# PV222
Security Architectures

Lecture 1
Security Models and Access Control Mechanisms

Lecture Overview

- Introduction to the concept of **Authorisation** as a natural extension of **User Authentication**.

- Describe the basic model for a **reference monitor** as a means of enforcing authorisation.

- Introduce the following standard **Access Control Models**, describing how each provides a different means of implementing authorisation in a computer system, and discuss their potential drawbacks:

  - *Access Control Matrix*

  - *Role-Based Access Control*

  - *Clark-Wilson*

- Describe a simplified version of the **Windows 2000 security mechanism**, and demonstrate how it implements authorisation.

The Basic Scenario

- Users run applications (programs)
- Uses interact with operating system
  - Graphical User Interface
- Applications require operating system services
  - File access
  - Printing
- Applications and operating system are executed by the CPU
- The CPU and the operating system manage access to resources and peripherals
  - Bring files into main memory from hard disk
  - Respond to input from hardware
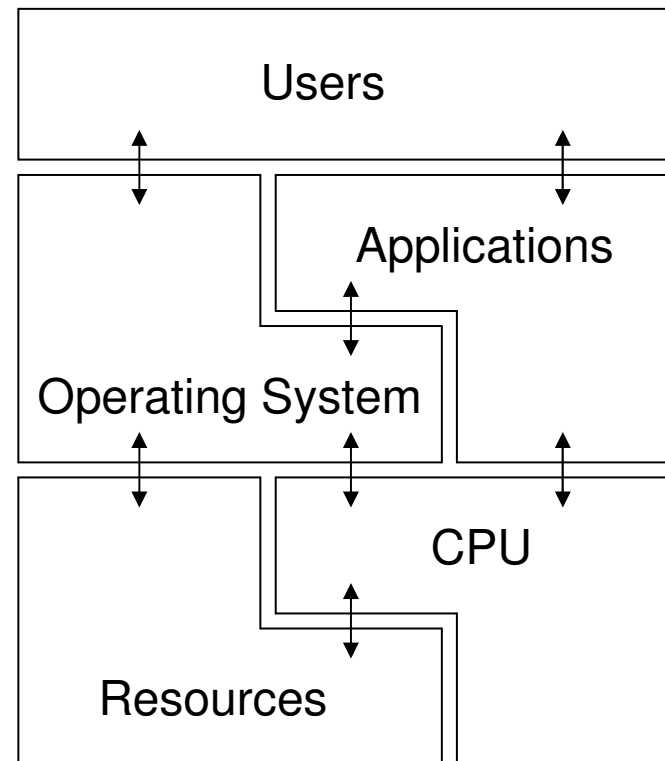
User-Computer Interaction

- `Alice` wishes to use a computer to edit the file `example.txt` using a word processor.
  - `Alice` logs on to the computer
  - `Alice` runs word processor program
    - CPU loads word processor program into main memory
  - `Alice` **opens** `example.txt`
    - CPU loads `example.txt` into main memory
    - Operating system renders `example.txt` on screen
  - `Alice` **edits** `example.txt`

Security Issues

- Is `Alice` allowed to use the computer?
- Is `Alice` allowed to run a word processor?
- Is `Alice` allowed to edit `example.txt`?
- Can the word processor program overwrite memory locations containing operating system programs or data?
- Does loading `example.txt` into main memory overwrite memory locations containing operating system programs or data?
- How do we ensure that the word processor only changes locations of memory containing `example.txt`?
- Is the word processor able to read contents of main memory that it shouldn't?

Computer Security Services

- **Authentication**:
  - Determines whether `Alice` is allowed to use the computer.
- **Authorization**:
  - Determines whether `Alice` is allowed to run a word processor.
  - Determines whether `Alice` is allowed to edit `example.txt`?
- **Memory protection**:
  - Prevents the word processor overwriting or reading memory locations to which it should not have access.

Computer Security Goals

- **Confidentiality**:
  - Unauthorised users and programs cannot read protected information.
    - Control who reads `example.txt`.
    - Control which programs can read main memory.
- **Integrity**:
  - Unauthorised users and programs cannot change protected information.
    - Control who changes `example.txt`.
    - Control which programs can change main memory.
- **Availability**:
  - Authorised users and programs can access protected information.
    - Protect integrity of operating system.
    - Control who deletes files.

Overview of Authentication

- ## What happens when a user tries to log on?
  - Computer system **verifies identity** of user.
  - Computer system provides user with work environment:
    - Command line
    - Desktop
  - Computer system provides user with **security context**.

Overview of Authentication

- ## How does a computer system check the identity of a user?

  - Compares information stored by system with information provided by user.

  - If the two bits of information match user is assumed to be authorised user and is authenticated.

  - Moral of the story is keep your information secret!

    - If `Alice` gives `Bob` her username and password, the computer system cannot distinguish between them.

Overview of Authorisation

- **What happens when a user tries to run a text editor?**

  - The operating system **checks** whether the user is allowed to run the text editor program.

    - Inputs to this decision are the security properties of the program and the security context of the user.

  - If the user is authorised, the machine code for the text editor is loaded into main memory.

Overview of Authorisation

- **How does the operating system check whether a user is authorised?**
  - Each user and resource is associated with security information.
    - This information:
      - is maintained by the operating system
      - must be protected!
  - User information typically includes:
    - Username and password.
    - Groups and roles to which the user belongs.
  - Resource information typically includes which users, roles and groups can access the resource.

Overview of Memory Protection

- **What happens when a program or other data is loaded into main memory?**
  - The operating system **ensures** that the machine code is loaded into a safe area of memory.
    - It does not overwrite memory locations being used by:
      - other processes
      - the operating system
  - Operating system makes use of memory management features provided by hardware.

Overview of Memory Protection

- **What happens when a program tries to access a location in main memory?**
  - The operating system **checks** that the memory address lies within a region that has been allocated to the process.
  - These regions are called **segments**.
  - **Segment registers** in the CPU are used to identify segment descriptors.
  - **Segment descriptors** contain information about segments:
    - Location of segment in main memory.
    - Size of segment.
    - Type of data that is stored in segment.
    - Which access operations are permitted for the segment.

Authentication

- Why is authentication important?
- "*Without identifying and authenticating the user logging on to the system, access to objects cannot be controlled, user rights and abilities cannot be enforced, and accountability cannot be maintained via auditing. For these reasons Windows 95 and Windows 98 can never be considered secure operating systems.*"
  - Windows 2000 Security Technical Reference
- Mandatory log on is a fundamental security requirement:
  - Part of the C2 requirements of the US Trusted Computer Security Criteria (TCSEC)

Authorisation

- Authorisation (access control) assumes the existence of an authentication process.
- The decision to grant an access request made 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 usually identified the user and any security groups to which that user belongs.

What is Access Control?

- Generic term for the process(es) by which a computer system controls the interaction between users and system resources.

- May implement (part of) a specific security policy that may be determined by:
  - organisational requirements
  - statutory requirements (medical records, for example)

- Policy requirements relevant to access control include:
  - confidentiality (restrictions on read access)
  - integrity (restrictions on write access)

Why Use Access Control?

- Prevent authorised users from having unlimited access to system resources.

- Limit access of unauthorised users that manage to impersonate authorised users.

- Access control is not required if access to resources does not need to be constrained.

  - Early stand-alone PCs (DOS, Windows 95) could not (and did not necessarily need to) enforce access control.

