Module I: Access Control

- Introduction to Access Control
- Purpose and fundamentals of access control
- Brief history
- Policies of Access Control, Models of Access Control and Mechanisms
- Discretionary Access Control (DAC)
- Non- Discretionary Access Control/Mandatory Access Control (MAC)
- Capabilities and Limitations of Access Control Mechanisms: Access Control List (ACL) and Limitations, Capability List and Limitations.

Introduction to Access Control

- Access control or authorization, in its broadest sense has existed as a concept for as long as humans have had assets worth protecting.
- In today's information technology, authorization is concerned with the ways in which users can access resources in the computer system, or informally speaking, with "who can do what."
- Access control is arguably most fundamental and most pervasive security mechanism in use today. Access control shows up in virtually all systems and imposes great architectural and administrative challenges at all levels of enterprise computing.
- Access control can take many forms. In addition to determining whether a user has rights to use a resource, the access control system may constrain when and how the resource may be used.

Purpose and fundamentals of access control

- To gain a better understanding of the purpose of access control, it is worth reviewing the risks to information systems. Information security risks can be broadly categorized into the following three types, confidentiality, integrity, and availability.
- **Confidentiality** refers to the need to keep information secure and private. This category may include anything from state secrets to confidential memoranda, financial information, and security information such as passwords.
- **Integrity** refers to the concept of protecting information from being improperly altered or modified by unauthorized users. For example, most users want to ensure that bank account numbers used by financial software cannot be changed by anyone else and that only the user or an authorized security administrator can change passwords.
- **Availability** refers to the notion that information is available for use when needed. Attacks that attempt to overload corporate Web servers, widely reported in the popular press, are attacks on availability.

Authorization versus authentication

- Authorization and authentication are fundamental to access control which are different but often confused.
- Authentication is the process of determining that a user's claimed identity is legitimate. Every computer user is familiar with passwords, the most common form of authentication. If Alice logs in as alice46 and then provides the correct password for user identification (ID) alice46, she has authenticated herself to the system. Less common forms of authentication include biometrics (e.g., fingerprint readers) and smart cards.

Authentication is based on one or more of the following factors:

- Something you know, such as the password, personal identification number (PIN), or lock combination;
- Something you have, such as a smart card, automatic teller machine (ATM) card, or key;
- Something you are, or a physical characteristic, such as a fingerprint or retinal pattern, or a facial characteristic.
- Clearly, authentication is normally stronger if two or more factors are used.
- While authentication is a process of determining who you are, Authorization determines what you are allowed to do. Authorization refers to a yes or no decision as to whether a user is granted access to a system resource.
- An information system must maintain some relationship between user IDs and system resources, possibly by attaching a list of authorized users to resources, or by storing a list of accessible resources with each user ID.

- Note that authorization necessarily depends on proper authentication. If the system cannot be certain of a user's identity, there is no valid way of determining if the user should be granted access.

Users, subjects, objects, operations, and permissions

- Almost any access control model can be stated formally using the notions of users, subjects, objects, operations, and permissions, and the relationships between these entities.
- The term **user** refers to people who interface with the computer system.
- In many designs, it is possible for a single user to have multiple login IDs, and these IDs may be simultaneously active. Authentication mechanisms make it possible to match the multiple IDs to a single human user, however.
- An instance of a user's dialog with a system is called a session. A computer process acting on behalf of a user is referred to as a **subject**. Note that in reality, all of a user's actions on a computer system are performed through some program running on the computer. A user may have multiple subjects in operation, even if the user has only one login and one session.
- For example, an e-mail system may be operating in the background, fetching e-mail from a server periodically, while the user operates a Web browser. Each of the user's programs is a subject, and each program's accesses will be checked to ensure that they are permitted for the user who invoked the program.
- An **object** can be any resource accessible on a computer system, including files, peripherals such as printers, databases, and fine-grained entities such as individual fields in database records. Objects are traditionally viewed as passive entities that contain or receive information, although even early access control models included the possibility of treating programs, printers, or other active entities as objects.
- An **operation** is an active process invoked by a subject. Early access control models that were concerned strictly with information flow (i.e., readandwrite access) applied the term subject to all active processes, but RBAC models require a distinction between subject and operation.
- For example, when an ATM user enters a card and correct PIN, the control program operating on the user's behalf is a subject, but the subject can initiate more than one operation—deposit, withdrawal, balance inquiry, or others.
- **Permissions** (or privileges) are authorizations to perform some action on the system. As used in this book, and in most computer security literature, the term permission refers to some combination of object and operation. A particular operation used on two different objects represents two distinct permissions, and similarly, two different operations applied to a single object represent two distinct permissions.
- For example, a bank teller may have permissions to execute debit and credit operations on customer records, through transactions, while an accountant may execute debit and credit operations on the general ledger, which consolidates the bank's accounting data.

