Security Pillar

AWS Well-Architected Framework

Security Pillar: AWS Well-Architected Framework

Table of Contents

Security Pillar - AWS Well-Architected Framework

Publication date: **May 2021** (*Document Revisions* (p. 41))

Abstract

The focus of this paper is the security pillar of the AWS Well-Architected Framework. It provides guidance to help you apply best practices, current recommendations in the design, delivery, and maintenance of secure AWS workloads.

Introduction

The AWS Well-Architected Framework helps you understand trade-offs for decisions you make while building workloads on AWS. By using the Framework, you will learn current architectural best practices for designing and operating reliable, secure, efficient, and cost-effective workloads in the cloud. It provides a way for you to consistently measure your workload against best practices and identify areas for improvement. We believe that having well-architected workloads greatly increases the likelihood of business success.

The framework is based on six pillars:

- Operational Excellence
- Security
- Reliability
- Performance Efficiency
- Cost Optimization
- Sustainability

This paper focuses on the security pillar. This will help you meet your business and regulatory requirements by following current AWS recommendations. It's intended for those in technology roles, such as chief technology officers (CTOs), chief information security officers (CSOs/CISOs), architects, developers, and operations team members.

After reading this paper, you will understand AWS current recommendations and strategies to use when designing cloud architectures with security in mind. This paper doesn't provide implementation details or architectural patterns but does include references to appropriate resources for this information. By adopting the practices in this paper, you can build architectures that protect your data and systems, control access, and respond automatically to security events.

Security Foundations

The security pillar describes how to take advantage of cloud technologies to protect data, systems, and assets in a way that can improve your security posture. This paper provides in-depth, best-practice guidance for architecting secure workloads on AWS.

Design Principles

In the cloud, there are a number of principles that can help you strengthen your workload security:

- **Implement a strong identity foundation:** Implement the principle of least privilege and enforce separation of duties with appropriate authorization for each interaction with your AWS resources. Centralize identity management, and aim to eliminate reliance on long-term static credentials.
- **Enable traceability:** Monitor, alert, and audit actions and changes to your environment in real time. Integrate log and metric collection with systems to automatically investigate and take action.
- **Apply security at all layers:** Apply a defense in depth approach with multiple security controls. Apply to all layers (for example, edge of network, VPC, load balancing, every instance and compute service, operating system, application, and code).
- **Automate security best practices:** Automated software-based security mechanisms improve your ability to securely scale more rapidly and cost-effectively. Create secure architectures, including the implementation of controls that are defined and managed as code in version-controlled templates.
- **Protect data in transit and at rest**: Classify your data into sensitivity levels and use mechanisms, such as encryption, tokenization, and access control where appropriate.
- **Keep people away from data:** Use mechanisms and tools to reduce or eliminate the need for direct access or manual processing of data. This reduces the risk of mishandling or modification and human error when handling sensitive data.
- **Prepare for security events:** Prepare for an incident by having incident management and investigation policy and processes that align to your organizational requirements. Run incident response simulations and use tools with automation to increase your speed for detection, investigation, and recovery.

Definition

Security in the cloud is composed of six areas:

1. Foundations
2. Identity and access management
3. Detection
4. Infrastructure protection
5. Data protection
6. Incident response

Shared Responsibility

Security and Compliance is a shared responsibility between AWS and the customer. This shared model can help relieve the customer's operational burden as AWS operates, manages, and controls the components from the host operating system and virtualization layer down to the physical security of the facilities in which the service operates. The customer assumes responsibility and management of

the guest operating system (including updates and security patches), and other associated application software in addition to the configuration of the AWS provided security group firewall. Customers should carefully consider the services they choose as their responsibilities vary depending on the services used, the integration of those services into their IT environment, and applicable laws and regulations. The nature of this shared responsibility also provides the flexibility and customer control that permits the deployment. As shown in the following chart, this differentiation of responsibility is commonly referred to as Security "of" the Cloud versus Security "in" the Cloud.

AWS responsibility "Security of the Cloud" – AWS is responsible for protecting the infrastructure that runs all of the services offered in the AWS Cloud. This infrastructure is composed of the hardware, software, networking, and facilities that run AWS Cloud services.

Customer responsibility "Security in the Cloud" – Customer responsibility will be determined by the AWS Cloud services that a customer selects. This determines the amount of configuration work the customer must perform as part of their security responsibilities. For example, a service such as Amazon Elastic Compute Cloud (Amazon EC2) is categorized as Infrastructure as a Service (IaaS) and, as such, requires the customer to perform all of the necessary security configuration 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.

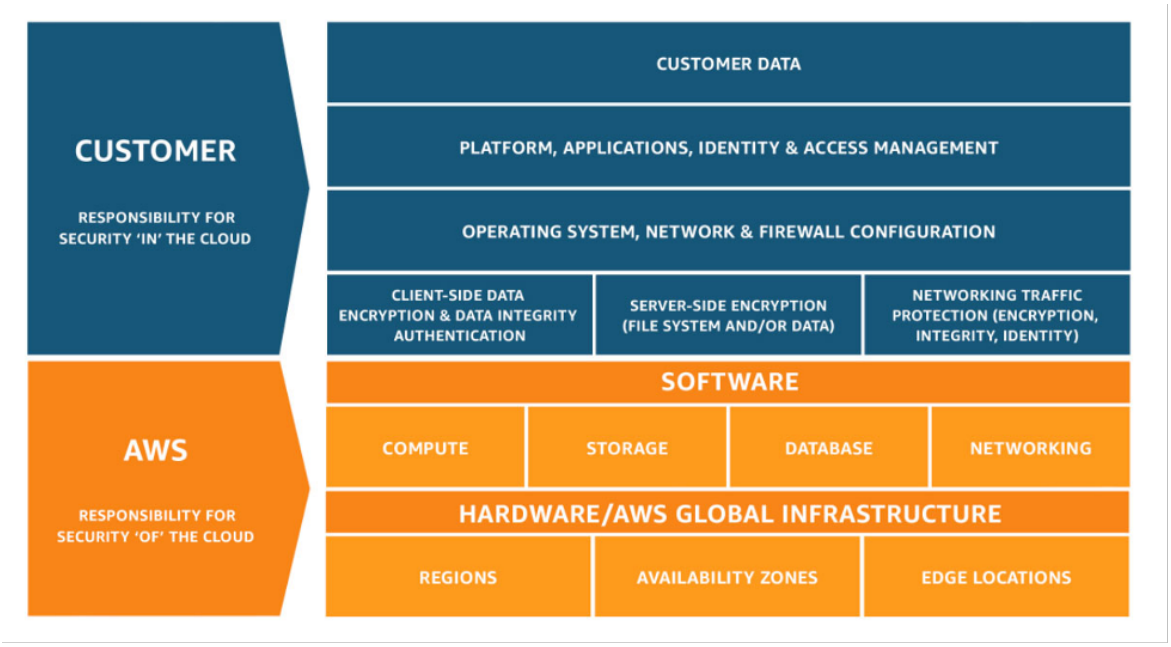

Figure 1: AWS Shared Responsibility Model.

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.

AWS Response to Abuse and Compromise

Abuse activities are observed behaviors of AWS customers' instances or other resources that are malicious, offensive, illegal, or could harm other internet sites. AWS works with you to detect and address suspicious and malicious activities from your AWS resources. Unexpected or suspicious behaviors from your resources can indicate that your AWS resources have been compromised, which signals potential risks to your business. Remember that you have alternate methods of contact in your AWS account. Be sure to use best practices when adding contacts, both for security and billing. Although your root account email is the primary target of communication from AWS, AWS also communicates security issues and billing issues to the secondary email addresses. Adding an email address that goes to only one person means that you have added a single point of failure to your AWS account. Make sure that you've added at least one distribution list to your contacts.

AWS detects abuse activities in your resources using mechanisms, such as:

- AWS internal event monitoring
- External security intelligence against AWS network address space
- Internet abuse complaints against AWS resources

Although the AWS abuse response team aggressively monitors and shuts down unauthorized activity running on AWS, the majority of abuse complaints refer to customers who have legitimate business on AWS. Some examples of common causes of unintentional abuse activities include:

- **Compromised resource** – An unpatched Amazon EC2 instance could be infected and become a botnet agent.
- **Unintentional abuse** – An overly aggressive web crawler might be classified as a denial-of-service attack by some internet sites.
- **Secondary abuse** – An end user of the service provided by an AWS customer might post malware files on a public Amazon S3 bucket.
- **False complaints** – Sometimes internet users mistakenly report legitimate activities as abuse.

AWS is committed to working with AWS customers to prevent, detect, and mitigate abuse, and to defend against future recurrences. We encourage you to review the AWS Acceptable Use Policy, which describes prohibited uses of the web services offered by Amazon Web Services and its affiliates. To support timely response to abuse notifications from AWS, make sure that your AWS account contact information is accurate. When you receive an AWS abuse warning, your security and operational staff should immediately investigate the matter. A delay can prolong the reputation impact and legal implications to yourself and others. More importantly, the implicated abuse resource might be compromised by malicious users, and ignoring the compromise could magnify damages to your business.

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 Single Sign-On (AWS SSO), 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 disabling 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

Operating Your Workloads Securely

Operating workloads securely covers the whole lifecycle of a workload from design, to build, to run, and to ongoing improvement. One of the ways to improve your ability to operate securely in the cloud is by taking an organizational approach to governance. Governance is the way that decisions are guided consistently without depending solely on the good judgment of the people involved. Your governance model and process are the way you answer the question "How do I know that the control objectives for a given workload are met and are appropriate for that workload?" Having a consistent approach to making decisions speeds up the deployment of workloads and helps raise the bar for the security capability in your organization.

To operate your workload securely, you must apply overarching best practices to every area of security. Take requirements and processes that you have defined in operational excellence at an organizational and workload level, and apply them to all areas. Staying up to date with AWS and industry recommendations and threat intelligence helps you evolve your threat model and control objectives. Automating security processes, testing, and validation help you scale your security operations.

Automation allows consistency and repeatability of processes. People are good at many things, but consistently doing the same thing repeatedly without mistakes is not one of them. Even with well-written runbooks, you run the risk that people won't consistently carry out repetitive tasks. This is especially true when people have diverse responsibilities and then have to respond to unfamiliar alerts. Automation, however, responds to a trigger the same way each time. The best way to deploy applications is through automation. The code that runs the deployment can be tested and then used to perform the deployment. This increases confidence in the change process and reduces the risk of a failed change.

To verify that the configuration meets your control objectives, test the automation and the deployed application in a non-production environment first. This way, you can test the automation to prove that it performed all the steps correctly. You also get early feedback in the development and deployment cycle, reducing rework. To reduce the chance of deployment errors, make configuration changes by code not by people. If you need to re-deploy an application, automation makes this much easier. As you define additional control objectives, you can easily add them to the automation for all workloads.

