Securing Access

SECURITY PRINCIPLES · IAM USERS, GROUPS & ROLES · POLICIES & EVALUATION

STUDY & REVIEW

01 MUST KNOW

if you remember five things, remember these

01 **Security is shared:** AWS is responsible for security **OF** the cloud (global infrastructure — Regions, Availability Zones, edge locations — plus foundation services, the host OS, and physical facilities). You are responsible for security **IN** the cloud (your data, IAM configuration, guest OS and firewall/network settings, and encryption). What you must do depends on the services you use.

02 **Least privilege:** grant only the permissions a task requires. **Start with a minimum set, grant more as needed, and revoke what is unused.** Starting minimal and adding is more secure than starting broad and trying to restrict later — it is the core of a strong identity foundation.

03 **Authentication vs. authorization: authentication** establishes *who* is requesting access (credentials — a person or an application); **authorization** decides *what* an authenticated requester may do (allow or deny). IAM does both and is **secure by default** — no access until permissions are explicitly granted.

04 **The four building blocks:** an IAM **user** is an identity with long-term credentials; an IAM **group** shares one set of permissions across many users; an IAM **role** hands out *temporary* credentials and is assumed by a person, app, or service; an IAM **policy** is the JSON document that defines the permissions. Best practice: attach policies to groups.

05 **How a request is decided:** by default **all requests are denied** (implicit deny); an **explicit allow** overrides that default; an **explicit deny** overrides any allow. Order of evaluation does not matter and the most restrictive result wins. The account **root user** has full access — protect it and use it only for tasks nothing else can do.

02 KEY TERMS

TERM	DEFINITION
IAM	AWS Identity and Access Management — configures fine-grained access to AWS resources; integrates with most services, supports federation and MFA
Principal	A person or application that signs in and makes requests to AWS (includes the root user, IAM users, assumed roles, and federated users)
IAM user	An identity representing one person or application; given a permanent set of credentials that stays until a forced rotation
IAM group	A collection of IAM users who are granted identical permissions through a policy attached to the group
IAM role	An identity that grants a temporary set of permissions; not tied to one person; assumed by a person, application, or service
IAM policy	A JSON document that defines which resources can be accessed and the level of access; attached to a user, group, role, or resource
IAM entity	The IAM objects AWS uses for authentication — users and roles
IAM identity	The IAM objects that can be authorized in policies to perform actions — users, groups, and roles
Root user	The single sign-in identity created with the account; has complete access to all services and resources — must be protected
MFA	Multi-factor authentication — an extra sign-in factor delivered to a hardware or virtual device (for example, Google Authenticator)
Identity-based policy	A policy attached to an IAM user, group, or role that indicates what that identity can do

TERM	DEFINITION
Resource-based policy	A policy attached to an AWS resource that indicates who is permitted to do what to the resource; must name a Principal

03 IAM USER VS. IAM ROLE

long-term identity vs. temporary, assumable identity

IAM USER

a permanent identity for one person or application

- Represents one specific person or application
- Has **long-term credentials** — a password and/or access keys that stay until rotated
- Credentials are unique to that user and authenticate it directly
- Best practice: place users in **groups** and attach the policy to the group

IAM ROLE

a temporary identity that is assumed, not owned

- Not tied to one person — assumable by a person, app, or AWS service
- Provides **temporary** session credentials; no long-term password or keys
- On assuming a role, the entity's prior permissions are set aside
- Used to delegate access: EC2 apps, cross-account, federated and mobile users

04 IDENTITY-BASED VS. RESOURCE-BASED POLICIES

the difference is where the policy is attached

ASPECT	IDENTITY-BASED	RESOURCE-BASED
Attached to	An IAM user, group, or role	An AWS resource (for example, an S3 bucket)
Answers	What can this identity access?	Who can access this resource?
Principal element	Implied — the identity it is attached to	Required — names the account, user, or role allowed or denied
Typical use	Grant a user access to a DynamoDB table	Grant cross-account access to an S3 bucket