Least privilege

- Least privilege is the time-honored administrative practice of selectively assigning permission to users such that the user is given no more permission than is necessary to perform his or her job function.
- The principle of least privilege avoids the problem of an individual having the ability to perform unnecessary and potentially harmful actions merely as a side effect of granting the ability to perform desired functions.
- Least privilege provides a rationale for where to install the separation boundaries that are to be provided by the access control mechanism. Ensuring adherence to the principle of least privilege is largely an administrative challenge that requires the

identification of job functions, the specification of the set of permissions required to perform each function, and the restriction of the user to a domain with those privileges and nothing more.

- Strict adherence to least privilege requires an individual to have different levels of permission at different times, depending on the task or function being performed. It must be recognized that in some environments and with some permissions, restricting permission because it is nominally unnecessary may inconvenience the user or place an additional burden on administrators.
- However, granting of excess privilege that potentially can be exploited to circumvent protection, whether for integrity or confidentiality, should be avoided whenever possible. It is also important that permissions not persist beyond the time that they are required for performance of duties.

Brief History

Although security issues had been addressed in some early time-sharing computer systems from the 1960s, the discipline of computer security began to progress rapidly in the early 1970s. At this time large resource-sharing systems were becoming commonplace in government, military, and large commercial organizations.

Access control in the mainframe era

The earliest work in defining a formal, mathematical description of access control is that the formal notions of subject and object and an access matrix that mediated the access of subjects to objects.
An access matrix is a simple conceptual representation in which the (i,j) entry in the matrix specifies the rights that subject i has to object j. An example is shown in Figure 1.1.
Subjects (processes invoked by users) are allowed to access objects such as files or peripherals according to the rights specified in the matrix. For example, user Bob is allowed read and write access to the payroll file, and read access to the accounts receivable and accounts payable file.

	General ledger	Payroll	Accounts receivable	Accounts payable
Alice	R,W		R	R
Bob		R,W	R	R
Charles	R		R	R

Figure 1.1 Access matrix.

A RAND Corporation report included the definition of a method to implement multilevel relating to documents classified by a security level, such as confidential, secret, or top-secret access control on a resource-sharing system, with separate considerations for local access and remote access where password-based authorization would be required.
This also discussed the basic requirements for controlling access to information based on a user's clearance level and the classification level of files stored on the system.
Bell and LaPadula formalized military access control rules into a mathematical model suitable for defining and evaluating computer security systems. As formulated in this model, multilevel secure systems implement the familiar government document classification rule:
Users are only allowed to access information that is classified at or below their own clearance level.
The Bell-LaPadula model was significant because it provided a formal (i.e., mathematical) model of the multilevel security policy, making it possible to analyze properties of the model in detail.

Two basic rules are required in the formal model: the simple security rule and the *-property, commonly known as "no read up" and "no write down."
The simple security rule is obvious: A user with a particular clearance level cannot be allowed to read information above that level (e.g., a user with secret clearance cannot read top-secret documents).
The *-property, which is essentially the reverse of the simple security rule, is required to maintain system security: A user operating at a particular clearance level can write information only at that level or above.
For example, if a user is logged in at secret level, programs or processes operated by that user are not permitted to write information at the confidential level, although it could be written to a higher level, such as top-secret. (Note that this rule makes sense where we are concerned with processes operating on a computer. Obviously, a human being could log in at a high level, then print out or memorize information and re-enter it after logging in at a lower level.)
Also included in the Bell-LaPadula model was the notion of categories, which refers to a vertical breakdown of security compartments across levels. In addition to having the proper clearance level, a user is required to be cleared for all of the categories attached to a classified document.
For example, a document might be classified [Secret, nuclear, NATO]. To access the document, a user would need a clearance of secret or above and must also be cleared for the two categories—nuclear and NATO.
This policy ensures that information cannot be downgraded either through unintentional or malicious actions of a process.

Department of Defense standards

The U.S. Department of Defense (DoD) published its Trusted Computer System Evaluation Criteria (TCSEC). This standard defined in detail two important access control modes for military systems: discretionary access control (DAC) and mandatory access control (MAC).
DAC is a mode in which the creators or owners of files assign access rights, and a subject with discretionary access to information can pass that information on to another subject.
By itself, DAC is insufficient for implementing the document classification scheme used by the military. Since users in the DAC model of security can give away rights to access objects.
To provide a truly secure scheme in which a system is guaranteed to remain secure, MAC is required.
MAC controls provide the multilevel security policy as formalized by the Bell-LaPadula model.
The key feature of MAC is that, as its name implies, it is required for the mediation of all accesses of objects on the system. Since the access control system mediates all access to objects using rules imposed externally, users cannot give away permissions for object access. Since users are limited in the actions they can take, the access controls can ensure that the system will remain in a secure state regardless of user actions.

