

UNIT 5

Access Control

Information Security — Course Notes (5 Hrs)

Topics covered: Access Control Principles • Subjects, Objects and Access Rights • Access Control Matrix and Capability Lists • Discretionary Access Control • Role Based Access Control • Attribute Based Access Control • Identity, Credential and Access Management • Trust Frameworks

1. Access Control Principles

In simple words, access control is the security mechanism that decides who is allowed to do what with a particular resource. Think of it like the security guard and ID-card system at the entrance of a building: the guard checks your card (identity) and decides which floors you are allowed to enter (access). In computer systems, access control protects files, databases, devices, and services from being used by people who should not be able to use them.

The Three Goals of Access Control (CIA Triad Support)

- Confidentiality — only authorised users can read sensitive information.
- Integrity — only authorised users can modify or delete information.
- Availability — authorised users can get the access they need, when they need it, without being blocked by unnecessary restrictions.

Core Principles Used When Designing Access Control

- Least Privilege — give every subject only the minimum access needed to do its job, nothing more. Example: a billing clerk should not be able to delete employee records.
- Separation of Duties — split sensitive tasks between two or more people so that no single person can misuse the system alone (e.g., the person who approves a payment should not also be the person who creates the payment request).
- Defence in Depth — use multiple layers of access control (login password, OTP, firewall, file permissions) so that if one layer fails, others still protect the resource.
- Fail-Safe Defaults — by default, access should be denied unless explicitly granted. This is safer than allowing everything and trying to block specific bad actions.
- Accountability — every access attempt (successful or failed) should be logged so actions can be traced back to a specific subject.

Note: Access control answers three connected questions in sequence — Identification (who are you claiming to be?), Authentication (can you prove it?), and Authorization (what are you allowed to do?). These three are often confused but are distinct steps.

Example: Identifying the Right Principle

A hospital wants nurses to view patient records only for patients currently assigned to them, not the entire hospital database. Which access control principle is this an example of?

Solution:

This is an example of the Principle of Least Privilege. Each nurse is given the minimum access necessary to perform their duty (viewing only their assigned patients) rather than blanket access to all hospital records. This limits the damage if a nurse's account is ever compromised, since the attacker would only see a small subset of records.

2. Subjects, Objects and Access Rights

Every access control decision involves three basic building blocks: who is asking for access (the subject), what is being accessed (the object), and how they are allowed to use it (the access right).

Subjects

A subject is an active entity that requests access to a resource. A subject is usually a user, but it can also be a running program or process acting on behalf of a user (for example, a web server process reading a file for you).

Objects

An object is a passive entity that holds or receives information, and on which an operation can be performed. Files, folders, database tables, printers, and even other processes can act as objects.

Access Rights (Permissions)

An access right defines the specific operation a subject is permitted to perform on an object. Common access rights include:

- Read (R) — view the content of the object.
- Write (W) — modify or create content in the object.
- Execute (X) — run the object if it is a program.
- Delete (D) — remove the object entirely.
- Own/Manage — control who else gets access, i.e. the right to grant or revoke rights to others.

Subject-Object-Access Right Relationship



Figure: Relationship between Subject, Object, and Access Right

Note: A subject in one context can become an object in another. For example, a process (subject) reading a file is also itself an object when another administrative process tries to terminate it.

Example: Classifying Subjects and Objects

In the statement 'Bob's print spooler process writes a job to the office printer,' identify the subject, the object, and the access right being exercised.

Solution:

Subject: Bob's print spooler process (the active entity initiating the action).

Object: the office printer (the passive resource being used).

Access right: Write (the process is sending/writing a print job to the printer).

3. Access Control Matrix and Capability Lists

The Access Control Matrix (ACM) is the simplest conceptual model for representing access rights. Imagine a big table: every row is a subject, every column is an object, and each cell tells you exactly what that subject can do to that object.

Access Control Matrix (Subjects x Objects)

	File1	File2	Printer
Alice	R, W	R	-
Bob	R	R, W	Execute
Carol	-	R	Execute

Figure: A sample Access Control Matrix

In practice, this matrix is usually huge and mostly empty (most users have no access to most files), so storing it directly wastes a lot of memory. Two practical ways of storing the same information are the Access Control List (ACL) and the Capability List.

