CS458: Introduction to Information Security

Notes 11: Access Control

Yousef M. Elmehdwi

Department of Computer Science

Illinois Institute of Technology

yelmehdwi@iit.edu

November 8th, 2018

Slides: Modified from Computer Security: Principles and Practice, 4th Edition. By: William Stallings and Lawrie Brown & https://sandilands.info/sgordon/teaching/Steven Gordon, Sirindhorn International Institute of Technology

Outline

- Access Control Concept
- Discretionary Access Control
    - Access Control Matrix
    - Access Control List
    - Capabilities

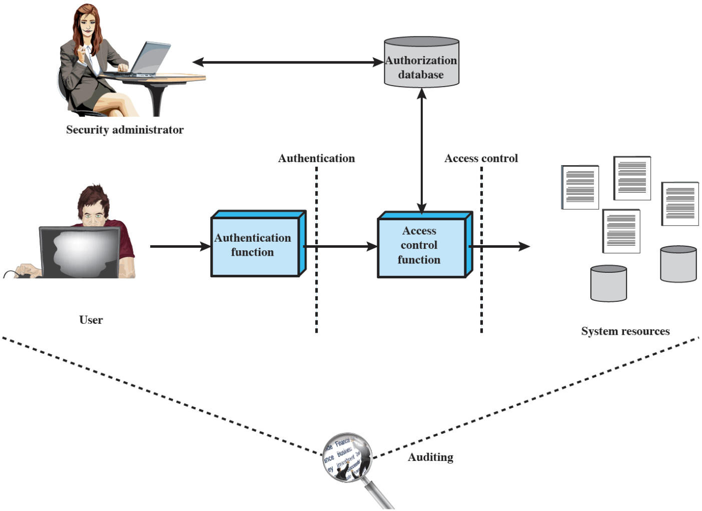
Access Control

- "The prevention of unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner"[1]
- "A process by which use of system resources is regulated according to a security policy and is permitted only by authorized entities (users, programs, processes, or other systems) according to that policy"[2]

[1] ITU-T Recommendation X.800 "Security architecture for Open Systems Interconnection"

[2] RFC 4949, Internet Security Glossary

Relationship Among Access Control and Other Security Functions



Access Control and Other Security Functions

- Access control part of a broader context
  - Authentication - verification that the credentials of a user or other entity are valid. Are you who you say you are? Who goes there?
    - Determine whether access is allowed
    - Authenticate human to machine
    - Authenticate machine to machine
  - Authorization - granting of a right or permission to a system entity to access a resource. Are you allowed to do that?
    - Follows authentication.
    - Once you have access, what can you do?
    - Enforces limits on actions
    - "access control" often used as synonym for authorization
  - Audit
    - independent review of system records and activities in order to test for adequacy of system control, ensure compliance to policy, detect breaches and recommend changes

Access Control Policies

- Discretionary Access Control (DAC)
  - Controls access based on the identity of the requestor and on access rules (authorizations) stating what requestors are (or are not) allowed to do
  - Regular users can adjust the policy, .i.e., entities may allow other entities to access resources
- Mandatory Access Control (MAC)
  - Controls access based on comparing security labels with security clearances
  - Regular user can not adjust the policy, i.e., entities cannot grant access to resources to other entities
- Role-based Access Control (RBAC)
  - Controls access based on the roles that users have within the system and on rules stating what accesses are allowed to users in given roles
    - i.e., roles of users in system and rules for roles are used to control access
- Attribute-based access control (ABAC)
  - Controls access based on attributes of the user, the resource to be accessed, and current environmental conditions

General Requirements of Access Control

- Reliable input
    - Access control assumes user has been authenticated, that other control inputs (e.g., object names, addresses) are correct
    - i.e., we assume that the authentication is reliable and it works.
- Fine and coarse specifications
    - We should be allowed to be very fine detailed in specifying who can access what, and also allowed very coarse specification of who can access what.
    - Management overhead vs. precision in policy
- Least privilege
    - Provide the least set of privileges needed for the task
- Separation of duty
    - More than one entity to complete a task (increased protection from fraud and errors)
    - Manages conflict of interest
    - Example: accountant and auditor

