Permissions

The objective of the following exercises would be to see the differences in file and folder permissions. The **chmod** command will be used to change file and directory permission to demonstrate the slight differences in permissions for files and directories.

1. Observe the result of **ls** and **cd** commands
   a. `cd /`
   b. `ls –l`
   c. `ls -al /home`
   d. What are the directory permissions for ***user1***, ***user2*** and ***test*** directories?

   ____
   ____
   ____

   e. Switch to ***user1*** using `su user1`
   f. `ls -al /home/user2` (Can you list directory?________)
   g. `cd /home/user2` (Can you change directory?________)

2. Change directory permissions of ***user2*** directory and try again as ***user1***.
   a. `su root`
   b. `chmod 740 /home/user2`
   c. Repeat steps *1e* to *1g* (Can you list or change directory?________)
   d. `su root`
   e. `chmod 750 /home/user2`
   f. Repeat steps *1e* to *1g* (Can you list or change directory?________)
   g. `touch  /home/user2/hello12.txt`
   (Can you create new file?________)
   h. `su root`
   i. `chmod  770  /home/user2`
   j. `su user1`
   k. Repeat step *2g*. (Can you create new file? ________)
   l. `ls –l /home/user2`

Alternative syntax for chmod command

The access permissions for the file *hello.txt* is to set the *su* bit only, allow all access permissions to owner, read and execute rights to the group and only read rights to others. In other works the 12 bit permission required on the file *hello.txt* is as follows: "100 111 101 100." This can be achieved in several ways using **chmod** command:

1. `chmod 4754 hello.txt`
2. `chmod u+srwx g+rx o+r hello.txt`
3. `chmod u=srwx, g=rx, o=r hello.txt`

(you are expected to learn both the ways to use `chmod`)

Exercise 2.3: New text files and linking files

Unix supports two kinds of link files--a *hard link* and a *symbolic link*. A *hard* link is a file with the actual address space of some ordinary file's data blocks. A *symbolic* link is just a reference to another file. It contains the pathname to some other file.

1. In the /test/temp/ directory, as root user, create a new text file ("hello") and fill it with some text using `touch`, `pico`, `vi` etc.
2. Create a link ***link_hello*** in the ***test*** folder pointing to ***hello.txt*** in the ***temp*** folder (refer to file structure in introduction)
   a. `cd /`
   b. `ln -s /test/temp/hello /test/link_hello`
   c. Is there any difference in file permissions of `link_hello` and `hello`?

   ____
   ____
   ____

   d. `cat /test/link_hello` What is the output?

   ____
   ____
   ____

Exercise 2.4: Default file permissions and Group access control

Whenever a new file is created using C program,defulat permissions can be assigned to it. UNIX system allows the user to filter out unwanted permissions by default. This default setting can be set by the user using the **umask** command. It is a system call that is also recognized by the shell. The command takes the permissions set during file creation and performs a bitwise AND to the bitwise negation of mask value. Some common **umask** values are 077 (only user has permissions), 022 (only owner can write), 002 (only owner and group members can write), etc.

1. In a terminal window, make sure you are a root user. If not the root user, then switch back to root user (use your password to switch).
2. Use umask command to check the current mask permission and assign a new mask.
   a. `umask`
   b. What is the current mask? How is it interpreted? (try `umask –S` or the man pages)

      ____

   c. `cd /test`
   d. `touch testmask1`
   e. `ls al`
   f. What are the permissions of the file *testmask1*

      ____

   g. `umask 0077`
   h. `touch testmask2`
   i. Now what are the permissions of the file *testmask2*

      ____

3. What is the effect of setting mask value to 0000?

   ____
   ____
   ____

<u>Exercise 5</u>: *setuid* bit, *setgid* bit and *sticky* bit

As explained in the background above, the highest three bits of the permission bits represent the *setuid* bit, *setgid* bit and the *sticky* bit. If the *setuid* bit is set then the uid will always be set to the owner of the file during execution. If the *setuid* bit is not set then the uid will be the user who executes the process. Similarly, if the *setgid* bit is set then the gid will be set to the group that owns the file during execution. If the *setgid* bit is not set then the gid will be the group that executes the process. The sticky bit is set to keep processes in the main memory.

