Introduction to Information Security (IY2760/DC3760): Introduction to computer security - access control

Dr Siaw-Lynn Ng

November 17, 2023

Contents

1 What is computer security?

So far...

- Communications between two computers:
  - Encryption for confidentiality, MACs for data origin authentication and integrity.

  - Key establishment and entity authentication to allow this.

- At the computer end:
  - Entity authentication to access the computer.

What happens when you are "in"?

- What happens when you are "in"? (*Computer security.*)
  - Some resources are private/restricted.
  - How do we express who can do what with which resource? (*Access control.*)
- How do we decide who can do what? (*Access control model.*)
- Where do we locate the access control? (*Hardware/OS security.*)

What is computer security?

- Informally: "Computer security is the *protection* of the items that a person values, called the *assets* of a computer or networked systems."
- Assets: hardware, software, data, reputation, etc.
- Value is subjective; based on the user's perspective.
- Computer security deals with the *prevention* and *detection* of *unauthorised access* by users of a system.

Computer security issues

- How do we control which people can use a computer system?
- How do we control which programs a user can run?
- How do we control which resources a process can access?
- How do we protect processes that share computer resources from each other?

Fundamental techniques

- Authentication:
  - This verifies user identities.
- Access control (authorisation):
  - Limits access to programs and resources to those authorised to have access (the main focus of this part of the course).

- Memory protection:
  - Segmented virtual memory model prevents a process reading or overwriting memory used by other processes.

Computer security: a potted history (NON-EXAMINABLE)

- 1940s: The first electronic computers.
- 1950s: The start of an industry.
- 1960s: Development of operating systems.
- 1970s: Age of the mainframe.
- 1980s: Age of the PC.
- 1990s: Age of the Internet.
- 2000s: Age of the Web.
- 2010s: Age of ubiquitous computing.

Computer security reports of the 60s and 70s (NON-EXAMINABLE)

- The beginning of multi-user multi-tasking computers led to investigations of security.
- Notable security reports of the 60s and 70s:
  - The Ware Report (1970). `https://www.rand.org/pubs/reports/R609-1.html`
  - The Anderson Report (1972). `https://apps.dtic.mil/sti/pdfs/AD0758206.pdf`
  - The Saltzer and Schroeder paper (1975). "The protection of information in computer systems" by Jerome H. Saltzer and Michael D. Schroeder. Proceedings of the IEEE, vol. 63, no. 9, pp. 1278-1308, Sept. 1975.
- Marked the start of the field of computer security.

2 Access control

2.1 Motivation and terminology

What is access control?

- A generic term for the process that controls interactions between users and resources.
- An access control system implements a security policy determined by
  - organisational requirements;
  - statutory requirements, for example, covering Personally Identifiable Information (PII), including medical records.

Why access control?

- Why access control?
  - Prevents users from having unlimited access to system resources.
  - Limit access of unauthorised users that manage to break in.
- Access control is not required if access to resources does not need to be constrained.
  - For example, early stand-alone PCs could not (and did not need to) enforce access control.

Access control: an analogy

- In the real world we have methods to control access to sensitive resources, for example:
  - locks,
  - guards.
- How do you gain access to protected resources?
  - Locks are opened with keys, and keys must be distributed to users;
  - Guards only allow certain people in - guards thus have to be given a list of trusted users.

Terminology: objects, subjects, principals

- *Objects*
  - Resources (passive entities) in computer system.
  - Examples: Files, directories, printers, sockets.
- *Subjects*
  - Active entities that access resources.
  - Examples: Processes, threads.

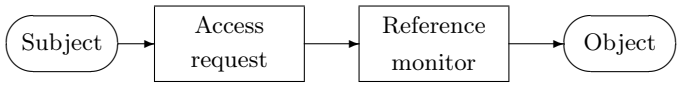
- *Principals*
  - Entities that represent a human user.
  - Examples: Users, groups, roles, cryptographic keys.
  - Principals can create subjects.

Access control: authorisation

- Authorisation relies on user authentication.
- The decision whether or not to grant an access request by a process is based on the security context of the process:
  - The security context is inherited from the user that initiated the process.
  - The security context of a user identifies the user and any groups to which the user belongs.

