**Identity and Access Management (IAM)**

**Identity and Access Management (IAM) Overview**

* **Identity and Access Management (IAM) Overview**: IAM is a framework of policies and technologies for ensuring that the right individuals have the appropriate access to technology resources. It involves managing user identities and their access to AWS services and resources.
* **Authentication and Authorization**: Authentication verifies the identity of a user, while authorization determines what resources the user can access. AWS IAM uses policies to manage permissions and control access.
* **Multifactor Authentication (MFA)**: This adds an extra layer of security by requiring users to provide multiple forms of verification before accessing resources.
* **Permissions and Policies**: IAM policies define permissions for users, groups, and roles. These policies specify what actions are allowed or denied on which resources.
* **Best Practices**: This includes creating individual user accounts, enabling MFA, rotating credentials regularly, and following the principle of least privilege, which means giving users only the permissions they need to perform their tasks.

**Principals**

* **Principals (Identities)**: These are entities that can take actions on AWS resources. They include users, groups, and roles.
  + **Users**: Individual people or services that interact with AWS. Users have credentials (username, password, and access keys) to log in and perform actions.
  + **Groups**: Collections of users that share the same permissions. Managing permissions at the group level simplifies administration.
  + **Roles**: Identities with a set of permissions that can be assumed by users or services. Roles are not permanently assigned to any entity and are useful for granting temporary access.
* **Permissions and Policies**: Permissions define what actions principals can perform on resources. Policies are JSON documents that specify these permissions. Policies can be attached to users, groups, or roles.
* **Federated Identity**: This allows users from an external identity system to access AWS resources without creating separate AWS user accounts. It often involves single sign-on (SSO).
* **Access Keys**: Used for programmatic access to AWS services via APIs or the command line interface (CLI). They consist of an access ID and a secret access key.
* **Best Practices**: Use groups for easier permission management, and decide whether to create a user or a role based on the specific needs of the entity requiring access.

These concepts help manage and secure access to AWS resources effectively.

**Root user**

* **Root User**: This is the initial account created when you set up an AWS account. It has unlimited capabilities and can perform any action within AWS.
* **Unlimited Capabilities**: The root user can change billing information, start and stop instances, work with buckets, and perform any other action in AWS.
* **Security Best Practices**:
  + **Avoid Daily Use**: Do not use the root user for everyday tasks to minimize the risk of accidental changes or security breaches.
  + **Create IAM Admin Users**: Use IAM admin users for daily administrative tasks instead of the root user.
  + **Secure Storage**: Store the root user credentials securely, using encryption and password protection.
* **Specific Tasks for Root User**: Some tasks require root user access, such as modifying the root user, changing the AWS support plan, closing the AWS account, creating a CloudFront key pair, enabling MFA on an S3 bucket, and restoring permissions for other IAM users.
* **Comparison with Other Systems**: Similar to root access in Linux or administrative access in Windows, the AWS root user has all-powerful capabilities but should be used sparingly.

These points help manage and secure the AWS root user effectively.

**Authentication**

* **Authentication**: This is the process of validating the identity of a user or process. It involves verifying credentials, such as a username and password, to ensure the entity is who they claim to be.
* **Single-Factor Authentication**: This involves one form of verification, such as a password.
* **Multi-Factor Authentication (MFA)**: This adds an extra layer of security by requiring multiple forms of verification, such as a password and a thumb scan or a retina scan.
* **User Authentication**: In AWS, users authenticate using a username and password to access the management console. For API and CLI access, users use an access key and a secret key.
* **Process Authentication**: Processes authenticate using roles, which have the necessary permissions to perform actions on AWS resources.
* **Anonymous Access**: Some AWS services, like S3 buckets, can be configured to allow anonymous access, meaning no authentication is required for certain actions.
* **Key Points for Exam**: Remember that for the management console, a username and password are used, while for API and CLI access, an access key and secret key are used.

These concepts are crucial for managing access and security in AWS.

Top of Form

**Authorization policies**

* **Authentication vs. Authorization**:
  + **Authentication** verifies who you are (e.g., logging in with a username and password).
  + **Authorization** determines what you can do once authenticated (e.g., accessing specific resources).
* **Policies**:
  + Policies are rules that define what actions are allowed or denied for users, groups, or roles in AWS.
  + They are written in JSON but can be managed through the AWS GUI.
* **Types of Policies**:
  + **Identity-based policies**: Linked to users, groups, or roles.
  + **Resource-based policies**: Linked directly to AWS resources, often used for cross-account access.
* **Policy Processing**:
  + By default, all actions are denied.
  + Explicit allows can override this default.
  + Permission boundaries and explicit denies can further restrict actions.
* **Granularity**:
  + Policies can be very granular, specifying exact actions allowed on specific resources.

