CS 356 – Lecture 6
Access Control

Spring 2013
Review

• Chapter 1: Basic Concepts and Terminology
  – Integrity, Confidentiality, Availability, Authentication, and Accountability
  – Types of threats: active vs. passive, insider/outsider
  – Lots of terminology and general concepts

• Chapter 2: Basic Cryptographic Tools
  – Symmetric key encryption and secure hashing
  – Public key cryptography
  – Random Numbers

• Chapter 3 – User Authentication
  – Passwords
  – Checking passwords and other user auth techniques
Chapter 4

Access Control
Access Control

ITU-T Recommendation X.800 defines access control as follows:

“The prevention of unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner.”
Access Control Principles

RFC 2828 defines computer security as:

“Measures that implement and assure security services in a computer system, particularly those that assure access control service”.
Relationship Among Access Control and Other Security Functions

![Diagram](image-url)

**Figure 4.1** Relationship Among Access Control and Other Security Functions
Access Control Policies

Figure 4.2  Multiple Access Control Policies. DAC, MAC, and RBAC are not mutually exclusive. A system may implement two or even three of these policies for some or all types of access. [SAND94]
Access Control Requirements

- reliable input
- support for fine and coarse specifications
- least privilege
- separation of duty
- open and closed policies
- policy combinations and conflict resolution
- administrative policies
- dual control
Access Control Basic Elements

- Subject – entity capable of accessing objects
  - Concept equates with that of process
  - Typically held accountable for the actions they initiate
  - Often have three classes: owner, group, world

- Object – resource to which access is controlled
  - Entity used to contain and/or receive information
  - Protection depends on the environment in which access control operates

- Access right – describes the way in which a subject may access an object
  - E.g. read, write, execute, delete, create, search
Discretionary Access Control (DAC)

- scheme in which an entity may enable another entity to access some resource
  - often provided using an access matrix
    - one dimension consists of identified subjects that may attempt data access to the resources
    - the other dimension lists the objects that may be accessed
  - each entry in the matrix indicates the access rights of a particular subject for a particular object
### Access Matrix

<table>
<thead>
<tr>
<th>SUBJECTS</th>
<th>File 1</th>
<th>File 2</th>
<th>File 3</th>
<th>File 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>User A</td>
<td>Own Read Write</td>
<td>Read</td>
<td>Own Read Write</td>
<td></td>
</tr>
<tr>
<td>User B</td>
<td>Read</td>
<td>Own Read Write</td>
<td>Write</td>
<td>Read</td>
</tr>
<tr>
<td>User C</td>
<td>Read Write</td>
<td>Read</td>
<td>Own Read Write</td>
<td></td>
</tr>
</tbody>
</table>

(a) Access matrix
Figures 4.3b and c

Example of Access Control Structures

(b) Access control lists for files of part (a)

(c) Capability lists for files of part (a)
### Authorization Table for Files

<table>
<thead>
<tr>
<th>Subject</th>
<th>Access Mode</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Own</td>
<td>File 1</td>
</tr>
<tr>
<td>A</td>
<td>Read</td>
<td>File 1</td>
</tr>
<tr>
<td>A</td>
<td>Write</td>
<td>File 1</td>
</tr>
<tr>
<td>A</td>
<td>Own</td>
<td>File 3</td>
</tr>
<tr>
<td>A</td>
<td>Read</td>
<td>File 3</td>
</tr>
<tr>
<td>A</td>
<td>Write</td>
<td>File 3</td>
</tr>
<tr>
<td>B</td>
<td>Read</td>
<td>File 1</td>
</tr>
<tr>
<td>B</td>
<td>Own</td>
<td>File 2</td>
</tr>
<tr>
<td>B</td>
<td>Read</td>
<td>File 2</td>
</tr>
<tr>
<td>B</td>
<td>Write</td>
<td>File 2</td>
</tr>
<tr>
<td>B</td>
<td>Write</td>
<td>File 3</td>
</tr>
<tr>
<td>B</td>
<td>Read</td>
<td>File 4</td>
</tr>
<tr>
<td>C</td>
<td>Read</td>
<td>File 1</td>
</tr>
<tr>
<td>C</td>
<td>Write</td>
<td>File 1</td>
</tr>
<tr>
<td>C</td>
<td>Read</td>
<td>File 2</td>
</tr>
<tr>
<td>C</td>
<td>Own</td>
<td>File 4</td>
</tr>
<tr>
<td>C</td>
<td>Read</td>
<td>File 4</td>
</tr>
<tr>
<td>C</td>
<td>Write</td>
<td>File 4</td>
</tr>
</tbody>
</table>
## Extended Access Control Matrix