Access control: Schematic view

- A User (actually a process acting on behalf of user) requests access (read, write, print, etc.) to a resource in the computer system.
- The reference monitor:
  - Establishes the validity of the request.
  - Returns a decision either granting or denying access.



Terminology: Trusted computing base

- The trusted computing base (TCB) is the set of protection mechanisms within a computer system.
  - Includes hardware, firmware, and software.
  - Responsible for enforcing security.
  - Correct enforcement depends on the mechanisms within the TCB and on the correct configuration by administrators.

The reference monitor

- The reference monitor is the entity that mediates all access requests by subjects.
- The security kernel:
  - The hardware, firmware and software parts of a TCB implementing the reference monitor.
  - Must mediate all accesses.
  - Must be protected from modification.
  - Must ideally be verifiable as correct.

An analogy: locks and keys

- In a paper-based office, certain documents should only be read by certain individuals.
- We could implement security by:
  - storing documents in filing cabinets; and
  - issuing keys to the relevant individuals for the appropriate cabinets.
- The reference monitor controls access to the set of filing cabinets:
  - An access request (an attempt to open a filing cabinet) is granted if the key fits the lock.

Another analogy: guest lists

- A club might restrict access to club members.
- It can enforce security by hiring a doorperson and giving the doorperson a membership list.
- Reference monitor = doorperson + membership list.
- An access request is granted only if a client:
  - can prove identity (authentication); and
  - is on the membership list.

Access modes

- Two basic modes of interaction between subject and object:
  - *observe* (i.e. read);
  - *alter* (i.e. write).
- Accessing object = initiating a flow of information:
  - When observing (reading) an object, information flows from object to subject.
  - When altering (writing to) an object, information flows from subject to object.

Other access modes: Execute access

- Sometimes an object can be accessed without using the observe or alter mode(s). For example:
  - executable files (programs);
  - directories;
  - cryptographic keys.
- The *execute* access depends on the context:
  - Execute access to a binary executable file grants permission to run the program;
  - Execute access to a Unix directory = permission to access directory. Read and execute access combined = permission to list contents of directory.

Administrative access rights

- Some rights are administrative, involving:
  - Changes to access control data structures.
  - Changing access rights of a user.
- Often related to ownership of resource.

Access control policies

- Access control systems enforce policies.
- Two types of access control: *discretionary* and *mandatory*.
- Discretionary policies are based on identities (or other characteristics of users).
- Mandatory policies are independent of user identities.

Discretionary access control policies

- Discretionary access control (DAC) policies are based on identities.
  - Ownership of resources is typically important.
  - An owner of a resource can set access control mechanisms to allow or deny another subject access to the resource.

Mandatory access control policies

- Mandatory access control (MAC) policies are independent of user identities.
  - Characteristics of resources are important.
  - Access is only allowed if user and object belong to same security domain;
  - A system-wide policy decrees who is allowed to have access to resources.

Delegation

- Sometimes it is necessary to allow a process to perform an access on behalf of another process.
- This creates requirement for *delegation* (temporary transfer) of access rights.
- Functionality for delegation is typically incorporated into access control systems.

2.2 Access control models

Access control models

- In general, a model is an abstract way of describing a system.
- A model includes elements that are used to represent the system:
  - Sets, relations and functions (entities, relations between entities, and operations that can be performed).
- In the context of access control, a model typically describes a reference monitor.
- Here we will look at:
  - Access control matrices.
  - Access control lists.
  - Capability lists.
  - An information flow model.

3 Access control models

3.1 Access control matrices

Access control matrix I

- Introduced by Lampson (1972) and extended by Harrison, Ruzzo and Ullman (1976-8):
  - Columns indexed by objects;
  - Rows indexed by subjects;
  - Matrix entries are (sets of) access operations.
- Foundation of many theoretical security models.

A small example I

	a.out	b.out	allfiles.txt
jason	{r,w}	{r,w,x}	{r,w}
mick		{r,x}	{r}

Access control matrix II

- Suppose subject s wants to access object o using access right a.
- We can represent the request as triple: (s, o, a).
- A request is granted (by the reference monitor) if and only if a belongs to the access matrix entry corresponding to subject s and object o.

