Lecture 6: Access Control

CS 392/681: Computer Security
Fall 2006
Polytechnic University

Nitesh Saxena

*Adopted from a previous lecture by Nasir Memon

Outline of the lecture
- What Access Control is?
- How it is implemented: Reference Monitor
- What are the different mechanisms
  - Access control matrix
  - Access control list
  - Capability
  - Lock and Key based
Access Control

- Any system consists of objects and subjects (active objects such as processes, users etc.) which access these objects.
- The security policy of a system defines
  - What a subject is allowed to do
  - What may be done with an object
- In other words – Access Control
- Two issues –
  - How do you specify access control to be performed?
  - How do you implement access control policy?
- Subjects, Objects and Interpretation of rights varies from system to system. For example, a process “reading” a process.

Reference Monitor Concept

- Above is just a conceptual model
- An actual system may not include an explicit reference monitor
- But we need to define functionality of the reference monitor and design mechanisms for its implementation.
Access Control Matrix (ACM) Model

- A Matrix $A$ with $a[s, o] \subseteq R$, where $s \subseteq S$ – the set of subjects, $o \subseteq O$ the set of objects and $R$ is the set of rights the subject has over the object.

<table>
<thead>
<tr>
<th>Process 1</th>
<th>File 1</th>
<th>File 2</th>
<th>Process 1</th>
<th>Process 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process 1</td>
<td>$r, w, o$</td>
<td>$r$</td>
<td>$r, w, x, o$</td>
<td>$w$</td>
</tr>
<tr>
<td>Process 2</td>
<td>$a$</td>
<td>$r, o$</td>
<td>$r$</td>
<td>$r, w, x, o$</td>
</tr>
</tbody>
</table>

ACM Example: LAN

- The set of objects, subjects and rights vary from system to system

<table>
<thead>
<tr>
<th>Hostname</th>
<th>Telegraph</th>
<th>Nob</th>
<th>Toadfax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Telegraph</td>
<td>Own</td>
<td>ftp</td>
<td>ftp</td>
</tr>
<tr>
<td>Nob</td>
<td>ftp, nfs, mail, own</td>
<td>ftp, nfs, mail</td>
<td></td>
</tr>
<tr>
<td>Toadfax</td>
<td>ftp, mail</td>
<td>ftp, nfs, own</td>
<td></td>
</tr>
</tbody>
</table>

1/30/2006 Module 5 - Access Control
ACM Example: Program

- Model is general. For example, programming language accesses:

<table>
<thead>
<tr>
<th></th>
<th>counter</th>
<th>Inc_ctr</th>
<th>Dec_ctr</th>
<th>manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inc_ctr</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dec_ctr</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>manager</td>
<td>call</td>
<td>call</td>
<td>call</td>
<td></td>
</tr>
</tbody>
</table>

- In fact, entries in ACM can even be functions that determine rights based on state, history, time of day etc. For example, a “lock” that ensures no access while write.

- Your text has even more elaborate examples.

Access Control Mechanisms

- Various access control mechanisms have been proposed:
  - Access Control Matrix
  - Access Control List
  - Capability based access control
  - Lock and Key based access control.
Discretionary and Mandatory Controls

- A *discretionary access control* mechanism is defined as those procedures and mechanisms that enforce specified mediation at the discretion of individual users.

- A *mandatory access control* mechanism is defined as those procedures and mechanisms that enforce specified mediation not at the discretion of individual users but rather at the discretion of a centralized system administration facility.

Access Control Matrix (ACM)

- An **Access Control Matrix** is a table in which
  - each row represents a subject,
  - each column represents an object, and
  - each entry is the set of access rights for that subject to that object.
ACM - Example

- Consider system with two files and two processes. Set of rights is - r,w,x,a,o (read, write, execute, append, own).

<table>
<thead>
<tr>
<th></th>
<th>File 1</th>
<th>File 2</th>
<th>Process 1</th>
<th>Process 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process 1</td>
<td>r,w,o</td>
<td>r</td>
<td>r,w,x,o</td>
<td>w</td>
</tr>
<tr>
<td>Process 2</td>
<td>a</td>
<td>r,o</td>
<td>r</td>
<td>r,w,x,o</td>
</tr>
</tbody>
</table>

- Can get very large and hence inefficient in general purpose scenarios - seldom used.