A Schematic View

- A user requests access (read, write, print, etc.) to a resource in the computer system.

  - `Alice` tries to open `example.txt` in order to edit the file.

- The reference monitor

  - establishes the validity of the request,

  - and returns a decision either granting or denying access.

Access
Request → Reference Monitor → Decision
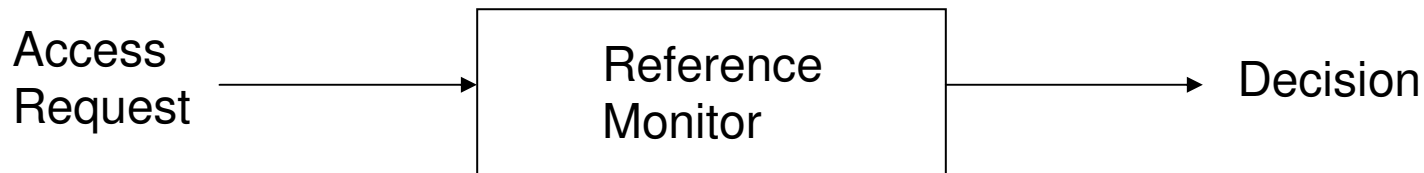
Reference Monitor

- The abstract requirements of a reference monitor are comprised of three fundamental principles:

  - *Completeness*: It must always be invoked and impossible to bypass.

  - *Isolation*: It must be tamper-proof.

  - *Verifiability*: It must be shown to be properly implemented.

Access Models and Information Flow

- **There are two basic modes of interaction between a subject and an object.**
  - Observe
  - Alter

- **Accessing an object can be regarded as initiating a flow of information.**
  - A subject may observe (read) an object:
    - Information flows from object to subject.
  - A subject may alter (write to) an object:
    - Information flows from subject to object.

Execute Access

- Sometimes an object can be used without using either observe or alter mode:
  - Executable files (programs)
  - Directories
  - Cryptographic keys

- The execute access right means different things in different contexts and in different systems.
  - Execute access on a binary executable file grants permission to run the program.
  - Execute access on a Unix directory grants permission to search the contents of the directory.

Least Privilege

- **Principle of Least Privilege**:
  - Assignment of permissions to users such that the user is given no more permission than in necessary to perform his or her job function.
  - Largely an administrative challenge that requires the identification of job functions and their associated permissions.
  - Strict adherence to least privilege requires an individual to have different levels of permission at different times.
  - Granting of excess privilege can potentially be exploited to circumvent protection.

DAC and MAC

- *Discretionary Access Control* (**DAC**):
  - Creators or owners of files assign access rights, and a subject with discretionary access to information can pass that information to another subject.
  - This is insufficient for implementing the document classification scheme used by the military.

- *Mandatory Access Control* (**MAC**):
  - A mandated set of accesses of all objects on the system.
  - Since rules are imposed externally, users cannot give away permissions for object access.

- Neither MAC nor DAC are suitable for commercial requirements.

Authorisation in Unix

- Unix allocates access rights on the basis of file ownership and group membership.

- The administrator defines a number of groups.
  - Typically these groups will be representative of the organisational structure.
  - Each group has a name and a numerical security identifier (GID).

- Each user is associated with:
  - a unique security identifier (UID),
  - one or more groups (identified by GID).

- Each object is associated with an owner (identified by UID) and a group (identified by GID).
  - The owner does not need to be a member of the group.

Authorisation in Unix

- For each object we identify three groups of users
    - `owner` (UID matches OID)
    - `group` (any user associated with a group that matches the GID of the object)
    - `world` (any other authenticated user)
- Each object is associated with an **access mask**:
    - A pattern of 9 bits in 3 groups of 3.
    - Determines which of read, write and execute access are available to each of the three groups.
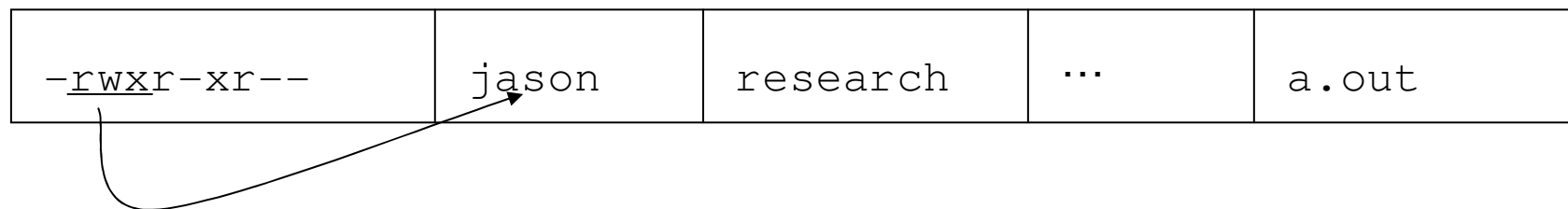
The Unix `ls` command

- Lists the contents of the current working directory including:
    - file permissions
        - `d` indicates that the file is a directory
        - first three permissions indicate those granted to the owner
        - next three permissions indicate those granted to members of the owner group
        - final three permissions indicate those granted to other authenticated users
    - file name, owner and group owner

| Permissions	Owner	Group	…	Name
drwxr-xr-x	jason	research	…	Research
-rw-r-----	jason	research	…	test.txt
-rwxr-xr--	jason	research	…	a.out

Making a Decision – I

- A user attempts to run `a.out` from the command line.

  - Running a program requires execute access (`x`).

- The Unix reference monitor checks the identity of the user (established when the user logged on).

- If the user is `jason` access is granted.

-rwxr-xr--	jason	research	...	a.out

Making a Decision – II

- A user attempts to run `a.out` from the command line.

- The Unix reference monitor checks the identity of the user (established when the user logged on).

- If the user is `geraint` (who is also a member of the `research` group) access is granted.

| -rwxr-xr--	jason	research	...	a.out
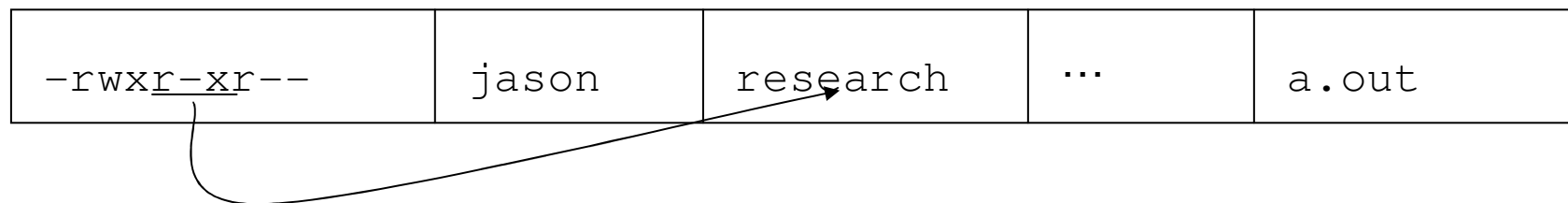
Making a Decision – III

- A user attempts to run `a.out` from the command line.

- The Unix reference monitor checks the identity of the user (established when the user logged on).

- If he user is `pauline` (who is not a member of the research group) access is denied.

