### **AWS Identity and Access Management (IAM)**

### **1. Overview of IAM**

* AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS services and resources.
* IAM enables you to manage permissions for users, groups, and roles, defining who can access what within your AWS environment.
* It’s a fundamental part of AWS security, allowing for fine-grained access control.

#### **2. Key Concepts in IAM**

* **Users:**
  + Represents a person or application that interacts with AWS services.
  + Each user can have individual security credentials (passwords, access keys).
  + Users are created to perform tasks within AWS and can be assigned specific permissions.
* **Groups:**
  + A collection of IAM users. Groups allow you to manage permissions for multiple users simultaneously.
  + Users in a group inherit the permissions that are attached to the group.
  + Examples of groups might include "Developers," "Admins," or "Billing."
* **Roles:**
  + A set of permissions that can be assumed by a user, application, or service to perform tasks on AWS resources.
  + Roles allow for temporary access and do not require credentials (passwords or access keys).
  + Commonly used for granting permissions to AWS services like EC2 or Lambda or for cross-account access.
* **Policies:**
  + Documents that define permissions. Policies specify what actions are allowed or denied on which resources.
  + Policies are written in JSON format and can be attached to users, groups, or roles.
  + Two types of policies:
    - **Managed Policies:** Created and managed by AWS or the user. These can be reused across multiple entities.
    - **Inline Policies:** Embedded directly into a user, group, or role. Typically used for specific, one-off permissions.
* **Principals:**
  + An entity (user, group, or role) that can make a request to perform an action on an AWS resource.
  + A principal is authenticated using credentials (passwords, access keys, etc.).
* **Actions and Resources:**
  + IAM policies define what actions (e.g., s3:PutObject) are allowed or denied on which resources (e.g., arn:aws:s3:::my-bucket).
* **Effect:**
  + IAM policies contain an "Effect" element that specifies whether the policy allows (Allow) or denies (Deny) access.
  + Explicit denies take precedence over allows, providing a way to block access even if other policies allow it.
* **Conditions:**
  + Policies can include conditions that limit when the policy is in effect, based on factors like IP addresses, time of day, or whether MFA is used.
  + Conditions use operators like StringEquals, IpAddress, and Bool.

#### **3. Authentication Methods in IAM**

* **Password-Based Authentication:**
  + For users accessing the AWS Management Console.
  + Password policies can be set to enforce complexity, rotation, and reuse rules.
* **Access Keys:**
  + For programmatic access via AWS CLI, SDKs, or APIs.
  + Access keys consist of an Access Key ID and Secret Access Key.
* **Multi-Factor Authentication (MFA):**
  + Adds an extra layer of security by requiring a second form of authentication (like a code from a mobile app).
  + MFA can be enforced for user accounts or roles that require elevated security.
* **Federation:**
  + Allows external identities (from corporate directories, Google, etc.) to access AWS without needing individual IAM users.
  + Supports SAML 2.0 for single sign-on (SSO) and identity federation.
* **Temporary Security Credentials:**
  + Generated by AWS Security Token Service (STS) for short-term access.
  + Often used in conjunction with IAM roles for services like EC2, Lambda, or cross-account access.

#### 

#### 

#### 

#### **4. Permissions and Access Control**

* **Principle of Least Privilege:**
  + Best practice to grant only the minimum permissions necessary for a user or role to perform their tasks.
* **Service Control Policies (SCPs):**
  + Used with AWS Organizations to define the maximum available permissions for accounts in the organization.
  + SCPs can prevent users or roles from performing restricted actions, even if an IAM policy allows it.

#### **5. IAM Best Practices**

* **Use IAM Roles for Applications and Services:**
  + Instead of embedding credentials in applications, use IAM roles to grant permissions dynamically.
  + This reduces the risk of credentials being compromised.
* **Enable MFA for Privileged Users:**
  + Require multi-factor authentication for root accounts and users with administrative access.
* **Regularly Rotate Credentials:**
  + Rotate access keys and passwords periodically, and remove any unused credentials.
* **Review and Audit Permissions:**
  + Regularly review IAM policies and permissions to ensure they align with the principle of least privilege.
  + Use AWS IAM Access Analyzer to identify resources that are shared with external entities.
* **Use AWS Managed Policies:**
  + Leverage AWS managed policies for common use cases to simplify permissions management and stay updated with best practices.
* **Enable CloudTrail for Logging:**
  + Use AWS CloudTrail to log all API calls and changes to IAM policies and roles for auditing and monitoring.
* **Least Privilege and Segregation of Duties:**
  + Implement segregation of duties by assigning distinct roles for administrative tasks, and ensure that users only have permissions necessary for their role.

