

IAM A Visual Guide

Identity and Access Management Fundamentals for Systems Engineers

Reiner Braun

Sample Book — Chapters 1 to 2

Introduction

This book is designed for readers who are taking their first steps with Identity and Access Management (IAM), including systems engineers who want to build the right foundation from the start.

Rather than asking you to memorize more than you need, the book focuses on the essential concepts that will support your future study and day-to-day work. More advanced topics are included only when they are likely to become useful to many readers, and even then they are introduced only far enough to explain their purpose, context, and overall role.

Processes, relationships, and structures that can be difficult to picture from text alone are supported by diagrams. By moving between explanations, visuals, and examples, you can build a clearer and more practical understanding of the subject.

Product names, service names, logos, icons, and other identifying marks mentioned in this book are trademarks or registered trademarks of their respective owners. This book is not an official publication, certified training resource, or endorsed learning material of any company or organization.

How to Use This Book

This is a beginner-friendly book focused on the essential fundamentals of IAM. The chapters are arranged so that each one supplies concepts used by the chapters that follow.

1. **Read the chapters in order.** Begin with the IAM big picture, then move through identities, authentication, authorization, lifecycle management, directories, infrastructure access, federation, cloud IAM, workload identity, privileged access, operations, incident response, and safe design. This sequence builds the necessary background step by step.
2. **Preview the chapter objective.** Review the **What You'll Be Able to Do** section at the beginning of each chapter before reading the chapter itself. In this edition, the opening bold overview serves that purpose by stating the practical outcome and central idea.
3. **Use diagrams to understand structure and flow.** Diagrams clarify relationships, processing sequences, trust boundaries, state changes, and responsibility divisions that are harder to hold in mind from prose alone. Read the surrounding text first, then use the diagram to confirm the mental model.
4. **Try the examples yourself.** Run code, commands, and hands-on examples in an appropriate test environment whenever possible. Check the stated prerequisites, predict the result before execution, and compare the actual result with the explanation.
5. **Use each chapter summary and review questions.** Use them to confirm that you have understood and retained the fundamentals. In this edition, each chapter's concluding synthesis and common-mistake section serve as the summary and review prompts. Explain the main ideas without looking at the text, then revisit any point you cannot explain clearly.
6. **Treat advanced-topic overviews as signposts.** You do not need to memorize them at this stage. Remember what problem each advanced topic addresses and where it fits, so that you can recognize when deeper study becomes necessary.

Table of Contents

Introduction

How to Use This Book

Chapter 1 — The IAM Big Picture

The access question

Four responsibilities of IAM

A complete access path

IAM is not the same as cybersecurity

IAM is not merely single sign-on

IAM is not merely a directory

Trust boundaries

The identity-related meaning of zero trust

Three distinctions that prevent many errors

A working mental model

Chapter 2 — Identities, Accounts, Principals, and Credentials

Digital identity

Common identity types

Identity, account, subject, and principal

Identifiers and attributes

Groups, roles, permissions, and entitlements

Credentials and authenticators

Sessions and tokens

Tenants, domains, realms, and trust boundaries

Identity proofing and enrollment

Mapping one person across systems

Ownership and purpose

Misunderstandings that create long-term risk

Chapter 1 — The IAM Big Picture

Identity and Access Management (IAM) is the discipline of ensuring that the right human or workload can perform the right action on the right resource, under the right conditions, with a reliable record of what happened. Every IAM design, configuration, and troubleshooting task is a variation of that sentence.

A login page is only one visible part of IAM. Behind it, systems must establish an identity, verify evidence, decide what is allowed, enforce that decision, and record the result. A well-designed IAM system makes legitimate work straightforward while limiting mistakes, misuse, and the impact of compromised credentials.

The access question

When an access request arrives, the system needs enough information to answer five questions:

- **Who or what is requesting access?**
- **What action is being requested?**
- **Which resource is the target?**
- **Under what conditions is the request being made?**
- **How will the decision and resulting activity be recorded?**

Consider an engineer who opens a log-search application and requests to view staging logs. The request is not simply “Can this user log in?” The application must determine which identity is present, whether that identity has permission to read the selected log group, whether any conditions apply, and whether the action should be audited.

This is similar to entering a controlled office building. Showing a badge at the entrance proves that the badge is valid, but it does not automatically permit entry to every floor, laboratory, or records room. Different doors apply different rules, and access events may be recorded for later review.

Four responsibilities of IAM

IAM work can be organized into four connected responsibilities.

Identity administration

Identity administration creates and maintains digital identities, accounts, attributes, group memberships, and lifecycle status. It answers questions such as:

- Has a new employee been given the required accounts?
- Has a contractor’s end date been recorded?
- Has an engineer’s team change updated the appropriate group memberships?
- Does every service account have a named owner and purpose?

Identity administration provides the information on which later access decisions depend. Incorrect or stale identity data can lead to incorrect access even when authentication and authorization mechanisms are working as designed.

Authentication

Authentication is the process of gaining confidence that a claimant controls an identity or an approved credential for that identity. Passwords, security keys, certificates, and passkeys are examples of authenticators used during this process.

Authentication answers, “Is this requester sufficiently proven to be the identity they claim?” It does not answer, “What may that identity do?”

Authorization

Authorization is the process of deciding whether an authenticated subject may perform a requested action on a resource. Authorization evaluates permissions, roles, policies, resource ownership, and conditions.

For example, two engineers may authenticate through the same company sign-in page. One may be authorized to read staging logs, while the other may also deploy to staging. Their authentication method can be identical even though their permissions differ.

Governance and accountability

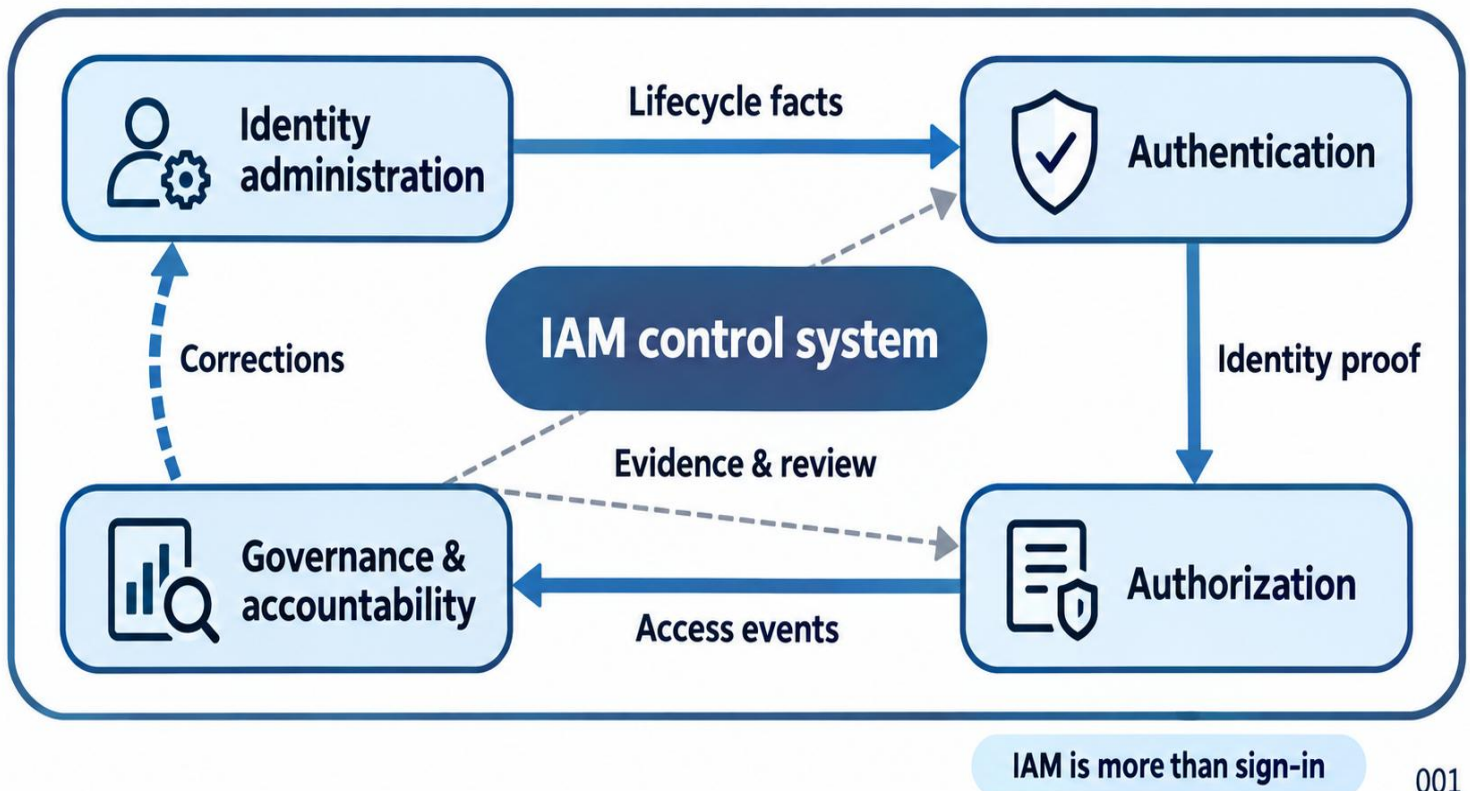
Governance is the set of processes used to ensure that access is justified, approved, reviewed, and removed when no longer needed. **Accountability** means that important identity and access events can be attributed to a specific actor and examined later.

