Access Control
Best way to defend any attack?
Best way to defend any attack?

- Prevent the attack
  - We can prevent the attacks on confidentiality, integrity and anonymity by
Best way to defend any attack?

- Prevent the attack
  
  • We can prevent the attacks on confidentiality, integrity and anonymity by
    
    - Restricting the access to the information to those who have a need to access and/or modify the information
Best way to defend any attack?

- Prevent the attack
  - We can prevent the attacks on confidentiality, integrity, and anonymity by
    - Restricting the access to the information to those who have a need to access and/or modify the information
  - Various tools:
    - Access control matrix
    - Access control lists
    - Role based access
Access Control Matrix

- A table that defines permissions
  - Row – associated with a subject (user, group)
  - Column – associated with an object (file, directory, document, device etc.)
  - Cell – defines access rights for a combination of subject/object

- Access Rights
  - Reading, writing, copying, executing, deleting
Example:

<table>
<thead>
<tr>
<th></th>
<th>/etc/passwd</th>
<th>/usr/bin/</th>
<th>/u/user3/</th>
<th>/admin/</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 2</td>
<td>R</td>
<td>R, X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User 3</td>
<td>R</td>
<td>R, X</td>
<td>R, W, X</td>
<td></td>
</tr>
<tr>
<td>User 4</td>
<td>R</td>
<td>R, X</td>
<td>R, X</td>
<td>R, X</td>
</tr>
</tbody>
</table>

R – Read
W – Write
X – Execute
Blank cell – No Access Rights
Access Control Matrix

Advantages

• Allows access for fast and easy determination of the access control rights for any subject-object pair
• Gives administrators a simple, visual way of seeing entire set of access control relationships at once
• Degree of control as specific as granularity subject-object pairs

Disadvantages

• Lack of scalability
  – Can get really big
  – $n \times m$ cells for $n$ subjects and $m$ objects
  – For 1000 subjects and 1,000,000 objects – 1 billion cells !!!!
Access Control Lists

- Takes an object-oriented approach
- Defines for each object $o$, a list $L$ – access control list (ACL)
- ACL enumerates:
  - all the subjects having access to object $o$
  - and, for each subject $s$, gives access rights for an object $o$
- takes each column of access control matrix and compresses it into a list (ignores empty cells)
Access Control Lists

- **/etc/password**
  - Root: R, W
  - User1: R
  - User 2: R
  - User 3: R

- **/usr/bin/**
  - Root: R, W, X
  - User1: R, X
  - User 2: R, X
  - User 3: R, X

- **/u/user3/**
  - Root: R, W, X
  - User 2: R, W, X
  - User 3: R, X

- **/admin/**
  - Root: R, W, X
  - User 3: R, X
Access Control Lists

Advantages

- Total size is much smaller than total number of cells in ACM
- ACL for an object can be stored with the same as it's metadata
- Easy to search the access rights object wise

Disadvantages

- Doesn't provide an efficient way to enumerate all the access rights of a given subject
- To find out the access rights of a subject, the ACLs of all the objects need to be looked for
  - May be needed when removing the subject
Capabilities

- Takes a subject-oriented approach
- Defines for each object $s$,
  - a list of objects for which $s$ has non-empty access rights
  - and, the corresponding access rights for each object
- Takes each row of access control matrix and compresses it into a list (ignoring empty cells)
Capabilities

Root

/etc/password: R, W, X; /usr/bin/: R, W, X;
/u/user3/: R, W, X; /admin/: R, W, X

User 1

/etc/password: R; /usr/bin/: R, X

User 2

/etc/password: R; /usr/bin/: R, X;
/u/user3/: R, W, X

User 3

/etc/password: R, W, X
/admin/: R, X
Capabilities

Advantages

• Total size is much smaller than total number of cells in ACM (similar to ACLs)
• ACL for a subject can be stored with the same as it's metadata
• Easy to search and update the access rights subject wise

Disadvantages

• Doesn't provide an efficient way to enumerate all the access rights of a given object
• To find out the access rights of an object, the capability lists of all the subjects need to be looked for
Role-Based Access Control

- Independent of the specific data structures that represents access control rights
- Can be used with ACM, ACL or Capabilities
- Administrators define roles (e.g. faculty, student, staff)
  - Access control rights are specified for roles, not directly for subjects
  - Rights appropriate for the class of users associated with that role
- Role-object pairs are assigned access rights
- Subjects are assigned to various roles
- One subject may have multiple roles (e.g. phd student + tutor)
Role-Based Access Control
Role-Based Access Control

- Role Hierarchies
  - A hierarchy can be defined over roles
  - Simplifies the definition and management of permissions due to inheritance property
    * Access rights propagate up the hierarchy
    * Role R1 > Role R2 then R1 has access rights of R2 as well
Access Control in Ubuntu

