Computer Systems Security

Lesson 05 - Authorization

Authorization

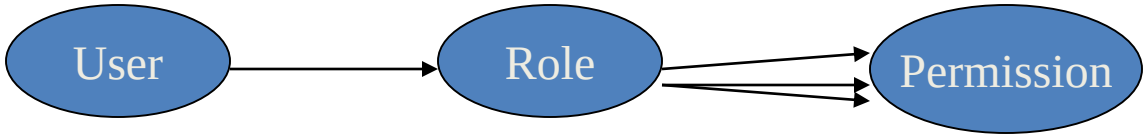
- Authorization:
  - Specifies what an authenticated user can do
  - Away of establishing and controlling access to resources.
  - Addresses the suite of privileges a user can have on the system or network
  - Specific to different areas of the system. e.g. user space vs kernel space in the OS.

Access Control Histroy

- RBAC – Role Based Access Control
- CBAC – Context Based Access Control
- CAAC – Context Aware Access Control

Role-Based Access Control

- Sandu et al. formalized Role-Based Access Control in 1996



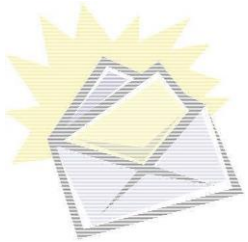
- User U acting in role R is granted permission P
  - Advantage: greatly improved efficiency
  - Disadvantage: cannot specify fine-grained rules

Context-Based Access Control

- What is “context”?
  - Circumstances in which an event occurs

Subject

Name
Age
ID
Location



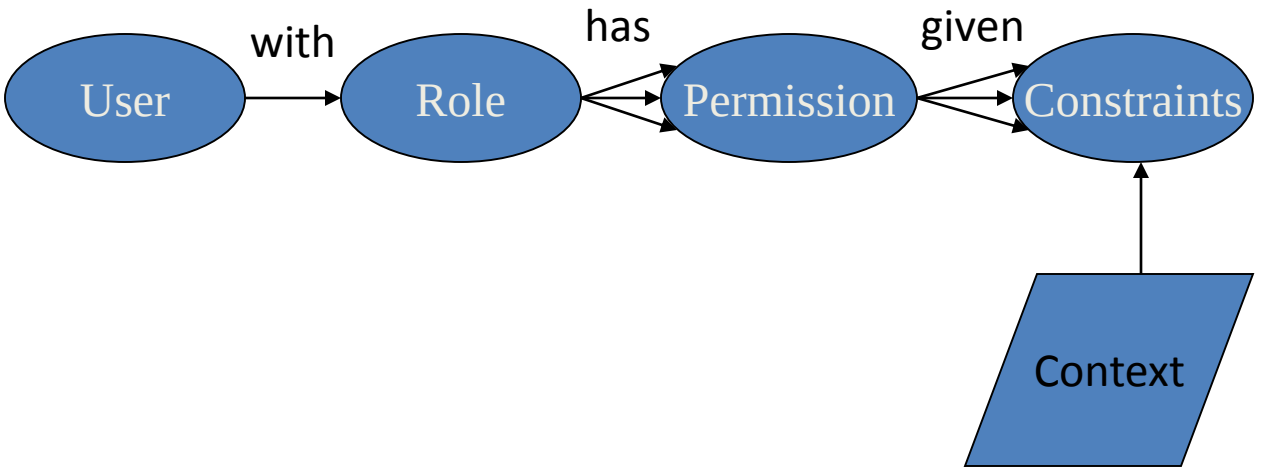
Object

Type
Owner

System

Time
Date
CPU Load

Context-Based Access Control



- Advantage: access control is context-aware
- Disadvantage: this is still a static model

RBAC → CBAC → CAAC

- RBAC and CBAC, even with extensions, cannot meet the access requirements of some modern environments
- CAAC is an extension to CBAC that is consistent with implementation via web services
- CAAC permits dynamic specification and dynamic enforcement of arbitrary access rules
- Context implementation is separated from the main business logic of target applications.