A small example II

	a.out	b.out	allfiles.txt
jason	{r,w}	{r,w,x}	{r,w}
mick		{r,x}	{r}

- The request (jason, allfiles.txt, w)?
  - It will be granted.
- The request (mick, allfiles.txt, w)?
  - It will be denied.

Disadvantages of the access control matrix

- An abstract formulation of access control.
- Not suitable for direct implementation:
  - The matrix is likely to be extremely sparse and therefore implementation is inefficient.
  - The management of the matrix is likely to be extremely difficult if there are tens of thousands of files and hundreds of users (resulting in millions of matrix entries).

3.2 Access control lists

Access control lists I

- An access control list (ACL) corresponds to a column in the access control matrix.
- Typically represented internally as a list of access rights associated with an object
  - For example, the ACL for b.out would be [(jason, {r,w,x}), (mick, {r,x})]

	a.out	b.out	allfiles.txt
jason	{r,w}	{r,w,x}	{r,w}
mick		{r,x}	{r}

Access control lists II

- ACLs focus on the objects and are typically implemented at operating system level.
- Disadvantage: How can we check the access rights of a particular subject efficiently?

3.3 Capability lists

Capability lists I

- A capability list corresponds to a row in the access control matrix.
  - For example, jason's capability list would be [(a.out, r,w), (b.out, r,w,x), (allfiles.txt, r,w)].

	a.out	b.out	allfiles.txt
jason	{r,w}	{r,w,x}	{r,w}
mick		{r,x}	{r}

- Access rights are kept with the subject.

Capability lists II

- Capability lists focus on subjects and are typically implemented in services and application software.
  - For example, database applications use capability lists to implement access control to tables.
- Disadvantage: How can we check which subjects can access a given object?

Back to the analogies

- ACL is analogous to a membership list:
  - The club is the (only) object.
  - Members appear on the list.
- A capability list is analogous to the set of keys issued to a user:
  - The filing cabinets are the objects.

3.4 An information flow model

An information flow model

- Accessing object = initiating a flow of information:
  - When observing (reading) an object, information flows from object to subject.
  - When altering (writing to) an object, information flows from subject to object.
- The notion of information flows can be used to construct an access control model.

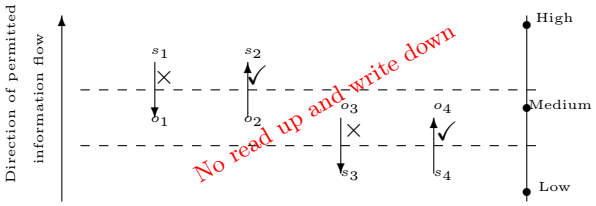
Information flow policy

- Every object and subject has a security level (*security label*).
  - Object label indicates sensitivity.
  - Subject label indicates clearance.
- The set of security labels is a partially ordered set.
  - Information flows must respect the partial ordering, so that information can only flow from entity a to entity b if $a \leq b$ (where $\leq$ is the partial ordering).

Aside: What is a partial order? (NOT EXAMINABLE)

- A partial order $\leq$ on a set L is a relation on (ordered) pairs of elements of L which is
  - *reflexive*: For all elements a of L, we have $a \leq a$.
  - *transitive*: For all elements a, b, c of L, if $a \leq b$ and $b \leq c$, then $a \leq c$.
  - *antisymmetric*: For all elements a, b of L, if $a \leq b$ and $b \leq a$, then $a = b$.
- If two elements a, b of L are not comparable, we say $a \not\leq b$.

An information flow policy example



- s_1 cannot write to o_1.
- s_2 can read o_2.
- s_3 cannot read o_3.
- s_4 can write to o_4.

Information flow policy properties

- What does such a policy prevent?
- Information leaks due to inappropriate reads.
  - It prevents unclassified users reading classified information.
- Information leaks due to inappropriate writes.
  - It prevents Trojan horses downgrading classified information.
  - It prevents classified information being printed to an unclassified printer.
- It addresses confidentiality. However, it does not address integrity requirements.