<table>
<thead>
<tr>
<th>SUBJECTS</th>
<th>S_1</th>
<th>S_2</th>
<th>S_3</th>
<th>OBJECTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>control</td>
<td>owner</td>
<td>owner control</td>
<td>files</td>
</tr>
<tr>
<td>S_1</td>
<td>control</td>
<td>owner</td>
<td>owner control</td>
<td>read *</td>
</tr>
<tr>
<td>S_2</td>
<td>control</td>
<td>write *</td>
<td>execute</td>
<td></td>
</tr>
<tr>
<td>S_3</td>
<td>control</td>
<td>write</td>
<td>stop</td>
<td></td>
</tr>
</tbody>
</table>

* - copy flag set

**Figure 4.4 Extended Access Control Matrix**
Figure 4.5 An Organization of the Access Control Function
<table>
<thead>
<tr>
<th>Rule</th>
<th>Command (by $S_o$)</th>
<th>Authorization</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>transfer ${\alpha^*}$ to $S, X$</td>
<td>'\alpha^*$' in $A[S_o, X]$</td>
<td>store ${\alpha^*}$ in $A[S, X]$</td>
</tr>
<tr>
<td>R2</td>
<td>grant ${\alpha^*}$ to $S, X$</td>
<td>'owner' in $A[S_o, X]$</td>
<td>store ${\alpha^*}$ in $A[S, X]$</td>
</tr>
<tr>
<td>R3</td>
<td>delete $\alpha$ from $S, X$</td>
<td>'control' in $A[S_o, S]$ or 'owner' in $A[S_o, X]$</td>
<td>delete $\alpha$ from $A[S, X]$</td>
</tr>
<tr>
<td>R4</td>
<td>$w \leftarrow$ read $S, X$</td>
<td>'control' in $A[S_o, S]$ or 'owner' in $A[S_o, X]$</td>
<td>copy $A[S, X]$ into $w$</td>
</tr>
<tr>
<td>R5</td>
<td>create object $X$</td>
<td>None</td>
<td>add column for $X$ to $A$; store 'owner' in $A[S_o, X]$</td>
</tr>
<tr>
<td>R6</td>
<td>destroy object $X$</td>
<td>'owner' in $A[S_o, X]$</td>
<td>delete column for $X$ from $A$</td>
</tr>
<tr>
<td>R7</td>
<td>create subject $S$</td>
<td>none</td>
<td>add row for $S$ to $A$; execute create object $S$; store 'control' in $A[S, S]$</td>
</tr>
<tr>
<td>R8</td>
<td>destroy subject $S$</td>
<td>'owner' in $A[S_o, S]$</td>
<td>delete row for $S$ from $A$; execute destroy object $S$</td>
</tr>
</tbody>
</table>
Protection Domains

- set of objects together with access rights to those objects
- more flexibility when associating capabilities with protection domains
- in terms of the access matrix, a row defines a protection domain
- user can spawn processes with a subset of the access rights of the user
- association between a process and a domain can be static or dynamic
- in user mode certain areas of memory are protected from use and certain instructions may not be executed
- in kernel mode privileged instructions may be executed and protected areas of memory may be accessed
UNIX File Access Control

UNIX files are administered using inodes (index nodes)

- control structures with key information needed for a particular file
- several file names may be associated with a single inode
- an active inode is associated with exactly one file
- file attributes, permissions and control information are sorted in the inode
- on the disk there is an inode table, or inode list, that contains the inodes of all the files in the file system
- when a file is opened its inode is brought into main memory and stored in a memory resident inode table

Directories are structured in a hierarchical tree

- may contain files and/or other directories
- contains file names plus pointers to associated inodes
UNIX
File Access Control

- unique user identification number (user ID)
- member of a primary group identified by a group ID
- belongs to a specific group
- 12 protection bits
  - specify read, write, and execute permission for the owner of the file, members of the group and all other users
- the owner ID, group ID, and protection bits are part of the file’s inode

Figure 4.6 UNIX File Access Control
Traditional UNIX
File Access Control

- “set user ID” (SetUID)
- “set group ID” (SetGID)
  - system temporarily uses rights of the file owner / group in addition to the real user’s rights when making access control decisions
  - enables privileged programs to access files / resources not generally accessible
- superuser
  - is exempt from usual access control restrictions
  - has system-wide access
Access Control Lists (ACLs) in UNIX

- modern UNIX systems support ACLs
  - FreeBSD, OpenBSD, Linux, Solaris
- FreeBSD
  - `setfacl` command assigns a list of UNIX user IDs and groups
  - any number of users and groups can be associated with a file
  - read, write, execute protection bits
  - a file does not need to have an ACL
  - includes an additional protection bit that indicates whether the file has an extended ACL
- when a process requests access to a file system object two steps are performed:
  - step 1 selects the most appropriate ACL
    - owner, named users, owning / named groups, others
  - step 2 checks if the matching entry contains sufficient permissions
What’s Next

• Read Chapter 1, 2, and 3
  – Chap 1: Focus on big picture and recurring concepts
  – Chap 2: Identify cryptographic tools and properties
  – Chap 3: How can you authenticate a user?

• Homework  Posted on Course Website
  – Due Tuesday

• Next Lecture Topics from Chapter 4
  – Role Based Access Control