Access Control - Operating Systems

Access control can be defined as a series of mechanisms to specify what users can do, which resources they can access, and what operations they can perform on a system. More generally, it permits managers of a system to direct or restrain the behavior, use, and content of a system. Access control mechanisms take as input security policies and attempted actions, and output an accept or reject on that action. The security policy that goes into the access control mechanism defines what a subject is allowed to do, and/or what may be with an object.

Access control has to server three main functions in order to be successful at what they do.

Identification
Identification is a method of establishing the subjects (people, processes, programs) identity through the use of user name or other public information that conforms to identification component requirements. Each value should be unique, for user accountability, and a standard naming scheme should be followed. The value should be non-descriptive of the user's position or tasks, and the value should not be shared between users.

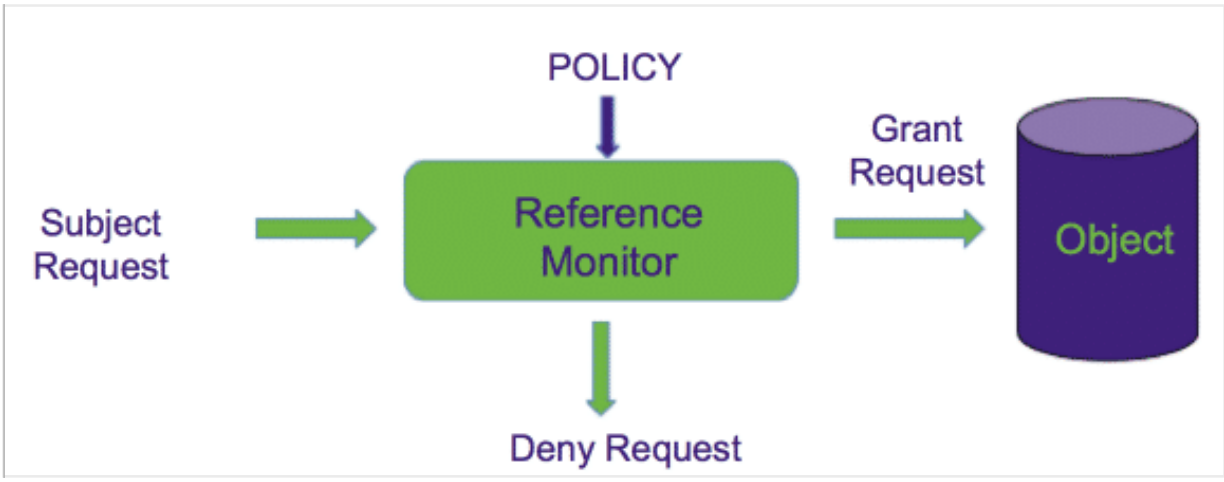
Authentication
A method of proving the identity using something a subject is, has, or knows. This might include the use of biometrics, passwords, passphrase, token, or other private information, as discussed in the previous lesson.

Authorization
A process that determines that the proven identity has some set of characteristics associated with it that gives it the right to access the requested objects. This access criteria can be thought of as roles or rules.

Access Control Models:

All access controls models implement a reference monitor.

A reference monitor is a secure, always-used and fully-testable module that controls all access to data objects or devices. The reference monitor verifies the nature of the request against a table of allowable access types for each process on the system. The reference monitor must always be used or else the attacker can attempt to bypass and fetch the designated object.



Discretionary Access Control (DAC):
A system that uses discretionary access control allows the owner of the resource to specify which subjects can access which resources. Said another way, access control is at the discretion of the owner of the resource.

Mandatory Access Control (MAC):
In Mandatory Access Control, access is based on a security labeling system. Users have security clearances and resources have security labels that contain data classifications. This model is used in environments where information classification and confidentiality is very important, such as the military.

Non-Discretionary (Role Based) Access Control:
Role Based Access Control (RBAC) is where one's role determines the access that's provisioned for them.
It is the best system for an organization that has high turnover, like typical businesses.

Context Based Access Control:
Context based access control is an access control method based on the context of a subject's request to an object, in addition to just the identities of the subject and object themselves. This is a little different from other methods, and requires more information to make a decision. For example, consider an organization with an employee, Bob. Usually, Bob reads information about the organization's transactions at the end of each week to ensure that nothing suspicious is happening. This is done each Friday during work hours.
Now, imagine that there is a read request to the transaction data coming from Bob's account on a Sunday night at 11 PM. The context of this request is very abnormal, since it is coming from outside of Bob's working hours, and on a weekend. If we were using context based access control, this request may be rejected. However, other types of access control (including DAC, MAC, RBAC and Content Dependent Access Control) may let this access occur, since Bob clearly has permission to read the transaction data in other circumstances.