`-rwxr-xr--`	`jason`	`research`	...	`a.out`

Making a Decision – IV

- **The decision is based on the most specific group membership with the user**
- **A user attempts to run `a.out` from the command line.**
  - If the user is `jason` access is denied:
    - even if he belongs to the `research` group,
    - despite the fact that he is the owner of the file.
  - If the user is `geraint` access is granted (as before).

-rw-r-xr--	jason	research	...	a.out
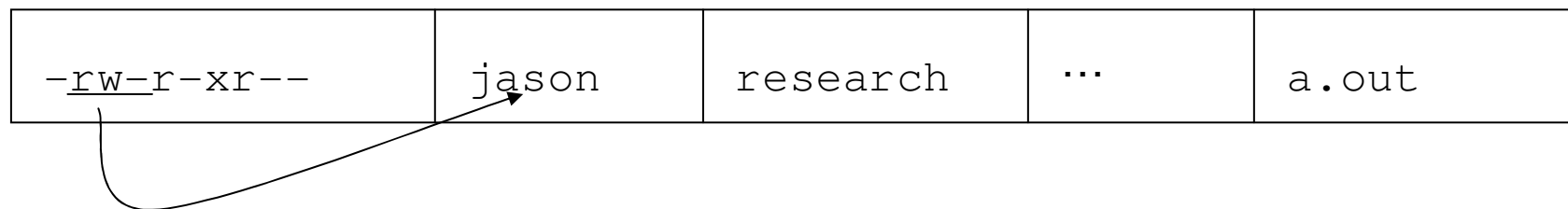
Exercise

- **For each of the three users** `jason,` `geraint` **and**
  `pauline`
  - determine whether he or she can read or write the file
    `rbac.txt`
  - determine whether he or she can read or write the file
    `myprog.exe`
  - (you may assume that `jason` is a member of the
    `research` **group**)

`-rw-r-----`	`jason`	`research`	`...`	`rbac.txt`
`-rwxr-x--x`	`geraint`	`research`	`...`	`myprog.exe`

What is an Access Control Model?

- *"The model has the ability to **represent abstractly** the elements of computer systems and of security that are relevant to a treatment of classified information stored in a computer system"*
    - Bell-LaPadula, 1976
- A model comprises elements that are used to represent the system such as sets, relations and functions.
- In the context of access control, a model typically describes a reference monitor.

Building a Secure System

- **Identify security requirements**
  - ❑ Low priority users cannot read high priority documents

- **Develop logical model of system**
  - ❑ How do we represent "low" and "high" in abstract terms?
  - ❑ How do we represent "users" and "documents" conceptually?

- **Implement system**
  - ❑ What data structures do we use to store security information about users and documents?
  - ❑ How do we identify users and documents within systems?

Why Are Models Useful?

- Formal results can be deduced from the model that make statements about the security of the system.
    - Specification of security policy.
    - Does the system maintain security policy?
- A model may also generate rules that can provide a blueprint for an implementation.
    - May assist in verifying that an implementation meets requirements.

Terminology

- A **subject** is an active entity in a computer system.

  - User, process, thread, cryptographic key.
  - We will assume that a subject is synonymous with a user.

- An **object** is a passive entity or resource in a computer system.

  - File, directory, printer, socket.

The Access Control Matrix

- **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.
  - Foundations of many theoretical security models.

Objects / Subjects	trash	a.out	allfiles.txt
jason	{r,w}	{r,w,x}	{r,w}
geraint		{r,x}	{r}

The Access Control Matrix

- A request s modelled as a triple (s, o, a):
  - the subject s wants to access objects o where a is an access right.

- A request is granted (by the reference monitor) if:
  - a belongs to the access matrix entry corresponding to subject s and object o.

The Access Control Matrix

- **The request (`jason, allfiles.txt, w`) is granted**
  - `w` belongs to the matrix entry

Objects / Subjects	trash	a.out	allfiles.txt
jason	{r,w}	{r,w,x}	{r,w}
geraint		{r,x}	{r}

The Access Control Matrix

- ## The request (`geraint, allfiles.txt, w`) is denied
  - `w` does not belong to the matrix entry

Subjects \ Objects	trash	a.out	allfiles.txt
jason	{r,w}	{r,w,x}	{r,w}
geraint		{r,x}	{r}

Disadvantages

- ## Not suitable for direct implementation:

  - ❑ The matrix is likely to be extremely sparse and therefore implementation is inefficient.

  - ❑ Management of the matrix is likely to be extremely difficult if there are 0000s of files and 00s of users (resulting in 000000s of matrix entries).

Access Control Lists

- An ACL corresponds to a column in the access control matrix

- `[a.out: (jason, {r, w, x}), (geraint, {r, x})]`

- How would a reference monitor that uses ACLs check the validity of the request `(jason, a.out, r)`?

Objects / Subjects	`trash`	`a.out`	`allfiles.txt`
`jason`	`{r,w}`	`{r,w,x}`	`{r,w}`
`geraint`		`{r,x}`	`{r}`

Access Control Lists

- ## Access control lists focus on the objects

  - ❑ Typically implemented at operating system level

  - ❑ Windows NT uses ACLs

- ## Disadvantage

  - ❑ How can we check the access rights of a particular subject efficiently ("before-the-act per-subject review")?

Capability Lists

- A capability list corresponds to a row in the access control matrix

- [`jason:` (`trash,` {`r,w`}), (`a.out,` {`r,w,x`}), (`allfiles.txt,` {`r,w`})]

- How would a reference monitor check the validity of the request (`jason,a.out,r`)?

Objects / Subjects	trash	a.out	allfiles.txt
jason	{r,w}	{r,w,x}	{r,w}
geraint		{r,x}	{r}

Capability Lists

- **Capability lists focus on the subjects.**
    - Typically implemented in services and application software.
    - Database applications often use capability lists to implement fine-grained access to tables and queries.
    - Renewed interest in capability-based access control for distributed systems.
- **Disadvantage**
    - How can we check which subjects can access a given object ("before-the-act per-object review")?

Role-Based Access Control (RBAC)

- **Role-Based Access Control is conceptually simple:**
  - Access to computer system objects is based on a user's role in an organisation.
  - Roles with different privileges and responsibilities have long been recognised in business organisations.
  - Commercial computer applications dating back to at least the 1970s implemented limited forms of access constraints based on the user's role within the organisation.

Roles – I

- The central concept in RBAC is that of a *role*:
  - Acts as a bridge between users and objects.
  - Reduces complexity of configuring authorisation policy.
- Early formalisation of RBAC had three basic rules: *Role Assignment*; *Role Authorisation*; *Transaction Authorisation*.
- **Role Assignment**:
  - A subject can execute a transaction only if the subject selected, or been assigned to, a role.
  - Login is not considered a transaction, but all other activities on the system, are conducted through transactions.
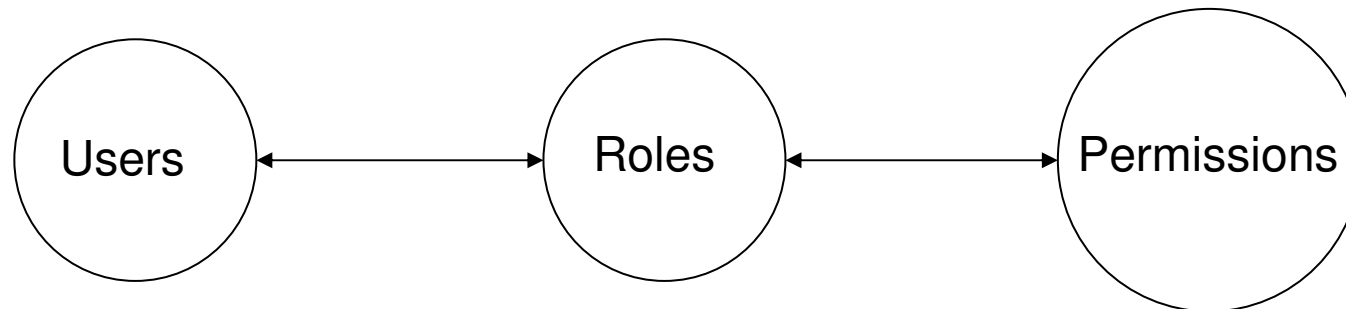
Roles – II

- **Role Authorisation**:
  - A subject's active role must be authorised for the subject.
  - With the rule 1 users can take on only roles for which they are authorised.