Access Control Lists

- Instead of using ACM, Access Control List (ACL). Essentially store each column of ACM with the object it represents.

- Definition: Let $S$ be set of subjects and $R$ the set of rights of a system. An access control list $l$ is a set of pairs $l = \{(s, r) : s \in S, r \subseteq R\}$. Let $acl$ be a function that determines the access control list associated with a particular object $o$. $acl(o) = \{(s_i, r_i) : 1 \leq i \leq n\}$ means that subject $s_i$ may access $o$ using any right in $r_i$. 

**ACL - Example**

- For ACM shown earlier, corresponding ACL’s are:

\[
\begin{align*}
\text{acl(file 1)} &= \{(\text{proc.1, \{r,w,o\}} \text{ (proc. 2, \{a\})}\} \\
\text{acl(file 2)} &= \{(\text{proc.1, \{r\}} \text{ (proc. 2, \{r,o\})}\} \\
\text{acl(proc.1)} &= \{(\text{proc.1,\{r,w,x,o\}} \text{ (proc.2, \{r\})}\} \\
\text{acl(proc.2)} &= \{(\text{proc.1,\{rw\}} \text{ (proc.2, \{r,w,x,o\})}\}
\end{align*}
\]

**Abbreviated ACL’s**

- To further reduce storage, one can abbreviate ACL’s as in UNIX.
- One can also assign default access to groups of subjects as well as specific rights to individual subjects.
  - Two ways of doing this: 1) What is not prohibited is permitted 2) What is not permitted is prohibited. Latter always better. Why?
  - Example: Unix hosts.deny and hosts.allow files.
Example - File Protection in Unix

- UNIX - allow read, write, execute, delete to each of the individual groups - owner, group, world.
  - Difficult for users in different groups to share files, since each user may belong to exactly one group.
- The Unix **set userid (suid)** scheme allows another user to temporarily acquire the protection level of a file's owner.
  - While executing the program to change their own password, Unix users actually acquire temporary modify access to the system's password file, but in a controlled way using suid.

Unix special users

- Special user with extra privileges - **root**.
  - UID is 0.
  - Can do (almost) anything!!
  - Holy grail of hackers!
- Other special users
  - **daemon** or **sys** - handles some network services
  - **ftp** - used for anonymous FTP access.
  - **uucp** - manages UUCP system.
  - **guest** - used for site visitors.
  - **lp** - used by printer system
  - Etc.
Unix Groups

- Every user belongs to one or more groups.
- The GID of primary group the user belongs to is stored in passwd file.
- Groups useful for access control features.
- /etc/groups contains a list of all groups in the system along with GID’s.
- Some special groups –
  - wheel - group of administrators
  - uucp, lp, etc. - groups corresponding to special users.

Unix file access control

- Each file entry in a directory is a pointer to a data structure called inode.

<table>
<thead>
<tr>
<th>mode</th>
<th>Type of file and access rights</th>
</tr>
</thead>
<tbody>
<tr>
<td>uid</td>
<td>User who owns the file</td>
</tr>
<tr>
<td>gid</td>
<td>Group which owns the file</td>
</tr>
<tr>
<td>atime</td>
<td>Access time</td>
</tr>
<tr>
<td>mtime</td>
<td>Modification time</td>
</tr>
<tr>
<td>itime</td>
<td>Inode alteration</td>
</tr>
<tr>
<td>Block count</td>
<td>Size of file</td>
</tr>
<tr>
<td></td>
<td>Pointer to physical location</td>
</tr>
</tbody>
</table>
Unix file permission bits

- Two examples of file permissions obtained by ls -l command
  - `rw------`
  - `drwxr-xr-x`
- First character indicates type of file
  - `-` plain file
  - `d` directory
  - `c` character device (tty or printer)
  - `b` block device
  - `l` symbolic link
  - `Etc.`

File permission bits (contd.)