Instead of having individual workload owners invest in security specific to their workloads, save time by using common capabilities and shared components. Some examples of services that multiple teams can consume include the AWS account creation process, centralized identity for people, common

logging configuration, and AMI and container base image creation. This approach can help builders improve workload cycle times and consistently meet security control objectives. When teams are more consistent, you can validate control objectives and better report your control posture and risk position to stakeholders.

Identify and prioritize risks using a threat model: Threat modeling provides a systematic approach to aid in finding and addressing security issues early in the design process. Earlier is better since mitigations have a lower cost compared to later in the lifecycle. Use a threat model to identify and maintain an up-to-date registry of potential threats.

The typical core steps of the threat modeling process are:

1. Identify assets, actors, entry points, components, use cases, and trust levels, and include these in a design diagram.
2. Identify a list of threats.
3. For each threat, identify mitigations, which might include security control implementations.
4. Create and review a risk matrix to determine if the threat is adequately mitigated.

Threat modeling is most effective when done at the workload (or workload feature) level, ensuring that all context is available for assessment. Revisit and maintain this matrix as your security landscape evolves.

Identify and validate control objectives: Based on your compliance requirements and risks identified from your threat model, derive and validate the control objectives and controls that you must apply to your workload. Ongoing validation of control objectives and controls help you measure the effectiveness of risk mitigation.

Keep up to date with security threats: To help you define and implement appropriate controls, recognize attack vectors by staying up to date with the latest security threats. Consume AWS Managed Services to make it easier to receive notification of unexpected or unusual behavior in your AWS accounts. Investigate using AWS Partner tools or third-party threat information feeds as part of your security information flow. The Common Vulnerabilities and Exposures (CVE) list contains publicly disclosed cyber security vulnerabilities that you can use to stay up to date.

Keep up to date with security recommendations: Stay up to date with both AWS and industry security recommendations to evolve the security posture of your workload. AWS Security Bulletins contain important information about security and privacy notifications.

Evaluate and implement new security services and features regularly: Evaluate and implement security services and features from AWS and AWS Partners that allow you to evolve the security posture of your workload. The AWS Security Blog highlights new AWS services and features, implementation guides, and general security guidance. What's New with AWS? is a great way to stay up to date with all new AWS features, services, and announcements.

Automate testing and validation of security controls in pipelines: Establish secure baselines and templates for security mechanisms that are tested and validated as part of your build, pipelines, and processes. Use tools and automation to test and validate all security controls continuously. For example, scan items such as machine images and infrastructure as code templates for security vulnerabilities, irregularities, and drift from an established baseline at each stage. AWS CloudFormation Guard can help you verify that CloudFormation templates are safe, save you time, and reduce the risk of configuration error.

Reducing the number of security misconfigurations introduced into a production environment is critical—the more quality control and reduction of defects you can perform in the build process, the better. Design continuous integration and continuous deployment (CI/CD) pipelines to test for security issues whenever possible. CI/CD pipelines offer the opportunity to enhance security at each stage of build and delivery. CI/CD security tooling must also be kept updated to mitigate evolving threats.

Track changes to your workload configuration to help with compliance auditing, change management, and investigations that may apply to you. You can use AWS Config to record and evaluate your AWS and third-party resources. It allows you to continuously audit and assess the overall compliance with rules and conformance packs, which are collections of rules with remediation actions.

Change tracking should include planned changes, which are part of your organization's change control process (sometimes referred to as MACD–Move/Add/Change/Delete), unplanned changes, and unexpected changes, such as incidents. Changes might occur on the infrastructure, but they might also be related to other categories, such as changes in code repositories, machine images and application inventory changes, process and policy changes, or documentation changes.

Resources

Refer to the following resources to learn more about operating your workload securely.

Videos

- Security Best Practices the Well-Architected Way
- Enable AWS adoption at scale with automation and governance
- AWS Security Hub: Manage Security Alerts & Automate Compliance
- Automate your security on AWS

Documentation

- AWS Security Home
- AWS Shared Responsibility Model
- Security Bulletins
- Security Blog
- What's New with AWS
- AWS security audit guidelines
- AWS Audit Manager
- How to approach threat modelling
- How to think about cloud security governance

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.

Separate workloads using accounts: Start with security and infrastructure in mind to enable your organization to set common guardrails as your workloads grow. This approach provides boundaries and controls between workloads. Account-level separation is strongly recommended for isolating production environments from development and test environments, or providing a strong logical boundary between workloads that process data of different sensitivity levels, as defined by external compliance requirements (such as PCI-DSS or HIPAA), and workloads that don't.

Secure AWS account: There are a number of aspects to securing your AWS accounts, including the securing of, and not using the root user, and keeping the contact information up to date. You can use AWS Organizations to centrally manage and govern your accounts as you grow and scale your workloads

in AWS. AWS Organizations helps you manage accounts, set controls, and configure services across your accounts.

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 disabling logging. You can also centrally aggregate data for rules that you've defined using AWS Config, enabling 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 GuardDuty, Security Hub, and AWS Config, support integrations with AWS Organizations including designating a specific account for administrative functions.

Resources

Refer to the following resources to learn more about AWS recommendations for deploying and managing multiple AWS accounts.

Videos

- Managing and governing multi-account AWS environments using AWS Organizations
- AXA: Scaling adoption with a Global Landing Zone
- Using AWS Control Tower to Govern Multi-Account AWS Environments

Documentation

- Establishing your best practice AWS environment
- AWS Organizations
- AWS Control Tower
- Working with AWS CloudFormation StackSets
- How to use service control policies to set permission guardrails across accounts in your AWS Organization

Hands-on

- Lab: AWS Account and Root User

Identity and Access Management

To use AWS services, you must grant your users and applications access to resources in your AWS accounts. As you run more workloads on AWS, you need robust identity management and permissions in place to ensure that the right people have access to the right resources under the right conditions. AWS offers a large selection of capabilities to help you manage your human and machine identities and their permissions. The best practices for these capabilities fall into two main areas.

Topics

Identity Management

There are two types of identities you need to manage when approaching operating secure AWS workloads.

- **Human Identities:** The administrators, developers, operators, and consumers of your applications require an identity to access your AWS environments and applications. These can be members of your organization, or external users with whom you collaborate, and who interact with your AWS resources via a web browser, client application, mobile app, or interactive command-line tools.
- **Machine Identities:** Your workload applications, operational tools, and components require an identity to make requests to AWS services, for example, to read data. These identities include machines running in your AWS environment, such as Amazon EC2 instances or AWS Lambda functions. You can also manage machine identities for external parties who need access. Additionally, you might also have machines outside of AWS that need access to your AWS environment.

Rely on a centralized identity provider: For workforce identities, rely on an identity provider that enables you to manage identities in a centralized place. This makes it easier to manage access across multiple applications and services, because you are creating, managing, and revoking access from a single location. For example, if someone leaves your organization, you can revoke access for all applications and services (including AWS) from one location. This reduces the need for multiple credentials and provides an opportunity to integrate with existing human resources (HR) processes.

For federation with individual AWS accounts, you can use centralized identities for AWS with a SAML 2.0-based provider with AWS IAM. You can use any provider—whether hosted by you in AWS, external to AWS, or supplied by the AWS Partner Network (APN)—that is compatible with the SAML 2.0 protocol. You can use federation between your AWS account and your chosen provider to grant a user or application access to call AWS API operations by using a SAML assertion to get temporary security credentials. Web-based single sign-on is also supported, allowing users to sign in to the AWS Management Console from your sign in portal.

For federation to multiple accounts in your AWS Organization, you can configure your identity source in AWS Single Sign-On (AWS SSO), and specify where your users and groups are stored. Once configured, your identity provider is your source of truth, and information can be synchronized using the System for Cross-domain Identity Management (SCIM) v2.0 protocol. You can then look up users or groups and grant them single sign-on access to AWS accounts, cloud applications, or both.

AWS SSO integrates with AWS Organizations, which enables you to configure your identity provider once and then grant access to existing and new accounts managed in your organization. AWS SSO provides you with a default store, which you can use to manage your users and groups. If you choose

to use the AWS SSO store, create your users and groups and assign their level of access to your AWS accounts and applications, keeping in mind the best practice of least privilege. Alternatively, you can choose to Connect to Your External Identity Provider using SAML 2.0, or Connect to Your Microsoft AD Directory using AWS Directory Service. Once configured, you can sign into the AWS Management Console, command line interface, or the AWS mobile app, by authenticating through your central identity provider.

For managing end-users or consumers of your workloads, such as a mobile app, you can use Amazon Cognito. It provides authentication, authorization, and user management for your web and mobile apps. Your users can sign in directly with a user name and password, or through a third party, such as Amazon, Apple, Facebook, or Google.

Leverage user groups and attributes: As the number of users you manage grows, you will need to determine ways to organize them so that you can manage them at scale. Place users with common security requirements in groups defined by your identity provider, and put mechanisms in place to ensure that user attributes that may be used for access control (for example, department or location) are correct and updated. Use these groups and attributes to control access, rather than individual users. This allows you to manage access centrally by changing a user's group membership or attributes once with a permission set, rather than updating many individual policies when a user's access needs change. You can use AWS SSO to manage user groups and attributes. AWS SSO supports most commonly used attributes whether they are entered manually during user creation or automatically provisioned using a synchronization engine, such as defined in the System for Cross-Domain Identity Management (SCIM) specification.

Use strong sign-in mechanisms: Enforce minimum password length, and educate your users to avoid common or reused passwords. Enforce multi-factor authentication (MFA) with software or hardware mechanisms to provide an additional layer of verification. For example, when using AWS SSO as the identity source, configure the "context-aware" or "always-on" setting for MFA, and allow users to enroll their own MFA devices to accelerate adoption. When using an external identity provider (IdP), configure your IdP for MFA.

Use temporary credentials: Require identities to dynamically acquire temporary credentials. For workforce identities, use AWS SSO, or federation with IAM, to access AWS accounts. For machine identities, such as EC2 instances or Lambda functions, require the use of IAM roles instead of IAM users with long term access keys.

For human identities using the AWS Management Console, require users to acquire temporary credentials and federate into AWS. You can do this using the AWS SSO user portal or configuring federation with IAM. For users requiring CLI access, ensure that they use AWS CLI v2, which supports direct integration with AWS Single Sign-On (AWS SSO). Users can create CLI profiles that are linked to AWS SSO accounts and roles. The CLI automatically retrieves AWS credentials from AWS SSO and refreshes them on your behalf. This eliminates the need to copy and paste temporary AWS credentials from the AWS SSO console. For SDK, users should rely on AWS Security Token Service (AWS STS) to assume roles to receive temporary credentials. In certain cases, temporary credentials might not be practical. You should be aware of the risks of storing access keys, rotate these often, and require MFA as a condition when possible.

For cases where you need to grant consumers access to your AWS resources, use Amazon Cognito identity pools and assign them a set of temporary, limited privilege credentials to access your AWS resources. The permissions for each user are controlled through IAM roles that you create. You can define rules to choose the role for each user based on claims in the user's ID token. You can define a default role for authenticated users. You can also define a separate IAM role with limited permissions for guest users who are not authenticated.