General Requirements of Access Control

- Open and closed policies
  - “closed”: access limited to those explicitly stated
    - e.g., default “deny” on firewall rule
  - “open”: access limitations are specified, all others allowed
- Policy combinations and conflict resolution
  - Different entities have different policy and we need to adjudicate with respect to both
  - We need to implement an access control system that implement those policies and make sure there is no conflicts
  - A given policy may have conflicting rules. How to resolve?
    - Unix example “deny global read or cross” at account level, “global read” permission at file level. what’s the policy decision?
- Administrative policies
  - Need to allow admin user to the system
- Dual control
  - Two entities required to implement policy change (e.g., should allow to have multiple admins)

Basic Elements of Access Control System

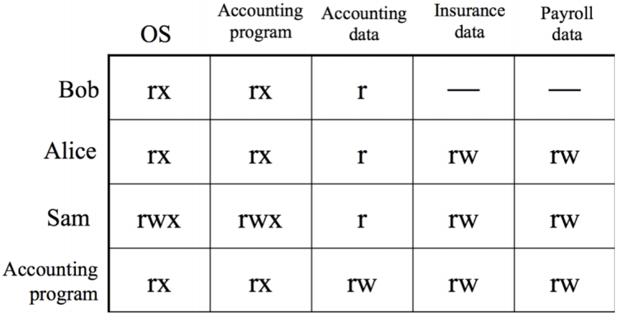
- Subject - entity capable of access resources (objects)
  - Often subject is a software process (often run by human users)
  - Classes of subject
    - **Owner**
      - This may be the creator of a resource, such as a file. For system resources, ownership may belong to a system administrator.
    - **Group**
      - In addition to the privileges assigned to an owner, a named group of users may also be granted access rights, such that membership in the group is sufficient to exercise these access rights.. A user may belong to multiple group
    - **World**
      - The least amount of access is granted to users who are able to access the system but are not included in the categories owner and group for this resource
- Object - resource to which access is controlled
  - e.g. records, blocks, pages, files, portions of files, directories, email boxes, programs, communication ports
  - i.e., entity used to contain and/or receive information
- Access right - describes way in which a subject may access an object
  - e.g. read, write, execute, delete, create, search

Discretionary Access Control

- Scheme in which an entity may be granted access rights that permit the entity, by its own violation/if they choose so, to enable another entity to access some resource
- Common access control scheme in operating systems and database management systems
- Often provided using an access matrix (Access control 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

Example of Access Control Structures

- Subjects (users) index the rows.
- Objects (resources) index the columns



	OS	Accounting program	Accounting data	Insurance data	Payroll data
Bob	rx	rx	r	—	—
Alice	rx	rx	r	rw	rw
Sam	rwx	rwx	r	rw	rw
Accounting program	rx	rx	rw	rw	rw

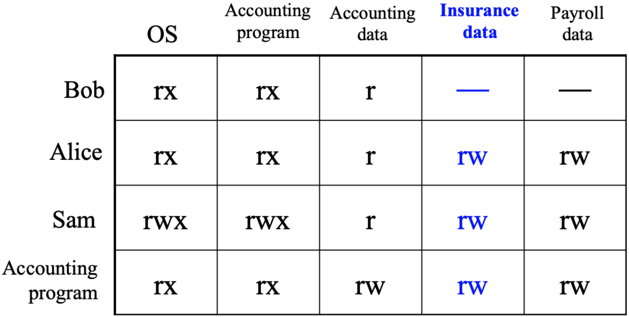
r - read, w - write, x - execute

Are you allowed to do that?

- Access control matrix has all relevant information to answer this questions.
  - Could be `100's` of users, `10,000's` of resources.
    - Then matrix with `1,000,000's` of entries.
  - How to manage such a large matrix?
    - We need to check this matrix before access to any resource by any user.
  - How to make this efficient/practical?
    - Split it into smaller pieces: by rows or by columns:
    - Access Control Lists (ACL)
      - For each object, list subjects and their access rights
    - Capability Lists
      - For each subject, list objects and the rights the subject have on that object
    - Alternative implementation: authorization table listing subject, access mode and object; easily implemented in database

Access Control Lists (ACLs)

- ACL stores access control matrix by column (Slice by column)
  - ACL for insurance data in blue.