Governance asks why access exists and whether it should continue. Accountability asks who changed or used access, when the event occurred, and what the result was.

These responsibilities reinforce one another. Strong authentication cannot compensate for excessive authorization. Precise authorization cannot remain safe if former workers keep active accounts. Good lifecycle processes cannot be verified without trustworthy logs.

Chapter 1: The IAM Big Picture

1-2 The four IAM responsibilities and their handoffs



001

Figure 1-1: The four IAM responsibilities and their handoffs

A complete access path

An access request commonly passes through several systems. Product names vary, but the responsibilities are broadly reusable.

1. An **authoritative source** supplies trusted lifecycle facts. For a workforce identity, this may be a human-resources system that records employment status, department, manager, and end date.
2. A **directory** stores identity-related objects and attributes so that other systems can find them efficiently.
3. An **identity provider**, often abbreviated as **IdP**, authenticates the user and provides identity information to an application or platform.
4. A **policy decision point** evaluates applicable rules and determines whether the request should be allowed or denied.
5. A **policy enforcement point** applies that decision at the application, API, operating system, database, or other protected resource.

6. An **audit log** records relevant events so that operators, security teams, and reviewers can reconstruct what happened.

Chapter 1: The IAM Big Picture

1-3 Generic end-to-end IAM access path

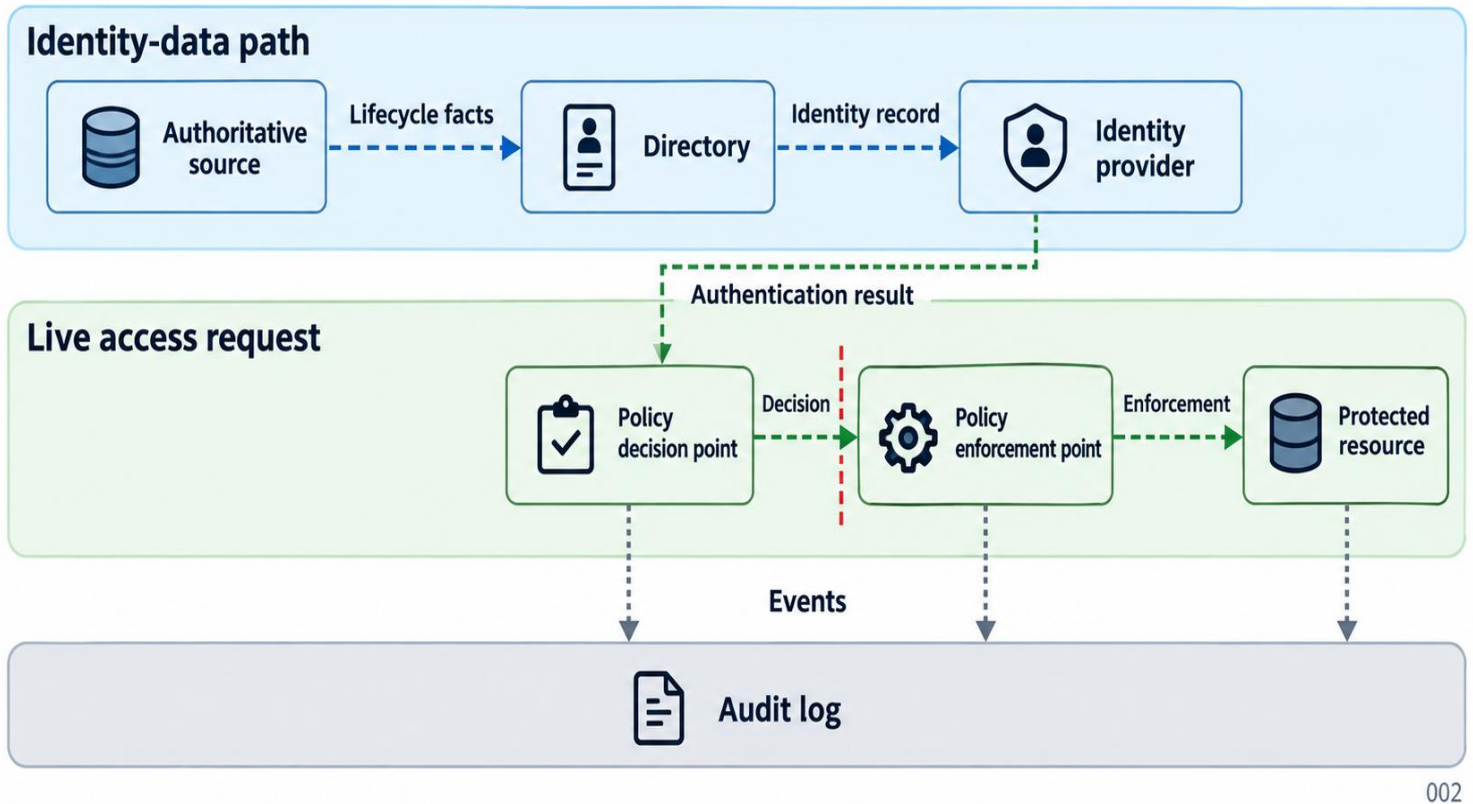


Figure 1-2: Generic end-to-end IAM access path

These functions may be distributed across several products or combined in one platform. The conceptual separation still matters because a failure can occur at any step.