4 Role-based access control

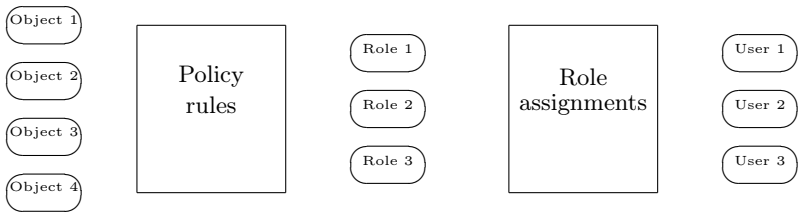
Issue with previous models

- Administration of access control systems can be time-consuming:
  - A system with 1000 users, 100 000 objects & 10 access rights has $1000 \times 100000 \times 10 = 10^9$ possible authorisation triples.
  - The *default* access rights for newly created objects, security groups and ACLs gives some reduction in administrative burden.
  - The potential for mistakes and omissions is high.
  - New paradigm required – scalable, easier to use.

Role-Based Access Control (RBAC)

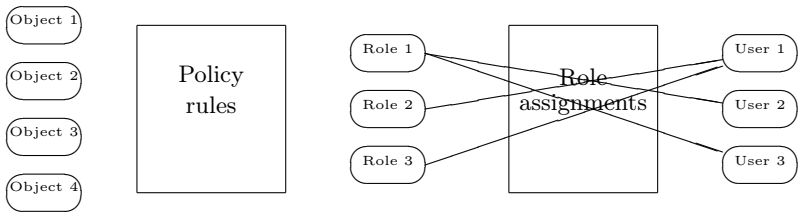
- The central concept in Role Based Access Control (RBAC) is the *role*:
  - A role is a collection of permissions.
  - It acts as a bridge between users and objects: access to objects is through roles.
  - Permissions are connected only to roles, not directly to users.
- It reduces the complexity of configuring the authorisation policy.

Privilege management



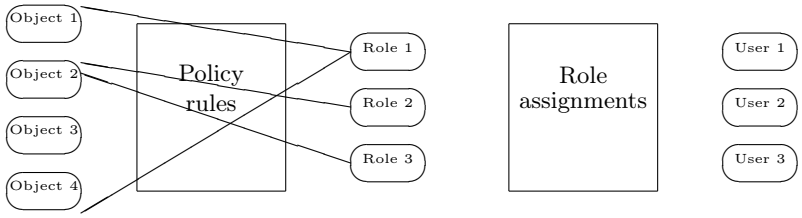
- With RBAC privileges are managed indirectly through roles.

RBAC core components: User Assignment relation



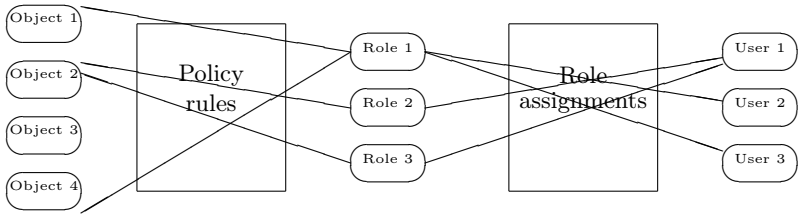
- The *User Assignment* (UA) relation is a many-to-many relation.
  - Users are assigned to one or more roles.
  - A user can be a member of many roles.
  - A role can have many users.

RBAC core components: Permission Assignment relation



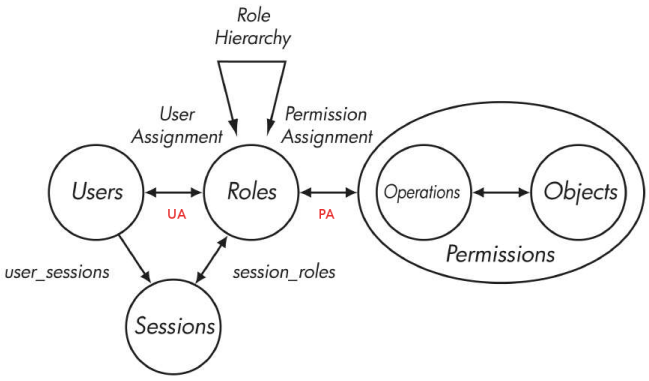
- The *Permission Assignment* (PA) relation is also a many-to-many relation.
  - Permissions are assigned to one or more roles.
  - A role can have many permissions.
  - The same permission can be assigned to many roles.