#### 

#### 

#### 

#### 

#### **6. IAM Identity Center (formerly AWS Single Sign-On)**

* Provides centralized access management for AWS accounts and applications.
* Allows users to sign in with their corporate credentials or use single sign-on (SSO) to access multiple AWS accounts.
* Simplifies management of access across multiple AWS accounts in an organization.

#### **7. Cross-Account Access and Trust Relationships**

* **Cross-Account Access:**
  + IAM roles can be used to grant access between AWS accounts.
  + For example, you can set up a role in Account A that Account B can assume to access resources in Account A.
* **Trust Policies:**
  + Define which entities (users, roles, or accounts) can assume a role.
  + Trust relationships allow you to specify the conditions under which access is granted.

#### **8. IAM Access Analyzer**

* A tool that helps identify resources shared with external entities.
* Scans policies attached to resources like S3 buckets, IAM roles, and Lambda functions to identify any that are accessible from outside your AWS account.
* Provides actionable insights to help you secure your AWS environment.

#### **9. IAM Quotas and Limits**

* **User Limits:**
  + 5,000 users per AWS account by default.
* **Role Limits:**
  + 1,000 roles per AWS account by default.
* **Policy Limits:**
  + Managed policies can have up to 6,144 characters.
  + Inline policies are limited by the entity they are attached to (e.g., 2,048 characters for roles).
* **Groups:**
  + A user can belong to up to 10 groups.

#### **10. IAM Policy Simulator**

* A tool that allows you to test and validate the effects of IAM policies before applying them.
* Helps to ensure that policies work as intended and do not grant excessive permissions.

#### 

#### **11. Advanced Features and Integrations**

* **Resource-Based Policies:**
  + These policies are attached directly to resources (like S3 buckets, SNS topics) and control access at the resource level.
* **Session Tags and Session Policies:**
  + Tags can be used to pass metadata through IAM roles, allowing you to control access based on session-specific attributes.
  + Session policies further restrict the permissions granted by a role when assumed.

#### **12. Common Use Cases**

* **Securing the Root Account:**
  + Best practice is to avoid using the root account for everyday tasks. Enable MFA and use IAM roles or users instead.
* **Delegating Access to Services:**
  + Use IAM roles to delegate permissions to services like EC2, Lambda, or ECS, allowing them to interact with other AWS services securely.
* **Cross-Account Access for Third-Party Audits:**
  + Create roles with restricted permissions that can be assumed by third-party auditors, allowing them to access only what is necessary.

### **13. IAM Roles with EC2 Instance Profiles**

* When you assign an IAM role to an EC2 instance, the role is associated with an instance profile.
* The instance profile allows the EC2 instance to assume the IAM role and gain the permissions assigned to it.
* Example: If an EC2 instance needs to access S3, you create an IAM role with the necessary permissions and associate it with the instance profile.

### **14. IAM Service-Linked Roles**

* Some AWS services automatically create IAM roles that are linked directly to the service. These are known as service-linked roles.
* Service-linked roles are pre-configured with the necessary permissions for the service to function, and you don't need to manually configure them.
* Example: AWS Elastic Beanstalk automatically creates a service-linked role that allows the service to manage resources on your behalf.

### **15. IAM Access Advisor**

* Access Advisor shows the services that a user or role has accessed and when those services were last accessed.
* This information can help you decide which permissions to keep or remove, ensuring that users or roles only have the necessary permissions.
* Integrated into the IAM console, Access Advisor helps in refining permissions based on actual usage.

### **16. IAM Identity Providers**

* IAM supports integration with external identity providers (IdPs) like Microsoft Active Directory, Okta, or Google Workspace.
* This allows users to federate into AWS using their existing corporate credentials.
* **SAML 2.0:** IAM supports Security Assertion Markup Language (SAML) 2.0, enabling single sign-on (SSO) for enterprise users.
* You can configure IAM to trust an external IdP for authentication, which simplifies user management and access.