- **Transaction Authorisation**:
  - A subject can execute a transaction only if the transaction is authorised for the subject's active role.
  - In concert with rules 1 and 2, this rule ensures that users can only execute transactions for which they are authorised.

Roles – III

- A key feature of the model is that all access is through roles.

- A role is essentially a collection of permissions.

- All users receive permissions only through roles to which they are assigned.

Users ←——→ Roles ←——→ Permissions

Roles – IV

- Within an organisation roles are relatively stable.

- On the other hand, users and permissions are both numerous and may change rapidly.

- Controlling all access through roles therefore simplifies the management and review of access controls.

- RBAC makes the *per-subject review* question much more easy to answer.

Roles v Groups – I

- As we saw previously, *groups* are used in ACLs for allowing shared access to resources.

- At a basic level, roles can be considered to be equivalent to groups:
  - a role can represent a collection of users;
  - a user can be a member of multiple roles.

- Also:
  - a single privilege can be associated with one or more groups or roles;
  - a single group or role can be associated with one or more privileges.

Roles v Groups – II

- At this level of discussion a role is not unlike that of a group within the context of an ACL.

- However, roles and groups have different semantics in access control models and different usage in their implementation.

- For example:
  - within some UNIX environments, only one group can be associated with a particular file;
  - other operating system environments allow multiple groups to coexist among the access control entries of a file;
  - while other access control systems prevent a user from being a member of more than one group at a time.

Roles v Groups – III

- Regardless of implementation, a role will always exhibit the properties defined by the RBAC model.
- A group may or may not exhibit these properties.
- For example, the properties of an RBAC role allow for the naming of many-to-many relations among users and permissions.
- For a group to meet this same property, the group structure must not place a restriction on the number of:
  - Groups that can be created.
  - Users that could become a member of any group.
  - Groups to which a user can have simultaneous membership.
  - Individual groups that can be included within access control entries of a single access control list.

Roles v Groups – IV

- Because RBAC is a model and not a mechanism, it may be implemented within many types of systems to include network and enterprise management systems.

- For instance, assigning a user to a role may grant the user a set of permissions within and across multiple operating systems and applications.

- From the enterprise perspective, it may be far more efficient to manage user permissions through global roles than through the individual groups of potentially many operating systems and applications.

Core RBAC

- Core RBAC recognises five administrative elements:
  - **Users**
  - **Roles**
  - **Permissions**, where permissions are composed of **Operations** applied to **Objects**.

- The core aspect of this is the many-to-many relationship between users, roles and permissions.

- For example, a single user can be associated with one or more roles, and a single role can have one or more user members.

Administrative Support

- With RBAC, users are not granted permissions to perform operations on an individual basis.

- Instead permissions are assigned to their roles.

- Role associations with new permissions can be established, while old permissions can be deleted as organisational functions change.

- This is considered advantageous to the practice of "cloning":

    - Duplication of permissions of a second user who performs a similar function to that of the first user.

- Another advantage of RBAC is that access requirements can be set at the same level as typical business processes in the enterprise.

Permissions – I

- Permissions can be thought of as the binding between computer operations and resource objects.

- One can consider permissions to represent an atomic unit of work within a computing environment.

- Permissions that are assigned to roles reflect policy decisions on the part of the organisation.

- These permission assignments can be detailed in terms of both *granularity of method* and *granularity of access*.

Permissions – II

- **In order to highlight the difference in granularity of method, consider the following example:**
  - In a bank, there are different access needs for a teller and an accounting supervisor:
    - A teller role needs to be able to perform a savings deposit operation – which requires read and write access to specific fields in the savings file.
    - An account supervisor that is allowed to perform correction operations – which also requires read and write operations to the same fields, but the supervisor should not be able to initiate withdrawals or deposits.

Permissions – III

- **In order to highlight the importance of granularity of access:**
  - A pharmacist might need to access a patient's record to check for interactions between medications and add notes on the medication.
  - However, they should not be able to read or alter other parts of the patient record.
- **The type of operations and the objects that RBAC controls are dependent on the type of system in which it will be implemented, e.g.:**
  - within an O/S, operations might include read, write, and execute;
  - within a DBMS, operations might include insert, delete, append, and update.

Permissions – IV

- This now gives us a slightly different view of our earlier diagram:
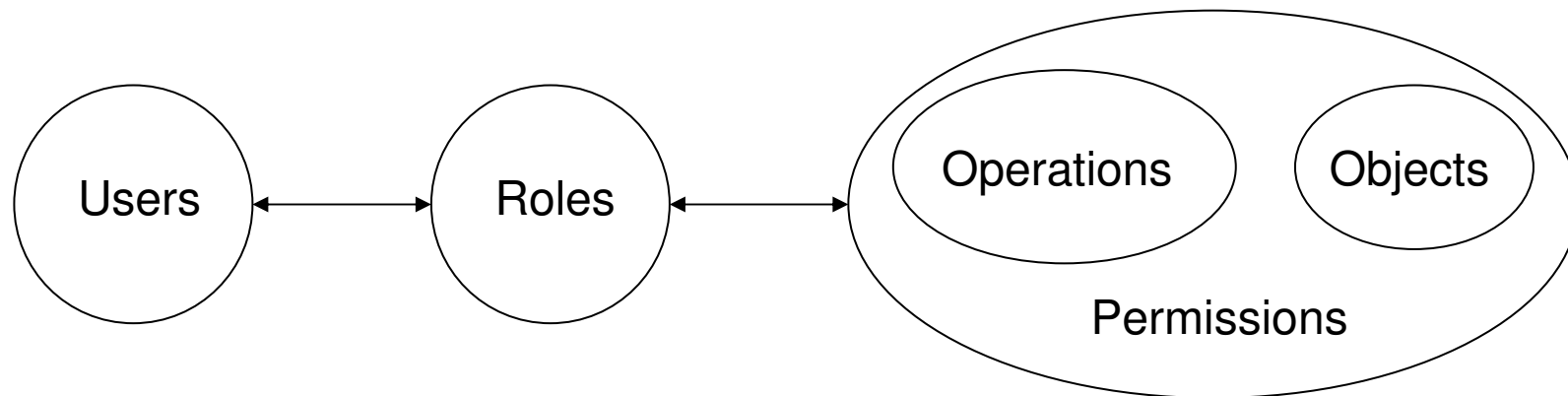
Role Activation – I

- In applying a dynamic security policy to a computing system, we speak of *subjects*.

  - Subjects are active entities whose access to roles, operations, and objects must be controlled.

