ABC Corp Security Policy on IAM

Index

1. Introduction

2. IAM Principles

3. Best Practices for IAM

4. Example IAM Policies

5. Additional Security Aspects

6. Conclusion

1. Introduction

At ABC Corp, security is a top priority. This document outlines our Identity and Access Management (IAM) policies and best practices, designed to protect our AWS environment and align with the AWS Well-Architected Security Pillar. We aim to minimize security risks while enabling efficient and secure operations.

2. IAM Principles

1. Principle of Least Privilege

IAM policies at ABC Corp grant the minimum necessary permissions required for a task. This reduces the risk of unauthorized access.

2. Strong Identity Foundation

We enforce strong authentication methods, such as Multi-Factor Authentication (MFA), to ensure only authorized users have access.

3. Traceability

All IAM activities are logged and monitored to provide an audit trail, aiding in security reviews and incident responses.

4. Regular Policy Review

IAM policies are reviewed regularly to ensure they remain aligned with security best practices and the changing needs of our business.

5. Automated Security Practices

Automation tools enforce consistent IAM policies across all AWS accounts, reducing human error and ensuring best practices are always applied.

3. Best Practices for IAM

1. Implement Role-Based Access Control (RBAC)

Use IAM roles to control access based on the roles users and services play within the organization, ensuring that each role has appropriate permissions.

2. Enforce MFA for All Users

Require MFA for all IAM users, especially those with administrative privileges, to add an extra layer of security.

3. Use IAM Access Analyzer

IAM Access Analyzer helps in identifying and refining overly permissive policies to adhere to the principle of least privilege.

4. Regularly Rotate Access Keys

For cases where long-term credentials are needed, ensure that access keys are rotated regularly to reduce the risk of key compromise.

5. Employ Policy Conditions

Use conditions in IAM policies to restrict actions based on factors such as IP address, time of day, or the presence of MFA.

4. Example IAM Policies

Good IAM Policy Example:

```
{
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Action": [
"s3:ListBucket",
"s3:GetObject"
],
"Resource": [
"arn:aws:s3:::example-bucket",
"arn:aws:s3:::example-bucket/*"
]
}
]
}
```

This policy grants specific S3 access following the least privilege principle.

Overly Permissive IAM Policy Example:

```
{
"Version": "2012-10-17",
```

"Statement": [

{

"Effect": "Allow",

"Action": "*",

"Resource": "*"

}

]

}

This policy is too broad, allowing all actions on all resources, which is a security risk.

5. Additional Security Aspects

1. Data Protection

We use AWS Key Management Service (KMS) to encrypt sensitive data at rest and in transit, ensuring that only authorized users can access it.

2. Incident Response

ABC Corp has an incident response plan that includes steps for detecting, responding to, and recovering from security incidents.

3. Infrastructure Security

Our infrastructure is regularly patched and updated to mitigate vulnerabilities, and we conduct regular security assessments.

6. Conclusion

ABC Corp is dedicated to maintaining a secure environment through the application of IAM best practices and regular reviews. By following the principles and practices outlined in this document, we ensure that our AWS resources are protected from unauthorized access, and that we can respond swiftly to any security incidents.

and management tasks. Customers that deploy an Amazon EC2 instance are responsible for management of the guest operating system (including updates and security patches), any application software or utilities installed by the customer on the instances, and the configuration of the AWS-provided firewall (called a security group) on each instance. For abstracted services, such as Amazon S3 and Amazon DynamoDB, AWS operates the infrastructure layer, the operating system, and platforms, and customers access the endpoints to store and retrieve data. Customers are responsible for managing their data (including encryption options), classifying their assets, and using IAM tools to apply the appropriate permissions.