- Next nine characters taken in groups of three indicate who can do what with the file
  - `R` – Permission to read
  - `W` – Permission to write
  - `X` – Permission to execute
- The three classes of permission correspond respectively to
  - Owner
  - Group
  - Other
File permission bits – special cases

- File permission bits do not apply to symbolic links.
- If you have x access but no r access you can execute the program without reading it (not on Linux).
- Execute permission in a directory means you can change to the directory. Secret Files!
- File permission bits also commonly specified in octal notation. 0777 mean -rwxrwxrwx, 0500 means -r_x------, etc.

Umask and default permissions

- umask (User file creation mode mask) is a four digit octal number used to determine file permissions for newly created files.
- It defines permission you do not want to be given (the bit-wise complement of the permission you want a file to have by default).
- 0666 – default mode means 022 umask.
- 0077 umask means ... 0022 means ...
- Set up at log in time in environment variables.
The suid bit

- Sometimes unprivileged users must perform tasks that are privileged.
  - Change password thereby modify /etc/passwd.
- UNIX allows certain programs to change UID to their owner when executed.
  - SUID programs - change UID to owner.
  - SGID programs - change GID to owners group.
- `ls -l` command indicates if SUID or SGID
  - `-rwsr-xr-x` indicates SUID
  - `-rwxr-sr-x` indicates SGID

Limitations of UNIX file permission system

- Abbreviated ACL’s in general and UNIX in particular may not be flexible enough for many circumstances.
- Consider the following example:
  - 5 users, Anne, Beth, Caroline, Della and Elizabeth.
  - Anne wants Beth to read her file and nothing else.
  - She wants Caroline to write
  - Della to only read and write
  - Elizabeth to only execute
  - Above not possible with Unix file permission bits!!
Augmenting abbreviated ACL’s

AIX uses extended permissions to augment base permissions.

attributes:
base permissions: owner (bishop): rw-
group (sys) : r--
others : ---

extended permissions enabled

<table>
<thead>
<tr>
<th>Action</th>
<th>Permission</th>
<th>Subject(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>specify</td>
<td>rw-</td>
<td>u:heberlei</td>
</tr>
<tr>
<td>permit</td>
<td>-w-</td>
<td>u:nelson, g=sys</td>
</tr>
<tr>
<td>permit</td>
<td>rw-</td>
<td>u:levitt</td>
</tr>
<tr>
<td>deny</td>
<td>-w-</td>
<td>u:heberlei, g=faculty</td>
</tr>
</tbody>
</table>

Issues to consider while designing an ACL based system

- Which subject can modify an object’s ACL?
- Does ACL apply to privileged user (root), if any?
- Does ACL support groups or wildcards?
- What are the default access rights?

Read text for examples of how above are handled in different systems.
Revoking Rights

- Revoking involves deletion of subject’s rights from object’s ACL.
- Typically owner of object has ability to provide or delete rights.

Capability based access control.

- Conceptually, capability is row of ACM i.e. list of rights for a subject.
- Definition: Let $O$ be set of objects, and $R$ the set of rights of a system. A capability list $c$ is a set of pairs $c = \{(o, r) : o \in O, r \subseteq R\}$. Let $cap$ be function that determines capability list $c$ associated with subject $s$. Then $cap(s) = \{(o_i, r_i) : 1 \leq i \leq n\}$ is that subject $s$ may access $o_i$ using any right in $r_i$. 
Example

For the ACM we saw earlier, capability lists are:

```
cap(proc. 1) = {(file1, {r,w,o}), (file2, {r}),
               (proc 1, {r,w,x,o}), (proc 2, {w})}.
```

```
cap(proc. 1) = {(file1, {r,w,o}), (file2, {r}),
               (proc 1, {r}), (proc 2, {r,w,x,o})}.
```

Capability Based Access Control - Implementation.

A capability is an unforgeable “token” giving the subject certain rights to an object.

- Analogous to movie ticket or ID card.
- Could be made transferable with appropriate entry in ticket.
- Used by Kerberos (WIN 2K).