Clark-Wilson model

One goal of the TCSEC was to encourage a market for secure operating systems and computer security products. Many writers argued that systems meeting the lower levels of TCSEC requirements would be sufficient for commercial use. The hope was that a uniform market for security products would develop, with the TCSEC providing guidance for both commercial and military security.
Despite efforts to promote TCSEC-compliant systems as commercial security solutions, most commercial firms recognized that DAC and MAC were not sufficient for their needs.

TCSEC-oriented systems are focused on the information flow and confidentiality of information. In a widely referenced 1987 paper, Clark and Wilson [7] argued that while confidentiality was important to commercial users, their primary concern pertains to integrity (i.e., ensuring that information is modified only in appropriate ways by authorized users).
When Clark and Wilson formalized business security practices into a security model, the result was quite different from the military security model formalized by Bell and LaPadula. The two central concepts in the Clark-Wilson model are the well-formed transaction and separation of duty (SoD). Well-formed transactions constrain the user to change data only in authorized ways. For example, 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.
Complementing the well-formed transaction is the ancient principle of SoD, which ensures the consistency of changes made to critical data. 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. Implementing these rules in a computer system has been found to be as challenging as implementing information flow policies. One of the motivations of RBAC was to make commercial security policies easier to manage.

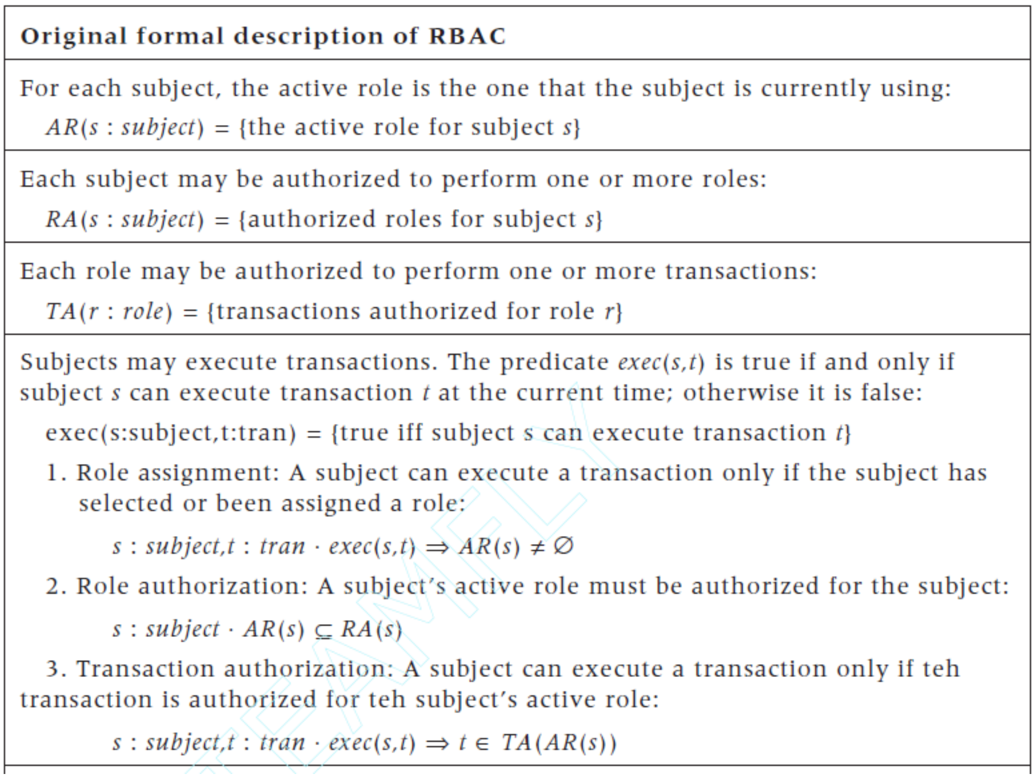
Origins of RBAC

RBAC is conceptually simple: Access to computer system objects is based on a user's role in an organization. Roles with different privileges and responsibilities have long been recognized in business organizations, and commercial computer applications.
A role was seen as a job or position within an organization. A role exists as a structure separate from that of the users who were assigned to the roles.
The model is described in terms of a particular commercial security policy, known as the Chinese wall. The model is developed by first defining what a Chinese wall means and then defining a set of rules (SoD requirements) such that no user can ever access data from the wrong side of the wall.
These role-based systems were relatively simple and application-specific. That is, there was no general-purpose model defining how access control could be based on roles, and little formal analysis of the security of these systems.
The systems were developed by a variety of organizations, with no commonly agreed upon definition or recognition in formal standards.
Conventional MAC, focused on preserving confidentiality, is also inadequate for these organizations. Although enforcing a need-to-know policy is important where classified information is of concern, there existed a general need to support subject-based security policies, such as access based on competency, the enforcement of conflict-of-interest rules, or access based on a strict concept of least privilege. Supporting such policies requires the ability to restrict access based on a user function or role within the enterprise.