  - A subject acting on a user's behalf carries out all the requests of a user.

  - Each subject is uniquely referenced by an identifier, which is used to determine whether the subject is authorised for a role and can become active in that role.

  - A user may be associated with multiple subjects at any moment in time.

  - Each subject may have a different combination of active roles.

Role Activation – II

- ## This feature of RBAC supports the principle of least privilege:

  - Any user that is assigned to multiple roles may activate any subset of these roles to suit his or her tasks.

- ## Properties of RBAC ensure that:

  - the active roles of a subject are a subset of the roles that are assigned to the subject's user;

  - and that the active roles of a subject are applied in the performance of object access checks.

Dynamic Mappings and Static Relations
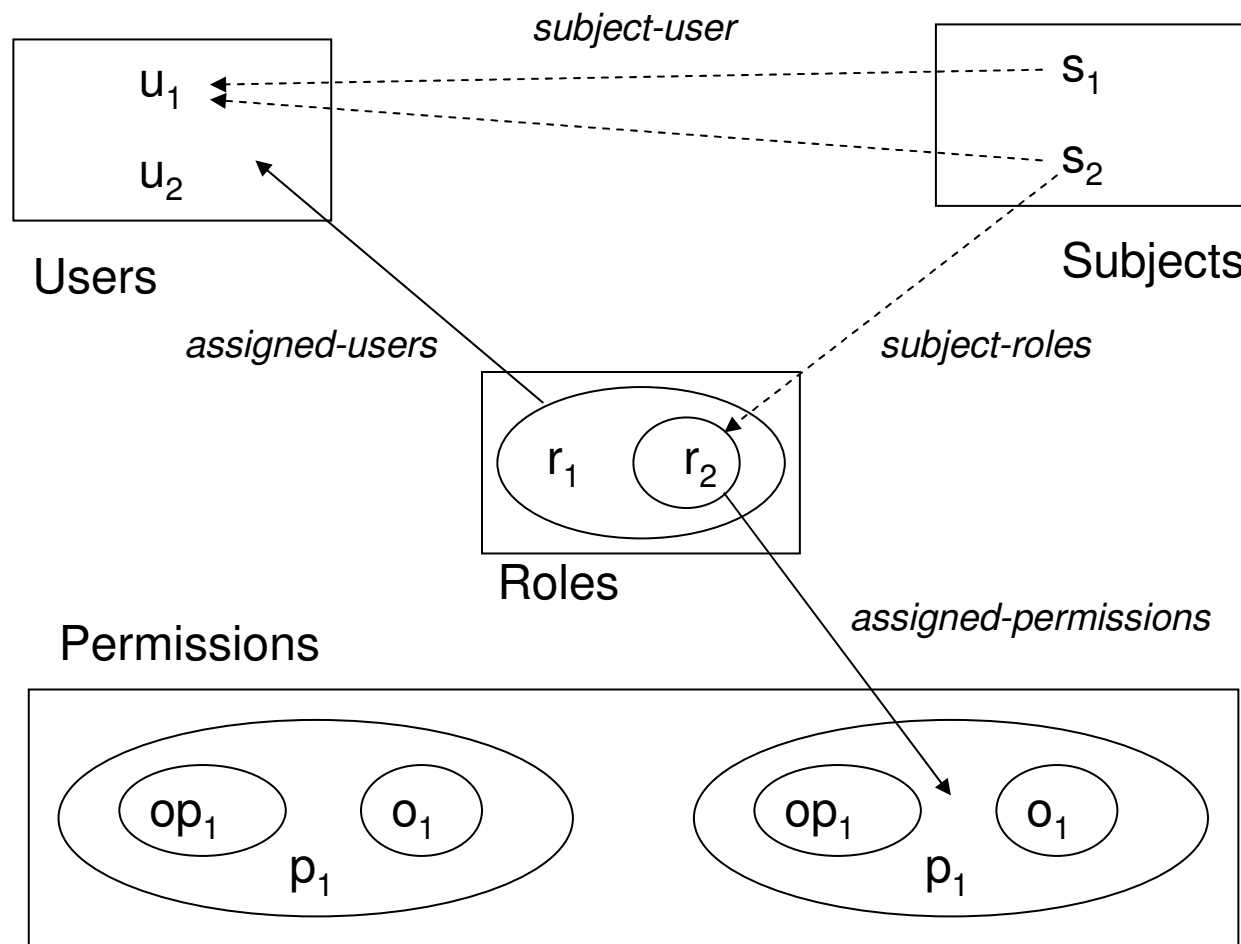
Enterprise and System Views

- *Privileges* are system specific, and permissions are mapped into privileges.

- Each system supports its own class of operations, and has its own class of resources.

- Although privileges are system-specific, *users* and *roles* can take on a common meaning across multiple systems.

- In general, users, roles, and permissions can be treated as global entities, while privileges that are ultimately assigned to a role are specific to local computing environments.

Role Hierarchies – I

- **Role hierarchies allow for:**

  - a more structured description of roles within an organisation;

  - easier administration of roles in relation to the privilege requirements of an organisation's tasks and processes.

- **These enhanced capabilities are a consequence of inheritance or containment relationships used to define role hierarchies:**

  - By virtue of a role's relative position in a role hierarchy, the permissions that are assigned to the role are known to contain, or be contained by, other roles in the hierarchy.
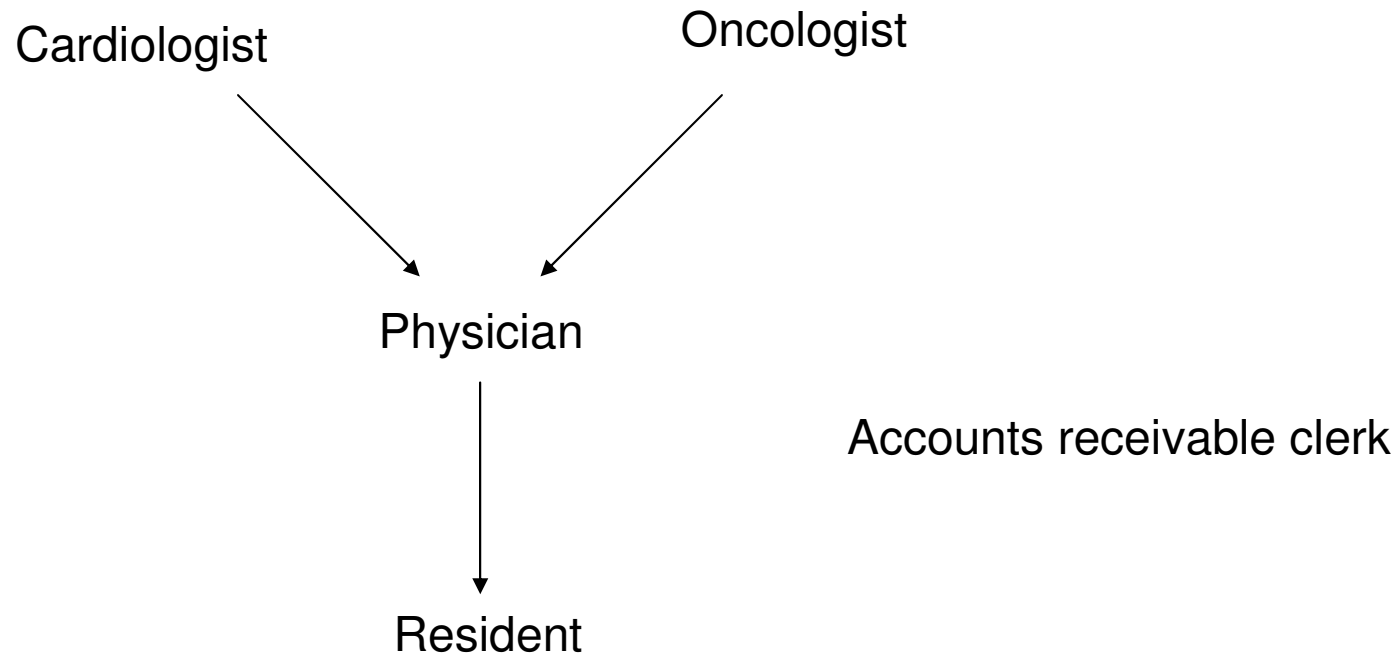
Role Hierarchies – II