Understanding these concepts will help you manage permissions and ensure security within your AWS environment.

Top of Form

**Multifactor authentication**

* **Multifactor Authentication (MFA)**: This is an additional security layer used to ensure that people trying to gain access to an online account are who they say they are. It requires more than just a username and password.
* **Factors of Authentication**:
  + **Something You Know**: Typically a password or PIN.
  + **Something You Have**: A physical device like a phone or a security token.
  + **Something You Are**: Biometrics like fingerprints, retina scans, or voice recognition.
* **Implementation in AWS**:
  + AWS supports various MFA methods, including virtual MFA devices (free), SMS MFA devices (in preview), and hardware MFA devices (with a fee).
  + Virtual MFA devices use apps on smartphones or tablets that support the TOTP standard.
  + SMS MFA sends a code to your mobile device.
  + Hardware MFA devices include security keys, key fobs, and display cards.
* **Best Practices**:
  + Always enable MFA for enhanced security, especially for the root user account.
  + Choose the right form factor based on your budget and security requirements.

By using MFA, you add an extra layer of security, making it harder for unauthorized users to access your AWS resources.

**Key rotation**

* **Access Keys**: These are used to access AWS services from the command line or APIs. They consist of an Access Key ID and a Secret Access Key.
* **Importance of Key Rotation**: Regularly rotating keys helps prevent unauthorized access, especially if keys are accidentally exposed (e.g., uploaded to a public repository like GitHub).
* **Key Rotation Process**:
  1. **Create a New Key**: Generate a new access key while keeping the old one active.
  2. **Update Applications**: Update all applications to use the new key.
  3. **Deactivate Old Key**: Once all applications are updated, deactivate the old key.
  4. **Validate**: Ensure all applications are functioning correctly with the new key.
  5. **Delete Old Key**: Finally, delete the old key once everything is confirmed to be working.
* **Using CLI for Key Rotation**: The video also shows how to use the AWS Command Line Interface (CLI) to list and create access keys, which can be more efficient than using the graphical interface.

Following these steps helps maintain security and ensures that your AWS environment remains protected.

**Multiple permissions**

* **Multiple Permissions**: Permissions in AWS IAM can come from different sources, such as user accounts and groups. All "allow" permissions are cumulative, meaning they add up. However, "deny" permissions override any allows, meaning if something is denied, it stays denied regardless of other permissions.
* **Permission Boundaries**: These do not grant permissions but set limits on what permissions can be granted. They act as a fence, ensuring that users cannot exceed the specified boundaries. For example, if a boundary allows actions only in S3 and EC2, any permissions granted outside these services (like RDS) will not be effective.
* **Effective Permissions**: The actual permissions a user has are the intersection of the permissions granted and the boundaries set. This ensures that no permissions outside the boundary are effective.

Understanding these concepts helps in managing and securing access within AWS effectively.

**AWS compliance program**

* **Compliance**: This refers to ensuring that an information system meets defined security best practices and standards. Compliance programs are set by governments, industries, and independent bodies.
* **AWS Compliance**: AWS provides extensive documentation on how to use their services to meet various compliance requirements. They also have third-party certifications to prove their compliance with standards like FIPS 140-2, ISO 9001, and PCI DSS Level 1.
* **AWS Compliance Website**: You can find detailed information on AWS compliance programs at aws.amazon.com/compliance. This includes certifications and attestations, as well as region-specific compliance requirements.
* **Key Takeaway for Exam**: Understand that AWS has been externally certified for many compliance programs, and you need to ensure your AWS setup complies with relevant standards. You don't need to memorize all services under each compliance program, just know how to find and use the documentation.

This video emphasizes the importance of compliance in maintaining secure AWS environments and how AWS supports users in meeting these standards.

**AWS Security Hub**

* **AWS Security Hub**: A centralized service that helps you manage and monitor your AWS security and compliance. It consolidates findings from various AWS services and third-party tools.
* **Compliance Checks**: Security Hub runs automated compliance checks to ensure your AWS environment meets regulatory and corporate policies, such as HIPAA and Sarbanes-Oxley.
* **Centralized Findings**: It gathers security findings from services like Amazon GuardDuty (intrusion detection), Amazon Inspector (vulnerability scans), and Amazon Macie (S3 bucket policies) into one place.
* **Automated Actions**: Security Hub can automatically take actions if severe security issues are detected, such as making a non-compliant database unavailable.
* **30-Day Free Trial**: You can try AWS Security Hub for 30 days for free to see if it fits your needs. After that, it charges based on the number of compliance checks and findings.