The generalized RBAC model is described, in a simple formal manner, the sets, relations, and mappings used in defining roles and role hierarchies, subject-role activation, and subject-object mediation, as well as the constraints on user-role membership and role-set activation.
Three basic rules were required:
1. Role assignment: Asubject can execute a transaction only if the subject has selected, or been assigned to, a role. The identification and authentication process (e.g., login) is not considered a transaction. All other user activities on the system are conducted through transactions. Thus, all active users are required to have some active role.

2. Role authorization: A subject's active role must be authorized for the subject. With rule 1, this rule ensures that users can take on only roles for which they are authorized.
3. Transaction authorization: A subject can execute a transaction only if the transaction is authorized for the subject's active role. In concert with rules 1 and 2, this rule ensures that users can execute only transactions for which they are authorized.

The formal description of the model is given in Figure 1.2. A key feature of this model is that all access is through roles. A role is essentially a collection of permissions, and all users receive permissions only through the roles to which they are assigned, as shown in Figure 1.3. Within an organization, roles are relatively stable, while users and permissions are both numerous and may change rapidly. Controlling all access through roles therefore simplifies the management and review of access controls.



Original formal description of RBAC

For each subject, the active role is the one that the subject is currently using:

$AR(s : subject)$ = {the active role for subject s}

Each subject may be authorized to perform one or more roles:

$RA(s : subject)$ = {authorized roles for subject s}

Each role may be authorized to perform one or more transactions:

$TA(r : role)$ = {transactions authorized for role r}

Subjects may execute transactions. The predicate $exec(s,t)$ is true if and only if subject s can execute transaction t at the current time; otherwise it is false:

exec(s:subject,t:tran) = {true iff subject s can execute transaction t}

1. Role assignment: A subject can execute a transaction only if the subject has selected or been assigned a role:

$s : subject, t : tran \cdot exec(s,t) \Rightarrow AR(s) \neq \varnothing$

2. Role authorization: A subject's active role must be authorized for the subject:

$s : subject \cdot AR(s) \subseteq RA(s)$

3. Transaction authorization: A subject can execute a transaction only if teh transaction is authorized for teh subject's active role:

$s : subject, t : tran \cdot exec(s,t) \Rightarrow t \in TA(AR(s))$

Figure 1.2 Formal description of RBAC

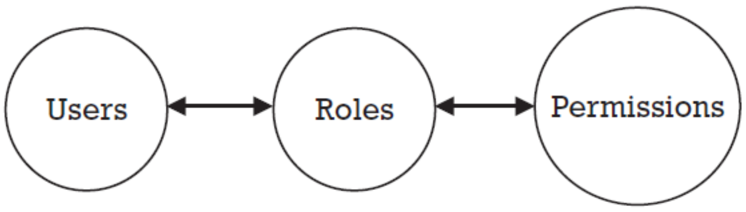


Figure 1.3 RBAC relationships.

Policy, models, and mechanisms

While authentication mechanisms ensure that system users are who they claim to be, these mechanisms say nothing about what operations users should or should not perform within the system. To afford protection to that effect, it is necessary to use access control.

Access control is concerned with determining the allowed activities of legitimate users, mediating every attempt by a user to access a resource in the system. A given IT infrastructure can implement access control systems in many places and at different levels. Operating systems use access control to protect files and directories. Database management systems (DBMSs) apply access control to regulate access to tables and views. Most commercially available application systems implement access control, often independent of the operating system or DBMS, or both, on which they may be installed.

The objectives of an access control system are often described in terms of protecting system resources against inappropriate or undesired user access. From a business perspective, this objective could just as well be described in terms of the optimal sharing of information. After all, the greater objective of IT is to make information available to users and applications. A greater degree of sharing gives rise to increased productivity. Although on the surface, access control appears to get in the way of this objective, in reality, a well-managed and effective access control system actually facilitates sharing. A sufficiently fine-grained access control mechanism can enable selective sharing of information where in its absence, sharing may be considered too risky altogether.