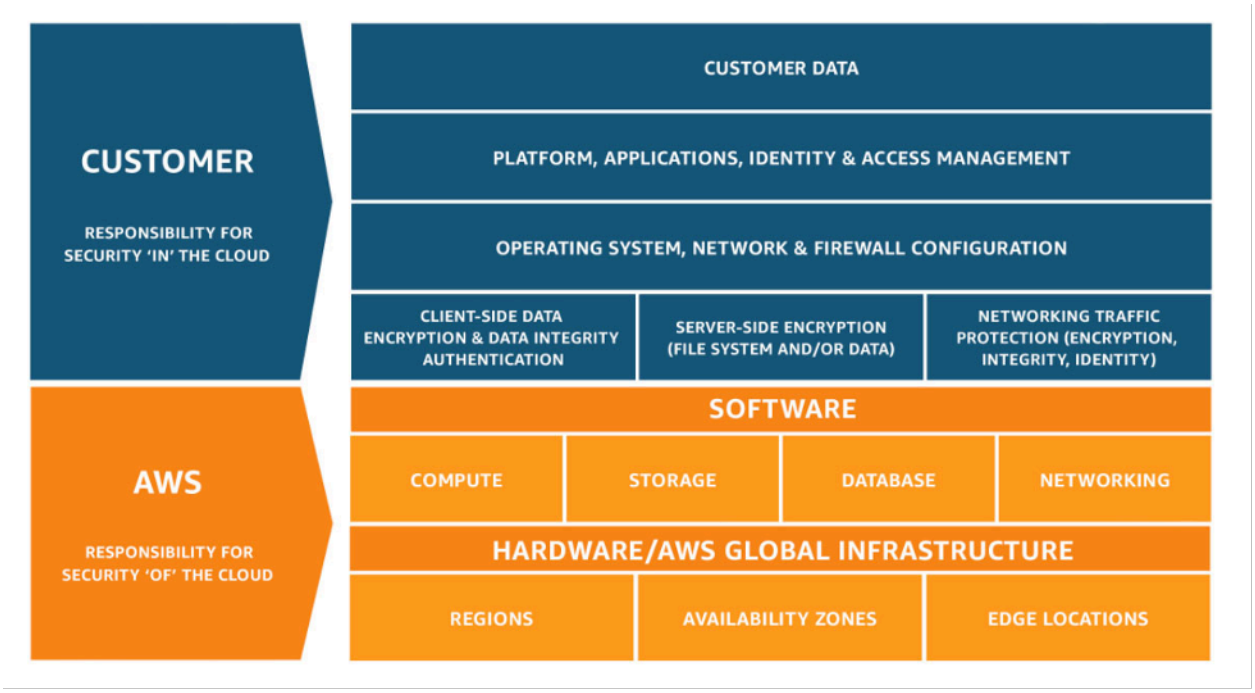

Figure 1: AWS Shared Responsibility Model.

This customer/AWS shared responsibility model also extends to IT controls. Just as the responsibility to operate the IT environment is shared between AWS and its customers, so is the management, operation, and verification of IT controls shared. AWS can help relieve customer burden of operating controls by managing those controls associated with the physical infrastructure deployed in the AWS environment that may previously have been managed by the customer. As every customer is deployed differently in AWS, customers can take advantage of shifting management of certain IT controls to AWS, which results in a (new) distributed control environment. Customers can then use the AWS control and compliance documentation available to them to perform their control evaluation and verification procedures as required. The following are examples of controls that are managed by AWS, AWS customers, or both.

Inherited Controls – Controls that a customer fully inherits from AWS.

- Physical and Environmental controls

Shared Controls – Controls that apply to both the infrastructure layer and customer layers, but in separate contexts or perspectives. In a shared control, AWS provides the requirements for the infrastructure and the customer must provide their own control implementation within their use of AWS services. Examples include:

- Patch Management – AWS is responsible for patching and fixing flaws within the infrastructure, but customers are responsible for patching their guest operating system and applications.
- Configuration Management – AWS maintains the configuration of its infrastructure devices, but customers are responsible for configuring their own guest operating systems, databases, and applications.
- Awareness and Training – AWS trains AWS employees, but customers must train their own employees.

Customer Specific – Controls that are solely the responsibility of the customer based on the application they are deploying within AWS services. Examples include:

- Service and Communications Protection or Zone Security, which might require a customer to route or zone data within specific security environments.