- File Permissions:
  - Everything is a file: Directories are files, files are files and devices are files
  - All of the files on a system have permissions that allow or prevent others from viewing, modifying or executing
  - If the file is of type Directory then it restricts different actions than files and devices
  - The super user "root" has the ability to access any file on the system
  - Each file has access restrictions with permissions, user restrictions with owner/group association.
  - Permissions are referred to as bits
Access Control in Ubuntu

There are three types of access restrictions:

<table>
<thead>
<tr>
<th>Permission</th>
<th>Action</th>
<th>chmod option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>(view)</td>
<td>r or 4</td>
</tr>
<tr>
<td>Write</td>
<td>(edit)</td>
<td>w or 2</td>
</tr>
<tr>
<td>Execute</td>
<td>(execute)</td>
<td>x or 1</td>
</tr>
</tbody>
</table>

There are also three types of user restrictions:

<table>
<thead>
<tr>
<th>User</th>
<th>Is output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Owner</td>
<td>-rwx------</td>
</tr>
<tr>
<td>Group</td>
<td>----rwx---</td>
</tr>
<tr>
<td>Other</td>
<td>------rwx</td>
</tr>
</tbody>
</table>

- The restriction type scope is not inheritable: the file owner will be unaffected by restrictions set for his group or everybody else
Access Control in Ubuntu

- Folder/Directory Permissions

- Directories have directory permissions. The directory permissions restrict different actions than with files or device nodes.

<table>
<thead>
<tr>
<th>Permission</th>
<th>Action</th>
<th>chmod option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>(view contents, i.e. ls command)</td>
<td>r or 4</td>
</tr>
<tr>
<td>Write</td>
<td>(create or remove files from dir)</td>
<td>w or 2</td>
</tr>
<tr>
<td>Execute</td>
<td>(changing - cd into directory)</td>
<td>x or 1</td>
</tr>
</tbody>
</table>

- write access for a directory allows deleting of files in the directory even if the user does not have write permissions for the file!!!!!!

- Folders (directories) must have 'execute' permissions set (x or 1)
  - folders (directories) will NOT FUNCTION as folders (directories) and WILL DISAPPEAR from view in the file browser.
Access Control in Ubuntu

Permissions in Action:

- Execute command: $ ls -l /etc/hosts

- And, I get the result:

  
  **rw-r--r-- 1 root root 220 Feb 8 2017 /etc/hosts**

  (permissions, number of links, owner, group, file size, date, file name)

- This means the file "/etc/hosts" is owned by the user root and belongs to the root group.

- What are the permissions from the above /etc/hosts? ls output?

  
  **-rw-r--r--**

  owner = Read & Write (rw-)
  group = Read (r--)
  other = Read (r--)

SA
Access Control in Ubuntu

- Changing Permissions
  - two ways to modify permissions:
    - with numbers or with letters.
  - Using letters is easier to understand for most people.
  - Command: chmod {options} filename
  - When modifying permissions be careful not to create security problems.
    - Some files are configured to have very restrictive permissions to prevent unauthorized access.
    - For example, the /etc/shadow file (file that stores all local user passwords) does not have permissions for regular users to read or otherwise access.
Access Control in Ubuntu

- For example: $ ls -l /etc/shadow
  -rw-r----- 1 root shadow 1195 Jul 19 17:01 /etc/shadow
- Permissions:
  - owner = Read & Write (rw-)
  - group = Read (r--)
  - other = None (---)
- Ownership:
  - owner = root
  - group = shadow
### Access Control in Ubuntu

#### chmod with Letters

<table>
<thead>
<tr>
<th>Options</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td>owner</td>
</tr>
<tr>
<td>g</td>
<td>group</td>
</tr>
<tr>
<td>o</td>
<td>other</td>
</tr>
<tr>
<td>a</td>
<td>all (same as ugo)</td>
</tr>
<tr>
<td>x</td>
<td>execute</td>
</tr>
<tr>
<td>w</td>
<td>write</td>
</tr>
<tr>
<td>r</td>
<td>read</td>
</tr>
<tr>
<td>+</td>
<td>add permission</td>
</tr>
<tr>
<td>-</td>
<td>remove permission</td>
</tr>
<tr>
<td>=</td>
<td>set permission</td>
</tr>
</tbody>
</table>
Access Control in Ubuntu

First create some empty files:
```bash
$ touch file1 file2 file3 file4
$ ls -l file*
```
total 0
```bash
-rw-rw-r-- 1 sarita sarita 0 Aug 11 09:57 file1
-rw-rw-r-- 1 sarita sarita 0 Aug 11 09:57 file2
-rw-rw-r-- 1 sarita sarita 0 Aug 11 09:57 file3
-rw-rw-r-- 1 sarita sarita 0 Aug 11 09:57 file4
```