RBAC core components



- A user activates a session (*security context*) by selecting one or more roles associated with the user.
- An access request is granted if one or more roles associated with user has the necessary permission.

RBAC model



This diagram is taken from a presentation by Ed Coyne, available on the NIST website. NIST RBAC website: https://csrc.nist.gov/projects/role-based-access-control

RBAC formal model I

- In a formal RBAC model we have:
  - a set of users U,
  - a set of roles R,
  - a set of operations A,
  - a set of objects O.
- Define the set of permissions: $P \subseteq O \times A$.
  - $O \times A$ represents the set consisting of ordered pairs of elements from O and A.
  - For example, if File1 is an object and "Read" is an operation, then (File1, Read) is a permission.

RBAC formal model II

- Define the user assignment relation: $UA \subseteq U \times R$.
- Define the permission assignment relation $PA \subseteq P \times R$.
- The set of sessions S is a subset of $U \times 2^R$.
  - 2^R represents the set of all subsets of R.
  - A session is a user and the set of roles the user selects.

Special types of RBAC

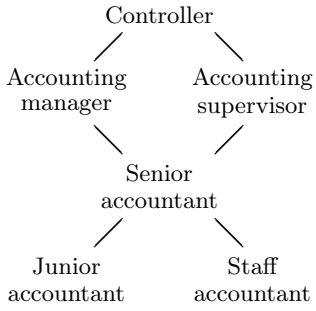
- We look briefly at two aspects of RBAC system:
  - hierarchical RBAC,
  - constrained RBAC.

4.1 Hierarchical RBAC

Hierarchical RBAC

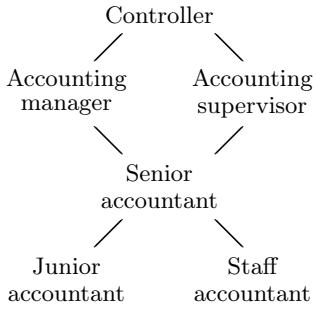
- A role hierarchy is used to impose a structure on set of roles to further reduce the administrative burden.
  - The role hierarchy should match the organisational structure.
  - More senior roles will inherit the rights of less senior roles.

Hierarchical RBAC: example I



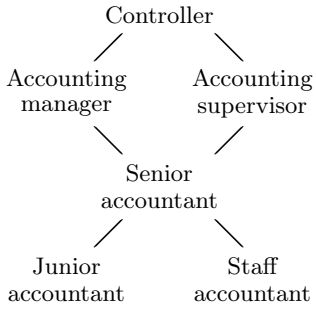
- Convention: senior roles higher in diagram.
- A line between two roles: upper role includes all the access rights of lower role.

Hierarchical RBAC: example II



- One role can inherit access rights of multiple roles.
- Eg. Senior accountant role includes all the access rights of junior accountant role and staff accountant role.

Hierarchical RBAC: example III



- Multiphe roles can inherit access rights of one role.
- Eg. Accounting manager role and accounting supervisor role both include all the access rights of senior accountant role.

- But accounting manager and accounting supervisor each has additional rights that the senior accountant role does not have.

Hierarchical RBAC: formal version

- A role hierarchy is a partially ordered set $(R, \leq)$.
- A user assigned to a role r (via User Assignment) is implicitly assigned to all roles $r' \leq r$.
  - A user assigned to role r is also assigned to all the roles that are lower than r.
- A permission assigned to a role r (via Permission Assignment) is implicitly assigned to all roles $r' > r$.
  - If a role r is assigned a permission then all the roles higher than r is also assigned that permission.

4.2 Constrained RBAC

Constrained RBAC

- Constraints in RBAC can be used to define *separation of duty (SoD)* requirements.
  - Some tasks need more than one user ('two-person rule', 'four-eyes rule', 'dual control').
  - The relevant permissions for such a task are assigned to different roles.
  - Membership of those roles is restricted by SoD constraints.