In the following exercise, the objective is to demonstrate how processes are affected when the *setuid* bit is set. The exercise must be begun with root privileges.

a. `which touch`
b. `ls –l /bin/touch`
c. `chmod 4755 /bin/touch`
d. `ls –l /bin/touch`
e. `ls –l /home/user2`
f. `chmod 700 /home/user2/HelloWorld`
g. `ls –l /home/user2` (observe timestamp and permissions)
h. `su user1`
i. `touch /home/user2/HelloWorld`

j. `ls –l /home/user2` (observe timestamp)
k. `su root`
l. `chmod 0755 /bin/touch`
m. `su user1`
n. `touch /home/user2/HelloWorld`

Why permission is it denied?

____
____
____

RESTORE SYSTEM

After the series of exercises, it is most essential that the system is restored to its normal state so that other students may undertake the exercises again. Below are the series of commands that are expected to restore the system to its original form.

- `su root`
- `umask 0022`
- `chmod 0755 /bin/touch`
- `userdel user1`
- `userdel user2`
- `rm –rf /home/user1`
- `rm –rf /home/user2`
- `rm –rf /test`
- `rm –rf /home/test/`

3. SOLARIS

Sun's Solaris operating system is in essence a UNIX-based OS, especially since a large part of early UNIX-based OSs were developed by Sun Systems. The file hierarchy is the same as in UNIX systems, with the exception that user accounts created are given a home directory with the path as */export/home/*`username`. All *users* are given UIDs when their accounts are created. Also, users can be grouped in different *groups* to assign similar permissions to each. The new features in Solaris systems include use of user templates, projects and the prominent RBAC access control. We shall discuss these in more detail.

User templates can be created in order to set up a template with certain fixed attributes for users. Then users that have similar properties like students, engineers etc can be created using user templates. Users can be assigned to certain *projects* which associates them with a certain workload component. This is very useful when it comes to resource allocation.

Role-based Access control was introduced in Solaris systems from Solaris 8 onwards. It was introduced with the intention of administrative purpose, which is why the roles are called administrative roles. The idea although similar to the *superuser* or *root*, was developed to overcome the shortcomings of having these accounts. The problem with having *superuser* or *root* accounts was that anyone who would be able to hijack these accounts, would be all-powerful in the system. But with the use of RBAC in the Solaris system, the *Principle of Least Privilege* could be enforced. By this, users could be given some administrative rights, but only to the extent that was required.

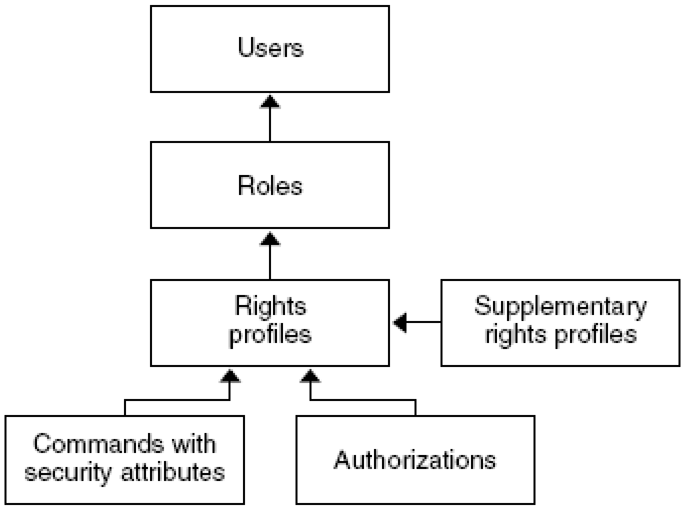
Users
Roles
Rights profiles
Supplementary rights profiles
Commands with security attributes
Authorizations

Figure 4: Role-based feature

In the Solaris system, rights are grouped together in to what are called *right profiles*. These right profiles are then assigned to roles, which users can assume in order to carry out some administrative operations. No roles are created by default, but 3 recommended roles are:

- Primary Administrator – All powerful and similar to the root or superuser
- System Administrator – Administrative privileges without security rights
- Junior Administrator – Administration over some operations like backup, printing etc.