Understanding these concepts helps you effectively use AWS Security Hub to maintain security and compliance in your AWS environment.

**Shared responsibility model**

* **AWS Responsibilities**:
  + **Security of the Cloud**: AWS is responsible for the physical security of data centers, including servers, network devices, and the hypervisor.
  + **Managed Services**: AWS secures managed services like DynamoDB, Redshift, and Aurora.
* **Customer Responsibilities**:
  + **Security in the Cloud**: Customers are responsible for securing their data, applications, and guest operating systems running on AWS infrastructure.
  + **Configuration Management**: Customers must manage the configuration of services like Virtual Private Cloud (VPC) and ensure proper IAM security for instances.
* **Shared Responsibilities**:
  + **Services like S3**: AWS manages the physical storage, but customers control access and permissions for their data.

Understanding this model helps ensure that both AWS and customers fulfill their respective security roles effectively.Top of Form

Bottom of Form

**Data security control**

* **Data Ownership and Control**: You own and control your data in AWS. This includes deciding where it's stored, how it's secured, and who can access it.
* **Governance**: Data in the cloud should be treated like on-premises data, with proper governance in place. This includes data ingestion, transformation, classification, utilization, retention, and removal.
* **Data Classification**: Classify data based on its sensitivity (e.g., public, internal, sensitive, restricted) to determine appropriate protection measures.
* **Encryption**: Sensitive data, such as Personally Identifiable Information (PII) and organizational trade secrets, should be encrypted to ensure its security.
* **Data Recovery and Retention**: Implement processes for data recovery (e.g., backups, replication) and retain data only as long as necessary, using archives or destruction as needed.

Understanding these concepts helps ensure that your data is properly managed and secured within AWS.

Top of Form

**Federated directory services**

* **Federated Identity**: A system of trust between two parties where one party trusts the other to authenticate users. Examples include logging in with Google or Facebook on other websites.
* **Supported Protocols**: AWS supports protocols like SAML 2.0, OpenID Connect (OIDC), and OAuth 2.0 for federated identity.
* **Federated Directory Services**: Extends federated identity by linking an entire directory of users and groups to AWS. This allows for centralized management of permissions and authentication.
* **AWS Single Sign-On (SSO)**: Now rebranded as IAM Identity Center, it connects existing directories to AWS and can also use an internal directory built into IAM Identity Center.
* **Integration with Active Directory**: IAM Identity Center can connect with Microsoft Active Directory, either through AWS Managed AD or self-managed AD with an Active Directory Connector.
* **AWS Organizations**: Required to enable IAM Identity Center, allowing for centralized management of users and groups across AWS accounts.

These concepts help streamline user management and enhance security within AWS environments.

**Identity and Access Management (IAM) Best Practices** Top of Form

**User accounts**

* **Creating User Accounts**: The video demonstrates how to create user accounts in AWS IAM. It emphasizes the importance of not using the root user account for regular administration. Instead, you should create an admin account for administrative tasks.
* **Admin Account Creation**: To create an admin account, navigate to the AWS Management Console, go to the IAM section, and add a new user. Assign a username, set a password, and attach the necessary permissions, such as the "AdministratorAccess" policy.
* **Groups and Permissions**: You can group users and assign policies to these groups. For example, creating an "Administrators" group and attaching the "AdministratorAccess" policy to it. Users added to this group will inherit these permissions.
* **Access Keys**: After creating a user, you can generate access keys for programmatic access. These keys consist of an access key ID and a secret access key, which can be downloaded and shared securely with the user.
* **Security Best Practices**: The video also highlights the importance of managing unused accounts and ensuring they are purged to avoid security risks.

These steps and best practices help you manage user accounts effectively in AWS IAM.