Separation of duty (SoD)

- From NIST CSRC glossary `https://csrc.nist.gov/glossary/`: No user should be given enough privileges to misuse the system on their own.
  - Eg. the person authorizing a paycheck should not also be the one who can prepare them.
- Two constraint types: static and dynamic.

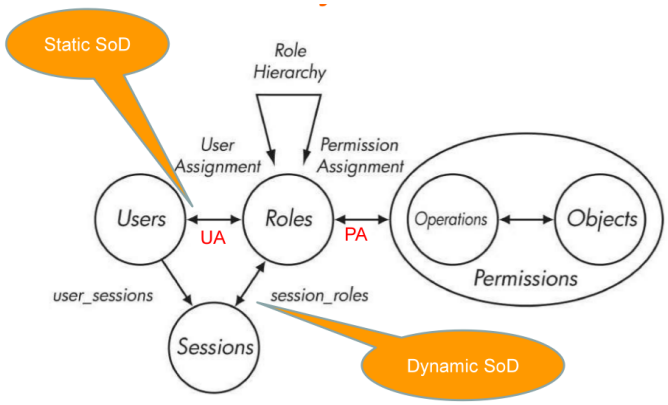
Static SoD constraints

- Static separation of duty limits the assignment of users to roles:
  - A static SoD constraint is specified as a pair (R', n) where $R' \subseteq R$ and n is an integer.
  - A user cannot have n roles in set R'.
  - For example, ({finClerk, poClerk}, 2) means a user cannot be assigned to both of the financial clerk and purchase order clerk roles.
- Static constraints are enforced by the UA relation.

Dynamic SoD constraints

- Dynamic separation of duty limits the activation of roles by users in sessions:
  - Dynamic SoD constraints are specified in the same way as static SoD constraints.
  - Dynamic constraints are enforced by the authentication mechanism.
  - When a user logs in and asks to be assigned certain roles, the constraint checking ensures that the user does not acquire a conflicting set of roles.
  - No user may activate n or more roles from the roles set in each user session.

Static and dynamic SoD



This diagram is taken from a presentation by Ed Coyne, available on the NIST website. NIST RBAC website: https://csrc.nist.gov/projects/role-based-access-control

RBAC administration I

- Changes may need to be made to the configuration of an RBAC system:
  - Assign (revoke) a user to (from) a role.
  - Assign (revoke) a permission to (from) a role.
  - Add (delete) a role.
  - Add (delete) an edge from the role hierarchy.
- Wide range of tools for admin, audit, etc. have been developed.

RBAC administration II

Two possible approaches:

1. Assign special administrative permissions to (administrative) roles.
   - Issues arise because it may become hard to decide whether the system is 'safe': whether or not certain undesirable situations can arise.
2. Use the structure of role hierarchy to limit power of (administrative) roles.
   - This remains an active area of current research.

Use of RBAC

- The use of RBAC to manage user privileges within a system or application is widely accepted as current best practice.
- Many systems effectively implement some form of RBAC, including: Microsoft - Active Directory, Microsoft SQL Server, Microsoft Azure, SELinux, FreeBSD, Solaris, Oracle DBMS, SAP R/3. . . ..

Potential conflicts

- Possible issues arise when constrained RBAC is used in conjunction with hierarchical RBAC.
- This is because a superior inherits the role assignments of all their subordinates.
  - This seriously complicates checking of static and dynamic constraints.
  - This also means that changes in the hierarchy also involve checking all the constraints.
  - This can be very time-consuming in a large system.

5 Examples of access control

5.1 Unix access control

Unix Preliminaries

- Unix was developed for friendly environments like research labs or universities.
- Security mechanisms were quite weak and elementary; improved gradually.
- Several flavours of Unix: vendor versions differ in the way some security controls are managed & enforced.
- Unix designed originally for small multi user computers in a network environment.
- Later scaled up to commercial servers and down to PCs.

Overview of security controls

- General pattern for security controls:
  - Information about users are stored in user accounts.
  - Privileges granted to a user are stored in accounts.
  - The system associates a user's privileges with a process after verifying the user's identity.
  - Permissions on resources are defined.
- When deciding whether to grant or deny an access request, the OS may refer to the user's identity , the user's privileges and the permissions set.