When considering any access control system one considers three abstractions of control: access control policies, access control models, and access control mechanisms. Policies are high-level requirements that specify how access is managed and who, under what circumstances, may access what information. While access control policies may be application-specific and thus taken into consideration by the application vendor, policies are just as likely to pertain to user actions within the context of an organizational unit or across organizational boundaries. For instance, specific policies may pertain to the resources that can be accessed by consultancies, or other business partners. Such policies may span multiple computing platforms and applications. Policies may pertain to resource usage within or across organizational units or may be based on need-to-know, competence, authority, obligation, or conflict-of-interest factors. Although there are several well-known access control policies, generating such a list is of limited value, since business objectives, tolerance for risk, corporate culture, and the regulatory responsibilities that influence policy differ from enterprise to enterprise, and even from organizational unit to organizational unit. The access control policies within a hospital may pertain to privacy and competency (e.g., only doctors and nurse practitioners may prescribe medication), and hospital policies will differ greatly from those of a military system, or a financial institution. Even within a specific business domain, policy will differ from institution to institution. Furthermore, access control policies are dynamic in nature, in that they are likely to change over time in reflection of ever evolving business factors, government regulations, and environmental conditions. However, because policy requirements can rarely be completely determined in advance, access control systems are best designed to flexibly accommodate a wide variety of changing policies.

At a high level, access control policies are enforced through a mechanism that translates a user's access request often in terms of a simple table lookup—to grant or deny access. Access control mechanisms come in a wide variety of forms, each with distinct policy advantages and disadvantages. Although no well-accepted standard yet exists for determining their policy support, access control mechanisms can be characterized in a number of different ways, each bearing policy implications. In general, access control mechanisms require that security attributes be kept about users and resources. User security attributes consist of things like user identifiers, groups, and roles to which users belong, or they can include security labels reflecting the level of trust bestowed on the user. Resource attributes can take on a wide variety of forms. For example, they can consist of sensitivity labels, types, or access control lists. In determining the user's ability to perform operations on resources, access control mechanisms compare the user's security attributes to those of the resource. Access control checks can be determined (evaluated) based on a previously determined set of rules.
For example, the security label of the user must be greater than or equal to the security label of the resource for the user to read the contents of the resource. Access control checks can also be determined based on an attribute-matching algorithm. The user may perform a read

operation on a resource if the user's identity, and read operation pair is included in the access control list of the resource. Other characteristics of access control mechanisms include attribute review and management capabilities. For example, can the access control system determine the permissions that are associated with a user or the users that can access a resource, or better yet both? Who can specify permissions? Can permission specification be delegated, and if so, does delegation infer further delegation?
From a consumer's perspective, determining the policy implications of a given access control mechanism is a formidable task. The fact that most enterprises need to deal with a wide variety of access control mechanisms only compounds this problem.
To provide greater policy support and control, a number of enterprise management and resource-provisioning vendors offer administrative capabilities over the native access control mechanisms of file management, database management, applications, and host and network operating systems. The result is an access control management system, on top of an access control management system, on top of potentially still another access control system. What are the policy implications of this arrangement?
Rather than attempting to evaluate and analyze access control systems exclusively at the mechanism level, security models are usually written to describe the security properties of an access control system. Access control models are written at a level of abstraction to accommodate a wide variety of implementation choices and computing environments, while providing a conceptual framework for reasoning about the policies they support. Access control models are of general interest to both users and vendors. They bridge the rather wide gap in abstraction between policy and mechanism. Models can be promoted for their support of policy, and mechanisms can be designed for their adherence to the properties of the model. Users see an access control model as an unambiguous and precise expression of requirements. Vendors and system developers see access control models as design and implementation requirements.
Access control models and mechanisms are often characterized in terms of their policy support. On one extreme an access control model may be rigid in its implementation of a single policy. On the other extreme, a security model will allow for the expression and enforcement of a wide variety of policies and policy classes. From the 1990s to the present, security researchers have sought to develop access control mechanisms and models that are largely independent of the policy for which they can be used. This is a generally considered to be a desirable objective in that it allows the use of a common mechanism for a wide variety of purposes.

Discretionary Access Control (DAC)

Discretionary Access Control (DAC) is a security model used in computer systems to regulate access to resources based on the discretion of the resource owner. In DAC, the resource owner determines who is allowed to access the resource and what permissions they have. This contrasts with other access control models like Mandatory Access Control (MAC), where access control decisions are determined by a central authority.
In DAC, each resource (such as files, directories, or devices) has an associated access control list (ACL) that specifies which users or groups are granted permission to access the resource and what actions they are allowed to perform (e.g., read, write, execute). The resource owner typically has the ability to modify the ACL to grant or revoke access as needed.

Key features of Discretionary Access Control include:

Resource Owner Control: The owner of a resource has complete control over who can access it and what actions they can perform. This allows for flexibility and customization in access control decisions.