To make sure capability cannot be forged:

- Maintained by OS.
- Stored in a region not accessible to users.
- Use cryptography (example, digital signatures) to prevent forgery.
**Example - Amoeba**

- On creation of an object, capability corresponding to object is returned to owner.
- To later use object, owner presents capability.
- Capability encoded name of object (24 bits), the server that created it (48 bits), rights (8 bits, initially all set), and 48 bit random “check” field.
- Random number stored in table of server that created object. When capability presented, number checked.
- Attacker who does not know random number cannot forge capability.
- If capability disclosed, system becomes vulnerable.

**Revoking rights in capability based system**

- Check each process and delete capability? Too inefficient. How to do this efficiently?
- One method: Use *indirection*. Capability does not name object but contains a pointer to object in global table. To revoke entry, just invalidate entry in global table.
- Amoeba: Change random check and issue new capability. This validates all existing capabilities.
Comparison of ACL and capability

- Two questions arise in access control systems:
  - Given a subject, what objects can access it and how?
  - Given an object, what subjects can access it and how?
- Former easier with capabilities and latter with ACL. Why?
- Latter more often asked, hence ACL’s used more often.
- With more distributed processing, perhaps the former question will be asked more in the future.

Access control with Locks and Keys

- Combines features of ACL’s and capabilities.
- A piece of information *(lock)* associated with the object.
- Another piece of information *(key)* associated with subjects authorized to access the object.
- Example implementation: Encrypt object and provide key to subject.
- Any of n subjects can access the objects
  - N keys k1, k2,...,kn; object o
  - Encrypted object: o’ = Enc(K1,o), Enc(K2,o),...,Enc(Kn,o)
- All of n subjects together can access the object
  - Encrypted object: o’ = Enc(K1,Enc(K2,Enc(...Enc(Kn,o)...)))
Confused Deputy Problem

- ACLs don’t solve the problem
- Capabilities do

Role-based Access Control

- Subjects have various roles
- Each role has certain access right
- A subject with a certain role has the access right corresponding to that particular role
- Example: hospital
  - Roles: doctor, nurses, staff
  - Doctors can right prescriptions, nurses can take blood samples, staff can make appointments
  - ABC is a doctor, XYZ is a nurse, PQR is a staff member
  - ABC can right prescriptions, XYZ can take blood samples, PQR can make appointments
Role-based Access Control

- Advantage

Ring based access control

- Used by MULTICS. Generalizes notion of user and supervisor modes.
- Segments in memory and disk are of two types - data and procedure.
- Segments have r, w, x, and a rights associated.
- In addition, notion of protection ring associated with segments. Ring is a number r, 0 <= r <= 63.
- A process is said to reside in ring r if it is currently executing a segment in ring r.
- Kernel resides in ring 0.
- Higher the ring number, lower the privileges.
- Crossing ring boundaries causes traps to the kernel which invokes ring gatekeeper.
Access brackets for data segments

- Each data segment as *access bracket*, which is a ring \((a_1, a_2)\) with \(a_1 \leq a_2\).
- Assume data segments permission allow desired access to a process in ring \(r\). The access bracket add additional constraints:
  - If \(r \leq a_1\); access permitted;
  - If \(a_1 < r \leq a_2\), \(r\) (read) and \(x\) permitted, \(w\) and \(a\) denied.
  - If \(a_2 < r\); all access denied.
Call brackets for procedures.

- In addition to access brackets, procedure segments may also have *call bracket* \((c_1, c_2)\).
- This leads to protection \((a_1, a_2, a_3)\) where \((a_1, a_2)\) - access bracket & \((a_2, a_3)\) - call bracket.

Access rules are:
- If \(r \leq a_1\) access permitted; but ring crossing fault occurs.
- If \(a_1 \leq r \leq a_2\) all access permitted and no fault occurs.
- If \(a_2 < r \leq a_3\) access permitted if made through valid gate.
- If \(a_3 < r\) all access denied.

Why call brackets?

- Consider a service routine in ring \(a\).
  - We need to allow user set \(A\) to invoke this routine.
  - We allow user set \(B\) to invoke only in specific ways
  - We do not allow user set \(C\) to invoke at all.
- Access brackets can take care of \(A\) and \(C\).
- To handle set \(B\) we need call brackets so that we can regulate their access.