05 HOW IAM EVALUATES A REQUEST

explicit deny always wins; default is deny

`01` Start: all requests denied by default → `02` Explicit **Deny** in any policy? If yes, deny wins → `03` Otherwise, an explicit **Allow**? If yes, allow → `04` No allow found → fall back to the implicit deny → `05` Net rule: allowed only if explicitly allowed and never explicitly denied

06 ANATOMY OF AN IAM POLICY

JSON document; each statement = one permission

ELEMENT	WHAT IT SPECIFIES
Version	The policy language version — for example, `2012-10-17`
Statement	Container for one or more permission statements (combined with a logical OR)
Effect	`Allow` or `Deny`
Principal	Resource-based: the account, user, role, or federated user to allow or deny. Identity-based: implied, so omitted
Action	The action(s) allowed or denied — for example, `s3:GetObject`
Resource	The resource(s) the actions apply to, given by ARN (AWS Resource Name)
Condition	Optional — circumstances that must be met for the statement to apply

07 EXAM TRAPS

where the knowledge check gets you

- × “An explicit allow always grants access.” — No. An **explicit deny overrides any explicit allow**, and the most restrictive policy wins — even a request an identity policy allows can be blocked by a deny elsewhere.
- × “If no policy mentions an action, it is allowed.” — No. By default **every request is denied** (implicit deny); access requires an explicit allow that is not overridden by a deny.
- × “An IAM role has a long-term password and access keys, like a user.” — No. A role has **no long-term credentials**; assuming it returns **temporary** session credentials.
- × “Use the root user for day-to-day administration.” — No. The root user has complete access — **enable MFA, protect it, and use it only for tasks no other user can do**; create an admin user for daily work.
- × “You can sign in to the AWS CLI with your username and password.” — No. The **console** uses username + password; the **CLI, SDKs, and API** require an **AWS access key** (access key ID + secret access key).
- × “Attach policies directly to each individual user.” — It works, but the best practice is to **attach the policy to a group and add users to the group**, so permissions are managed once and follow membership.
- × “A resource-based policy can omit the Principal, like an identity-based one.” — No. A **resource-based policy must name the Principal**; only in an identity-based policy is the Principal implied.
- × “Encryption in transit and at rest are the same mechanism.” — Different: **in transit** protects moving data with a cryptographic protocol such as TLS; **at rest** protects stored data with client-side or server-side encryption.

08 SELF-QUIZ

answers are upside-down below

Q1 In the shared responsibility model, who patches the guest operating system on an EC2 instance — AWS or the customer?

Q2 What is the difference between authentication and authorization?

Q3 Which IAM building block provides temporary credentials and can be assumed by a person, application, or service?

Q4 What credential authenticates a call made through the AWS CLI or an API, and what two parts make it up?

Q5 By default, is a request that no policy explicitly addresses allowed or denied?

Q6 When an explicit allow and an explicit deny both apply to the same action, which one wins?

Q7 Which policy type is attached to an AWS resource and must specify a Principal?

Q8 What is the best-practice way to grant the same permissions to many users?

Q9 Name three of the seven core elements of an IAM policy statement.

Q10 Sample exam: a developer team whose membership will change needs identical access for 9 months — what design is best?

ANSWER KEY (rotate the page)

Q1 The customer (security IN the cloud) **Q2** Authentication establishes who is requesting (via credentials); authorization determines what they are allowed to do (allow or deny) **Q3** An IAM role **Q4** An AWS access key — an access key ID plus a secret access key **Q5** Denied — an implicit (default) deny **Q6** The explicit deny — the most restrictive policy is applied **Q7** A resource-based policy **Q8** Attach the policy to an IAM group and add the users to that group **Q9** Any three of: Version, Statement, Effect, Principal, Action, Resource, Condition **Q10** An IAM user per developer, all placed in one IAM group, with the policies attached to the group