1. Add owner execute bit:
```bash
$ chmod u+x file1
$ ls -l file1
```
```bash
-rwxrw-r-- 1 sarita sarita 0 Aug 11 09:57 file1
```
Access Control in Ubuntu

➢ Add other write & execute bit:
$ chmod o+wx file2
$ ls -l file2
-rw-rw-rwx 1 sarita sarita 0 Aug 11 09:57 file2

➢ Remove group read bit:
$ chmod g-r file3
$ ls -l file3
-rw--w-r-- 1 sarita sarita 0 Aug 11 09:57 file3

➢ Add read, write and execute to everyone:
$ chmod ugo+rwx file4
$ ls -l file4
-rwxrwxrwx 1 sarita sarita 0 Aug 11 09:57 file4
Access Control in Ubuntu

- chmod with Numbers

<table>
<thead>
<tr>
<th>Options</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>#--</td>
<td>owner</td>
</tr>
<tr>
<td>-#-</td>
<td>group</td>
</tr>
<tr>
<td>--#</td>
<td>other</td>
</tr>
<tr>
<td>1</td>
<td>execute</td>
</tr>
<tr>
<td>2</td>
<td>write</td>
</tr>
<tr>
<td>4</td>
<td>read</td>
</tr>
</tbody>
</table>
Access Control in Ubuntu

- chmod with Numbers
  - Owner, Group and Other is represented by three numbers.
  - To get the value for the options determine the type of access needed for the file then add.
  - For example if you want a file that has -rwxrw-r-- permissions you will use the following:

$ chmod 764 filename

<table>
<thead>
<tr>
<th>Owner</th>
<th>Group</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>read, write and execute</td>
<td>read &amp; write</td>
<td>read</td>
</tr>
<tr>
<td>4 + 2 + 1 = 7</td>
<td>4 + 2 = 6</td>
<td>4</td>
</tr>
</tbody>
</table>

- Another example: for file with permissions: --w-r-x--x

$ chmod 251 filename

<table>
<thead>
<tr>
<th>Owner</th>
<th>Group</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>write</td>
<td>read &amp; execute</td>
<td>execute</td>
</tr>
<tr>
<td>2</td>
<td>4 + 1 = 5</td>
<td>1</td>
</tr>
</tbody>
</table>
Access Control in Ubuntu

Create four files file5, file6, file7, file8
Suppose they are:
- `-rw-rw-r-- 1 <owner> <group> 0 Aug 11 10:57 file5`
- `-rw-rw-r-- 1 <owner> <group> 0 Aug 11 10:57 file6`
- `-rw-rw-r-- 1 <owner> <group> 0 Aug 11 10:57 file7`
- `-rw-rw-r-- 1 <owner> <group> 0 Aug 11 10:57 file8`

What are the current permissions in terms of numbers?

Try changing the permissions using numbers
- Add owner execute bit for file5
- Add other write & execute bit for file6
- Remove group read bit for file7
- Add read, write and execute to everyone for file8
Access Control in Ubuntu

- What if you do not have the permission to change the permissions?
  
  For example:
  
  `-rwxr-xr-x` 1 root root 5049 Jan 10 2014 znew
  <username>@<hostname>:/bin$ chmod 777 znew
  chmod: changing permissions of ‘znew’: Operation not permitted

- You can change the permissions of such files, if you have root access.

- We use “sudo” to run a command with root access.
  
  <username>@<hostname>:/bin$ sudo chmod 777 znew
  [sudo] password for <username>:
  sarita@PC201:/bin$ ls -l znew
  `-rwxrwxrwx` 1 root root 5049 Jan 10 2014 znew

  !!!! You can revert the permissions back to original after trying, so that you do not mess up with your system accidentally.
Access Control in Ubuntu

- Changing the File Owner and Group:
  - A file's owner can be changed using the `chown` command.
    
    ```
    $ sudo chown <new owner name> <filename>
    ```
    e.g. $ sudo chown tux foobar

  - To change a file's group, either `chgrp` or `chown with special syntax` can be used.
    
    ```
    $ sudo chgrp <new group name> <filename>
    ```
    or
    ```
    $ sudo chown :<new group name> <filename>
    ```

  - To change the file's owner and the group with a single command:
    ```
    $ sudo chown <new owner name>:<new group name> <filename>
    ```
Access Control in Ubuntu

➢ Recursive Permission Changes:
  ➢ To change the permissions of multiple files and directories with one command.
  ➢ Recursive chmod with -R and sudo
    ➢ $ sudo chmod <permissions> -R <path to some directory>

Recursively changing the file permissions can be extremely dangerous ！！！！！！

$ sudo chmod 777 -R / home/user1/Desktop/tempfiles
(what this command is doing?)