**Password policies**

1. **Default Password Policies in AWS**:
   * **Minimum Length**: Passwords must be at least 8 characters long.
   * **Maximum Length**: Passwords can be up to 128 characters.
   * **Character Types**: Passwords must include at least three of the following: uppercase letters, lowercase letters, numbers, and special characters (e.g., !, @, #).
2. **Creating Custom Password Policies**:
   * You can set your own rules for password length and character requirements.
   * You can choose to require uppercase, lowercase, numbers, and/or special characters.
   * You can also set password expiration policies to ensure passwords are changed periodically.
3. **Best Practices for Passwords**:
   * **Change Passwords Periodically**: The frequency depends on your organization’s security needs.
   * **Use Unique Passwords**: Avoid using the same password for AWS and other accounts.
   * **Avoid Easily Guessed Passwords**: Don’t use personal information like names or favorite sports teams.
4. **Implementing Policies in AWS**:
   * Navigate to the IAM section in the AWS Management Console.
   * Go to Account Settings and configure your password policies to match your organization’s security requirements.

These practices help ensure that your AWS environment remains secure by enforcing strong password policies.

**Credential rotation**

* **Credential Rotation**: This refers to the process of regularly changing passwords to enhance security.
* **Benefits of Credential Rotation**:
  + **Reduces Vulnerabilities**: Limits the time an attacker has to exploit a compromised password.
  + **Security Awareness**: Reminds users of the importance of security.
  + **Prevents Password Reuse**: Reduces the likelihood of using the same password across multiple systems.
* **Implementing Credential Rotation in AWS IAM**:
  + **Password Expiration**: Set passwords to expire after a certain number of days (e.g., 30 days).
  + **Prevent Password Reuse**: Specify the number of previous passwords that cannot be reused.
  + **Administrator Reset**: Optionally require an administrator to reset expired passwords.
* **Best Practices**:
  + **Understand Organizational Policies**: Ensure your credential rotation policies align with your organization’s security requirements.
  + **Communicate Changes**: Inform users about password policies and expiration periods to avoid confusion.

These practices help maintain a secure AWS environment by ensuring passwords are regularly updated and not reused.

**Principle of least privilege**

* **Principle of Least Privilege**: This security principle states that users should be granted only the access necessary to perform their job functions, and no more. This minimizes the risk of accidental or intentional misuse of permissions.
* **Reducing Vulnerabilities**:
  + **Accidental Actions**: Limiting permissions prevents users from accidentally performing actions they shouldn't.
  + **Malicious Actions**: Restricting access reduces the potential for malicious activities by limiting what an attacker can do if they gain access.
* **Reviewing Permissions**: Regularly review and adjust permissions to ensure they align with the principle of least privilege. AWS provides tools like policy summaries to help analyze and manage permissions effectively.
* **Policy Summaries**: These summaries in AWS IAM help you understand what a specific policy allows, making it easier to ensure permissions are appropriately set.
* **Granular Control**: Use conditions in policies to fine-tune permissions, ensuring users can only perform specific actions under certain conditions.

Implementing the principle of least privilege helps maintain a secure environment by ensuring users have only the access they need and nothing more.

**IAM roles**

* **Understanding IAM Roles**: IAM roles in AWS are used to delegate access to users, applications, or services that don't normally have access to your AWS resources. Roles are similar to user accounts but are intended to be assumable by anyone who needs them.
* **Creating IAM Roles**:
  + Navigate to the IAM section in the AWS Management Console.
  + Go to the "Roles" section and click on "Create role".
  + Choose the type of trusted entity (e.g., AWS service, another AWS account, web identity, or SAML 2.0 federation).
* **Role Types**:
  + **AWS Service Role**: Allows an AWS service (like EC2) to take actions on other AWS services.
  + **Another AWS Account**: Grants access to resources in your account from another AWS account.
  + **Web Identity**: Uses external identity providers (like Cognito or OpenID) for authentication.
  + **SAML 2.0 Federation**: Allows users from a federated identity system to access AWS resources.
* **Best Practices**:
  + **Least Privilege**: Only grant the permissions necessary for the role to perform its tasks.
  + **Naming Conventions**: Use clear and consistent naming conventions for roles to easily identify their purpose.
  + **Review and Manage Roles**: Regularly review roles and their permissions to ensure they are still needed and correctly configured.
* **Example**: Creating a role for an EC2 instance to interact with S3:
  + Select "AWS service" and choose "EC2".
  + Attach the "AmazonS3FullAccess" policy.
  + Name the role (e.g., "EC2\_S3\_FullAccess") and create it.

These concepts help you effectively manage access and permissions in your AWS environment.