Access Control List (ACL) — stored by column (per object)

Each object keeps a list of which subjects can access it and with what rights. This is like a guest list pinned to each door: the door 'remembers' who is allowed in.

Capability List — stored by row (per subject)

Each subject keeps a list (a set of 'tickets' or capabilities) of which objects it can access and with what rights. This is like a keychain: the person carries proof of every door they are allowed to open.

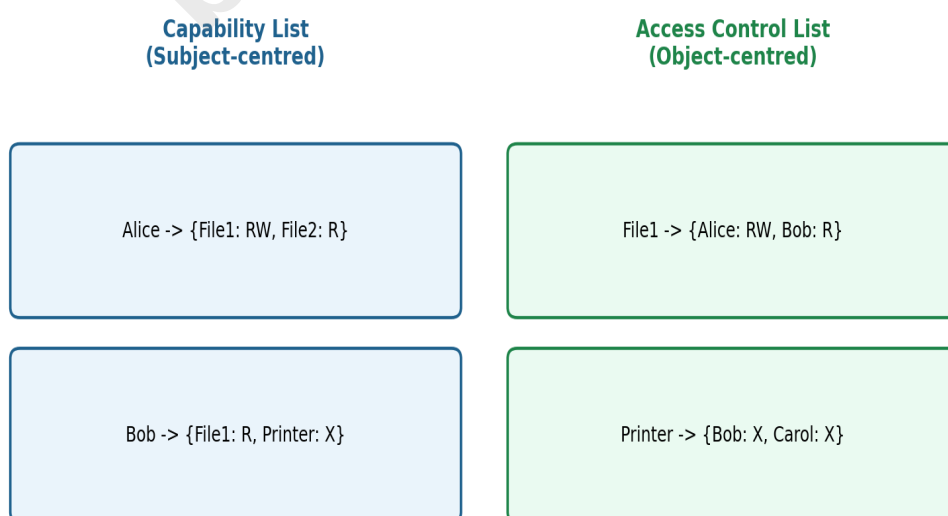


Figure: Capability List vs Access Control List representation

Aspect	Access Control List (ACL)	Capability List
Organised by	Object	Subject
Easy question to answer	Who can access this file?	What can this user access?
Revoking a user's access everywhere	Hard — must search every object	Easy — remove the subject's list
Revoking access to one object	Easy — edit that object's list	Hard — must search every subject
Common real-world example	Unix/Linux file permissions, Windows NTFS ACLs	Capability-based OS tokens, Kerberos tickets

Note: ACLs and Capability Lists store exactly the same information as the Access Control Matrix — they are just two different ways of slicing the same table for efficient storage and lookup.

Example: ACL vs Capability List Trade-off

A system administrator needs to immediately revoke an employee's access to every file and resource the moment the employee resigns. Which storage approach — ACL or Capability List — makes this task easier, and why?

Solution:

A Capability List makes this easier. Since access rights are stored per subject (per user), the administrator only needs to delete or invalidate that one employee's capability list/token set, and all of their access disappears at once. With ACLs, the administrator would instead have to search through every single object's access list and manually remove the employee's entry from each one, which is slower and more error-prone.

4. Discretionary Access Control (DAC)

In Discretionary Access Control (DAC), the owner of a resource decides who else gets access to it, and what they can do with it. The word 'discretionary' simply means 'left to someone's judgement' — here, the owner's judgement. This is the model used by everyday operating systems such as Windows and Linux for personal files.

Key Characteristics

- The resource owner has full control and can grant or revoke access to other users at will.
- Access rights can be passed on — a user who has been given access can sometimes grant that same access to another user (depending on system configuration).

- It is flexible and easy to manage for small, informal environments.
- It is considered less secure for highly sensitive environments, because a careless or malicious owner can accidentally (or deliberately) share data too widely.

Note: A real example of DAC is sharing a Google Drive file: the file owner chooses exactly who can view, comment, or edit it, and can change that decision at any time.