For machine identities, you should rely on IAM roles to grant access to AWS. For EC2 instances, you can use roles for Amazon EC2. You can attach an IAM role to your EC2 instance to enable your applications running on Amazon EC2 to use temporary security credentials that AWS creates, distributes, and rotates automatically through the Instance Metadata Service (IMDS). The latest version of IMDS helps protect against vulnerabilities that expose the temporary credentials and should be implemented. For

accessing EC2 instances using keys or passwords, AWS Systems Manager is a more secure way to access and manage your instances using a pre-installed agent without the stored secret. Additionally, other AWS services, such as AWS Lambda, enable you to configure an IAM service role to grant the service permissions to perform AWS actions using temporary credentials. In situations where you cannot use temporary credentials, use programmatic tools, such as AWS Secrets Manager, to automate credential rotation and management.

Audit and rotate credentials periodically: Periodic validation, preferably through an automated tool, is necessary to verify that the correct controls are enforced. For human identities, you should require users to change their passwords periodically and retire access keys in favor of temporary credentials. As you are moving from IAM users to centralized identities, you can generate a credential report to audit your IAM users. We also recommend that you enforce MFA settings in your identity provider. You can set up AWS Config Rules to monitor these settings. For machine identities, you should rely on temporary credentials using IAM roles. For situations where this is not possible, frequent auditing and rotating access keys is necessary.

Store and use secrets securely: For credentials that are not IAM-related and cannot take advantage of temporary credentials, such as database logins, use a service that is designed to handle management of secrets, such as AWS Secrets Manager. Secrets Manager makes it easy to manage, rotate, and securely store encrypted secrets using supported services. Calls to access the secrets are logged in CloudTrail for auditing purposes, and IAM permissions can grant least-privilege access to them.

Resources

Refer to the following resources to learn more about AWS best practices for protecting your AWS credentials.

Videos

- Mastering identity at every layer of the cake
- Managing user permissions at scale with AWS SSO
- Best Practices for Managing, Retrieving, & Rotating Secrets at Scale

Documentation

- The AWS Account Root User
- Getting started with AWS Single Sign-On
- Setting an Account Password Policy for IAM Users
- Getting Started with AWS Secrets Manager
- Using Instance Profiles
- Temporary Security Credentials
- Identity Providers and Federation

Permissions Management

Manage permissions to control access to human and machine identities that require access to AWS and your workloads. Permissions control who can access what, and under what conditions. Set permissions to specific human and machine identities to grant access to specific service actions on specific resources. Additionally, specify conditions that must be true for access to be granted. For example, you can allow developers to create new Lambda functions, but only in a specific Region. When managing your AWS environments at scale, adhere to the following best practices to ensure that identities only have the access they need and nothing more.

There are a number of ways to grant access to different types of resources. One way is by using different policy types.

Identity-based policies in IAM are *managed* or *inline*, and attach to IAM identities, including users, groups, or roles. These policies let you specify what that identity can do (its permissions). Identity-based policies can be further categorized:

- **Managed policies** – Standalone identity-based policies that you can attach to multiple users, groups, and roles in your AWS account. There are two types of managed policies:
  - **AWS-managed policies** – Managed policies that are created and managed by AWS.
  - **Customer-managed policies** – Managed policies that you create and manage in your AWS account. Customer-managed policies provide more precise control over your policies than AWS-managed policies.
- **Inline policies** – Policies that you add directly to a single user, group, or role. Inline policies maintain a strict one-to-one relationship between a policy and an identity. Inline policies are deleted when you delete the identity.

In most cases, you should create your own customer-managed policies following the principle of least privilege.

Resource-based policies are attached to a resource. For example, an S3 bucket policy is a resource-based policy. These policies grant permission to a principal that can be in the same account as the resource or in another account. For a list of services that support resource-based policies, see AWS services that work with IAM.

Permissions boundariesuse a managed policy to set the maximum permissions that an administrator can set. This enables you to delegate the ability to create and manage permissions to developers, such as the creation of an IAM role, but limit the permissions they can grant so that they cannot escalate their permission using what they have created.

Attribute-based access control (ABAC) enables you to grant permissions based on attributes. In AWS, these are called tags. Tags can be attached to IAM principals (users or roles) and to AWS resources. Using IAM policies, administrators can create a reusable policy that applies permissions based on the attributes of the IAM principal. For example, as an administrator you can use a single IAM policy that grants developers in your organization access to AWS resources that match the developers' project tags. As the team of developers adds resources to projects, permissions are automatically applied based on attributes. As a result, no policy update is required for each new resource.

Organizations service control policies (SCP) define the maximum permissions for account members of an organization or organizational unit (OU). SCPs *limit* permissions that identity-based policies or resource-based policies grant to entities (users or roles) within the account, but *do not grant* permissions.

Session policies assume a role or a federated user. Pass session policies when using the AWS CLI or AWS API Session policies to limit the permissions that the role or user's identity-based policies grant to the session. These policies *limit* permissions for a created session, but *do not grant* permissions. For more information, see Session Policies.

Define permission guardrails for your organization: As you grow and manage additional workloads in AWS, you should separate these workloads using accounts and manage those accounts using AWS Organizations. We recommend that you establish common permission guardrails that restrict access to all identities in your organization. For example, you can restrict access to specific AWS Regions, or prevent your team from deleting common resources, such as an IAM role used by your central security team. You can get started by implementing example service control policies, such as preventing users from disabling key services.

You can use AWS Organizations to group accounts and set common controls on each group of accounts. To set these common controls, you can use services integrated with AWS Organizations. Specifically, you

can use service control policies (SCPs) to restrict access to group of accounts. SCPs use the IAM policy language and enable you to establish controls that all IAM principals (users and roles) adhere to. You can restrict access to specific service actions, resources and based on specific condition to meet the access control needs of your organization. If necessary, you can define exceptions to your guardrails. For example, you can restrict service actions for all IAM entities in the account except for a specific administrator role.

Grant least privilege access: Establishing a principle of least privilege ensures that identities are only permitted to perform the most minimal set of functions necessary to fulfill a specific task, while balancing usability and efficiency. Operating on this principle limits unintended access and helps ensure that you can audit who has access to which resources. In AWS, identities have no permissions by default with the exception of the root user, which should only be used for a few specific tasks.

You use policies to explicitly grant permissions attached to IAM or resource entities, such as an IAM role used by federated identities or machines, or resources (for example, S3 buckets). When you create and attach a policy, you can specify the service actions, resources, and conditions that must be true for AWS to allow access. AWS supports a variety of conditions to help you scope down access. For example, using the `PrincipalOrgID` condition key, the identifier of the AWS Organizations is verified so access can be granted within your AWS Organization. You can also control requests that AWS services make on your behalf, like AWS CloudFormation creating an AWS Lambda function, by using the `CalledVia` condition key. This enables you to set granular permissions for your human and machine identities across AWS.

AWS also has capabilities that enable you to scale your permissions management and adhere to least privilege.

Analyze public and cross account access: In AWS, you can grant access to resources in another account. You grant direct cross-account access using policies attached to resources (for example, S3 bucket policies) or by allowing an identity to assume an IAM role in another account. When using resource policies, you want to ensure you grant access to identities in your organization and are intentional about when you make a resource public. Making a resource public should be used sparingly, as this action allows anyone to access the resource. IAM Access Analyzer uses mathematical methods (that is, provable security) to identify all access paths to a resource from outside of its account. It reviews resource policies continuously, and reports findings of public and cross-account access to make it easy for you to analyze potentially broad access.

Share resources securely: As you manage workloads using separate accounts, there will be cases where you need to share resources between those accounts. We recommend that you share resources using AWS Resource Access Manager (AWS RAM). This service enables you to easily and securely share AWS resources within your AWS Organization and Organizational Units. Using AWS RAM, access to shared resources is automatically granted or revoked as accounts are moved in and out of the Organization or Organization Unit with which they are shared. This helps ensure that resources are only shared with the accounts that you intend.

Reduce permissions continuously: Sometimes, when teams and projects are just getting started, you might choose to grant broad access (in a dev/test environment) to inspire innovation and agility. We recommend that you evaluate access continuously and, especially in a production environment, restrict access to only the permissions required and achieve least privilege. AWS provides access analysis capabilities to help you identify unused access. To help you identify unused users and roles, AWS analyzes access activity and provides access key and role last used information. You can use the last accessed timestamp to identify unused users and roles, and remove them. Moreover, you can review service and action last accessed information to identify and tighten permissions for specific users and roles. For example, you can use last accessed information to identify the specific S3 actions that your application role requires and restrict access to only those. These feature are available in the console and programmatically to enable you to incorporate them into your infrastructure workflows and automated tools.

Establish emergency access process: You should have a process that allows emergency access to your workload, in particular your AWS accounts, in the unlikely event of an automated process or pipeline issue. This process could include a combination of different capabilities, for example, an emergency AWS

cross-account role for access, or a specific process for administrators to follow to validate and approve an emergency request.

Resources

Refer to the following resources to learn more about current AWS best practices for fine-grained authorization.

Videos

- Become an IAM Policy Expert in 60 Minutes or Less
- Separation of Duties, Least Privilege, Delegation, & CI/CD

Documentation

- Grant least privilege
- Working with Policies
- Delegating Permissions to Administer IAM Users, Groups, and Credentials
- IAM Access Analyzer
- Remove unnecessary credentials
- Assuming a role in the CLI with MFA
- Permissions Boundaries
- Attribute-based access control (ABAC)

Hands-on

- Lab: IAM Permission Boundaries Delegating Role Creation
- Lab: IAM Tag Based Access Control for EC2
- Lab: Lambda Cross Account IAM Role Assumption
- IAM Tutorial: Use SAML session tags for ABAC

Detection

Detection consists of two parts: detection of unexpected or unwanted configuration changes, and the detection of unexpected behavior. The first can take place at multiple places in an application delivery lifecycle. Using infrastructure as code (for example, a CloudFormation template), you can check for unwanted configuration before a workload is deployed by implementing checks in the CI/CD pipelines or source control. Then, as you deploy a workload into non-production and production environments, you can check configuration using native AWS, open source, or AWS Partner tools. These checks can be for configuration that does not meet security principles or best practices, or for changes that were made between a tested and deployed configuration. For a running application, you can check whether the configuration has been changed in an unexpected fashion, including outside of a known deployment or automated scaling event.

For the second part of detection, unexpected behavior, you can use tools or by alerting on an increase in a particular type of API call. Using Amazon GuardDuty, you can be alerted when unexpected and potentially unauthorized or malicious activity occurs within your AWS accounts. You should also explicitly monitor for mutating API calls that you would not expect to be used in your workload, and API calls that change the security posture.

Detection enables you to identify a potential security misconfiguration, threat, or unexpected behavior. It's an essential part of the security lifecycle and can be used to support a quality process, a legal or compliance obligation, and for threat identification and response efforts. There are different types of detection mechanisms. For example, logs from your workload can be analyzed for exploits that are being used. You should regularly review the detection mechanisms related to your workload to ensure that you are meeting internal and external policies and requirements. Automated alerting and notifications should be based on defined conditions to enable your teams or tools to investigate. These mechanisms are important reactive factors that can help your organization identify and understand the scope of anomalous activity.