Example: a new engineer reads staging logs

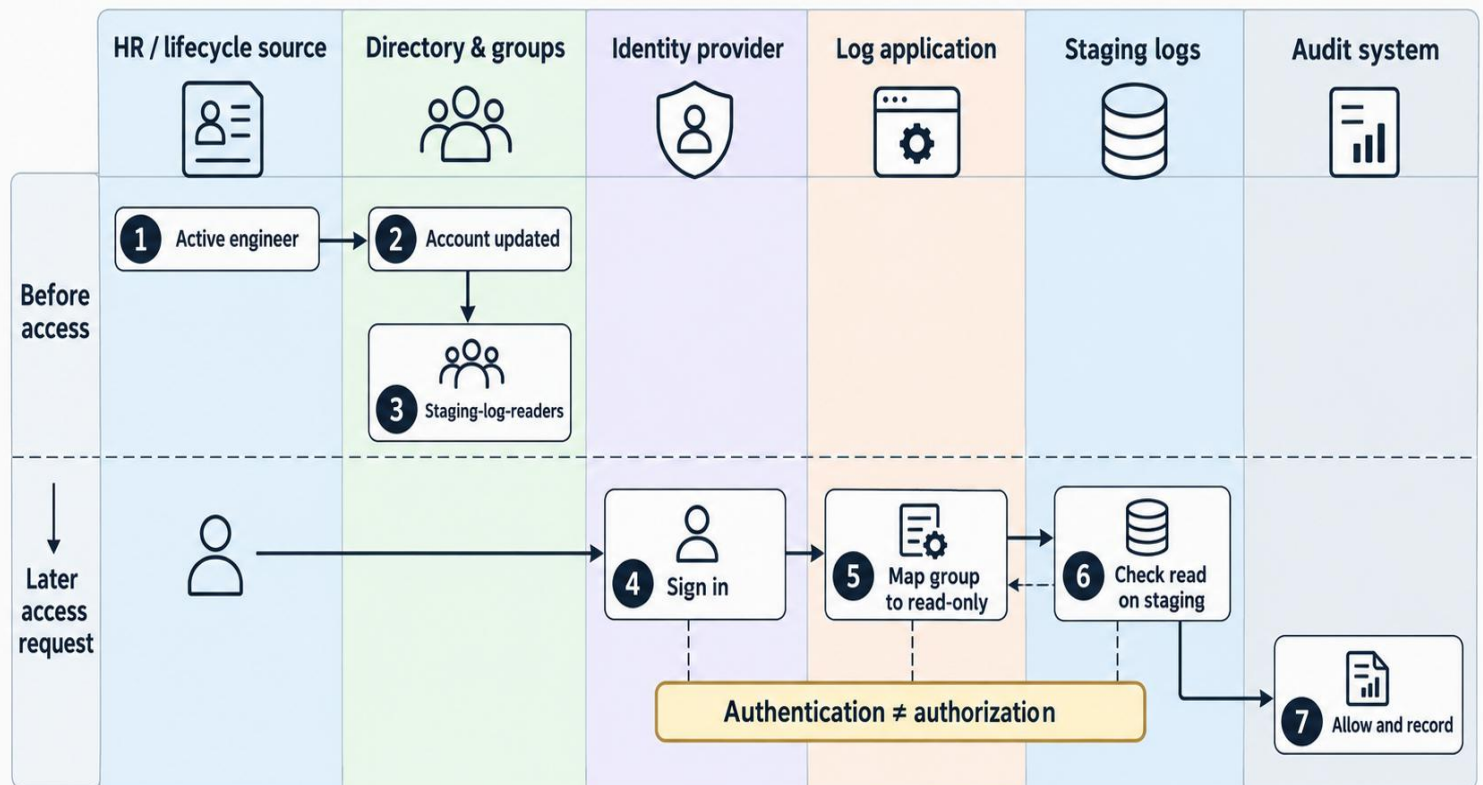
A newly assigned engineer needs read-only access to the staging environment.

1. The human-resources system records the engineer as an active employee in the platform team.
2. An automated process creates or updates the company directory account.
3. The engineer is added to a group representing staging log readers.
4. The engineer signs in through the corporate identity provider and completes the required authentication.
5. The log application receives trusted identity information and maps the group membership to its read-only role.
6. The application checks that the requested operation is a read action against a staging log resource.
7. The application allows the request and records the actor, action, resource, time, and result.

If the engineer can sign in but receives an access-denied response, authentication may be working correctly while authorization, group synchronization, role mapping, or resource policy is not. Separating these layers prevents the common mistake of changing passwords or disabling security controls when the actual problem is a missing permission.

Chapter 1: The IAM Big Picture

1-3 New engineer reads staging logs



003

Figure 1-3: New engineer reads staging logs

IAM is not the same as cybersecurity

Cybersecurity is the broader discipline of protecting systems, networks, applications, and data from threats and failures. IAM is one major part of cybersecurity, but it does not replace controls such as secure software development, network segmentation, vulnerability management, backups, encryption, or incident response.

IAM controls who may attempt an operation and under which conditions. It cannot, by itself, ensure that an authorized application contains no software vulnerabilities or that an authorized administrator will never make a harmful mistake.

The reverse is also true: strong network controls do not eliminate the need for IAM. A user who is connected to an internal network should not automatically receive unrestricted access to every internal service.

IAM is not merely single sign-on

Single sign-on (SSO) is a user experience in which one authentication session can be used to access multiple applications without entering credentials repeatedly. SSO improves usability and can centralize authentication controls, but it is not a complete IAM system.

Each application still needs to decide what the signed-in identity may do. An SSO session might allow a user to open a project-management application, but the application must still distinguish project viewers, editors, billing administrators, and organization owners.

SSO also does not automatically create, update, suspend, or delete application accounts. Those lifecycle functions require provisioning or another account-management process.

IAM is not merely a directory

A **directory service** is a system designed to store and retrieve identity-related objects such as users, groups, devices, and their attributes. A directory can be an important source of identity data, but storing a user record does not by itself perform every IAM function.

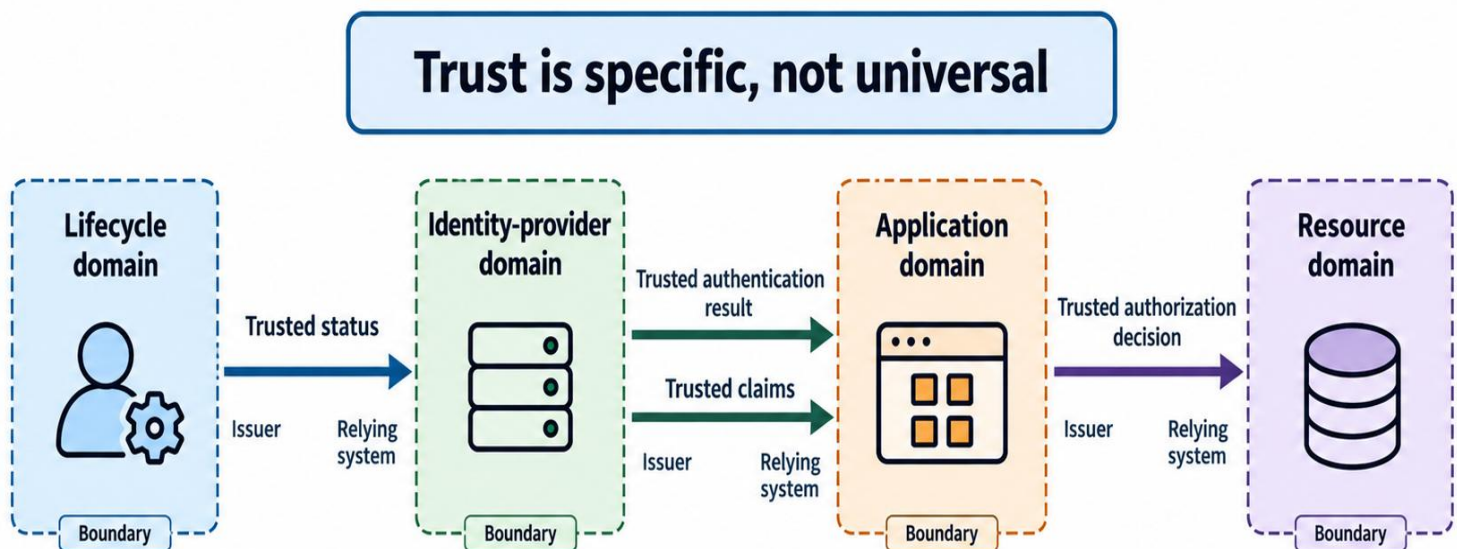
A directory may support authentication, or it may be connected to a separate identity provider. Applications may use directory groups for authorization, but they may also maintain their own roles and permissions. Governance processes must still determine whether the stored access is appropriate.