Principals in UNIX

- Principals: user identifiers (UIDs) and group identifiers (GIDs)
- Users belong to one or more groups.
- Information about principals is stored in user accounts and home directories.
  - User accounts stored in the `/etc/passwd` file: `username:password:UID:GID:name:homedir:shell`.
  - Example: `dieter:RT.QsZEEsxT92:1026:53:Dieter Gollmann :/home/staff/dieter :/bin/bash`

UNIX Subjects

- The subjects are *processes*.
- Each process has a process ID (PID).
- For example: Login.
  - Users are identified by usernames and authenticated by passwords.
  - When system is booted, the login process is started.
  - When a user logs in, this process verifies username and password.
  - If verification succeeds, the session UID/GID are changed to that of the user.

UNIX Objects

- The objects include
  - Files.
  - Directories.
  - Memory devices.
  - I/O devices.
- These are objects of access control and treated as resources, which are organized in a tree structured file system.

File permissions

- Each file is associated with three different groups of permissions: Owner ('user'), Group or Others.
- Access rights for each file are Read (r), Write (w), Execute (x).
- For example: `r w - r - - r - -` means read and write access for the Owner, read access for Group, read access for Other.

- Represented internally as a 9-bit access mask: If a bit is set then the corresponding permission is granted.
  - For example, `r w - r - - r - -` is represented as 110 100 100.
- Other additional bits for permissions for directories.

UNIX access control

- Access control is based on the attributes of subjects (processes) and objects (files).
- Unix associates access rights with each file corresponding to owner, group and other/world.
- Permissions are checked in the following order
  1. If your UID indicates you are the owner of the file, the permissions for *owner* decide whether you can get access.
  2. If you are not the owner but your GID indicates your group owns the file, the permissions for *group* decide it.
  3. If you are neither, the permissions for *other* decide it.
- (Superusers are not subject to this kind of access control.)

5.2 Windows access control

Windows access control

- Access control in Windows applies to objects:
  - Files, registry keys (entries in systems database), printers, etc.
- More complex in general than access control in a file system.
  - Access rights beyond simple read, write, execute.
- Means for structuring policies in complex systems: groups, roles, inheritance.

Principals

- Principals are the active entities that can be granted or denied access.
  - This may be managed locally on a single system or centrally managed for a group of systems belonging to a *domain*.
- Principals for access control:
  - An individual principal (with security identifier SID).

  - Group: a collection of SIDs managed by the domain controller (DC). A group has its own group SID, so groups can be nested.
  - Alias (local group): collection of user and group SIDs managed by DC or locally. This cannot be nested. Aliases are also used to implement logical roles.

Subjects

- Subjects are the active entities in an operational system.
  - Eg. processes and threads.
- Security credentials for a process are stored in an access token.
- Access tokens are bound to authenticated users.

Objects

- Objects get an owner when they are created.
  - Executive objects: processes, threads.
  - File system objects: files, directories.
  - Other objects: Registry keys, printers, etc.
- Securable objects have a security descriptor.
  - Describes owner and group.
  - Has discretionary access control lists (DACL) to describe who is granted or denied access.
  - Has system access control list (SACL) to allow audit.
  - Enables highly granular access control.

Permissions (access rights)

- Describe what one can do with an object.
- Standard permissions:
  - DELETE
  - READ_CONTROL: read access (to security descriptor) for owner, group, DACL.
  - WRITE_DAC: write access to DACL.
  - WRITE_OWNER: write access to owner.
  - SYNCHRONIZE: the right to use the object for synchronization. This enables a thread to wait until the object is in the signaled state.
- Specific permissions can be tailored to each class of objects.

Access Control Approaches

- Several ways to do Windows access control (with increasing granularity and complexity).
- Impersonation: access control based on the principal requesting access.
  - A process 'impersonates' user SID of its token.
  - Coarse but simple to implement.
  - Typical O/S concept; does not work well at application level.
- Role-centric: use groups and aliases to give a process suitable access rights for its task.
- Object-centric: objects at the application level get a security descriptor. This can get complex.