- **Role inheritance relation**:
  - If a role *A* inherits role *B*, it means that all of *B*'s permissions are available via role *A*.
  - In other words, *B*'s permissions are a proper subset of the permissions of *A*.
- This allows administrators to better formulate access policies in terms of organisation specific functions and business structures.
- For instance:
  - The permissions that are authorised for a role can be decomposed into lower-level roles representing the functions, duties, and tasks that comprise the role.
  - Once these lower-level roles are created, they may be reused in the creation of higher-level roles.

Role Hierarchies – III

- **The motivation for role hierarchies is that individual roles within an organisation have overlapping functions:**
  - Users belonging to different roles may be authorised for common permissions.
  - In extreme circumstances, there are general functions performed by all or most users within an organisation.
  - In the absence of role hierarchies, it is inefficient and administratively cumbersome to specify these general permissions repeatedly for a large number of roles.
  - With a role hierarchy, the collection of permissions that comprise a job function can be defined by multiple subordinate roles, each of which may be reused.

Role Hierarchies – An Example

Cardiologist

Oncologist

Physician

Accounts receivable clerk

Resident

Role Hierarchies – An Example

- **In this example:**
  - The roles *cardiologist* and *oncologist* inherit the roles *physician* and *resident*.
  - That is, any user that is assigned to the *cardiologist* is authorised for the permissions that are assigned to the role *cardiologist* and authorised for the permissions that are assigned to the roles *physician* and *resident*.
  - Not all roles have to be hierarchically related.
  - The roles *cardiologist*, *oncologist*, and *accounts receivable clerk* are not hierarchically related.

Comparing RBAC to DAC – I

- For many enterprises within industry and civilian government, end users do not "own" the information to which they are allowed access, as is assumed by DAC policies.

- For these organisations, the corporation or agency is the actual "owner" of system objects.

- It may not be appropriate to allow users to give away access rights to the objects.

- With RBAC, access decisions are based on the roles individual users have as part of an organisation.

Comparing RBAC to DAC – II

- An RBAC policy is based on the functions or the actions that a user is allowed to perform within the context of an organisation.

- The users cannot normally pass their permissions on to others at their discretion.

- Security policies often support higher level organisational objectives:

  - e.g. the laws for medical privacy.

- Supporting such policies requires centrally controlled and maintained access rights.

Comparing RBAC to MAC

- **Due to the centralised control, RBAC is sometimes compared to MAC.**

- **However, RBAC is different from MAC:**

    - Traditional MAC supports military policies for unauthorised access to classified information.

    - Particularly to the protection of the *confidentiality* of sensitive information.

    - RBAC is usually put in place to enforce *confidentiality*, or *integrity*, or both.

Efficiency of RBAC

- RBAC has the potential to offer greater administrative efficiency for:
  - giving permissions to new users;
  - reviewing and removing old privileges;
  - changes in a user's job assignment;
  - removal of privileges for leaving employees.
- There is usually a direct relationship between the cost of administration and the number of associations that must be managed.
- The larger the number of associations, the costlier and more error-prone access control administration.
- In most organisations RBAC reduces the number of associations that must be managed.

Clark-Wilson

- In 1987, Clark and Wilson documented the differences between commercial and military security requirements, arguing that the primary concern for most commercial applications is *integrity*, rather than *confidentiality*.

- For example, the 600-year-old principle of double-entry bookkeeping helps to ensure the accuracy and integrity of accounts.

- They proposed two principles as most important in ensuring information integrity:
  - *Well-formed transactions*
  - *Separation of duty*

Well-Formed Transactions

- *Well-Formed Transactions*:
 - Constrain the user to change data only in authorised ways.
  - Thus ensuring that all data that starts in a valid state will remain so after the execution of a transaction.
  - e.g.:
    - A bank teller cannot modify an arbitrary part of a customer record, only those data fields that are incorporated into the particular transaction being run, such as a savings deposit or withdrawal.

Separation of Duty

- *Separation of Duty*:
  - Prevents any one individual possessing all the privileges required to carry out a sensitive transaction.
  - This goal is achieved indirectly by separating all operations into multiple subparts and requiring that a different person perform each subpart.
  - e.g.:
    - A division manager, for example, can request an expenditure, but another person must approve it, and a third audits the completed transaction to ensure that fraud has not occurred.

Access Control Triple – I

- The basic unit of access control in the Clark-Wilson model is the *access control triple*:
  - *user*;
  - *transformation procedure*;
  - *constrained data item*.

User		Operation		Object
	↔		↔	
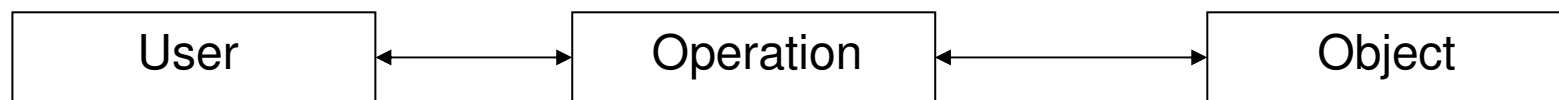
Access Control Triple – II

- A transformation procedure (**TP**) is a transaction, and a constrained data item (**CDI**) is one for which integrity must be preserved.

- Unconstrained data items (**UDIs**) are those that are not protected by the integrity model.

- Integrity Verification Procedures (**IVPs**) ensure that a data item is in a valid state.

Integrity Rules – I

- Clark and Wilson proposed nine rules to ensure the integrity of data.

- Certification Rules:

    1. IVPs must ensure that all CDIs are in a valid state.

    2. Each TP must be certified to be valid (that is, valid CDIs must be transformed into valid CDIs by the TP) and must be certified to access a specific set of CDIs.

    3. The access rules must satisfy any separation of duty requirements.

    4. All TPs must write to an append-only log.

    5. Any TP that takes a UDI as input must either convert it into a CDI or fail.

Integrity Rules – II

- **Enforcement Rules:**

  1. The system must maintain and protect the list of CDIs that a TP is certified to access.

  2. The system must maintain and protect the list of TPs that a user can execute.

  3. The system must authenticate each user requesting the use of a TP.

  4. Only a certifier of a TP can change the certification. The certifier must not be allowed to execute the TP.

Windows 2000 Authentication

- The `Winlogon` process initiates the authentication procedure by intercepting the user's **secure attention sequence** (`Ctrl+Alt+Del`).