Trust boundaries

A **trust boundary** is a point at which a system accepts information or authority from another security domain or component. Crossing a trust boundary requires explicit rules about what is trusted and why.

Chapter 1: The IAM Big Picture

1-7 Trust boundaries in an IAM decision chain



004

Figure 1-4: Trust boundaries in an IAM decision chain

Suppose an application accepts a signed statement from a corporate identity provider. The application is trusting several things:

- The identity provider has authenticated the user according to an acceptable policy.
- The statement was issued by the expected system.
- The statement has not been altered.
- The statement was intended for this application.
- The identity attributes and group information are sufficiently accurate for the application's decisions.

Trust should be narrow. An application may trust an identity provider to state who signed in, but it does not need to let the identity provider define every application permission. Similarly, a directory may trust an authoritative human-resources source for employment status but not for technical administrator privileges.

An IAM system is only as strong as the systems and processes it trusts. A carefully protected identity provider can still be undermined by a weak account-recovery process, an inaccurate lifecycle source, an unreviewed administrator role, or an application that fails to enforce authorization.

The identity-related meaning of zero trust

Zero trust is an architectural approach that avoids granting broad trust solely because a request originates from a familiar network location. In IAM terms, access should be evaluated using identity and relevant context rather than assuming that “inside the network” means safe.

This does not mean that nothing is ever trusted. Systems must trust certain identity providers, devices, certificates, policies, and administrators. The point is that trust is explicit, limited, and continually supported by evidence.

A zero-trust-oriented access decision may consider:

- The identity of the user or workload.
- The strength and recency of authentication.
- The state of the device.
- The sensitivity of the requested resource.
- The requested action.
- The network or geographic context.
- Known risk signals.

Chapter 1: The IAM Big Picture

1-8 Identity-centered zero-trust request evaluation

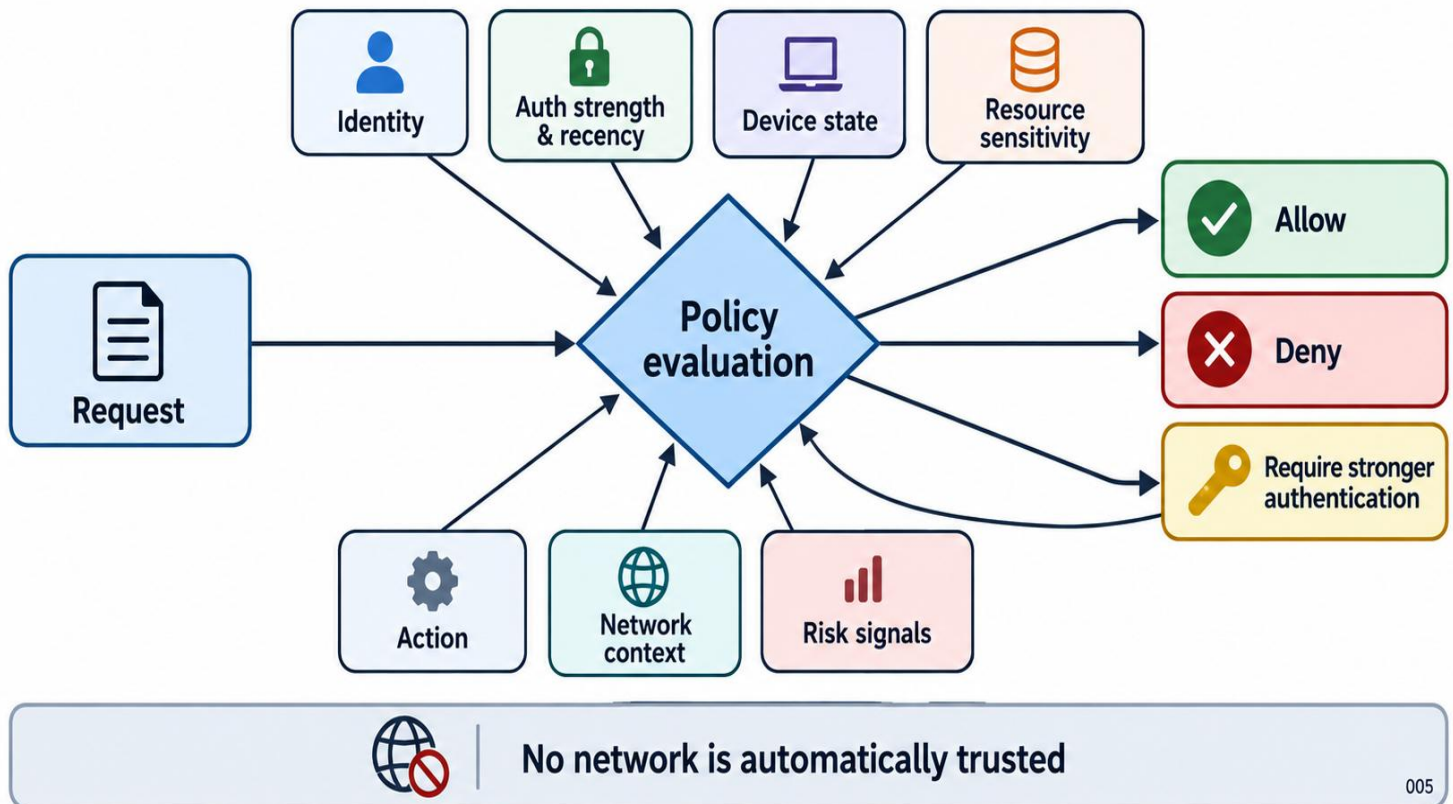


Figure 1-5: Identity-centered zero-trust request evaluation

For example, routine access to a development wiki may require a normal authenticated session. Changing a production access policy may require a managed device, a stronger authentication method, and recent reauthentication.

Three distinctions that prevent many errors

Authentication is not authorization

A successful sign-in establishes an identity with some level of confidence. It does not grant every permission. When troubleshooting, first determine whether the failure occurred before or after authentication.

Convenience is not authority

An email address, job title, or visible button can be convenient input, but none should be treated as sufficient authorization by itself. Authorization must be based on controlled policy and enforced at the protected service or resource.

Centralization is not completeness

A central identity provider can improve control and consistency, but applications, operating systems, cloud platforms, and databases still contain authorization settings, local accounts, sessions, and credentials that must be managed.

A working mental model

For any IAM task, start with this sentence:

A subject requests an action on a resource under certain conditions, and a trusted enforcement point must allow or deny the request and record the outcome.

Then ask where the subject came from, how it authenticated, which policies apply, where the decision is enforced, how access changes over time, and what evidence will remain. This model scales from a local file permission to enterprise federation and cloud administration.

Chapter 2 — Identities, Accounts, Principals, and Credentials

IAM becomes easier when each object is named precisely: an identity describes an entity, an account represents that identity in a particular system, a principal acts in a security context, and a credential helps prove control. Treating these terms as interchangeable leads to incorrect designs and confusing incident investigations.