Governance

Security governance, as a subset of the overall approach, is meant to support business objectives by defining policies and control objectives to help manage risk. Achieve risk management by following a layered approach to security control objectives–each layer builds upon the previous one. Understanding the AWS Shared Responsibility Model is your foundational layer. This knowledge provides clarity on what you are responsible for on the customer side and what you inherit from AWS. A beneficial resource is AWS Artifact, which gives you on-demand access to AWS' security and compliance reports and select online agreements.

Meet most of your control objectives at the next layer. This is where the platform-wide capability lives. For example, this layer includes the AWS account vending process, integration with an identity provider such as AWS IAM Identity Center, and the common detective controls. Some of the output of the platform governance process is here too. When you want to start using a new AWS service, update service control policies (SCPs) in the AWS Organizations service to

provide the guardrails for initial use of the service. You can use other SCPs to implement common security control objectives, often referred to as security invariants. These are control objectives or configuration that you apply to multiple accounts, organization units, or the whole AWS organization. Typical examples are limiting the Regions that infrastructure runs in or preventing the deactivation of detective controls. This middle layer also contains codified policies such as config rules or checks in pipelines.

The top layer is where the product teams meet control objectives. This is because the implementation is done in the applications that the product teams control. This could be implementing input validation in an application or ensuring that identity passes between microservices correctly. Even though the product team owns the configuration, they can still inherit some capability from the middle layer.

Wherever you implement the control, the goal is the same: manage risk. A range of risk management frameworks apply to specific industries, regions, or technologies. Your main objective: highlight the risk based on likelihood and consequence. This is the *inherent risk*. You can then define a control objective that reduces either the likelihood, consequence, or both. Then, with a control in place, you can see what the resulting risk is likely to be. This is the *residual risk*. Control objectives can apply to one or many workloads. The following diagram shows a typical risk matrix. The likelihood is based on frequency of previous occurrences and the consequence is based on the financial, reputational and time cost of the event.

Likelihood	Risk Level				
Very Likely	Low	Medium	High	Critical	Critical
Likely	Low	Medium	Medium	High	Critical
Possible	Low	Low	Medium	Medium	High
Unlikely	Low	Low	Medium	Medium	High
Very unlikely	Low	Low		Medium	High
Consequence	Minimal	Low	Medium	High	Severe

Figure 2: Risk level likelihood matrix

AWS account management and separation

We recommend that you organize workloads in separate accounts and group accounts based on function, compliance requirements, or a common set of controls rather than mirroring your organization's reporting structure. In AWS, accounts are a hard boundary. For example, account-level separation is strongly recommended for isolating production workloads from development and test workloads.

Manage accounts centrally: AWS Organizations automates AWS account creation and management, and control of those accounts after they are created. When you create an account through AWS Organizations, it is important to consider the email address you use, as this will be the root user that allows the password to be reset. Organizations allows you to group accounts into organizational units (OUs), which can represent different environments based on the workload's requirements and purpose.

Set controls centrally: Control what your AWS accounts can do by only allowing specific services, Regions, and service actions at the appropriate level. AWS Organizations allows you to use service control policies (SCPs) to apply permission guardrails at the organization, organizational unit, or account level, which apply to all AWS Identity and Access Management (IAM) users and roles. For example, you can apply an SCP that restricts users from launching resources in Regions that you have not explicitly allowed. AWS Control Tower offers a simplified way to set up and govern multiple accounts. It automates the setup of accounts in your AWS Organization, automates provisioning, applies guardrails (which include prevention and detection), and provides you with a dashboard for visibility.

Configure services and resources centrally: AWS Organizations helps you configure AWS services that apply to all of your accounts. For example, you can configure central logging of all actions performed across your organization using AWS CloudTrail, and prevent member accounts from deactivating logging. You can also centrally aggregate data for rules that you've defined using AWS Config, allowing you to audit your workloads for compliance and react quickly to changes. AWS CloudFormation StackSets allow you to centrally manage AWS CloudFormation stacks across accounts and OUs in your organization. This allows you to automatically provision a new account to meet your security requirements.

Use the delegated administration feature of security services to separate the accounts used for management from the organizational billing (management) account. Several AWS services, such as