In AWS, there are a number of approaches you can use when addressing detective mechanisms. The following sections describe how to use these approaches:

- Configure
- Investigate

Topics

Configure

Configure service and application logging: A foundational practice is to establish a set of detection mechanisms at the account level. This base set of mechanisms is aimed at recording and detecting a wide range of actions on all resources in your account. They allow you to build out a comprehensive detective capability with options that include automated remediation, and partner integrations to add functionality.

In AWS, services that can implement this base set include:

- AWS CloudTrail provides event history of your AWS account activity, including actions taken through the AWS Management Console, AWS SDKs, command line tools, and other AWS services.

- AWS Config monitors and records your AWS resource configurations and allows you to automate the evaluation and remediation against desired configurations.
- Amazon GuardDuty is a threat detection service that continuously monitors for malicious activity and unauthorized behavior to protect your AWS accounts and workloads.
- AWS Security Hub provides a single place that aggregates, organizes, and prioritizes your security alerts, or findings, from multiple AWS services and optional third-party products to give you a comprehensive view of security alerts and compliance status.

Building on the foundation at the account level, many core AWS services, for example Amazon Virtual Private Cloud (Amazon VPC), provide service-level logging features. VPC Flow Logs enable you to capture information about the IP traffic going to and from network interfaces that can provide valuable insight into connectivity history, and trigger automated actions based on anomalous behavior.

For EC2 instances and application-based logging that doesn't originate from AWS services, logs can be stored and analyzed using Amazon CloudWatch Logs. An agent collects the logs from the operating system and the applications that are running and automatically stores them. Once the logs are available in CloudWatch Logs, you can process them in real-time, or dive into analysis using CloudWatch Logs Insights.

Equally important to collecting and aggregating logs is the ability to extract meaningful insight from the great volumes of log and event data generated by complex architectures. See the *Monitoring* section of the Reliability Pillar whitepaper for more detail. Logs can themselves contain data that is considered sensitive–either when application data has erroneously found its way into log files that the CloudWatch Logs agent is capturing, or when cross-region logging is configured for log aggregation and there are legislative considerations about shipping certain kinds of information across borders.

One approach is to use Lambda functions, triggered on events when logs are delivered, to filter and redact log data before forwarding into a central logging location, such as an S3 bucket. The unredacted logs can be retained in a local bucket until a "reasonable time" has passed (as determined by legislation and your legal team), at which point an S3 lifecycle rule can automatically delete them. Logs can further be protected in Amazon S3 by using S3 Object Lock, where you can store objects using a write-once-read-many (WORM) model.

Analyze logs, findings, and metrics centrally: Security operations teams rely on the collection of logs and the use of search tools to discover potential events of interest, which might indicate unauthorized activity or unintentional change. However, simply analyzing collected data and manually processing information is insufficient to keep up with the volume of information flowing from complex architectures. Analysis and reporting alone don't facilitate the assignment of the right resources to work an event in a timely fashion.

A best practice for building a mature security operations team is to deeply integrate the flow of security events and findings into a notification and workflow system such as a ticketing system, a bug/issue system, or other security information and event management (SIEM) system. This takes the workflow out of email and static reports, and allows you to route, escalate, and manage events or findings. Many organizations are also integrating security alerts into their chat/collaboration and developer productivity platforms. For organizations embarking on automation, an API-driven, low-latency ticketing system offers considerable flexibility when planning "what to automate first".

This best practice applies not only to security events generated from log messages depicting user activity or network events, but also from changes detected in the infrastructure itself. The ability to detect change, determine whether a change was appropriate, and then route that information to the correct remediation workflow is essential in maintaining and validating a secure architecture, in the context of changes where the nature of their undesirability is sufficiently subtle that their execution cannot currently be prevented with a combination of IAM and Organizations configuration.

GuardDuty and Security Hub provide aggregation, deduplication, and analysis mechanisms for log records that are also made available to you via other AWS services. GuardDuty ingests, aggregates,

and analyzes information from sources such as CloudTrail management and data events, VPC DNS logs, and VPC Flow Logs. Security Hub can ingest, aggregate, and analyze output from GuardDuty, AWS Config, Amazon Inspector, Amazon Macie, AWS Firewall Manager, and a significant number of third-party security products available in the AWS Marketplace, and if built accordingly, your own code. Both GuardDuty and Security Hub have an Administrator-Member model that can aggregate findings and insights across multiple accounts, and Security Hub is often used by customers who have an on-premises SIEM as an AWS-side log and alert preprocessor and aggregator from which they can then ingest Amazon EventBridge via a Lambda-based processor and forwarder.

Resources

Refer to the following resources to learn more about current AWS recommendations for capturing and analyzing logs.

Videos

- Threat management in the cloud: Amazon GuardDuty & AWS Security Hub
- Centrally Monitoring Resource Configuration & Compliance

Documentation

- Setting up Amazon GuardDuty
- AWS Security Hub
- Getting started: Amazon CloudWatch Logs
- Amazon EventBridge
- Configuring Athena to analyze CloudTrail logs
- Amazon CloudWatch
- AWS Config
- Creating a trail in CloudTrail
- Centralize logging solution

Investigate

Implement actionable security events: For each detective mechanism you have, you should also have a process, in the form of a runbook or playbook, to investigate. For example, when you enable Amazon GuardDuty, it generates different findings. You should have a runbook entry for each finding type, for example, if a trojan is discovered, your runbook has simple instructions that instruct someone to investigate and remediate.

Automate response to events: In AWS, investigating events of interest and information on potentially unexpected changes into an automated workflow can be achieved using Amazon EventBridge. This service provides a scalable rules engine designed to broker both native AWS event formats (such as CloudTrail events), as well as custom events you can generate from your application. Amazon GuardDuty also allows you to route events to a workflow system for those building incident response systems (Step Functions), or to a central Security Account, or to a bucket for further analysis.

Detecting change and routing this information to the correct workflow can also be accomplished using AWS Config Rules and Conformance Packs. AWS Config detects changes to in-scope services (though with higher latency than Amazon EventBridge) and generates events that can be parsed using AWS Config Rules for rollback, enforcement of compliance policy, and forwarding of information to systems, such as change management platforms and operational ticketing systems. As well as writing your

own Lambda functions to respond to AWS Config events, you can also take advantage of the AWS Config Rules Development Kit, and a library of open source AWS Config Rules. Conformance packs are a collection of Config Rules and remediation actions you deploy as a single entity authored as a YAML template. A sample conformance pack template is available for the Well-Architected Security Pillar.

Resources

Refer to the following resources to learn more about current AWS best practices for integrating auditing controls with notification and workflow.

Videos

- Remediating Amazon GuardDuty and AWS Security Hub Findings
- Best Practices for Managing Security Operations on AWS
- Achieving Continuous Compliance using AWS Config

Documentation

- Amazon EventBridge
- AWS Config Rules
- AWS Config Rules Repository (open source)
- AWS Config Rules Development Kit

Hands-on

- Solution: Real-Time Insights on AWS Account Activity
- Solution: Centralized Logging

Infrastructure Protection

Infrastructure protection encompasses control methodologies, such as defense in depth, that are necessary to meet best practices and organizational or regulatory obligations. Use of these methodologies is critical for successful, ongoing operations in the cloud.

Infrastructure protection is a key part of an information security program. It ensures that systems and services within your workload are protected against unintended and unauthorized access, and potential vulnerabilities. For example, you'll define trust boundaries (for example, network and account boundaries), system security configuration and maintenance (for example, hardening, minimization and patching), operating system authentication and authorizations (for example, users, keys, and access levels), and other appropriate policy-enforcement points (for example, web application firewalls and/or API gateways).

Regions, Availability Zones, AWS Local Zones, and AWS Outposts

Make sure you are familiar with Regions, Availability Zones, AWS Local Zones, and AWS Outposts, which are components of the AWS secure global infrastructure.

AWS has the concept of a Region, which is a physical location around the world where we cluster data centers. We call each group of logical data centers an Availability Zone (AZ). Each AWS Region consists of multiple, isolated, and physically separate AZs within a geographic area. If you have data residency requirements, you can choose the AWS Region that is close to your desired location. You retain complete control and ownership over the Region in which your data is physically located, which can be helpful for meeting your regional compliance and data residency requirements. Each AZ has independent power, cooling, and physical security. If an application is partitioned across AZs, you are better isolated and protected from issues such as power outages, lightning strikes, tornadoes, earthquakes, and more. AZs are physically separated by a meaningful distance, many kilometers, from any other AZ, although all are within 100 km (60 miles) of each other. All AZs in an AWS Region are interconnected with high-bandwidth, low-latency networking, using fully redundant, dedicated metro fiber providing high-throughput, low-latency networking between AZs. All traffic between AZs is encrypted. AWS customers focused on high availability can design their applications to run in multiple AZs to achieve even greater fault-tolerance. AWS Regions meet the highest levels of security, compliance, and data protection.

AWS Local Zones place compute, storage, database, and other select AWS services closer to end users. With AWS Local Zones, you can easily run highly demanding applications that require single-digit millisecond latencies to your end users, such as media and entertainment content creation, real-time gaming, reservoir simulations, electronic design automation, and machine learning. Each AWS Local Zone location is an extension of an AWS Region where you can run your latency-sensitive applications, using AWS services such as Amazon EC2, Amazon VPC, Amazon EBS, Amazon File Storage, and Elastic Load Balancing in geographic proximity to end users. AWS Local Zones provide a high-bandwidth, secure connection between local workloads and those running in the AWS Region, allowing you to seamlessly connect to the full range of in-region services through the same APIs and tool sets.

AWS Outposts bring native AWS services, infrastructure, and operating models to virtually any data center, co-location space, or on-premises facility. You can use the same AWS APIs, tools, and infrastructure across on-premises facilities and the AWS Cloud to deliver a truly consistent hybrid experience. AWS Outposts is designed for connected environments and can be used to support workloads that must remain on premises due to low latency or local data processing needs.

In AWS, there are a number of approaches to infrastructure protection. The following sections describe how to use these approaches.

Topics

Protecting Networks

Users, both in your workforce and your customers, can be located anywhere. You need to pivot from traditional models of trusting anyone and anything that has access to your network. When you follow the principle of applying security at all layers, you employ a Zero Trust approach. Zero Trust security is a model where application components or microservices are considered discrete from each other and no component or microservice trusts any other.

The careful planning and management of your network design forms the foundation of how you provide isolation and boundaries for resources within your workload. Because many resources in your workload operate in a VPC and inherit the security properties, it's critical that the design is supported with inspection and protection mechanisms backed by automation. Likewise, for workloads that operate outside a VPC, using purely edge services and/or serverless, the best practices apply in a more simplified approach. Refer to the AWS Well-Architected Serverless Applications Lens for specific guidance on serverless security.