Top of Form

**Amazon STS**

* **AWS Security Token Service (STS)**: This service allows you to create temporary security credentials for trusted users to access your AWS resources. These credentials are short-term and expire after a set period, ranging from a few minutes to several hours.
* **Temporary Security Credentials**:
  + **Short-Term**: Created when needed and expire after use, enhancing security.
  + **Dynamic**: Generated on-the-fly, not stored with the user, reducing the risk of credential exposure.
* **Benefits**:
  + **No Long-Term Credentials**: Avoid embedding long-term credentials in applications, which can be a security risk.
  + **Identity Federation**: Allows users from external identity providers (like Active Directory) to access AWS resources without creating separate AWS identities.
  + **EC2 Roles**: Commonly used for EC2 instances to access other AWS services (e.g., S3 buckets) securely.
* **AssumeRole API Call**: This is the key API call used to obtain temporary credentials. It allows applications to assume a role and gain the necessary permissions to access AWS resources.
* **Best Practices**:
  + **Use Regional Endpoints**: To reduce latency, point STS API calls to endpoints in the same region as your resources.
  + **Limit Exposure**: Use temporary credentials to minimize the risk of long-term credential exposure.

These concepts help you securely manage access to your AWS resources using temporary credentials.

Top of Form

**Policy conditions**

* **IAM Policies**: These are rules that define permissions for users, groups, and roles within AWS. They determine what actions are allowed or denied on specific resources.
* **Policy Conditions**: Conditions are additional constraints added to policies to control under what circumstances the policy applies. They provide more granular control over permissions.
* **Creating a Policy with Conditions**:
  + **Service Selection**: Choose the AWS service (e.g., S3) the policy will apply to.
  + **Actions**: Define what actions (e.g., read, write) are allowed.
  + **Resources**: Specify the resources (e.g., S3 buckets) the policy applies to.
  + **Conditions**: Add conditions to further restrict actions. Common conditions include:
    - **MFA Required**: Actions are allowed only if multi-factor authentication is used.
    - **Source IP**: Actions are allowed only from specific IP addresses.
    - **Time-Based Conditions**: Actions are allowed only within a certain time window.
    - **String Matching**: Actions are allowed only if a specific string (e.g., "Marketing") is present in the resource.
* **Visual Editor**: AWS provides a Visual editor to create policies without needing to write JSON code, making it easier to define and manage policies.
* **Review and Apply**: Once the policy is created, review it, give it a meaningful name and description, and apply it to the necessary users, groups, or roles.

These concepts help you create more secure and precise IAM policies by adding conditions that control when and how permissions are granted.

**CloudTrail**

* **Logging and Event Histories**: CloudTrail is a logging service in AWS that records all actions taken within your AWS account. This includes actions from the Management Console, AWS SDK, and command line. These logs are crucial for governance, compliance, auditing, and troubleshooting.
* **Integration with CloudWatch**: CloudTrail can be integrated with CloudWatch to set up alerts based on specific events. This means you can get notifications when certain actions occur, which is useful for monitoring and maintaining your infrastructure.
* **Storing Logs**: By default, CloudTrail retains logs for 90 days. However, you can create a trail to store these logs in an S3 bucket for longer retention. This is useful for long-term analysis and troubleshooting.
* **Creating a Trail**: When creating a trail, you can specify the types of events to log (e.g., read-only, write-only, or all events) and where to store them (e.g., in a new or existing S3 bucket).
* **Security and Compliance**: CloudTrail helps ensure security and compliance by providing a detailed record of all actions taken within your AWS account. This can help you detect unusual activity and ensure that your infrastructure is being used as intended.

These concepts are essential for managing and securing your AWS environment, especially in a software development context where maintaining a secure and compliant infrastructure is critical.

**AWS Control Tower**

* **AWS Control Tower**: This is a service that helps you manage multiple AWS accounts under one umbrella. It simplifies the process of setting up and governing a secure, multi-account AWS environment.
* **Landing Zone**: This is a well-architected, multi-account environment that implements security best practices. Think of it as a secure starting point for your AWS accounts.
* **Guardrails**: These are policies that help you govern your environment. There are two types:
  + **Preventive Guardrails**: These prevent certain actions from happening, ensuring consistency and security.
  + **Detective Guardrails**: These notify you when specific events occur, helping you monitor and respond to changes.
* **Account Factory**: This feature allows you to create accounts based on predefined templates, ensuring consistency across your organization.
* **Dashboard**: Provides insights into your AWS environment, showing you account activities, enabled guardrails, and triggered alerts.
* **Integration with Other Services**: AWS Control Tower uses other AWS services like AWS CloudTrail for logging, AWS Config for configuration management, and Amazon S3 for storage. While AWS Control Tower itself doesn't have additional charges, using these services may incur costs.

In your role in software development, understanding these concepts can help you manage and secure multiple AWS accounts more effectively, ensuring compliance and governance across your projects.