Context-Aware Access Control

- Presented 2004 by Juhnze Hu
- Terminology:
  - **Data Object:** the smallest unit to be accessed in an application
  - **Data Type:** a group of data objects with the same attributes
  - **Data Set:** the set of all data objects
  - **User Set:** the set of potential entities that access the data objects

Authorization

- Types of authorization systems:
  - User rights
  - Role-based authorization
  - Access Control Lists (ACLs)
  - Rule-based authorization

User Rights

- Different from “permissions”(granting access to resources and specifying what users can do with them)
- Provides authorization to do things that affect the entire system.
- Example:
  - Creating users, groups,
  - Assigning users to groups
  - Log on to a system

User Rights

- Implicit user rights:
  - Granted to default groups
  - Cannot be removed
  - Granted to root UNIX (wheel group)
  - System admins can grant the rights to use specific resources as “root” without getting access to the root password.
- Example:
  - sudo command in UNIX/LINUX

Role-based authentication

- Default roles:
  - Administrator
  - User
- Administrators:
  - Granted special privileges and access to a larger array of resources than ordinary users
    - (create users, assign passwords, shutdown/reboot, access system files etc)
- Users:
  - Log in and read files

Access Control Matrix

- Definition:
  - An abstract model of protection state in computer systems
  - Describes the rights of users over files in a matrix format.
  - Forms basis for ACLs and capabilities
  - The set of all protected entities is called the set of objects O,
  - The set of subjects S is the set of active objects such as processes and users.

Access Control Matrix

- Representation (for a matrix X):
  - Each row is a subject (e.g. a process)
  - Each column is an object (e.g. a file)
  - Each matrix entry is the access right that a subject has for an object
- The relationship between entities is described with:
  - Rights drawn from a set of rights R in each entry a[*s, o*]. Where *s* ϵ S, *o* ϵO, and $a[s, o] \subseteq R$
  - The subject s has the set of rights a[*s,o*] over the object o.

Access Control Matrix

- Example:
  - Subjects: To be processes P1, P2
  - Objects to be files f1, f2, f3, f4
  - Access rights: read, write, execute, own

	f1	f2	f3	f4
P1	rwo	r	rwxo	w
P2	r	-	ro	rwxo

- The set of protection states of the system is represented by the triple (S,O,A).

Access Control Matrix

- ACM operations:
  - Create subject,
  - create object,
  - destroy subject,
  - destroy object,
  - add access right,
  - delete access right

Access Control Matrix

- Problems:
  - The number of subjects and objects will be large hence the matrix will utilize a significant amount of storage.
  - Most entries in the matrix will either be blank (indicating no access) or be the same (default settings)
  - Complexity of matrix storage space management during addition or deletion of subjects and objects

Access Control Lists

- Definition:
  - A list of permissions attached to an object.
  - Specifies which users or system processes are granted access to objects and operations allowed on any given object
  - A variant of Access Control Matrix (ACM)
  - Stores each column of the ACM with the object it represents
  - Each object has an associated set of pairs (a subject, rights pair)

Access Control List

- Interpretation:
  - If S is the set of subjects and R the set of rights in a system, then an access control list (ACL) *l* is the set of pairs $l = \{(s, r) : s \in S, r \subseteq R\}$
  - The named subject can access the associated object using any of the rights in the set of rights given.

Access Control List

- If *acl* is a function that determines the access control list *l* associated with an object o, then:

$$acl(o) = \{(s_i, r_i) : 1 \leq i \leq n\}$$

- means:
  - Subject s_i may access object o using any right in r_i.
  - *acl(f1)*={(user1,{r,w,x}), (user2, {r})}

Access Control Lists

- Implementation considerations:
  - Which subjects can modify an object's ACL?
  - Do ACLs apply to a privileged user?
  - Does the ACL support groups or wildcards?
  - How are contradictory ACLs handled if any?
  - If a default setting is allowed, do ACL permissions modify it?