Create network layers: Components such as EC2 instances, RDS database clusters, and Lambda functions that share reachability requirements can be segmented into layers formed by subnets. For example, an RDS database cluster in a VPC with no need for internet access should be placed in subnets with no route to or from the internet. This layered approach for the controls mitigates the impact of a single layer misconfiguration, which could allow unintended access. For AWS Lambda, you can run your functions in your VPC to take advantage of VPC-based controls.

For network connectivity that can include thousands of VPCs, AWS accounts, and on-premises networks, you should use AWS Transit Gateway. It acts as a hub that controls how traffic is routed among all the connected networks, which act like spokes. Traffic between an Amazon VPC and AWS Transit Gateway remains on the AWS private network, which reduces external threat vectors such as distributed denial of service (DDoS) attacks and common exploits, such as SQL injection, cross-site scripting, cross-site request forgery, or abuse of broken authentication code. AWS Transit Gateway inter-region peering also encrypts inter-region traffic with no single point of failure or bandwidth bottleneck.

Control traffic at all layers: When architecting your network topology, you should examine the connectivity requirements of each component. For example, if a component requires internet accessibility (inbound and outbound), connectivity to VPCs, edge services, and external data centers.

A VPC allows you to define your network topology that spans an AWS Region with a private IPv4 address range that you set, or an IPv6 address range AWS selects. You should apply multiple controls with a defense in depth approach for both inbound and outbound traffic, including the use of security groups (stateful inspection firewall), Network ACLs, subnets, and route tables. Within a VPC, you can create subnets in an Availability Zone. Each subnet can have an associated route table that defines routing rules for managing the paths that traffic takes within the subnet. You can define an internet routable subnet by having a route that goes to an internet or NAT gateway attached to the VPC, or through another VPC.

When an instance, RDS database, or other service is launched within a VPC, it has its own security group per network interface. This firewall is outside the operating system layer and can be used to define rules for allowed inbound and outbound traffic. You can also define relationships between security groups. For example, instances within a database tier security group only accept traffic from instances within the application tier, by reference to the security groups applied to the instances involved. Unless you are using non-TCP protocols, it shouldn't be necessary to have an EC2 instance directly accessible by the internet (even with ports restricted by security groups) without a load balancer, or CloudFront. This helps protect it from unintended access through an operating system or application issue. A subnet can also have a network ACL attached to it, which acts as a stateless firewall. You should configure the network

ACL to narrow the scope of traffic allowed between layers, note that you need to define both inbound and outbound rules.

Some AWS services require components to access the internet for making API calls, where AWS API endpoints are located. Other AWS services use VPC endpoints within your Amazon VPCs. Many AWS services, including Amazon S3 and Amazon DynamoDB, support VPC endpoints, and this technology has been generalized in AWS PrivateLink. We recommend you use this approach to access AWS services, third-party services, and your own services hosted in other VPCs securely. All network traffic on AWS PrivateLink stays on the global AWS backbone and never traverses the internet. Connectivity can only be initiated by the consumer of the service, and not by the provider of the service. Using PrivateLink for external service access allows you to create air-gapped VPCs with no internet access and helps protect your VPCs from external threat vectors. Third-party services can use AWS PrivateLink to allow their customers to connect to the services from their VPCs over private IP addresses. For VPC assets that need to make outbound connections to the internet, these can be made outbound only (one-way) through an AWS managed NAT gateway, outbound only internet gateway, or web proxies that you create and manage.

Implement inspection and protection: Inspect and filter your traffic at each layer. You can inspect your VPC configurations for potential unintended access using VPC Network Access Analyzer. You can specify your network access requirements and identify potential network paths that do not meet them. For components transacting over HTTP-based protocols, a web application firewall can help protect from common attacks. AWS WAF is a web application firewall that lets you monitor and block HTTP(s) requests that match your configurable rules that are forwarded to an Amazon API Gateway API, Amazon CloudFront, or an Application Load Balancer. To get started with AWS WAF, you can use AWS Managed Rules in combination with your own, or use existing partner integrations.

For managing AWS WAF, AWS Shield Advanced protections, and Amazon VPC security groups across AWS Organizations, you can use AWS Firewall Manager. It allows you to centrally configure and manage firewall rules across your accounts and applications, making it easier to scale enforcement of common rules. It also enables you to rapidly respond to attacks, using AWS Shield Advanced, or solutions that can automatically block unwanted requests to your web applications. Firewall Manager also works with AWS Network Firewall. AWS Network Firewall is a managed service that uses a rules engine to give you fine-grained control over both stateful and stateless network traffic. It supports the Suricata compatible open source intrusion prevention system (IPS) specifications for rules to help protect your workload.

Automate network protection: Automate protection mechanisms to provide a self-defending network based on threat intelligence and anomaly detection. For example, intrusion detection and prevention tools that can adapt to current threats and reduce their impact. A web application firewall is an example of where you can automate network protection, for example, by using the AWS WAF Security Automations solution (https://github.com/awslabs/aws-waf-security-automations) to automatically block requests originating from IP addresses associated with known threat actors.

Resources

Refer to the following resources to learn more about AWS best practices for protecting networks.

Video

- AWS Transit Gateway reference architectures for many VPCs
- Application Acceleration and Protection with Amazon CloudFront, AWS WAF, and AWS Shield
- DDoS Attack Detection at Scale

Documentation

- Amazon VPC Documentation

- Getting started with AWS WAF
- Network Access Control Lists
- Security Groups for Your VPC
- Recommended Network ACL Rules for Your VPC
- AWS Firewall Manager
- AWS PrivateLink
- VPC Endpoints
- Amazon Inspector
- AWS Network Firewall

Hands-on

- Lab: Automated Deployment of VPC
- Lab: Automated Deployment of Web Application Firewall

Protecting Compute

Compute resources include EC2 instances, containers, AWS Lambda functions, database services, IoT devices, and more. Each of these compute resource types require different approaches to secure them. However, they do share common strategies that you need to consider: defense in depth, vulnerability management, reduction in attack surface, automation of configuration and operation, and performing actions at a distance. In this section, you will find general guidance for protecting your compute resources for key services. For each AWS service used, it's important for you to check the specific security recommendations in the service documentation.

Perform vulnerability management: Frequently scan and patch for vulnerabilities in your code, dependencies, and in your infrastructure to help protect against new threats.

Starting with the configuration of your compute infrastructure, you can automate creating and updating resources using CloudFormation. CloudFormation allows you to create templates written in YAML or JSON, either using AWS examples or by writing your own. This allows you to create secure-by-default infrastructure templates that you can verify with CloudFormation Guard, to save you time and reduce the risk of configuration error. You can build your infrastructure and deploy your applications using continuous delivery, for example with AWS CodePipeline, to automate the building, testing, and release.

You are responsible for patch management for your AWS resources, including EC2 instances, Amazon Machine Images (AMIs), and many other compute resources. For EC2 instances, AWS Systems Manager Patch Manager automates the process of patching managed instances with both security related and other types of updates. You can use Patch Manager to apply patches for both operating systems and applications. (On Windows Server, application support is limited to updates for Microsoft applications.) You can use Patch Manager to install Service Packs on Windows instances and perform minor version upgrades on Linux instances. You can patch fleets of EC2 instances or your on-premises servers and virtual machines (VMs) by operating system type. This includes supported versions of Windows Server, Amazon Linux, Amazon Linux 2, CentOS, Debian Server, Oracle Linux, Red Hat Enterprise Linux (RHEL), SUSE Linux Enterprise Server (SLES), and Ubuntu Server. You can scan instances to see only a report of missing patches, or you can scan and automatically install all missing patches.

Reduce attack surface: Reduce your exposure to unintended access by hardening operating systems and minimizing the components, libraries, and externally consumable services in use. Start by reducing unused components, whether they are operating system packages, applications, etc. (for EC2-based workloads) or external software modules in your code (for all workloads). You can find many hardening and security configuration guides for common operating systems and server software. For example, you can start with the Center for Internet Security and iterate.

In Amazon EC2, you can create your own AMIs, which you have patched and hardened, to help you meet the specific security requirements for your organization. The patches and other security controls you apply on the AMI are effective at the point in time in which they were created—they are not dynamic unless you modify after launching, for example, with Systems Manager.

You can simplify the process of building secure AMIs with EC2 Image Builder. EC2 Image Builder significantly reduces the effort required to create and maintain golden images without writing and maintaining automation. When software updates become available, Image Builder automatically produces a new image without requiring users to manually initiate image builds. EC2 Image Builder allows you to easily validate the functionality and security of your images before using them in production with AWS-provided tests and your own tests. You can also apply AWS-provided security settings to further secure your images to meet internal security criteria. For example, you can produce images that conform to the Security Technical Implementation Guide (STIG) standard using AWS-provided templates.

Using third-party static code analysis tools, you can identify common security issues such as unchecked function input bounds, as well as applicable CVEs. You can use Amazon CodeGuru for supported languages. Dependency checking tools can also be used to determine whether libraries your code links against are the latest versions, are themselves free of CVEs, and have licensing conditions that meet your software policy requirements.

Using Amazon Inspector, you can perform configuration assessments against your instances for known common vulnerabilities and exposures (CVEs), assess against security benchmarks, and automate the notification of defects. Amazon Inspector runs on production instances or in a build pipeline, and it notifies developers and engineers when findings are present. You can access findings programmatically and direct your team to backlogs and bug-tracking systems. EC2 Image Builder can be used to maintain server images (AMIs) with automated patching, AWS-provided security policy enforcement, and other customizations. When using containers implement ECR Image Scanning in your build pipeline and on a regular basis against your image repository to look for CVEs in your containers.

While Amazon Inspector and other tools are effective at identifying configurations and any CVEs that are present, other methods are required to test your workload at the application level. Fuzzing is a well-known method of finding bugs using automation to inject malformed data into input fields and other areas of your application.

Enable people to perform actions at a distance: Removing the ability for interactive access reduces the risk of human error, and the potential for manual configuration or management. For example, use a change management workflow to manage EC2 instances using tools such as AWS Systems Manager instead of allowing direct access, or via a bastion host. AWS Systems Manager can automate a variety of maintenance and deployment tasks, using features including automation workflows, documents (playbooks), and the run command. AWS CloudFormation stacks build from pipelines and can automate your infrastructure deployment and management tasks without using the AWS Management Console or APIs directly.

Implement managed services: Implement services that manage resources, such as Amazon RDS, AWS Lambda, and Amazon ECS, to reduce your security maintenance tasks as part of the shared responsibility model. For example, Amazon RDS helps you set up, operate, and scale a relational database, automates administration tasks such as hardware provisioning, database setup, patching, and backups. This means you have more free time to focus on securing your application in other ways described in the AWS Well-Architected Framework. AWS Lambda lets you run code without provisioning or managing servers, so you only need to focus on the connectivity, invocation, and security at the code level–not the infrastructure or operating system.

Validate software integrity: Implement mechanisms (e.g. code signing) to validate that the software, code and libraries used in the workload are from trusted sources and have not been tampered with. For example, you should verify the code signing certificate of binaries and scripts to confirm the author, and ensure it has not been tampered with since created by the author. AWS Signer can help ensure the trust and integrity of your code by centrally managing the code-signing lifecycle, including signing certification and public and private keys. You can learn how to use advanced patterns and best practices

for code signing with AWS Lambda. Additionally, a checksum of software that you download, compared to that of the checksum from the provider, can help ensure it has not been tampered with.