- The user enters a username and password and authenticates to a Windows 2000 domain using the Kerberos protocol.

- A successful authentication process results in an **access token** being returned to the `Winlogon` process.

- The access token is bound to the authenticated user:
  - Each program that is run by the user is associated with the user's access token.

Windows 2000 Subjects

- Each user and group has a security identifier (SID).

- When a user logs on and authenticates successfully to a Windows 2000 system, the local security authority creates an **access token**.

  - The access token can be thought of as the subject in access control decisions.

- An access token includes the following information:

  - The SID of the authenticated user.

  - The SIDs of the groups to which the user belongs.

Windows 2000 Objects
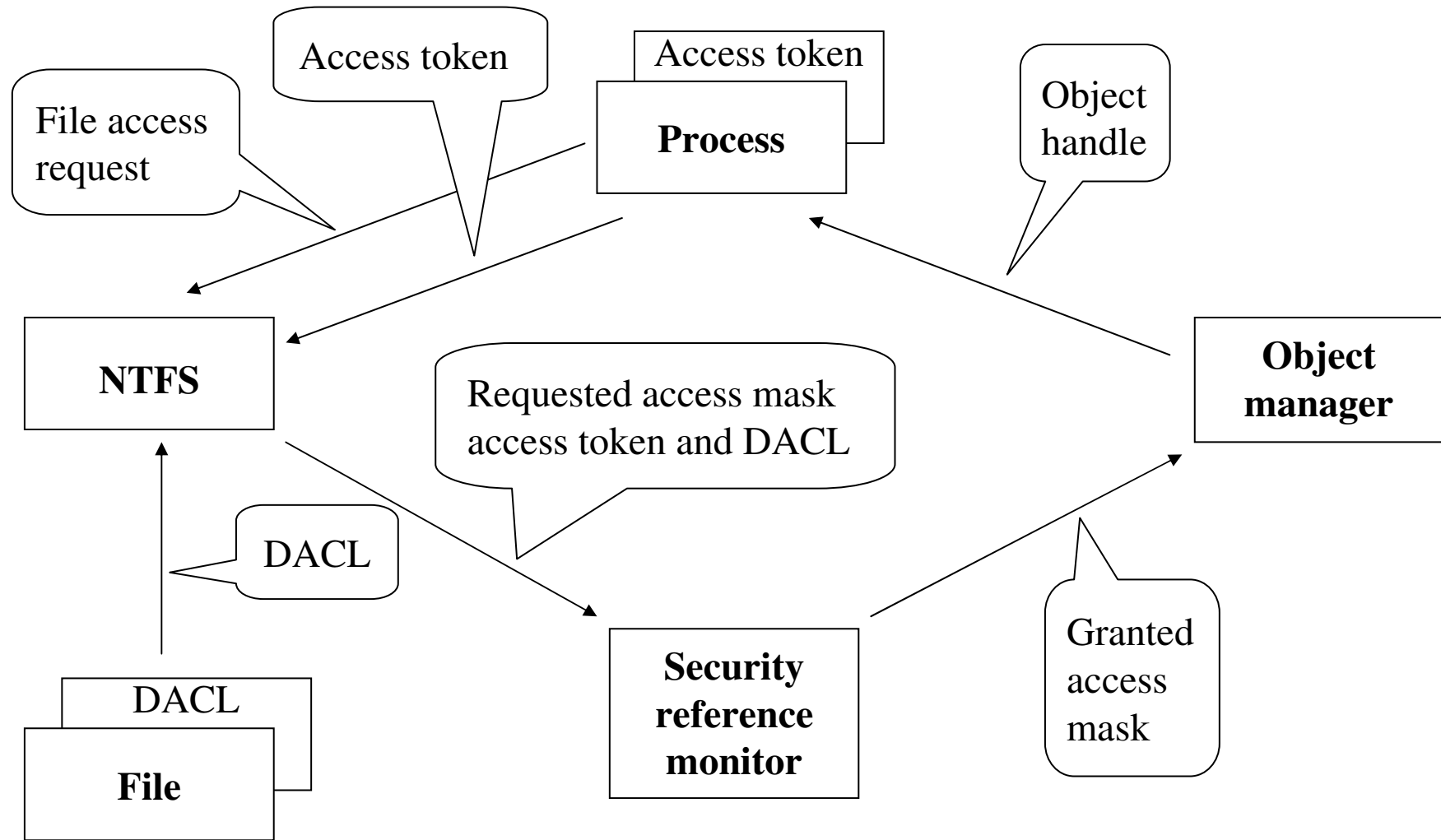
- Each object has a **security descriptor**.
- Each object has a **type** which determines the access rights associated with the object.
  - File access rights include read, write, etc.
- The security descriptor contains a discretionary access control list (DACL).
  - The DACL is a list of access control entries (ACEs).
  - Each ACE contains a security identifier and an access mask.

Access Masks

- Windows 2000 recognises several different access rights and encodes them in a 32-bit **access mask**.
- Access masks are used in a variety of contexts.
  - An access request is handled as an access mask.
    - A requested permission has the corresponding bit set.
  - The security reference monitor constructs a granted access mask.
    - A granted permission has the corresponding bit set.
  - Each access control entry contains an access mask.
    - A permission granted by the ACE has the corresponding bit set.
  - An object handle contains an access mask.