Digital identity

A **digital identity** is a set of information used by systems to represent a person, workload, device, or organization. It is not the physical person or machine itself. It is the system's maintained representation of that entity.

A workforce identity might include an employee number, name, status, department, manager, and employment type. A workload identity might include an application identifier, owning team, deployment environment, and approved purpose.

An identity can appear differently in different systems. One engineer may be represented by an employee record, a directory object, a cloud identity, a local operating-system user, and an account in a software service. These records may be linked, but they are not the same object.

Common identity types

IAM is not limited to employees. Most environments contain several identity types with different lifecycle and authentication needs.

- A **human identity** represents an employee, contractor, partner, customer, or other person.
- A **workload identity** represents software such as an application, background service, scheduled job, container, or deployment pipeline.
- A **device identity** represents a managed workstation, server, phone, network appliance, or other device.
- An **external identity** represents a person or organization whose primary identity is managed outside the local organization.
- A **shared identity** is used by more than one person, such as a legacy shared administrator account.

Shared identities should be uncommon because they weaken accountability. When several people use one account, logs may show which account acted but not which person was responsible. A shared identity may occasionally be required by a legacy system or emergency design, but it needs compensating controls such as controlled checkout, strong logging, limited use, and credential rotation.

Identity, account, subject, and principal

These terms overlap in everyday conversation, but they describe different roles.

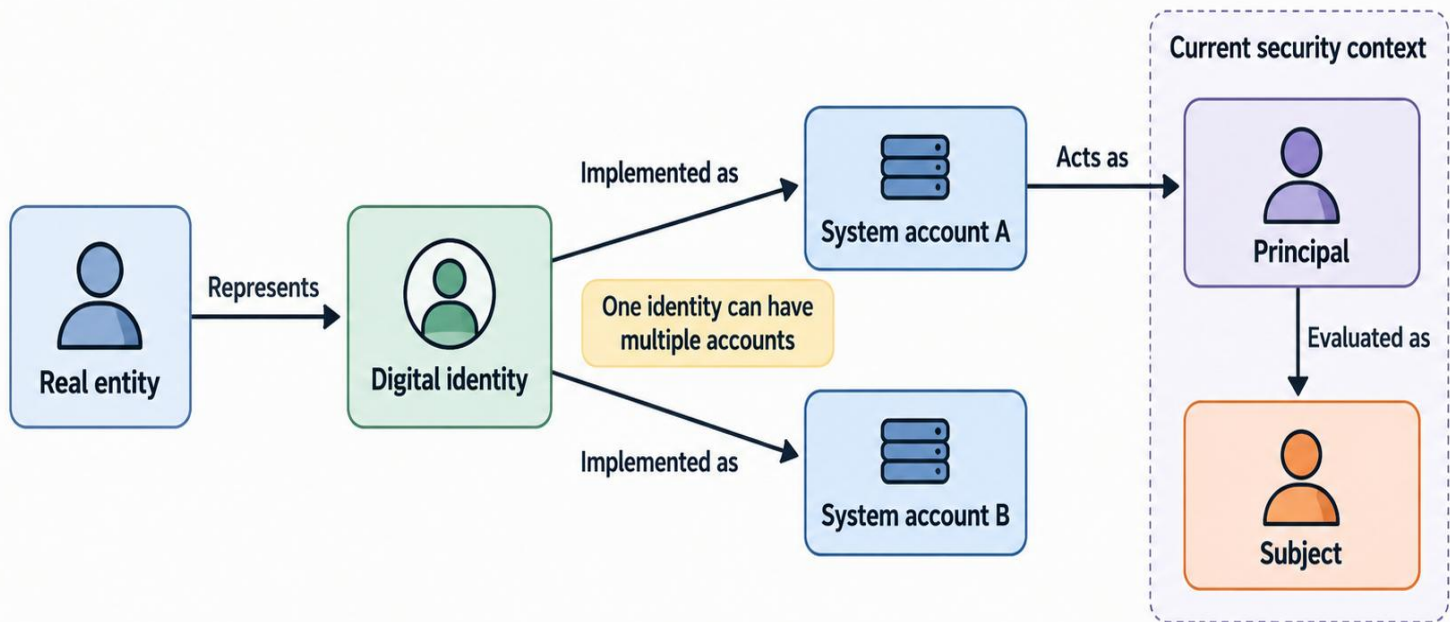
Term	Practical meaning	Example
Identity	The maintained representation of an entity.	The organization's record of engineer Maya Chen.
Account	A system-specific object through which an identity accesses or uses that system.	Maya's account in the code repository.
Subject	The entity about which a security decision is being made.	The user requesting to read a repository.
Principal	A subject represented in a form that a security system can authenticate or authorize.	A signed-in user principal or an application principal.
Local system user	An operating-system account defined on one host rather than in a central directory.	The Linux user <code>mchen</code> stored in the host's local account database.

A person can have several legitimate accounts. For example, an administrator may use one everyday account for email and documentation and a separate administrative account for elevated tasks. This separation reduces the chance that routine activity exposes powerful credentials.

Multiple accounts are not automatically a problem. **Unowned, unexplained, or unmanaged accounts are the problem.** Every account should have a clear owner, purpose, lifecycle, and appropriate authentication method.

Chapter 2: Identities, Accounts, Principals, and Credentials

2-3 Identity, account, subject, and principal relationships



006

Figure 2-1: Identity, account, subject, and principal relationships

Identifiers and attributes

An **identifier** is a value used to distinguish an identity or account from others within a defined scope. An employee number, directory object identifier, username, or application client identifier can serve this purpose.

An **attribute** is a named property associated with an identity or account, such as department, manager, employment type, email address, or device-compliance state.

Identifiers should be chosen for stability and uniqueness within their intended scope. Human-readable values are convenient but often change. A person may change their name, department, or primary email address while remaining the same identity.

An email address is therefore often a useful login name or contact attribute, but it should not automatically be treated as a permanent global identifier. Email addresses can be renamed, reused, or duplicated across unrelated organizations.

A stable internal identifier allows systems to preserve the relationship even when visible attributes change. For example, an application can continue associating records with the same employee object after a name change rather than mistakenly creating a second user.

Groups, roles, permissions, and entitlements

These concepts describe different parts of access assignment.

A **group** is a collection of identities or accounts managed together. Groups are commonly used to represent teams, populations, or access sets.

A **role** is a reusable collection of responsibilities or permissions associated with a function. A role should describe what access is needed, not merely repeat an organizational title.

A **permission** is an allowed action on a resource or resource type, such as reading a log stream or restarting a service.

An **entitlement** is a grantable unit of access exposed by a system, such as membership in a privileged group, assignment of an application role, or access to a shared folder. In governance work, the term often refers to the specific access item that can be requested, approved, reviewed, and revoked.