Constrained User Interfaces:
One way that we can enforce access control is by constraining the user interface used to get access. This can be done by not allowing certain types of access on the interface, or not including the ability to request certain types of access, or access to certain objects. There are three major types of constrained user interfaces. They include menus and shells, database views, and physically constrained interfaces.

Access Control Matrix (ACM):

	File 1	**File 2**	**Process 1**	**Process 2**
Process 1	r,w,o	r	r,w,x,o	w
Process 2	a	r,o	r	r,w,x,o

Access Control List (ACL):
An access control list (ACL) is a set of permissions that correspond to an object (the object will store the permission listing)

Example: acl(File A): {(Alice: write), (Bob: read, execute)}

As we can see in the example above, Alice can write to File A, and Bob can read and execute File A.
Since Carla is not mentioned on the ACL, she has no rights to access File A in any way.

ACL <=> ACM:
acl(file 1) = {(proc.1, {r,w,o}) (proc. 2, {a})}
acl(file 2) = {(proc.1, {r}) (proc. 2, {r,o})}
acl(proc.1) = {(proc.1,{r,w,x,o}) (proc.2, {r})
acl(proc.2) = {(proc.1,{w}) (proc.2, {r,w,x,o})

UNIX Access Model:

In UNIX, each user has a user ID (UID), which is a uniquely identifying number that is linked to their username. In addition to each normal user there are a few special user accounts. The most important of these user accounts is root, which has a UID of 0 and can do almost anything in the system. If an attacker is able to access the root account, they have unhindered access to the system and can accomplish any goal them may have.

In addition to root there are other special users in UNIX. These include daemon or sys, which handles some important services including some networking, ftp, which is used for anonymous FTP access, uucp, which manages the UNIX-to-UNIX copy system, guest which is used for site visitors, lp which is used by printer systems, and more.

Groups:
In addition to users, UNIX also has the concept of groups. Each group has a unique group ID (GID), and each user belongs to one or more groups. Groups are useful for access control features, since they allow an easy way to apply some coarse grained access control rules to users quickly.

UNIX File Access Control:
Permissions are stored as access control lists in a structure called an inode. The inode also stores other information about a file, such as the user who owns it, the group who owns it, the time it was last accessed, the time it was

last modified, the time the inode was last modified, the size of the file, and where on the disk the file sits.

mode	Type of file and access rights
uid	Users who own the file
gid	Group which owns the file
atime	Access time
mtime	Modification time
itime	Inode alteration
Block count	Size of file (sort of)
	Pointer to physical location

(First 3 chars, 'rwx' refers to owner permissions, next 3 'rw-' refers to group, and '- - -' is for everyone else being no permissions)
- rwx rw- ---

In octal format, the execute permission is 1, the write permission is 2, and the read permission is 4. 0 means no permissions at all.
Each octal digit that follows chmod is a combination of: 4,2,1,0

TYPES OF FILES:

- Character files: hardware file which reads/writes data in character by character. Some examples include auxiliary devices like keyboards, mouses, printers: Examples': ls -a /dev/ | grep ^c'
- Block files: A block file is a hardware file which read/write data in blocks instead of character by character. These are used when we want to read/write data in a bulk fashion such as USB, CDROMS, etc. Examples': ls -a /dev/ | grep ^d'
- Symbolic Links: a pointer to another file... think shortcut icons on your desktop

WINDOWS ACCESS CONTROL:

Tokens:
A security descriptor is the set of information required for the authorization of access. It consists of the ACLs associated with an object, as well as

ownership information, group information, revision numbers, etc. Tokens, in contrast, when a process uses a token it is using the security attributes of another subject. Think of this as a verified source honors me with a ticket, therefore I'm trusted with the same rights as the verified source.

SMART PHONES:

Since these devices are in single user access, the access model is not designed to care about multi-user permission management but instead, treat applications as users and ensure processes are self contained and non-overstepping. Each app runs within it's own sandbox (like JVM) and has a UID tied to it. The application then reads from a manifest file to determine what it has permission to do on host device.