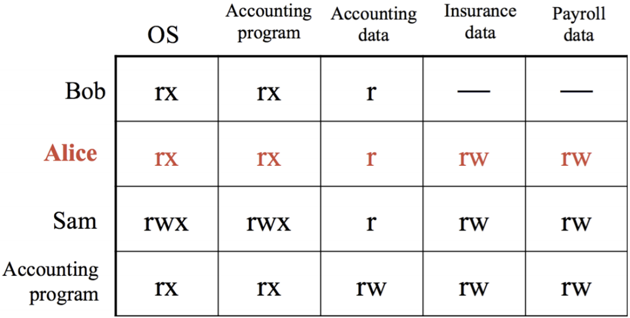


	OS	Accounting program	Accounting data	**Insurance data**	Payroll data
Bob	rx	rx	r	—	—
Alice	rx	rx	r	rw	rw
Sam	rwx	rwx	r	rw	rw
Accounting program	rx	rx	rw	rw	rw

r - read, w - write, x - execute

Capabilities (or C-Lists)

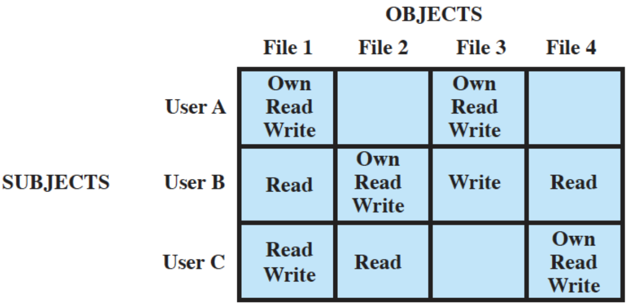
- Store access control matrix by row (Slice by row) yields Capabilities tickets
  - Capability for Alice in red



	OS	Accounting program	Accounting data	Insurance data	Payroll data
Bob	rx	rx	r	—	—
Alice	rx	rx	r	rw	rw
Sam	rwx	rwx	r	rw	rw
Accounting program	rx	rx	rw	rw	rw

r - read, w - write, x - execute

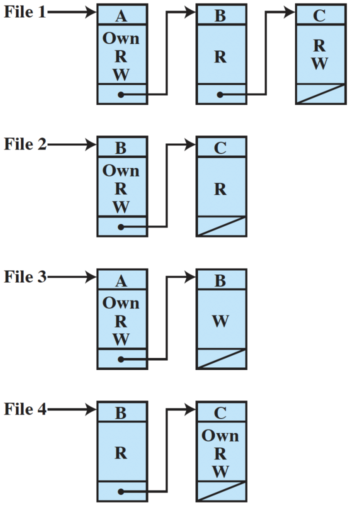
Example of DAC Access Matrix



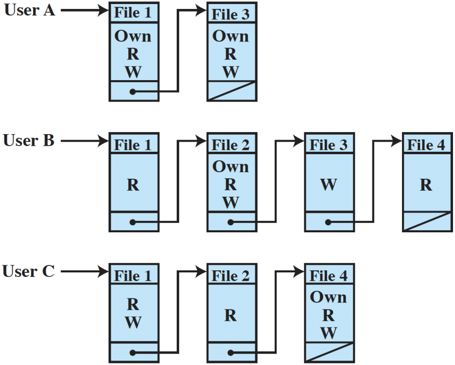
		OBJECTS			
		File 1	File 2	File 3	File 4
SUBJECTS	User A	Own Read Write		Own Read Write	
	User B	Read	Own Read Write	Write	Read
	User C	Read Write	Read		Own Read Write

- User A owns files 1 and 3 and has read and write access rights to those files.
- User B has read access rights to file 1, and so on.