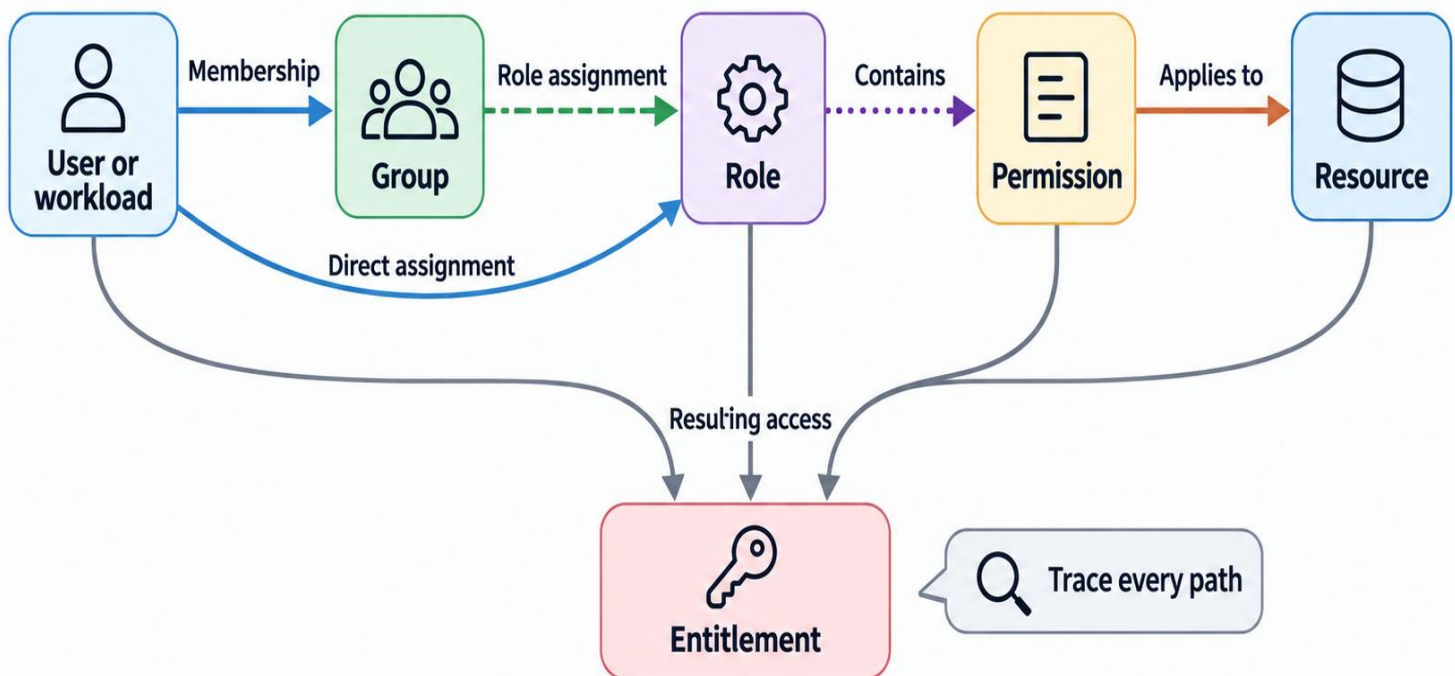
One common arrangement is:

1. A user is added to a group.
2. The group is assigned a role.
3. The role contains permissions.
4. Those permissions allow actions on defined resources.

Not every system separates the layers this way. Some applications assign roles directly to users. Others treat group membership itself as the entitlement. The important point is to trace how an identity eventually receives an effective permission.

Chapter 2: Identities, Accounts, Principals, and Credentials

2-5 How groups, roles, permissions, and entitlements relate



007

Figure 2-2: How groups, roles, permissions, and entitlements relate

Credentials and authenticators

A **credential** is data or an object used as evidence in authentication or as a means of obtaining an authenticated security context. An **authenticator** is the mechanism a claimant possesses or controls and uses to prove identity.

In practical conversation, the terms are often used together. Common examples include:

- Passwords or personal identification numbers.
- Private cryptographic keys.
- Client certificates.

- Passkeys.
- Hardware security keys.
- One-time-code devices or applications.
- Recovery codes.

A username is normally an identifier, not a secret credential. Knowing a username helps a system locate the account, but it should not prove control of that account.

Credentials require lifecycle management. They must be issued securely, stored appropriately, replaced when lost or compromised, revoked when no longer valid, and associated with a known owner and purpose.

Sessions and tokens

After authentication, a system usually creates a temporary representation of the authenticated state so that the user or workload does not need to authenticate for every request.

A **session** is a bounded period during which a system remembers an authenticated or authorized context. A browser application may represent a session with a protected cookie and server-side state.

A **token** is a data object that represents claims, authorization, or delegated access for a limited purpose and time. Tokens may be opaque identifiers that only the issuing system understands, or they may contain structured claims.

A session or token is not the identity itself. It is evidence that a trusted process established a security context. If stolen, it may allow an attacker to act without knowing the original password. This is why sign-out, expiry, revocation, secure storage, and transport protection matter.

The distinction resembles a hotel check-in. A guest proves identity at the front desk and receives a room key. The key is not the guest's identity, but possession of it may open specific doors for a limited period. Losing the key can create risk even though the guest's passport remains safe.

Chapter 2: Identities, Accounts, Principals, and Credentials

2-7 From authentication event to session and token

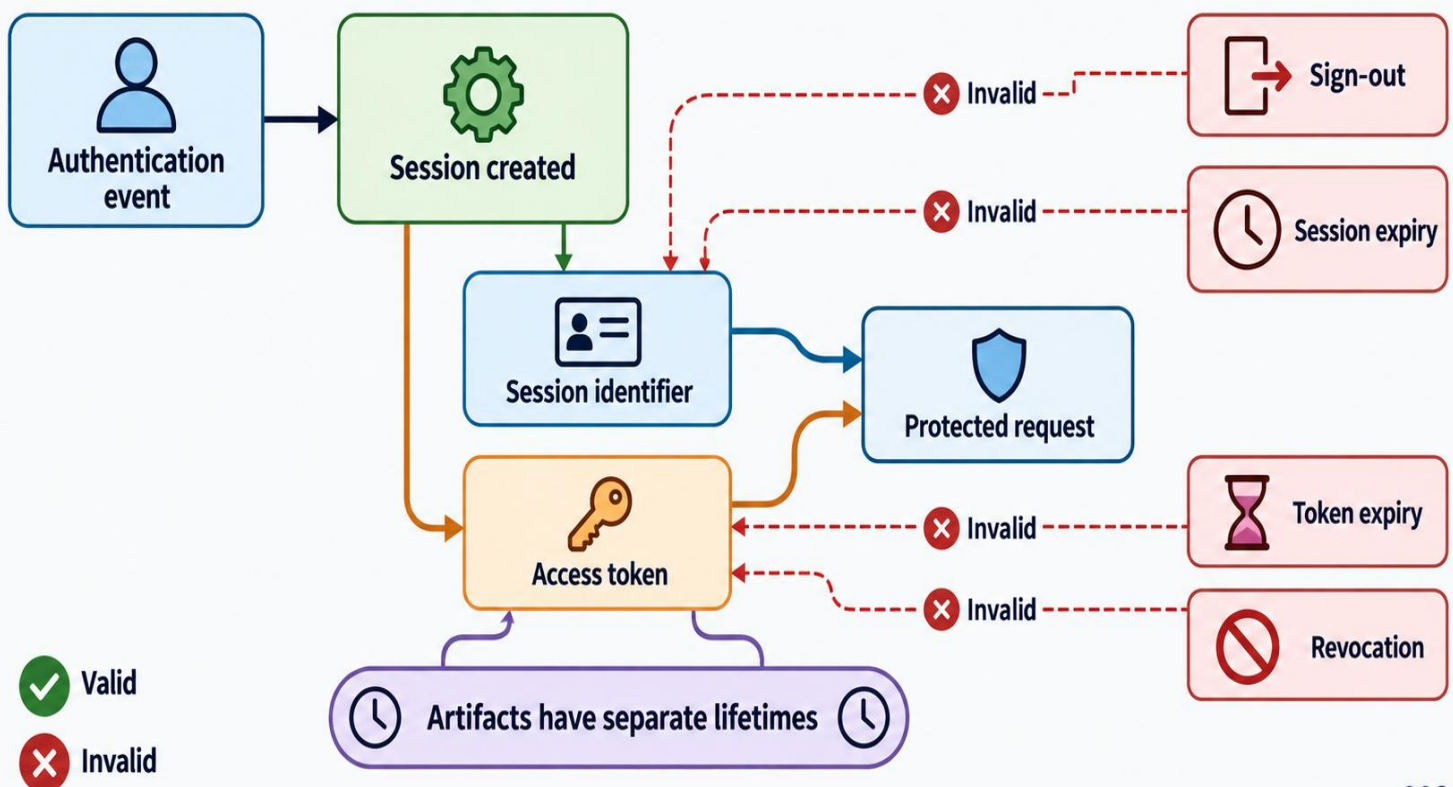


Figure 2-3: From authentication event to session and token

Tenants, domains, realms, and trust boundaries

Identity systems often divide their administrative or naming space.