**Service control policies**

* **Service Control Policies (SCPs)**: SCPs manage permissions in an AWS organization, providing centralized control over accounts and their capabilities. They enforce guardrails or limits on what actions can be performed.
* **Allow and Deny Capabilities**: SCPs can include both allow and deny statements. However, many organizations primarily use deny statements to restrict actions.
* **Inheritance**: Deny statements in SCPs apply to all lower levels in the hierarchy. For example, a deny statement at the root level affects all organizational units (OUs) and accounts below it. Allow statements work similarly unless overridden.
* **Implicit Deny**: If the default FullAWSAccess SCP is not attached, the system defaults to implicit deny, meaning no actions are allowed unless explicitly permitted.
* **Deny List and Allow List Strategies**:
  + **Deny List Strategy**: Keeps the default FullAWSAccess SCP in place and applies deny SCPs at lower levels to restrict specific actions.
  + **Allow List Strategy**: Removes the default FullAWSAccess SCP, resulting in an implicit deny at the root level. Allow SCPs are then applied at lower levels to permit specific actions.
* **JSON Structure**: SCPs are defined using JSON format, with key-value pairs and nested elements. This structure allows for easy creation, backup, and reimporting of SCPs.

These concepts are essential for managing permissions and ensuring security within your AWS organization.

**Key security services**

* **AWS Shield**:
  + **Purpose**: Protects against Distributed Denial of Service (DDoS) attacks.
  + **Versions**:
    - **Standard**: Free and provides basic protection.
    - **Advanced**: Paid version with enhanced protection.
  + **Protection Layers**: Covers network (Layer 3), transport (Layer 4), and application (Layer 7) layers.
* **Web Application Firewall (WAF)**:
  + **Purpose**: Acts as an application layer firewall, mainly for HTTP/HTTPS traffic.
  + **Components**:
    - **Web Access Control Lists (ACLs)**: Define rules for allowed communications.
    - **Rule Groups**: Reusable sets of rules to manage traffic.
* **Secrets Manager**:
  + **Purpose**: Manages and retrieves credentials securely.
  + **Benefits**:
    - **No Hard-Coded Credentials**: Credentials are not stored in the application code.
    - **Automatic Rotation**: Credentials can be rotated without manual intervention.
    - **Secure Retrieval**: Applications retrieve credentials over an encrypted channel (HTTPS with TLS).

These services help enhance the security of your AWS environment by protecting against attacks, managing web traffic, and securely handling credentials.

Top of Form

**Additional security services**

* **Amazon Cognito**:
  + **Purpose**: Provides authentication, authorization, and user management for web and mobile apps.
  + **User Pools**: Directories of user profiles, including those authenticated through social identity providers (e.g., Facebook, Google).
  + **Identity Pools**: Provide access to AWS resources, often used in conjunction with user pools for managing user profiles and access.
* **Amazon GuardDuty**:
  + **Purpose**: Security monitoring service that uses machine learning to identify threats.
  + **Key Components**:
    - **Account**: Standard AWS account where GuardDuty is enabled.
    - **Detector**: Represents GuardDuty service and identifies potential threats.
    - **Data Sources**: Includes CloudTrail logs, VPC flow logs, DNS logs, etc.
    - **Findings**: Potential security issues identified by GuardDuty.
    - **Suppression Rules**: Define attributes to suppress certain findings, reducing unnecessary alerts.
* **Amazon Macie**:
  + **Purpose**: Data privacy service that discovers, monitors, and protects sensitive data in S3 buckets.
  + **Features**:
    - **Machine Learning and Pattern Matching**: Identifies sensitive data.
    - **Regex-Based Custom Data Identifiers**: Allows creation of custom patterns to identify specific sensitive data.
    - **Integration with AWS Organizations**: Enables central management across multiple accounts.

These services enhance the security and privacy of your AWS environment by providing robust tools for authentication, threat detection, and data protection.

Top of Form

Bottom of Form

Bottom of Form

Top of Form

Bottom of Form

Top of Form

Bottom of Form

Top of Form

Bottom of Form

Bottom of Form

Bottom of Form

Top of Form

Bottom of Form

Top of Form

Bottom of Form

Top of Form

Bottom of Form

Top of Form

Bottom of Form

Bottom of Form

Bottom of Form

Top of Form

Bottom of Form

Top of Form

Bottom of Form

Top of Form

Bottom of Form

Top of Form

Bottom of Form

Bottom of Form

Bottom of Form

Top of Form

Bottom of Form