Access Control Lists (ACL): Each resource has an ACL that lists the users or groups that have permissions to access it and the specific permissions granted to each entity.
Permission Granularity: DAC systems typically offer fine-grained control over permissions, allowing owners to specify different levels of access for different users or groups.
Flexibility: DAC provides flexibility in managing access control policies since decisions are decentralized and left to the discretion of resource owners. This can be advantageous in environments where resources are frequently changing or where there is a need for granular control.
Ownership Changes: Ownership of resources can be transferred, and with it, the control over access permissions. This allows for flexibility in managing resources as organizational roles and responsibilities change.

However, Discretionary Access Control also has some limitations and potential security risks:
Over-reliance on Resource Owners: DAC relies heavily on the resource owners to make appropriate access control decisions. If resource owners are not adequately trained or vigilant, they may grant inappropriate access permissions, leading to security vulnerabilities.
Lack of Centralized Control: Since access control decisions are decentralized, there may be challenges in enforcing consistent security policies across an organization or system.
Difficulty in Managing Large-scale Systems: In large-scale systems with numerous resources and users, managing access control lists and ensuring they remain up-to-date can become challenging and prone to errors.
Limited Auditability: DAC systems may have limited capabilities for auditing access activities since access decisions are made by individual resource owners rather than a centralized authority.

Advantages of Discretionary Access Control (DAC)

Discretionary Access Control (DAC) offers several advantages in managing access to resources within a computer system:
Flexibility: DAC provides flexibility in defining access control policies since access decisions are decentralized and left to the discretion of resource owners. This flexibility allows organizations to tailor access permissions to suit their specific needs and requirements.
User Autonomy: DAC empowers resource owners to control access to their resources. Users have the autonomy to determine who can access their files, directories, or other resources, and what actions those users can perform. This autonomy can improve user satisfaction and productivity by enabling them to manage their resources according to their preferences.
Simplicity: DAC systems are often simpler to implement and manage compared to other access control models like Mandatory Access Control (MAC). Since access decisions are based on the discretion of resource owners, there is typically less administrative overhead involved in managing access control policies.
Scalability: DAC can be easily scaled to accommodate changes in organizational structure or resource ownership. As new users join the organization or existing users' roles change, resource owners can adjust access permissions accordingly without requiring intervention from a central authority.
Ease of Collaboration: DAC facilitates collaboration by allowing resource owners to grant access to specific resources to other users or groups as needed. This enables teams to work

together on shared documents, projects, or other resources, without the need for complex access control procedures.
Adaptability: DAC systems can adapt to dynamic environments where access requirements may change frequently. Resource owners can modify access permissions in real-time to accommodate changing business needs or security considerations.

Example of Discretionary Access Control (DAC)
Here's an example of Discretionary Access Control (DAC) in action within a typical computer system:

Consider a scenario where you have a user named Alice who owns a folder containing sensitive documents related to a project. Alice wants to control access to these documents based on her discretion using DAC.
Resource Creation: Alice creates a folder named "ProjectX" on a shared network drive to store the documents related to the project.
Access Control List (ACL) Configuration: Alice sets up an Access Control List (ACL) for the "ProjectX" folder, specifying who can access the folder and what permissions they have. Initially, she grants herself full access (read, write, and execute permissions) to the folder.
Granting Access to Others: Alice decides to collaborate with two colleagues, Bob and Carol, on the project. She updates the ACL to grant Bob and Carol read and write permissions to the "ProjectX" folder. This allows Bob and Carol to view the documents and make changes to them as necessary.
Revoking Access: Later, Alice completes the project and no longer needs Bob and Carol to have access to the documents. She removes their entries from the ACL, revoking their access to the "ProjectX" folder.
Changing Permissions: Alice realizes that she needs assistance from her supervisor, Dave, to review the documents before presenting them to the client. She updates the ACL to grant Dave read-only access to the folder, ensuring that he can view the documents but not make any changes to them.
Ownership Transfer: Eventually, Alice leaves the company, and her responsibilities are transferred to another employee named Eve. Before leaving, Alice transfers ownership of the "ProjectX" folder to Eve, who becomes the new owner. Eve now has full control over the folder and can manage access permissions according to her discretion.
In this example, Discretionary Access Control allows Alice, as the resource owner, to control access to the "ProjectX" folder based on her discretion. She can grant or revoke access to the folder, change permissions, and transfer ownership as needed, providing flexibility and autonomy in managing access to the sensitive documents.