A **tenant** is an isolated organizational instance within a shared service, typically with its own identities, policies, and administrators. A cloud-based business application may host many customer tenants on the same platform.

A **domain** is a scoped administrative or naming area. Depending on the technology, it may refer to a directory domain, a Domain Name System namespace, an email suffix, or another boundary. The exact meaning must be determined from context.

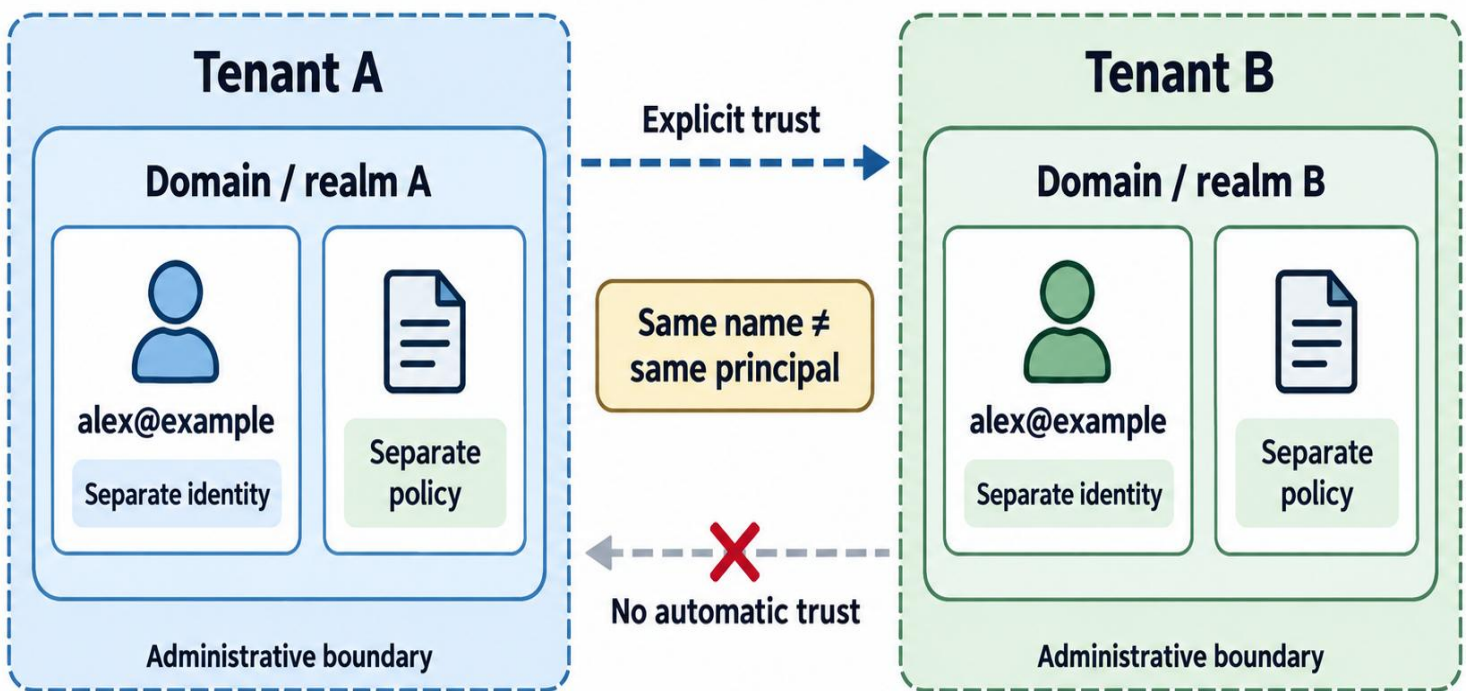
A **realm** is an authentication or security domain within which a set of identities and trust rules applies. The term is common in Kerberos and some application platforms.

These boundaries help answer “unique where?” and “trusted by whom?” The username `alex` may be unique within one local host but not across an enterprise. An account trusted in one tenant is not automatically trusted in another.

Crossing a boundary requires an integration or trust relationship. A system should not accept a familiar-looking identifier from another domain without verifying the issuer and the intended trust.

Chapter 2: Identities, Accounts, Principals, and Credentials

2-8 Tenants, domains, realms, and trust boundaries



009

Figure 2-4: Tenants, domains, realms, and trust boundaries

Identity proofing and enrollment

Identity proofing is the process of establishing that an identity record corresponds to the intended real-world person or entity. **Enrollment** is the process of registering credentials or authenticators for that identity.

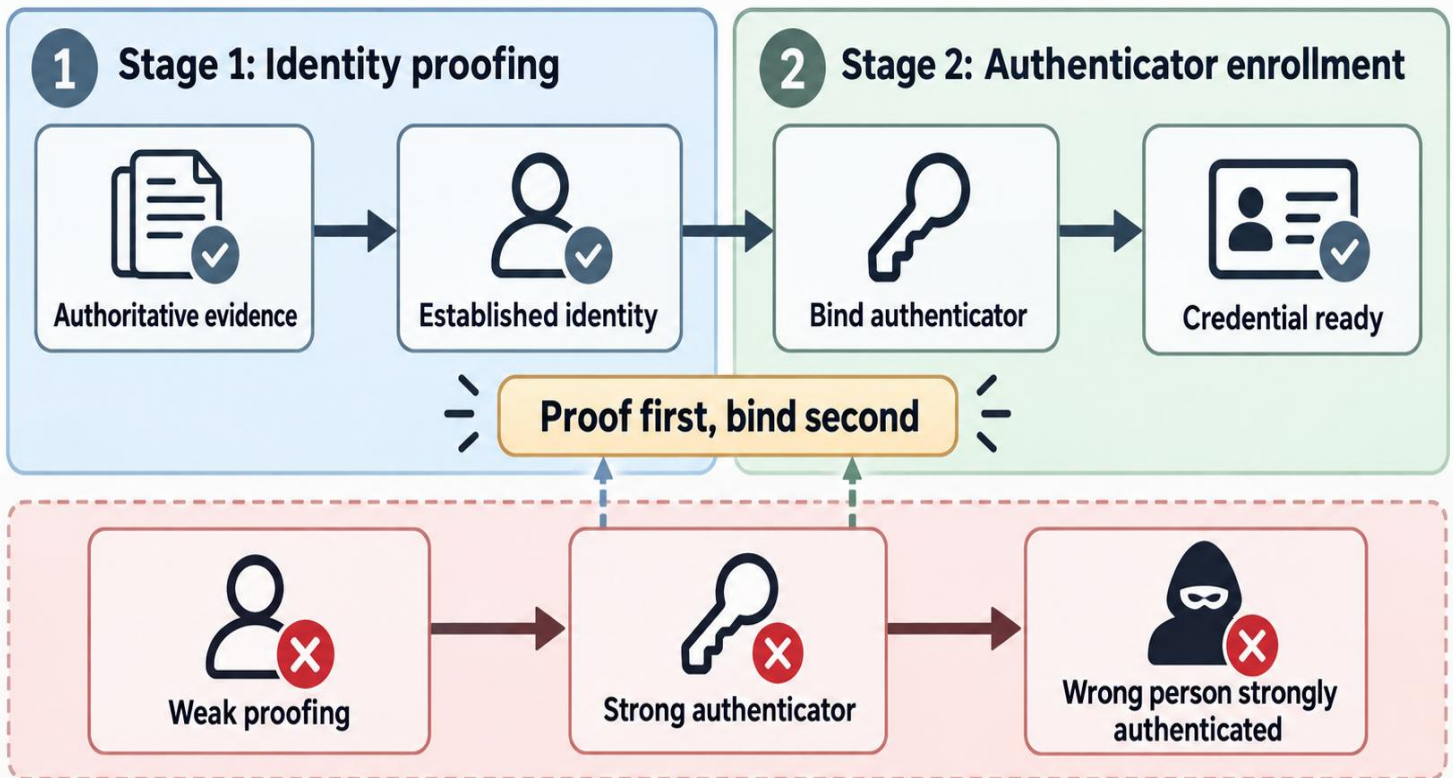
For a new employee, proofing may begin with hiring and onboarding checks, followed by a controlled process that links the employee record to a directory identity. Enrollment then gives the employee an initial sign-in method and registers stronger authenticators.