Automate compute protection: Automate your protective compute mechanisms including vulnerability management, reduction in attack surface, and management of resources. The automation will help you invest time in securing other aspects of your workload, and reduce the risk of human error.

Resources

Refer to the following resources to learn more about AWS best practices for protecting compute.

Video

- Security best practices for the Amazon EC2 instance metadata service
- Securing Your Block Storage on AWS
- Securing Serverless and Container Services
- Running high-security workloads on Amazon EKS
- Architecting Security through Policy Guardrails in Amazon EKS

Documentation

- Security Overview of AWS Lambda
- Security in Amazon EC2
- AWS Systems Manager
- AWS Image Builder
- Amazon Inspector
- Writing your own AWS Systems Manager documents
- Replacing a Bastion Host with Amazon EC2 Systems Manager
- Accessing EC2 instances using key pairs
- AWS Signer

Hands-on

- Lab: Automated Deployment of EC2 Web Application

Data Protection

Before architecting any workload, foundational practices that influence security should be in place. For example, data classification provides a way to categorize data based on levels of sensitivity, and encryption protects data by way of rendering it unintelligible to unauthorized access. These methods are important because they support objectives such as preventing mishandling or complying with regulatory obligations.

In AWS, there are a number of different approaches you can use when addressing data protection. The following section describes how to use these approaches.

Topics

Data Classification

Data classification provides a way to categorize organizational data based on criticality and sensitivity in order to help you determine appropriate protection and retention controls.

Identify the data within your workload: You need to understand the type and classification of data your workload is processing, the associated business processes, data owner, applicable legal and compliance requirements, where it's stored, and the resulting controls that are needed to be enforced. This may include classifications to indicate if the data is intended to be publicly available, if the data is internal use only such as customer personally identifiable information (PII), or if the data is for more restricted access such as intellectual property, legally privileged or marked sensitive, and more. By carefully managing an appropriate data classification system, along with each workload's level of protection requirements, you can map the controls and level of access/protection appropriate for the data. For example, public content is available for anyone to access, but important content is encrypted and stored in a protected manner that requires authorized access to a key for decrypting the content.

Define data protection controls: By using resource tags, separate AWS accounts per sensitivity (and potentially also per caveat / enclave / community of interest), IAM policies, Organizations SCPs, AWS KMS, and AWS CloudHSM, you can define and implement your policies for data classification and protection with encryption. For example, if you have a project with S3 buckets that contain highly critical data or EC2 instances that process confidential data, they can be tagged with a "Project=ABC" tag. Only your immediate team knows what the project code means, and it provides a way to use attribute-based access control. You can define levels of access to the AWS KMS encryption keys through key policies and grants to ensure that only appropriate services have access to the sensitive content through a secure mechanism. If you are making authorization decisions based on tags you should make sure that the permissions on the tags are defined appropriately using tag policies in AWS Organizations.

Define data lifecycle management: Your defined lifecycle strategy should be based on sensitivity level as well as legal and organization requirements. Aspects including the duration for which you retain data, data destruction processes, data access management, data transformation, and data sharing should be considered. When choosing a data classification methodology, balance usability versus access. You should also accommodate the multiple levels of access and nuances for implementing a secure, but still usable, approach for each level. Always use a defense in depth approach and reduce human access to data and mechanisms for transforming, deleting, or copying data. For example, require users to strongly authenticate to an application, and give the application, rather than the users, the requisite access

permission to perform “action at a distance.” In addition, ensure that users come from a trusted network path and require access to the decryption keys. Use tools, such as dashboards and automated reporting, to give users information from the data rather than giving them direct access to the data.

Automate identification and classification: Automating the identification and classification of data can help you implement the correct controls. Using automation for this instead of direct access from a person reduces the risk of human error and exposure. You should evaluate using a tool, such as Amazon Macie, that uses machine learning to automatically discover, classify, and protect sensitive data in AWS. Amazon Macie recognizes sensitive data, such as personally identifiable information (PII) or intellectual property, and provides you with dashboards and alerts that give visibility into how this data is being accessed or moved.

Resources

Refer to the following resources to learn more about data classification.

Documentation

- Data Classification Whitepaper
- Tagging best practices
- Amazon S3 Object Tagging

Protecting Data at Rest

Data at rest represents any data that you persist in non-volatile storage for any duration in your workload. This includes block storage, object storage, databases, archives, IoT devices, and any other storage medium on which data is persisted. Protecting your data at rest reduces the risk of unauthorized access, when encryption and appropriate access controls are implemented.

Encryption and tokenization are two important but distinct data protection schemes.

Tokenization is a process that allows you to define a token to represent an otherwise sensitive piece of information (for example, a token to represent a customer’s credit card number). A token must be meaningless on its own, and must not be derived from the data it is tokenizing–therefore, a cryptographic digest is not usable as a token. By carefully planning your tokenization approach, you can provide additional protection for your content, and you can ensure that you meet your compliance requirements. For example, you can reduce the compliance scope of a credit card processing system if you leverage a token instead of a credit card number.

Encryption is a way of transforming content in a manner that makes it unreadable without a secret key necessary to decrypt the content back into plaintext. Both tokenization and encryption can be used to secure and protect information as appropriate. Further, masking is a technique that allows part of a piece of data to be redacted to a point where the remaining data is not considered sensitive. For example, PCI-DSS allows the last four digits of a card number to be retained outside the compliance scope boundary for indexing.

Implement secure key management: By defining an encryption approach that includes the storage, rotation, and access control of keys, you can help provide protection for your content against unauthorized users and against unnecessary exposure to authorized users. AWS KMS helps you manage encryption keys and integrates with many AWS services. This service provides durable, secure, and redundant storage for your AWS KMS keys. You can define your key aliases as well as key-level policies. The policies help you define key administrators as well as key users. Additionally, AWS CloudHSM is a cloud-based hardware security module (HSM) that enables you to easily generate and use your own encryption keys in the AWS Cloud. It helps you meet corporate, contractual, and regulatory compliance requirements for data security by using FIPS 140-2 Level 3 validated HSMs.

Enforce encryption at rest: You should ensure that the only way to store data is by using encryption. AWS KMS integrates seamlessly with many AWS services to make it easier for you to encrypt all your data at rest. For example, in Amazon S3 you can set default encryption on a bucket so that all new objects are automatically encrypted. Additionally, Amazon EC2 and Amazon S3 support the enforcement of encryption by setting default encryption. You can use AWS Managed Config Rules to check automatically that you are using encryption, for example, for EBS volumes, RDS instances, and S3 buckets.

Enforce access control: Different controls including access (using least privilege), backups (see Reliability whitepaper), isolation, and versioning can all help protect your data at rest. Access to your data should be audited using detective mechanisms covered earlier in this paper including CloudTrail, and service level log, such as S3 access logs. You should inventory what data is publicly accessible, and plan for how you can reduce the amount of data available over time. Amazon S3 Glacier Vault Lock and S3 Object Lock are capabilities providing mandatory access control—once a vault policy is locked with the compliance option, not even the root user can change it until the lock expires. The mechanism meets the Books and Records Management requirements of the SEC, CFTC, and FINRA. For more details, see this whitepaper.

Audit the use of encryption keys: Ensure that you understand and audit the use of encryption keys to validate that the access control mechanisms on the keys are appropriately implemented. For example, any AWS service using an AWS KMS key logs each use in AWS CloudTrail. You can then query AWS CloudTrail, by using a tool such as Amazon CloudWatch Insights, to ensure that all uses of your keys are valid.

Use mechanisms to keep people away from data: Keep all users away from directly accessing sensitive data and systems under normal operational circumstances. For example, use a change management workflow to manage EC2 instances using tools instead of allowing direct access or a bastion host. This can be achieved using AWS Systems Manager Automation, which uses automation documents that contain steps you use to perform tasks. These documents can be stored in source control, be peer reviewed before running, and tested thoroughly to minimize risk compared to shell access. Business users could have a dashboard instead of direct access to a data store to run queries. Where CI/CD pipelines are not used, determine which controls and processes are required to adequately provide a normally disabled break-glass access mechanism.

Automate data at rest protection: Use automated tools to validate and enforce data at rest controls continuously, for example, verify that there are only encrypted storage resources. You can automate validation that all EBS volumes are encrypted using AWS Config Rules. AWS Security Hub can also verify a number of different controls through automated checks against security standards. Additionally, your AWS Config Rules can automatically remediate noncompliant resources.

Resources

Refer to the following resources to learn more about AWS best practices for protecting data at rest.

Video

- How Encryption Works in AWS
- Securing Your Block Storage on AWS
- Achieving security goals with AWS CloudHSM
- Best Practices for Implementing AWS Key Management Service
- A Deep Dive into AWS Encryption Services

Documentation

- Protecting Amazon S3 Data Using Encryption

- Amazon EBS Encryption
- Encrypting Amazon RDS Resources
- Protecting Data Using Encryption
- How AWS services use AWS KMS
- Amazon EBS Encryption
- AWS Key Management Service
- AWS CloudHSM
- AWS KMS Cryptographic Details Whitepaper
- Using Key Policies in AWS KMS
- Using Bucket Policies and User Policies
- AWS Crypto Tools

Protecting Data in Transit

Data in transit is any data that is sent from one system to another. This includes communication between resources within your workload as well as communication between other services and your end users. By providing the appropriate level of protection for your data in transit, you protect the confidentiality and integrity of your workload's data.

Implement secure key and certificate management: Store encryption keys and certificates securely and rotate them at appropriate time intervals with strict access control. The best way to accomplish this is to use a managed service, such as AWS Certificate Manager (ACM). It lets you easily provision, manage, and deploy public and private Transport Layer Security (TLS) certificates for use with AWS services and your internal connected resources. TLS certificates are used to secure network communications and establish the identity of websites over the internet as well as resources on private networks. ACM integrates with AWS resources, such as Elastic Load Balancers, AWS distributions, and APIs on API Gateway, also handling automatic certificate renewals. If you use ACM to deploy a private root CA, both certificates and private keys can be provided by it for use in EC2 instances, containers, etc.

Enforce encryption in transit: Enforce your defined encryption requirements based on appropriate standards and recommendations to help you meet your organizational, legal, and compliance requirements. AWS services provide HTTPS endpoints using TLS for communication, thus providing encryption in transit when communicating with the AWS APIs. Insecure protocols, such as HTTP, can be audited and blocked in a VPC through the use of security groups. HTTP requests can also be automatically redirected to HTTPS in Amazon CloudFront or on an Application Load Balancer. You have full control over your computing resources to implement encryption in transit across your services. Additionally, you can use VPN connectivity into your VPC from an external network to facilitate encryption of traffic. Third-party solutions are available in the AWS Marketplace, if you have special requirements.