Non-Discretionary Access Control/ Mandatory Access Control (MAC)
Non-Discretionary Access Control (NDAC), also known as Mandatory Access Control (MAC), is an access control model where access decisions are not based on the discretion of the resource owner but are instead determined by a central authority or security policy. Unlike Discretionary Access Control (DAC), where resource owners have the autonomy to control access to their resources, in NDAC/MAC, access control decisions are centrally managed and enforced, typically by the operating system or security kernel. Here's an explanation of NDAC/MAC along with an example:

Explanation of Non-Discretionary Access Control (NDAC)/Mandatory Access Control (MAC):
In NDAC/MAC systems, access to resources is determined by security labels or classifications assigned to both subjects (users or processes) and objects (resources). These

security labels represent sensitivity levels, categories, or classifications of the subjects and objects.
Access decisions are based on security policies defined by a central authority or security administrator, and they are typically more rigid and strict compared to DAC. These policies are often based on predefined rules that dictate which subjects are allowed to access specific objects based on their security labels.

Examples of NDAC/MAC mechanisms include:
Role-Based Access Control (RBAC): RBAC is a form of NDAC where access permissions are assigned to subjects based on their roles within an organization. Access is determined by the roles assigned to subjects rather than their individual identities.
Label-Based Access Control (LBAC): LBAC assigns security labels to both subjects and objects based on their sensitivity levels or classifications. Access decisions are then made based on the comparison of these labels, with subjects only being allowed access to objects that have compatible or lower security labels.

Example of Non-Discretionary Access Control (NDAC)/Mandatory Access Control (MAC):
Let's consider a government agency handling classified information with different levels of sensitivity: Top Secret, Secret, and Unclassified. The agency implements a MAC system to enforce access control based on these classifications.
Security Labels: Each document in the agency is assigned a security label indicating its classification level (e.g., Top Secret, Secret, or Unclassified).
Subject Labels: Each user in the agency is assigned a security clearance level (e.g., Top Secret, Secret, or Unclassified).
Access Decisions: The MAC system enforces access decisions based on the comparison of subject labels (user clearances) and object labels (document classifications). For example:
- A user with a "Secret" clearance can only access documents labeled as "Secret" or "Unclassified."
- A user with a "Top Secret" clearance can access documents labeled as "Top Secret," "Secret," or "Unclassified."
- In this example, access control decisions are not based on the discretion of individual users but are instead enforced by the MAC system according to predefined security policies and classifications.

Advantages:
Enhanced Security: NDAC/MAC provides a higher level of security compared to Discretionary Access Control (DAC) by enforcing access decisions based on centrally defined security policies. This reduces the risk of unauthorized access, data leaks, and insider threats.
Consistency and Uniformity: NDAC/MAC ensures consistent and uniform access control across the entire system or organization. Access decisions are based on predetermined security labels or classifications, which helps maintain a standardized level of security compliance.
Protection against Data Breaches: Since access control decisions are not subject to the discretion of individual users, NDAC/MAC helps prevent accidental or intentional data breaches by enforcing strict access restrictions based on security classifications.
Minimal Dependency on Individual Users: NDAC/MAC reduces the reliance on individual users to make access control decisions. Access policies are centrally managed and enforced, reducing the likelihood of human error or manipulation.

Enforcement of Need-to-Know Principle: NDAC/MAC systems can enforce the need-to-know principle more effectively by restricting access to sensitive information based on the clearance levels or security labels of users. This ensures that users only have access to the information necessary for their roles.

Limitations:
Complexity and Administration Overhead: Implementing and managing NDAC/MAC systems can be complex and resource-intensive. Defining and maintaining security policies, assigning security labels, and managing user clearances require significant administrative effort and expertise.
Limited Flexibility: NDAC/MAC systems are inherently rigid and less flexible compared to DAC. Access decisions are based on predefined security policies, which may not easily accommodate dynamic changes in user roles or access requirements.
Difficulty in Collaboration: NDAC/MAC can hinder collaboration and information sharing, especially in environments where users need to access resources across different security domains. Access restrictions based on security labels may impede the flow of information between users with different clearance levels.
Risk of Over-Privilege: NDAC/MAC systems may result in over-privilege, where users are granted access to resources beyond what is necessary for their roles. This can occur if security policies are not carefully designed or if users are assigned overly broad security clearances.
Scalability Challenges: NDAC/MAC systems may face scalability challenges, particularly in large and complex environments with numerous users and resources. Managing security labels, user clearances, and access policies becomes more cumbersome as the system scales up.