Windows 2000 Authorisation

Access token

Access token

Process

Object handle

File access request

NTFS

Requested access mask access token and DACL

Object manager

DACL

DACL

File

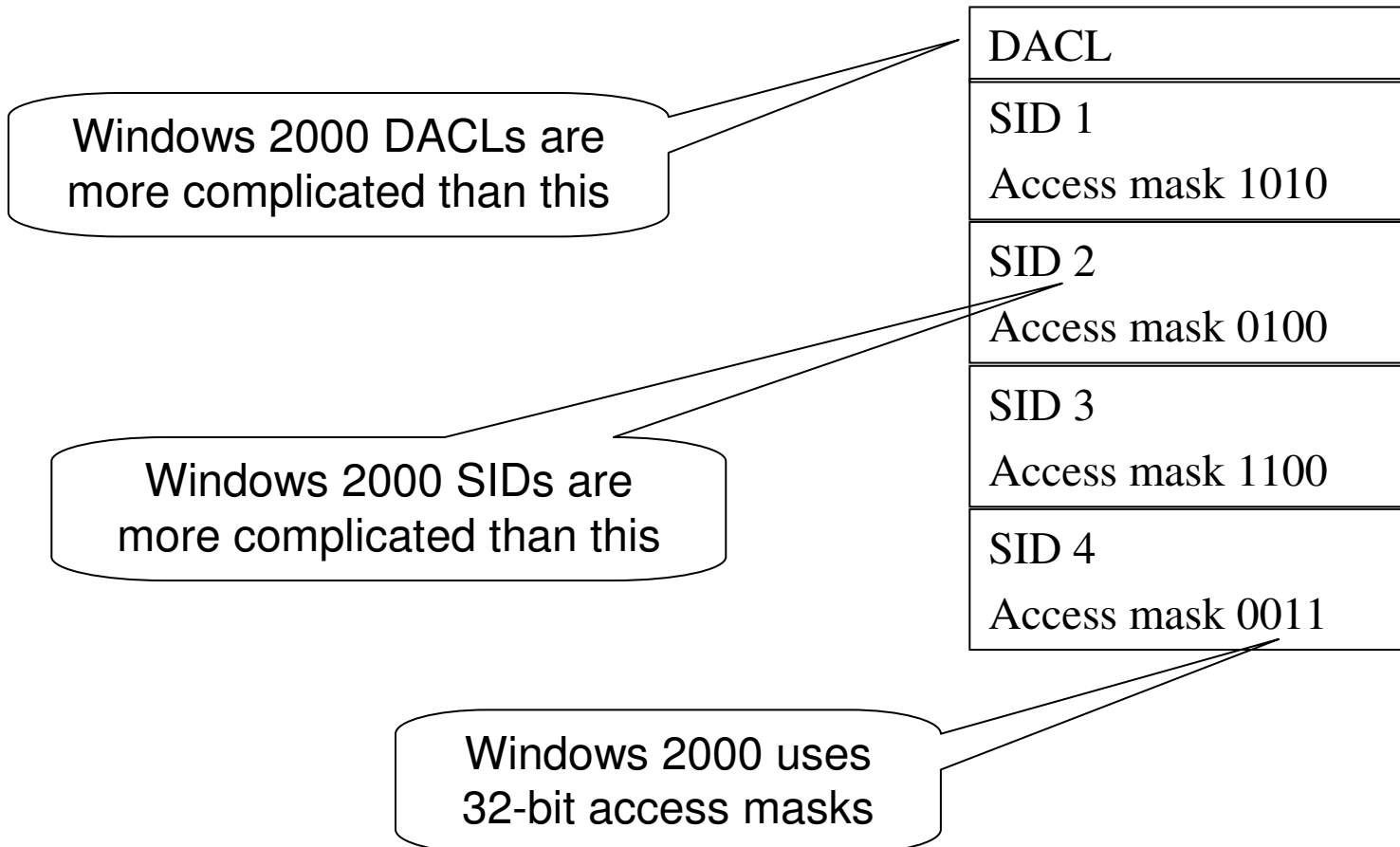**Security reference monitor**

Granted access mask

Windows 2000 Authorisation

- When a process makes a file access request the access token is presented to the NTFS service.
- The NTFS service forwards the request (presented as a requested access mask), the access token and the file's access control list (DACL) to the security reference monitor (SRM).
- The SRM builds a granted access mask:
  - If the granted access mask equals the requested mask then access is granted.
  - If the granted access mask is not equal to the requested access mask then access is denied.
- If access is granted the object manager returns an object handle to the process.
  - The object handle contains a copy of the granted access mask.
  - A process can only access a file if it has an object handle for the file.

Building the Granted Access Mask

- **The SRM examines each ACE.**

- **If the access token contains an SID that matches the ACE SID then**

  - The matching entries in the requested access mask and the ACE access mask are added to the granted access mask.

- **When the end of the DACL is reached the requested and granted access masks are compared.**

  - If they are equal then access is granted.

  - If they are not equal then access is denied.

Simplified Walk-Through

Windows 2000 DACLs are more complicated than this

Windows 2000 SIDs are more complicated than this

Windows 2000 uses 32-bit access masks

DACL
SID 1
Access mask 1010
SID 2
Access mask 0100
SID 3
Access mask 1100
SID 4
Access mask 0011

Building the Granted Access Mask

- **Request access mask 0110**

- **Initial granted access mask 0000**

- **Access token contains SIDs 1, 4 and 5**

- **Access denied**

Granted access mask	DACL
0010	SID 1 Access mask 1010
0010	SID 2 Access mask 0100
0010	SID 3 Access mask 1100
0010	SID 4 Access mask 0011

Building the Granted Access Mask

- Request access mask 0011

- Initial granted access mask 0000

- Access token contains SIDs 1, 4 and 5

- Access granted

Granted access mask	DACL
0010	SID 1 Access mask 1010
0010	SID 2 Access mask 0100
0010	SID 3 Access mask 1100
0011	SID 4 Access mask 0011

Exercise

- Using the DACL in the previous slide and given a user with an access token containing SIDs 1, 2 and 3, would the following requests be granted?
  - 0011
  - 0101
  - 0110
  - 1110

Additional Authorisation Features

- **Impersonation tokens**
  - Used when a server acts on behalf of a client process.

- **DACLs may contain**
  - Access-denied ACEs
  - Inherit-only ACEs

- **Access tokens may contain**
  - Restricted SIDs
  - Disabled SIDs
  - Deny-only SIDs
  - Privileges

Other Access Control Models

- **Bell-LaPadula**
  - Designed by the military to enforce confidentiality.
  - Subjects and Objects are assigned to security levels.
  - Subjects are limited to which objects they can access.
  - Canonical means of implementing Mandatory Access Control.

- **Chinese Wall model**
  - Main consideration is to prevent conflicts of interest arising.
  - Uses "history matrix" to record users' actions of time.
  - A user may not access a file belonging to company A (Amoco) if she has previously accessed a file belonging to a company B (BP) who is a competitor of company A.

Things to Remember

- **Security issues:**
  - Authentication
  - Authorisation
  - Memory protection

- **Authorisation:**
  - Control of access to computer resources.
  - Compares user's security profile with that of the target object.
  - Importance of access control models for reasoning about authorisation and security policies.

- **Computer security can be compromised in many ways!**

Further Reading

- D.E. Bell and L. LaPadula. *Secure Computer Systems: Mathematical Foundations*, Mitre Corporation Technical Report MTR-2547 (1973)

- J. McLean. Security models. In *Encyclopedia of Software Engineering*, *Wiley*, 1994.

- D.D. Clark and D.R. Wilson. A comparison of military and commercial computer security policies, *Proceedings of IEEE Symposium on Security and Privacy*, 184-194 (1987).

- D.Brewer and M. Nash. The Chinese Wall Security Policy. In *Proceedings of IEEE Symposium on Security and Privacy*, 1989.

- R.S. Sandhu et al. Role-based access control models, IEEE Computer 29(6), 38-47 (1996)

- A Guide to Understanding Identification and Authentication in Trusted Systems
  - http://www.radium.ncsc.mil/tpep/library/rainbow/NCSC-TG-017.pdf

- US Military Glossary of Computer Terms
  - http://www.radium.ncsc.mil/tpep/library/rainbow/NCSC-TG-004.pdf

On-Line Resources

- **CERT (Computer Emergency Response Team)**
  - http://www.cert.org/
- **SysAdmin, Audit, Network, Security (SANS) Institute**
  - http://www.sans.org/
- **Security Tracker**
  - http://www.secuirtytracker.com/
- **Overview of authentication in Windows 2000**
  - http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/windows2000serv/reskit/distsys/part2/dsgch12.asp
- **Overview of access control in Windows 2000**
  - http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/windows2000serv/reskit/distsys/part2/dsgch12.asp

Acknowledgements

- Slides based on:
  - Original lecture notes by Jason Crampton
  - Chapters 1-3 of *Role-Based Access Control*, David F. Ferraiolo, D. Richard Kuhn and Ramaswamy Chandramouli, Artech House.

Conclusions

- **After today's lecture you should:**
  - Be able to identify how the **reference monitor** is a simple and powerful abstraction for providing authorisation in a computer system.
  - Be able to describe how each of the following **Access Control Models** implements authorisation in a computer system:
    - *Access Control Matrix*
    - *Role-Based Access Control*
    - *Clark-Wilson*
  - Be able to provide a high-level description of how **Windows 2000** implements authorisation.