Authenticate network communications: Using network protocols that support authentication allows for trust to be established between the parties. This adds to the encryption used in the protocol to reduce the risk of communications being altered or intercepted. Common protocols that implement authentication include Transport Layer Security (TLS), which is used in many AWS services, and IPsec, which is used in AWS Virtual Private Network (AWS VPN).

Automate detection of unintended data access: Use tools such as Amazon GuardDuty to automatically detect suspicious activity or attempts to move data outside of defined boundaries. For example, GuardDuty can detect S3 read activity that is unusual with the Exfiltration:S3/ObjectRead.Unusual finding. In addition to Amazon GuardDuty, Amazon VPC Flow Logs, which capture network traffic information, can be used with Amazon EventBridge to trigger detection of abnormal connections–both successful and denied. S3 Access Analyzer can help assess what data is accessible to who in your S3 buckets.

Secure data from between VPC or on-premises locations: You can use AWS PrivateLink to create a secure and private network connection between Amazon Virtual Private Cloud (Amazon VPC) or on-premises connectivity to services hosted in AWS. You can access AWS services, third-party services, and services in other AWS accounts as if they were on your private network. With AWS PrivateLink, you can access services across accounts with overlapping IP CIDRs without needing an Internet Gateway or NAT. You also do not have to configure firewall rules, path definitions, or route tables. Traffic stays on the Amazon backbone and doesn't traverse the internet, therefore your data is protected. You can maintain compliance with industry-specific compliance regulations, such as HIPAA and EU/US Privacy Shield. AWS PrivateLink seamlessly works with third-party solutions to create a simplified global network, allowing you to accelerate your migration to the cloud and take advantage of available AWS services.

Resources

Refer to the following resources to learn more about AWS best practices for protecting data in transit.

Video

- Networking best practices and tips with the Well-Architected Framework
- Deep Dive on AWS Certificate Manager Private CA

Documentation

- AWS Certificate Manager
- HTTPS Listeners for Your Application Load Balancer
- AWS VPN
- API Gateway Edge-Optimized

Incident Response

Even with mature preventive and detective controls, your organization should implement mechanisms to respond to and mitigate the potential impact of security incidents. Your preparation strongly affects the ability of your teams to operate effectively during an incident, to isolate, contain and perform forensics on issues, and to restore operations to a known good state. Putting in place the tools and access ahead of a security incident, then routinely practicing incident response through game days, helps ensure that you can recover while minimizing business disruption.

Topics

Design Goals of Cloud Response

Although the general processes and mechanisms of incident response, such as those defined in the NIST SP 800-61 Computer Security Incident Handling Guide, remain true, we encourage you to evaluate these specific design goals that are relevant to responding to security incidents in a cloud environment:

- **Establish response objectives:** Work with your stakeholders, legal counsel, and organizational leadership to determine the goal of responding to an incident. Some common goals include containing and mitigating the issue, recovering the affected resources, preserving data for forensics, and attribution.
- **Document plans:** Create plans to help you respond to, communicate during, and recover from an incident.
- **Respond using the cloud:** Implement your response patterns where the event and data occurs.
- **Know what you have and what you need:** Preserve logs, snapshots, and other evidence by copying them to a centralized security cloud account. Use tags, metadata, and mechanisms that enforce retention policies. For example, you might choose to use the Linux dd command or a Windows equivalent to make a complete copy of the data for investigative purposes.
- **Use redeployment mechanisms:** If a security anomaly can be attributed to a misconfiguration, the remediation might be as simple as removing the variance by redeploying the resources with the proper configuration. When possible, make your response mechanisms safe to execute more than once and in environments in an unknown state.
- **Automate where possible:** As you see issues or incidents repeat, build mechanisms that programmatically triage and respond to common situations. Use human responses for unique, new, and sensitive incidents.
- **Choose scalable solutions:** Strive to match the scalability of your organization's approach to cloud computing, and reduce the time between detection and response.
- **Learn and improve your process:** When you identify gaps in your process, tools, or people, implement plans to fix them. Simulations are safe methods to find gaps and improve processes.

In AWS, there are a number of different approaches you can use when addressing incident response. The following section describes how to use these approaches:

- **Educate** your security operations and incident response staff about cloud technologies and how your organization intends to use them.
- **Prepare** your incident response team to detect and respond to incidents in the cloud, enable detective capabilities, and ensure appropriate access to the necessary tools and cloud services. Additionally, prepare the necessary runbooks, both manual and automated, to ensure reliable and consistent responses. Work with other teams to establish expected baseline operations, and use that knowledge to identify deviations from those normal operations.
- **Simulate** both expected and unexpected security events within your cloud environment to understand the effectiveness of your preparation.
- **Iterate** on the outcome of your simulation to improve the scale of your response posture, reduce time to value, and further reduce risk.

Educate

Automated processes enable organizations to spend more time focusing on measures to increase the security of their workloads. Automated incident response also makes humans available to correlate events, practice simulations, devise new response procedures, perform research, develop new skills, and test or build new tools. Despite increased automation, your team, specialists, and responders within a security organization still require continuous education. We encourage you to review and incorporate the following areas when thinking about educating your security teams:

Development Skills: Equipping security professionals with programming skills will accelerate your organization's automation efforts. This includes not only ensuring education around programming languages, such as Python, but also ensuring familiarity with source control system, version control, and CI/CD processes. When developers have this understanding, they'll increase efficiency and reduce errors when automating.

AWS Services: It's important for your security team to be proficient with the security services offered by AWS. Understanding how to use cloud native tools will reduce response time and build team confidence. In addition, establish a cadence of education about new services and capabilities in order to continually iterate your capabilities. Just as the threat landscape changes, so do the tools.

Application Awareness: Train your incident response team on the specifics of the workloads and environments that they own. This includes understanding what logs are emitted, what information the logs contain, the traffic flow of the application, and what authentication and authorization mechanisms are in use. This is a critical component as deep understanding of your organization's infrastructure and applications provides an advantage to protecting them.

The best way to learn is hands-on, through running incident response game days. This allows for experts in your team to hone the tools and techniques while teaching others. This is covered in more detail in the Simulate section.

Finally, don't forget to maintain required education for your entire organization. Security awareness is an important line of defense. Users should be trained to report suspicious behavior to your security team for further investigation.

Prepare

During an incident, your incident response teams must have access to various tools and the workload resources involved in the incident. Make sure that your teams have appropriate pre-provisioned access to perform their duties before an event occurs. All tools, access, and plans should be documented and tested before an event occurs to make sure that they can provide a timely response.

Identify key personnel and external resources: When you define your approach to incident response in the cloud, in unison with other teams (such as your legal counsel, leadership, business stakeholders, AWS

Support Services, and others), you must identify key personnel, stakeholders, and relevant contacts. To reduce dependency and decrease response time, make sure that your team, specialist security teams, and responders are educated about the services that you use and have opportunities to practice hands-on.

We encourage you to identify external AWS security partners that can provide you with outside expertise and a different perspective to augment your response capabilities. Your trusted security partners can help you identify potential risks or threats that you might not be familiar with.

Develop incident management plans: Create plans to help you respond to, communicate during, and recover from an incident. For example, you can start at incident response plan with the most likely scenarios for your workload and organization. Include how you would communicate and escalate both internally and externally. Create incident response plans in the form of playbooks, starting with the most likely scenarios for your workload and organization. These might be events that are currently generated. If you need a starting place, you should look at AWS Trusted Advisor and Amazon GuardDuty findings. Use a simple format such as markdown so it's easily maintained but ensure that important commands or code snippets are included so they can be executed without having to lookup other documentation.

Start simple and iterate. Work closely with your security experts and partners to identify the tasks required to ensure that the processes are possible. Define the manual descriptions of the processes you perform. After this, test the processes and iterate on the runbook pattern to improve the core logic of your response. Determine what the exceptions are, and what the alternative resolutions are for those scenarios. For example, in a development environment, you might want to terminate a misconfigured Amazon EC2 instance. But, if the same event occurred in a production environment, instead of terminating the instance, quarantine the instance by snapshotting volumes, and capturing memory. In addition, verify with stakeholders that critical data will not be lost, and evidence is collected before shutdown. Include how you would communicate and escalate both internally and externally. When you are comfortable with the manual response to the process, automate it to reduce the time to resolution.

Pre-provision access: Ensure that incident responders have the correct access pre-provisioned into AWS and other relevant systems to reduce the time for investigation through to recovery. Determining how to get access for the right people during an incident delays the time it takes to respond, and can introduce other security weaknesses if access is shared or not properly provisioned while under pressure. You must know what level of access your team members require (for example, what kinds of actions they are likely to take) and you must provision access in advance. Access in the form of roles or users created specifically to respond to a security incident are often privileged in order to provide sufficient access. Therefore, use of these user accounts should be restricted, they should not be used for daily activities, and usage alerted on.

Pre-deploy tools: Ensure that security personnel have the right tools pre-deployed into AWS to reduce the time for investigation through to recovery.

To automate security engineering and operations functions, you can use a comprehensive set of APIs and tools from AWS. You can fully automate identity management, network security, data protection, and monitoring capabilities and deliver them using popular software development methods that you already have in place. When you build security automation, your system can monitor, review, and initiate a response, rather than having people monitor your security position and manually react to events. An effective way to automatically provide searchable and relevant log data across AWS services to your incident responders is to enable Amazon Detective.

If your incident response teams continue to respond to alerts in the same way, they risk alert fatigue. Over time, the team can become desensitized to alerts and can either make mistakes handling ordinary situations or miss unusual alerts. Automation helps avoid alert fatigue by using functions that process the repetitive and ordinary alerts, leaving humans to handle the sensitive and unique incidents. Integrating anomaly detection systems, such as GuardDuty, CloudTrail Insights, and CloudWatch Anomaly Detection, can reduce the burden of common threshold-based alerts.

You can improve manual processes by programmatically automating steps in the process. After you define the remediation pattern to an event, you can decompose that pattern into actionable logic, and

write the code to perform that logic. Responders can then execute that code to remediate the issue. Over time, you can automate more and more steps, and ultimately automatically handle whole classes of common incidents.

For tools that execute within the operating system of your EC2 instance, you should evaluate using the AWS Systems Manager Run Command, which enables you to remotely and securely administrate instances using an agent that you install on your Amazon EC2 instance operating system. It requires the AWS Systems Manager Agent (SSM Agent), which is installed by default on many Amazon Machine Images (AMIs). Be aware, though, that once an instance has been compromised, no responses from tools or agents running on it should be considered trustworthy.

Prepare forensic capabilities: It's important for your incident responders to have an understanding of when and how the forensic investigation fits into your response plan. Your organization should define what evidence is collected and what tools are used in the process. Identify and prepare forensic investigation capabilities that are suitable, including external specialists, tools, and automation. A key decision that you should make upfront is if you will collect data from a "live" system. Some data, such as the contents of volatile memory or active network connections, will be lost if the system is powered off or rebooted.