Example: Identifying DAC in Practice

Priya creates a spreadsheet on her work laptop and shares 'Edit' access with two colleagues directly from the file's sharing settings, without needing approval from the IT department. What access control model is being used here, and why?

Solution:

This is Discretionary Access Control. Priya, as the owner of the spreadsheet, is using her own discretion (judgement) to decide who else may access the file and at what level (Edit), without any centrally enforced policy or administrator approval being required. This ownership-driven, flexible sharing is the defining feature of DAC.

5. Role Based Access Control (RBAC)

Instead of giving permissions directly to each individual user (which becomes unmanageable in a large organisation), Role Based Access Control (RBAC) assigns permissions to roles (job functions like 'Manager', 'Clerk', 'Auditor'), and then assigns users to the appropriate roles. The user automatically receives all the permissions tied to their role.

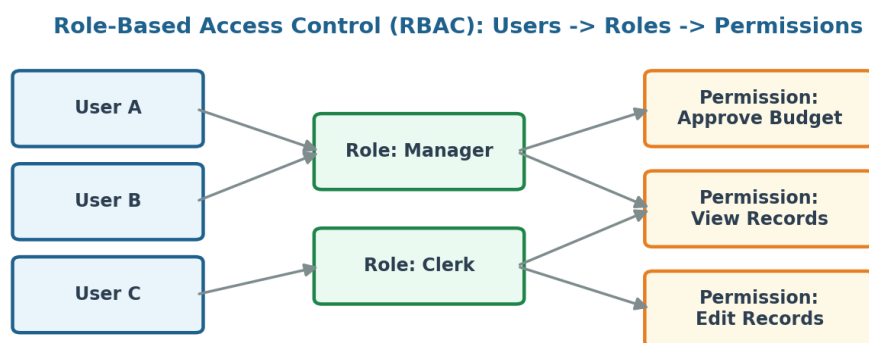


Figure: Users are assigned Roles, and Roles are assigned Permissions

Why RBAC is Useful

- Simplifies administration — when an employee changes job, you simply change their role instead of editing dozens of individual permissions.
- Supports Least Privilege naturally — roles are designed to contain only the access needed for that job function.
- Scales well — a company with thousands of employees might only need a few dozen well-designed roles.
- Reduces errors — manually assigning permissions to each new hire is error-prone; assigning a pre-built role is much safer.

Core RBAC Components

- Users — the people (or system accounts) needing access.
- Roles — named job functions, e.g. 'HR Manager', 'Database Admin'.
- Permissions — the actual operations allowed (read, write, approve, etc.).
- Sessions — a user may activate one or more of their assigned roles during a particular login session.

Example: Designing Roles

A university has 500 students, 60 faculty members, and 10 administrative staff who all need access to the student records system, but each group needs very different permissions. Explain how RBAC would be applied here instead of assigning permissions to each of the 570 individuals separately.

Solution:

Under RBAC, the university would create just three roles: Student, Faculty, and Admin Staff. Each role is assigned the permissions appropriate to it — for example, Students can view only their own grades; Faculty can view and update grades for students in their own courses; Admin Staff can view and edit all student records. Every one of the 570 people is then simply assigned to the matching role. If a new student joins, the administrator only assigns the 'Student' role rather than configuring permissions individually, saving enormous administrative effort.

6. Attribute Based Access Control (ABAC)

Attribute Based Access Control (ABAC) goes a step further than RBAC. Instead of relying only on a fixed role, ABAC makes access decisions dynamically using attributes — properties describing the subject, the object, and the environment at the moment access is requested. A central policy engine evaluates rules written in terms of these attributes and decides Allow or Deny in real time.

Types of Attributes

- Subject attributes — role, department, security clearance, employment status.
- Object (resource) attributes — file type, owner, classification level (e.g. 'Confidential'), creation date.
- Environment attributes — time of day, location, device type, network used (e.g. office Wi-Fi vs public Wi-Fi).