Example of MAC/ Non-Discretionary Access Control
An example of Non-Discretionary Access Control (NDAC), also known as Mandatory Access Control (MAC), can be found in military and government settings where sensitive information is classified into different levels of security clearance. Let's consider a scenario within a military organization:
Scenario: Military Information Security Clearance
• **Security Classifications:** In a military organization, information is classified into different levels of sensitivity, such as Top Secret, Secret, and Unclassified.
• **Security Clearance Levels:** Personnel within the military are assigned security clearances based on their roles, responsibilities, and the sensitivity of the information they need to access. Clearance levels may include Top Secret, Secret, and Confidential.
• **Access Control Enforcement:** The military organization implements an NDAC/MAC system to control access to classified information. Access to information is determined by comparing the security clearance level of individuals with the security classification level of the information.
Example Access Scenarios:
Scenario 1 - Top Secret Clearance: A military intelligence officer with a Top Secret clearance is granted access to Top Secret, Secret, and Unclassified information. They can access highly sensitive information crucial for national security.
Scenario 2 - Secret Clearance: An operations officer with a Secret clearance is granted access to Secret and Unclassified information but is denied access to Top Secret information. They can access sensitive operational plans and reports, but not the most classified intelligence.
Scenario 3 - Unclassified Clearance: A logistics officer with an Unclassified clearance is only granted access to Unclassified information. They have access to general administrative documents and reports but cannot access any classified information.

Access Revocation and Changes: If a military personnel's clearance level changes (e.g., due to a change in role or a security investigation), their access to information is automatically adjusted by the NDAC/MAC system. For example, if a personnel's clearance is downgraded from Top Secret to Secret, they will lose access to Top Secret information.
In this example, the NDAC/MAC system enforces access control based on predefined security policies and classifications. Access to information is determined by the security clearance level of individuals, ensuring that sensitive information is only accessed by authorized personnel with the appropriate clearance level.

Capabilities and Limitations of Access Control Mechanisms: Access Control List (ACL) and Limitations, Capability List and Limitations:

- Many systems use Lampson's access control matrix to represent and interpret the particular security policy. From a theoretical perspective, the access control matrix has traditionally been used to represent the secure state of an access control system.
- The access matrix is an array containing one row per subject in the system and one column per object. Table 2.1 illustrates a simple access control matrix. The entries in the matrix specify the operations of, or the type of access that each subject has to, each object.
- The basic function of an access control system is to ensure that only the operations specified by the matrix can be executed.

Table 2.1 Example Access Control Matrix

Subject/ Object	File_1	File_2	File_3	Process_1
Chris	Read, write	—	Write	—
Janet	—	Execute	—	Suspend
Barbara	—	Read	Read	—
Frank	Read	—	—	—

- Although an access control matrix is an interesting construct from a theoretical perspective, for a system with a large number of users and objects, the matrix will become very large and will be sparsely populated.
- There are two primary representations of the access control matrix as implemented in computer systems today: **ACLs and capability lists.**

Capability lists and Limitations:

In a capability system, access to an object is allowed if the subject that is requesting access possesses a capability for the object. A capability is a protected identifier that both identifies the object and specifies the access rights to be allowed to the accessor who possesses the capability.

Table 2.2 Capability List

Subject		
Chris	File_1: Read, Write	File_3: Write
Janet	File_2: Execute	Process_1: Suspend
Barbara	File_2: Read	File_3: Read
Frank	File_1: Read	

In a capabilitybased system, access to protected objects is granted to the would-be accessor possesses a capability for the object. This approach corresponds to storing the matrix by rows. Table 2.2 presents the capability lists corresponding to the access control matrix. Each subject is associated with a capability list, which stores its approved operations to all concerned objects.
A subject possessing a capability is proof of the subject having the access privileges. The

principle advantage of capabilities is that it is easy to review all accesses that are authorized for a given subject. On the other hand, it is difficult to review the subjects that can access a particular object. To do so would entail an examination of each and every capability list. It is also difficult to revoke access to an object given the need for a similar examination. For this reason, capability lists have been criticized in their support of DAC policies and therefore, not commercially popular.

ACL and Limitations:

ACLs implement the access control matrix by representing the columns as lists of users attached to a protected object. Each object is associated with an ACL that stores the subjects and the subject's approved operations for the object. The list is checked by the access control system to determine if access is granted or denied.

Table 2.3 ACL

Object		
File_1	Chris: Read, write	Frank: Read
File_2	Janet: Execute	Barbara: Read
File_3	Chris: Write	Barbara: Read
Process_1	Janet: Suspend	

Table 2.3 presents the ACLs corresponding to the access control matrix in Table 2.1.

The principal advantage of ACLs is that they make it easy to review the users that have access to an object as well as the operations that users can apply to the object. In addition, it is easy to revoke access to an object by simply deleting an ACL entry. These advantages make ACLs ideal for implementing DAC policies that are object-oriented.

Another advantage is that the lists need not be excessively long, if groups of users with common accesses to the object are attached to the object instead of the group's individual members.

Although the use of groups adds the need for additional administrative functions for managing membership within groups, the availability of groups generally makes the administration of ACLs more efficient.

Generally speaking, the creation and management of groups should be strictly controlled, since becoming a member of a group can change the objects accessible to any member.