It is to be noted, that roles are totally a function of the organizations needs.
The RBAC model introduces some interesting concepts in Solaris. An *authorization* is a permission to perform a certain class of actions. A *privilege* is a discrete right that can be assigned to a user, system or object. A *security attribute* is an attribute that allows a certain process to successfully carry out an operation. *Privileged applications* can use security attributes to override system controls and perform operations, like `setuid` and `setgid` in UNIX systems. A rights profile is a collection of administrative capabilities, and can consist of authorizations, security attributes and even other rights profiles. Finally a role is defined as a special identity for running privileged applications. Roles run privileged applications from a separate shell called *profile shell* which is a different shell that can recognize security attributes. All of these entities work together as shown in Figure 4.

File Access in Solaris can be controlled through regular UNIX commands as mentioned in the previous section. But a more granular way of controlling access to files is with the help of ACLs. Unlike in UNIX, where all users, groups and others are assigned permissions to a given file, ACLs allow control of access by specific users or groups. This allows finer access control. Access to devices is controlled with the help of *device*

policies and *device allocation*, which are enforced at the kernel and user allocation time respectfully.
In the following lab exercises, some of the tasks are to be performed on the Sun Management Console while the rest are to be performed from command line, through a terminal window. The Sun Management Console (SMC) can be executed at a terminal window as

```
$ /usr/sbin/smc &
```

Lab Procedures

Login: `stadmin`
Password: `intro2810`

<u>Exercise 3.1:</u> Creating User accounts and assuming roles.

1. Start the SMC and click on This Computer icon on the left panel.
2. Then double click on System Configuration and double click Users.
3. Enter the username and password used to login.
4. The next screen asks you to assume the role `studentadmin` if you wish to continue. You will learn to assign roles in the next exercise. Are there any other roles that you can assume? ___________________
5. Enter the password as `intro2150.`
6. Then double click on **User Accounts**. Once the user accounts come up, go to **Action** on the menu and select **Add User**->**from Wizard**.
7. Follow the procedure, creating a user with username `telcom2810` and password `sec2810`. Select the group as `students`.

<u>Exercise 3.2</u>: Creating roles and assigning them to user accounts.

1. Under Users in the SMC, select Administrative Roles and go to **Action->Add Role**
2. Create a role with the name `newadmin` and password `123abc`. Add the Basic Solaris User rights for the role. Add the user `telcom2810` as a user who can assume this role
3. Logout and login as `telcom2810`. Then launch SMC. Does it allow you to login? ___________
4. Try to create a new user account. What do you see?
   ____
   ____
5. Logout and login as `stadmin`.

<u>Exercise 3.3</u>: Working with file permissions and ACLs

1. Open a terminal and type in the command
   ```
   $ cd /public
   $ vi testing.txt
   ```
   to launch the *vi* editor, and enter some text into the file. To enter text, hit '***Insert***' to go into insert mode and enter text. Then press '**Esc**'. Save the file and exit by typing '***:wq***'.

2. Then logout and login as `telcom2810` and open a terminal.
3. Enter the command
   ```
   $ vi /public/testing.txt
   ```
4. Enter some text and try to save as before.
   What do you see? ____
5. Then logout and login as `stadmin` again and open a terminal.
6. Enter the command to set an ACL to allow `telcom2810` to write to the file. The command for this is as follows:
   ```
   $setfacl –s user:r-x group;r-x other r-x user:telcom2810:rw-
   ```
7. Repeat steps 2, 3 and 4. What do you see?
   ____
   ____
   ____

Questions

1. How different is the access control architecture from that of UNIX-based platforms?
2. How different is access control management in Windows, compared to UNIX? Which would you say is easier? Which is more efficient?