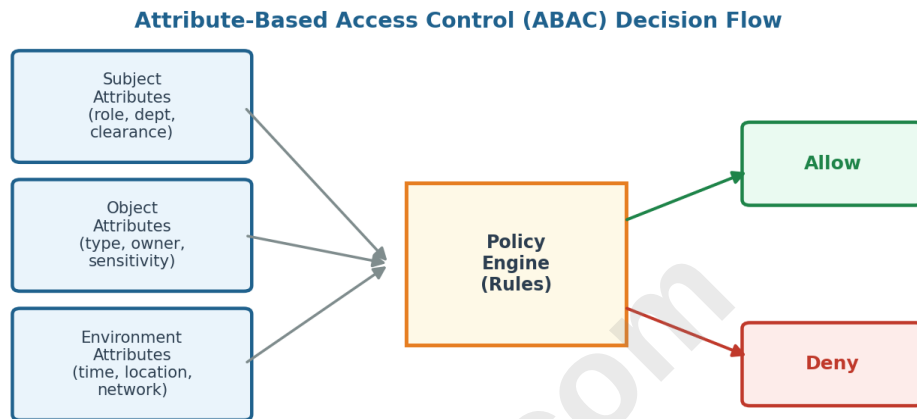


Figure: ABAC policy decision flow combining multiple attribute types

Example Policy Rule (in plain English)

Note: "Allow access to files marked Confidential ONLY IF the subject's clearance level is Secret or higher, AND the request is made between 9 AM and 6 PM, AND the connection comes from the corporate network." — this single rule combines a subject attribute, an environment time attribute, and an environment location attribute.

RBAC vs ABAC — Quick Comparison

Aspect	RBAC	ABAC
Decision based on	Fixed role membership	Multiple dynamic attributes
Flexibility	Lower — roles must be predefined	Higher — fine-grained, context-aware
Ease of setup	Simpler to design and audit	More complex to design and test
Best suited for	Organisations with stable, clear job functions	Environments needing context-sensitive rules (time, location, risk)

Example: Applying an ABAC Rule

An employee with 'Secret' clearance tries to open a Confidential file at 11 PM from a personal phone connected to public Wi-Fi at a cafe. Using the example policy rule above, will ABAC allow or deny this request, and why?

Solution:

ABAC will deny this request. Although the subject attribute (clearance level: Secret) satisfies the rule, the environment attributes do not: the time (11 PM) falls outside the allowed 9 AM-6 PM window, and the connection is from public Wi-Fi rather than the corporate network. Because ABAC rules require ALL specified conditions to be true together (it is a logical AND), failing even one condition causes the policy engine to deny access, regardless of how trustworthy the subject's clearance level looks on its own.

7. Identity, Credential and Access Management (ICAM)

Identity, Credential and Access Management (ICAM) is the overall discipline and set of tools an organisation uses to manage a person's digital identity from the moment they join until the moment they leave, including proving who they are and controlling what they can access throughout their time with the organisation.

The Three Components

- Identity Management — creating, maintaining, and retiring a unique digital identity for each user (e.g. an employee ID and user account).
- Credential Management — issuing and managing the proof used to authenticate that identity, such as passwords, smart cards, security tokens, or biometric data.
- Access Management — enforcing what an authenticated identity is actually allowed to do, using models like DAC, RBAC, or ABAC.