Your response team can combine tools, such as AWS System Manager, Amazon EventBridge, and AWS Lambda, to automatically run forensic tools within an operating system and VPC traffic mirroring to obtain a network packet capture, to gather non-persistent evidence. Conduct other activities, such as log analysis or analyzing disk images, in a dedicated security account with customized forensic workstations and tools accessible to your responders.

Routinely ship relevant logs to a data store that provides high durability and integrity. Responders should have access to those logs. AWS offers several tools that can make log investigation easier, such as Amazon Athena, Amazon OpenSearch Service (OpenSearch Service), and CloudWatch Log Insights. Additionally, preserve evidence securely using S3 Object Lock. This service follows the WORM (write-once-read-many) model and prevents objects from being deleted or overwritten for a defined period of time. As forensic investigation techniques require specialist training, you might need to engage external specialists.

Simulate

Run game days: Game days, also known as simulations or exercises, are internal events that provide a structured opportunity to practice your incident management plans and procedures during a realistic scenario. These events should exercise responders using the same tools and techniques that would be used in a real-world scenario - even mimicking real-world environments. Game days are fundamentally about being prepared and iteratively improving your response capabilities. Some of the reasons you might find value in performing game day activities include:

- Validating readiness
- Developing confidence – learning from simulations and training staff
- Following compliance or contractual obligations
- Generating artifacts for accreditation
- Being agile – incremental improvement
- Becoming faster and improving tools
- Refining communication and escalation
- Developing comfort with the rare and the unexpected

For these reasons, the value derived from participating in a simulation activity increases an organization's effectiveness during stressful events. Developing a simulation activity that is both realistic and beneficial

can be a difficult exercise. Although testing your procedures or automation that handles well-understood events has certain advantages, it is just as valuable to participate in creative Security Incident Response Simulations (SIRS) activities to test yourself against the unexpected and continuously improve.

Create custom simulations tailored to your environment, team, and tools. Find an issue and design your simulation around it. This could be something like a leaked credential, a server communicating with unwanted systems, or a misconfiguration that results in unauthorized exposure. Identify engineers who are familiar with your organization to create the scenario and another group to participate. The scenario should be realistic and challenging enough to be valuable. It should include the opportunity to get hands on with logging, notifications, escalations, and executing runbooks or automation. During the simulation, your responders should exercise their technical and organizational skills, and leaders should be involved to build their incident management skills. At the conclusion of the simulation, celebrate the efforts of the team and look for ways to iterate, repeat, and expand into further simulations.

AWS has created Incident Response Runbook templates that you can use not only to prepare your response efforts, but also as a basis for a simulation. When planning, a simulation can be broken into five phases.

Evidence gathering: In this phase, a team will get alerts through various means, such as an internal ticketing system, alerts from monitoring tooling, anonymous tips, or even public news. Teams then start to review infrastructure and application logs to determine the source of the compromise. This step should also involve internal escalations and incident leadership. Once identified, teams move on to containing the incident.

Contain the incident: Teams will have determined there has been an incident and established the source of the compromise. Teams now should take action to contain it, for example, by disabling compromised credentials, isolating a compute resource, or revoking a role's permission.

Eradicate the incident: Now that they've contained the incident, teams will work towards mitigating any vulnerabilities in applications or infrastructure configurations that were susceptible to the compromise. This could include rotating all credentials used for a workload, modifying Access Control Lists (ACLs) or changing network configurations.

Recover from the incident: After teams have implemented best practices, they can now focus on ensuring the complete removal of the compromise. They can do this by restoring files or pieces of data that were compromised during the incident from backups or previous versions. Also, they must ensure that all suspicious activity has ceased and continue to monitor to ensure a stable state.

Post-incident debrief: This session allows teams to share learnings and increase the overall effectiveness of the organization's incident response plan. Here the teams should review handling of the incident in detail, document lessons learned, update runbooks based on learnings, and determine if new risk assessments are required.

Iterate

Automate containment and recovery capability: Automate containment and recovery of an incident to reduce response times and organizational impact.

Once you create and practice the processes and tools from your playbooks, you can deconstruct the logic into a code-based solution, which can be used as a tool by many responders to automate the response and remove variance or guess-work by your responders. This can speed up the lifecycle of a response. The next goal is to enable this code to be fully automated by being invoked by the alerts or events themselves, rather than by a human responder, to create an event-driven response. These processes should also automatically add relevant data to your security systems. For example, and incident involving traffic from an unwanted IP address can automatically populate an AWS WAF block list or Network Firewall rule group to prevent further activity.

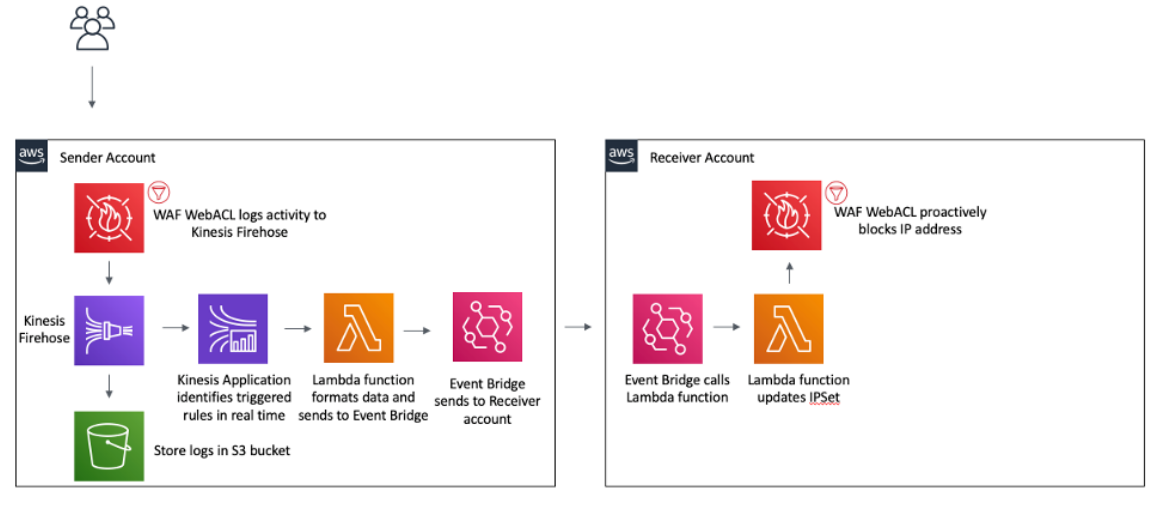

Figure 3: AWS WAF automate blocking of known malicious IP addresses.

With an event-driven response system, a detective mechanism triggers a responsive mechanism to automatically remediate the event. You can use event-driven response capabilities to reduce the time-to-value between detective mechanisms and responsive mechanisms. To create this event-driven architecture, you can use AWS Lambda, which is a serverless compute service that runs your code in response to events and automatically manages the underlying compute resources for you. For example, assume that you have an AWS account with the AWS CloudTrail service enabled. If AWS CloudTrail is ever disabled (through the `cloudtrail:StopLogging` API call), you can use Amazon EventBridge to monitor for the specific `cloudtrail:StopLogging` event, and invoke an AWS Lambda function to call `cloudtrail:StartLogging` to restart logging.

Resources

Refer to the following resources to learn more about current AWS best practices for incident response.

Videos

- Prepare for & respond to security incidents in your AWS environment
- Automating Incident Response and Forensics
- DIY guide to runbooks, incident reports, and incident response

Documentation

- AWS Incident Response Guide
- Amazon Detective
- AWS Step Functions
- Amazon EventBridge
- CloudEndure Disaster Recovery
- Blog: How to perform automated incident response in a multi-account environment
- Blog: How to automate incident response in the AWS Cloud for EC2 instances
- Blog: Automatically updating AWS WAF rule in real time using Amazon EventBridge

- Blog: What is a cyber range and how do you build one on AWS?

Hands-on

- Lab: Incident Response with AWS Console and CLI
- Lab: Incident Response Playbook with Jupyter - AWS IAM
- Blog: Orchestrating a security incident response with AWS Step Functions

Conclusion

Security is an ongoing effort. When incidents occur, they should be treated as opportunities to improve the security of the architecture. Having strong identity controls, automating responses to security events, protecting infrastructure at multiple levels, and managing well-classified data with encryption provides defense in depth that every organization should implement. This effort is easier thanks to the programmatic functions and AWS features and services discussed in this paper.

AWS strives to help you build and operate architectures that protect information, systems, and assets while delivering business value.

Contributors

The following individuals and organizations contributed to this document:

- Adam Cerini, Senior Solution Architect, Amazon Web Services
- Ben Potter, Principal Security Lead Well-Architected, Amazon Web Services
- Bill Shinn, Senior Principal, Office of the CISO, Amazon Web Services
- Brigid Johnson, Senior Software Development Manager, AWS Identity, Amazon Web Services
- Byron Pogson, Senior Solution Architect, Amazon Web Services
- Charlie Hammell, Principal Enterprise Architect, Amazon Web Services
- Darran Boyd, Principal Security Solutions Architect, Financial Services, Amazon Web Services
- Dave Walker, Principal Specialist Solutions Architect, Security and Compliance, Amazon Web Services
- John Formento, Senior Solution Architect, Amazon Web Services
- Paul Hawkins, Senior Security Strategist, Amazon Web Services
- Sam Elmalak, Senior Technology Leader, Amazon Web Services
- Aden Leirer, Content Program Manager Well-Architected, Amazon Web Services

Further Reading

For additional help, please consult the following sources:

- AWS Well-Architected Framework whitepaper
- AWS Architecture Center

Document Revisions

To be notified about updates to this whitepaper, subscribe to the RSS feed.

update-history-change	update-history-description	update-history-date
Minor update (p. 41)	Additional AWS PrivateLink information added and corrected broken links.	May 19, 2022
Minor update (p. 29)	Added AWS PrivateLink.	May 6, 2022
Minor update (p. 41)	Removed non-inclusive language.	April 22, 2022
Minor update (p. 41)	Added information about VPC Network Access Analyzer.	February 2, 2022
Minor update (p. 1)	Added Sustainability Pillar to introduction.	December 2, 2021
Minor update (p. 34)	Corrected link to Incident Response Runbook templates.	July 28, 2021
Minor update (p. 41)	Fixed broken link.	May 27, 2021
Minor update (p. 41)	Editorial changes throughout.	May 17, 2021
Major update (p. 41)	Added section on governance, added detail to various sections, added new features and services throughout.	May 7, 2021
Minor update (p. 41)	Updated links.	March 10, 2021
Minor update (p. 41)	Fixed broken link.	July 15, 2020
Updates for new Framework (p. 41)	Updated guidance on account, identity, and permissions management.	July 8, 2020
Updates for new Framework (p. 41)	Updated to expand advice in every area, new best practices, services and features.	April 30, 2020
Whitepaper updated (p. 41)	Updates to reflect new AWS services and features, and updated references.	July 1, 2018
Whitepaper updated (p. 41)	Updated System Security Configuration and Maintenance section to reflect new AWS services and features.	May 1, 2017
Initial publication (p. 41)	Security Pillar - AWS Well-Architected Framework published.	November 1, 2016

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.