Access Control List

- File-Access permissions:
  - Supported in both windows and UNIX
  - Implementation differs on the two platforms

Access Control Lists

- Windows:
  - NTFS file system maintains an ACL for each file and folder
  - ACL composed of a list of access control entries (ACEs)
  - Each ACE includes a security identifier (SID) and permissions granted to that SID

Access Control Lists

- Much of OS security functions are accorded through the ACLs.
- An object's security descriptor can contain two ACLs:
  - Discretionary Access Control Lists (DACL)
    - Identifies the users and groups who are allowed or denied access
  - System Access Contol Lists (SACL)
    - Control how access is audited

MAC and DAC

- MAC and DAC
  - ACLs can be refined into required and optional settings
  - Discretionary Access Control:
    - Provides an entity or object with access privileges it can pass on to other entities.
    - Are also called IBAC (Identity Based Access Control)
    - Individual users may determine access controls

MAC and DAC

- Mandatory Access Control:
  - Require that access control policy decisions be beyond the control of individual owners of an object.
  - Control is left with the system administrators and root users.
  - Access permissions cannot be passed from one user to another
  - Used in enforcing system-wide policy
  - Better suited for environments with rigid information

Windows File-Access Permissions

- Permissions may be either access or deny.
- SIDs may represent:
  - User accounts, computer accounts or groups.
- ACEs may be assigned by admins, owners of the file or users with privileges to apply permissions.
- During logon: A list is composed that includes the user's SID, the SIDs of the groups to which the user belongs and the privileges the user has

Windows File-Access Permissions

- When logged in:
  - An access token is created for the user and attached to any running process the user might start in the system.
- Accessing a resource:
  - Security subsystem compares the list of ACEs on the resource against the list of SIDs and privileges in the access token.
  - If there is a match for both the SIDs and the access rights requested, authorization is granted except when the access is marked as “deny”.
  - Mismatch results in implicit denial

Windows File-Access Permissions

- Permissions:
  - Full control,
  - Modify,
  - Read and Execute,
  - List folder contents,
  - Read,
  - Write
  - Special permissions (granular selection of permissions)

UNIX File-Access Permissions

- UNIX Systems:
  - Traditionally don't use ACLs
  - Limit access to files based on user account and group
  - Classification: owner, group, others
  - Owner privileges include determining who can read, write or execute the file.
  - Directories can also have permissions assigned as above
  - Less granularity in these permissions

UNIX File-Access Permissions

- Limitations:
  - Impossible to grant read access to a single individual in addition to the file owner.
  - Impossible to grant read access to one group and write access to another
- Some UNIX system provide ACLs e.g. Solaris
- Traditional UNIX file permissions include:
  - Read,
  - Write
  - Execute
  - Denied

Rule-Based permission

- Require development of rules that stipulate what a specific user can do on a system.
- Example:
  - User A can access resource R but not resource S.
  - User A can only read file P if accessing from a PC in a given IP address range.

Capabilities

- Is conceptualized like a row of an ACM.
  - Each subject has associated with it, a set of pairs.
  - Each pair contains an object and a set of rights
  - The subject associated with this list can access the named object as specified by the given rights
  - Let O be the set of objects and R the set of rights in a system. A capability list c is a set of pairs

$$c = \{(o, r) : o \in O, r \subseteq R\}$$

Capabilities

- If *cap()* is a function that determines the capability list c associated with a particular subject s then:

$$cap(s) = \{(o_i, r_i) : 1 \leq i \leq n\}$$

- Means that subject s can access object o_i using any right in r_i.
- Example:
  - cap(user1)={(file1,{rwo}), (file2,{rwx}), (process1,{r})}

Capabilities

- Capabilities encapsulate object identity:
  - When a process presents a capability on behalf of a subject, the OS examines the capability to determine both the object and the access to which the object is entitled.

Capabilities

- Protection mechanisms:
  - Tagged architecture:
    - Has a set of bits associated with each hardware word
    - Has two states : set (read but not modify) and unset (read and modify)
  - Protected memory
    - Protection bits associated with paging or segmentation
    - All capabilities are store in a page (segment) that a process can read but not alter

Capabilities

- Protection mechanisms (contd…)
  - Cryptography:
    - Use cryptographic checksums to ensure that capabilities are not illegally alterered.
    - Each capability has an associated cryptographic checksum
    - The checksum is enciphered using a cryptosystem whose key is known to the OS

Capabilities

- Copying and amplification
  - Ability to copy implies ability to give rights
  - Each capability is associated with a copy flag
  - The copy flag must be set for a process to copy a capability to another process
  - Amplification: Increasing of privileges