Identity, Credential and Access Management (ICAM) Lifecycle

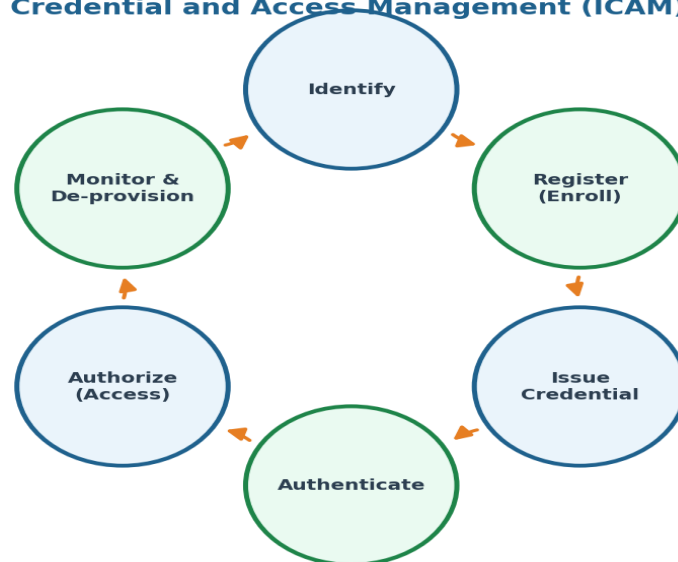


Figure: The ICAM lifecycle from identification to de-provisioning

Why ICAM Matters

- Ensures every access request can be tied back to a real, verified identity (accountability).
- Automatically de-provisions (removes) access the moment an employee leaves, closing a common security gap.
- Supports Single Sign-On (SSO), letting one verified identity access many systems without re-entering credentials repeatedly.
- Forms the foundation that RBAC and ABAC depend on — you cannot correctly assign roles or evaluate attributes for a user whose identity was never properly verified in the first place.

Example: Spotting an ICAM Failure

Six months after an employee resigns, an audit discovers their account can still log in and access company files. Which part of the ICAM lifecycle failed, and what is the security risk?

Solution:

This is a failure of the de-provisioning stage of the ICAM lifecycle — the final step where access and credentials should be revoked when a user leaves the organisation. The security risk is significant: a former employee (or anyone who later obtains their credentials) could access confidential systems and data with no legitimate business reason to do so, and any access during this period would also be very hard to monitor or attribute correctly, since the organisation no longer expects activity from that account.

8. Trust Frameworks

A Trust Framework is an agreed set of rules, standards, and technical requirements that allow different organisations to trust each other's identity and credential checks, instead of every organisation re-verifying identity from scratch. In simple words: it lets Organisation B say 'I trust that Organisation A already verified this person properly, so I will accept their login.'

Why Trust Frameworks are Needed

Imagine you need a separate username and password for your bank, your university, your hospital, and your government tax portal — and each one independently re-checks your identity documents. This is wasteful and inconvenient. A trust framework defines common rules so that an identity verified once by a trusted provider can be reused safely across many services.

Typical Elements of a Trust Framework

- Identity Assurance Levels — agreed levels describing how strongly an identity was verified (e.g. low, medium, high assurance).
- Technical Standards — common protocols such as SAML, OAuth 2.0, or OpenID Connect that let systems exchange identity information securely.
- Legal and Policy Agreements — contracts defining liability, privacy obligations, and data-handling rules between participating organisations.
- Accreditation/Certification — an independent process confirming that an identity provider actually follows the agreed rules before it is trusted by others.

Note: A familiar everyday example of a trust framework in action is logging into a third-party website using 'Sign in with Google' or 'Sign in with your government Digital ID' — the website trusts Google's or the government's identity verification instead of building its own from scratch.

Example: Recognising a Trust Framework Benefit

A citizen uses a single government-issued Digital ID to log into their tax portal, their national health portal, and their vehicle registration portal, without separately registering on each one. Which concept enables this, and what problem does it solve?

Solution:

This is enabled by a Trust Framework connecting the three government portals to a common, accredited Digital ID identity provider. It solves the problem of redundant, repeated identity verification: instead of each portal independently checking the citizen's documents and maintaining a separate credential, all three portals agree to trust the single verified identity issued under the shared framework, improving both convenience for the citizen and consistency of identity assurance across government services.

Key Takeaways — Unit 5: Access Control

- Access control governs who can do what to a resource, built on Identification, Authentication, and Authorization.
- Core design principles: Least Privilege, Separation of Duties, Defence in Depth, Fail-Safe Defaults, and Accountability.
- Every access decision involves a Subject, an Object, and an Access Right.
- The Access Control Matrix is the conceptual model; ACLs (per object) and Capability Lists (per subject) are the two practical ways to store it.
- DAC lets the resource owner decide access; flexible but less controlled.
- RBAC assigns permissions to roles, then users to roles — simple and scalable for stable job functions.
- ABAC evaluates subject, object, and environment attributes dynamically through a policy engine — flexible and context-aware.
- ICAM manages the full identity lifecycle: identification, credentialing, authentication, authorization, and de-provisioning.
- Trust Frameworks allow organisations to rely on each other's identity verification through shared standards, assurance levels, and legal agreements.

— End of Unit 5 —