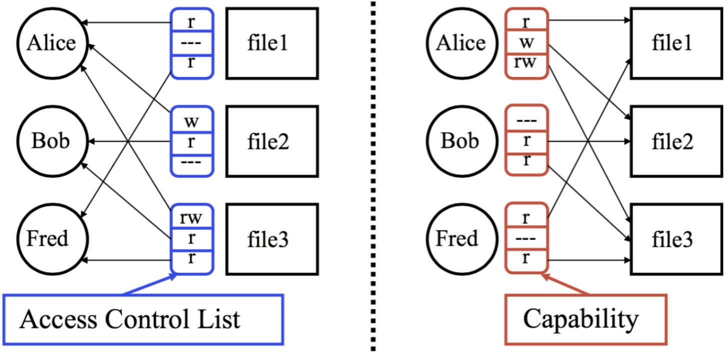
Example of Access Control Lists



Example of Capability Lists



ACLs vs Capabilities



- Note that arrows point in opposite directions.
- ACLs: Given an object, which subjects can access it?
- Capabilities: Given a subject, what objects can it access?
- With ACLs, still need to associate users to files

Capabilities Integrity

- A capability ticket specifies authorized objects and operations for a particular subject
- Subject presents "capability" in order to access object
    - Capability data structure encapsulates object ID with allowed rights.
- Unlike ACLs, capabilities are not completely contained by the OS
    - Can be transmitted, e.g., tokens in Kerberos
- Capability integrity is a big concern
    - The integrity of the ticket must be protected, and guaranteed
        - Operating system hold all tickets on behalf of users
          e.g, held in a region of memory inaccessible to users
        - Include an unforgeable token in the capability
          e.g, random password, or a cryptographic message authentication code

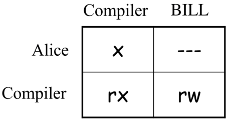
Authorization Table

- Not sparse, like the access matrix
  - Each row describes one access right of one subject to one resource
  - Sorting or accessing the table by subject is equivalent to a capability list
  - Sorting or accessing the table by object is equivalent to an ACL
  - Easy implemented with relational database

Subject	Access Mode	Object
A	Own	File 1
A	Read	File 1
A	Write	File 1
A	Own	File 3
A	Read	File 3
A	Write	File 3
B	Read	File 1
B	Own	File 2
B	Read	File 2
B	Write	File 2
B	Write	File 3
B	Read	File 4
C	Read	File 1
C	Write	File 1
C	Read	File 2
C	Own	File 4
C	Read	File 4
C	Write	File 4

Confused Deputy

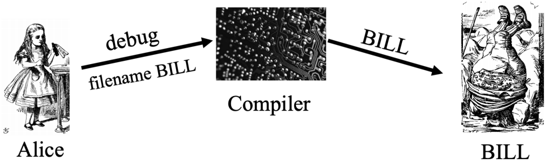
- Well-known problem arises due to poor configuration of ACL in DAC
- Confused deputy is a program that is fooled by some other party into misusing its own authority
- ACL: Two resources
  - Compiler and BILL file (billing info/BILL).
  - Compiler can has access to file BILL and can write to it.
- Alice can invoke compiler with a debug filename
- Alice can not access and not allowed to write to BILL



	Compiler	BILL
Alice	x	---
Compiler	rx	rw

- Compiler is acting as a subject rather than an object because it's able to execute and activate access to the resource file BILL

ACL's and Confused Deputy



- Compiler is **deputy** acting on behalf of Alice.
- Compiler is **confused**
  - When executed on behalf of Alice, the compiler still access according to its own privileges of the file Bill
  - However, Alice is not allowed to write to the file BILL.
- Compiler has confused its rights with Alice's.

Confused Deputy

- Compiler acting for Alice is confused.
- There has been a separation of authority from the purpose for which a given resource is used.
  - ⇒ may create possible conflicting access control privileges that apply to a resource
- With ACLs, more difficult to prevent this
- With Capabilities, easier to prevent problem.
  - Must maintain association between authority and intended purpose
  - Capabilities make it easy to delegate authority as needed

ACLs vs Capabilities

- ACLs
    - Good when users manage their own files.
    - Protection is data-oriented.
    - Easy to change rights to a resource.
- Capabilities
    - Easier to delegate - avoid the confused deputy.
    - Easy to add/delete users.
    - More difficult to implement

Key Points

- Access control part of broader system
- Access Control Matrix or Access Matrix
  - Means to model access control systems
- Real implementations
  - Access control lists
  - Capability lists