Proofing and enrollment are important because strong authentication only proves control of an enrolled credential. If a credential was issued to the wrong person, the authentication system can work perfectly and still authenticate the wrong individual.

Recovery and replacement are effectively re-enrollment events. A help desk that resets credentials must gain sufficient confidence that the requester is the legitimate account owner. A weak recovery process can become an easier attack path than the primary sign-in method.

Chapter 2: Identities, Accounts, Principals, and Credentials

2-9 Identity proofing and authenticator enrollment



010

Figure 2-5: Identity proofing and authenticator enrollment

Mapping one person across systems

Consider Maya Chen, an engineer joining a platform team.

Human-resources record

The human-resources system holds the authoritative workforce identity. It records a stable employee identifier, active status, team, manager, start date, and other employment facts.

Directory account

An automated process creates a directory account linked to the employee identifier. The account has a sign-in name, display name, email attribute, group memberships, and authentication methods.

Cloud role session

Maya signs in through the company identity provider and requests a staging role. The cloud platform creates a temporary role session. That session is a principal with time-limited permissions; it is not a new human identity.

Linux user identity

A centrally managed Linux server resolves Maya to a numeric user identifier and group memberships. File access and process ownership are evaluated using the operating system's identity representation.

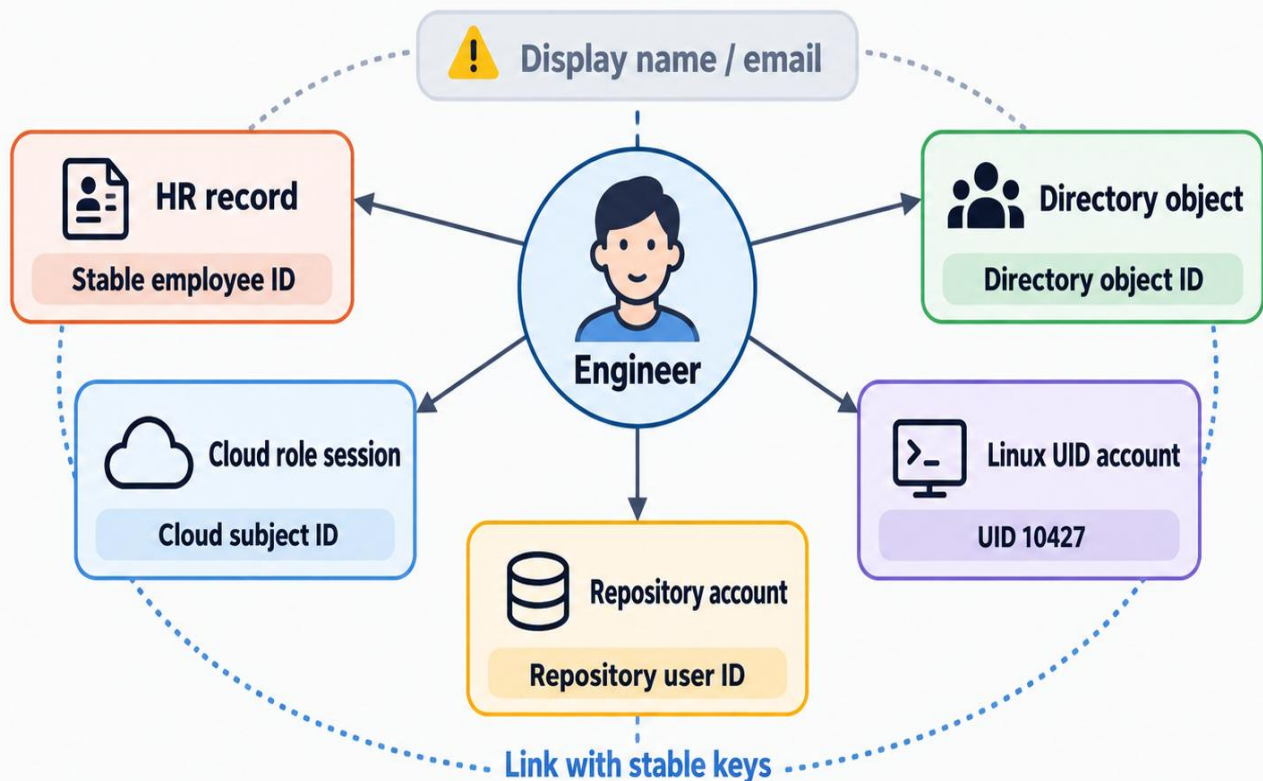
Software-repository account

A repository service maintains an account linked through federation or provisioning. The account may belong to project teams and receive repository roles that differ from cloud and Linux permissions.

The records should be linkable, but each system has its own account, authorization model, session state, and logs. Disabling only one account may leave other access paths active.

Chapter 2: Identities, Accounts, Principals, and Credentials

2-10 One engineer represented across multiple systems



011

Figure 2-6: One engineer represented across multiple systems

Ownership and purpose

Every identity and account should answer two operational questions:

- **Who is responsible for it?**
- **Why does it exist?**

For a human identity, the owner is usually the person, while a manager or sponsor is responsible for approving continued access. For a workload identity, an engineering team or service owner must be accountable because software cannot review its own permissions or respond to credential exposure.

Purpose should be specific enough to guide permissions and lifecycle decisions. "Automation account" is weak. "Deploys the payments API from the approved production pipeline" is better because it indicates where the account should be used and what access it may require.

Misunderstandings that create long-term risk

One person does not imply one account

Separate administrative and everyday accounts can be a good design. The requirement is not a single account; it is complete ownership, traceability, and lifecycle management.

One account does not always imply one person

Workload and device accounts are legitimate. Shared human accounts, however, reduce accountability and should be replaced where practical.

A visible name is not a stable identity key

Names, email addresses, and usernames can change. Use a stable system identifier to maintain relationships, and treat visible attributes as descriptive data.

A credential is not a permission

Possessing a password or private key may prove control of an account. Authorization still determines what the resulting principal may do.

A session can outlive the event that created it

Changing a password or disabling one sign-in method may not immediately invalidate all active sessions or tokens. Lifecycle and incident procedures must account for temporary credentials already in circulation.

Precise vocabulary is not academic decoration. It enables accurate access reviews, safer automation, and faster troubleshooting. When an incident occurs, the team must know whether it is revoking an identity, disabling an account, replacing a credential, terminating a session, or removing an entitlement. Those actions are related, but they are not interchangeable.