AWS Whitepaper

Security Overview of Amazon API Gateway

Security Overview of Amazon API Gateway: AWS Whitepaper

Table of Contents

Security Overview of Amazon API Gateway

Publication date: **November 12, 2020** (*Document revisions*)

Abstract

This whitepaper presents a deep dive into Amazon API Gateway and integrated Amazon Web Services (AWS) services through a security lens. It provides a well-rounded picture of the service for new adopters, and a deeper understanding of Amazon API Gateway for current users.

The intended audience for this whitepaper includes Chief Information Security Officers (CISOs), information security groups, security analysts, enterprise architects, compliance teams, and anyone interested in understanding the security features of API Gateway and its related services.

Are you Well-Architected?

The AWS Well-Architected Framework helps you understand the pros and cons of the decisions you make when building systems in the cloud. The six pillars of the Framework allow you to learn architectural best practices for designing and operating reliable, secure, efficient, cost-effective, and sustainable systems. Using the AWS Well-Architected Tool, available at no charge in the AWS Management Console, you can review your workloads against these best practices by answering a set of questions for each pillar.

For more expert guidance and best practices for your cloud architecture—reference architecture deployments, diagrams, and whitepapers—refer to the AWS Architecture Center.

Introduction

Today, more business workloads use Amazon API Gateway to enable API-driven architectures, improving scalability, performance, and cost efficiency, without managing the underlying infrastructure. These workloads scale to thousands of concurrent requests per second. API Gateway is used by thousands of AWS customers to serve trillions of requests every month.

The managed environment model of API Gateway intentionally hides many implementation details from the user. This makes some existing best practices for cloud security irrelevant, and creates the need for new best practices. This paper presents a detailed view of these best practices.

About Amazon API Gateway

Amazon API Gateway is a fully-managed service that enables developers to create, publish, maintain, monitor, and secure APIs at any scale. APIs act as the front door for applications to access data, business logic, or functionality from backend services. Using API Gateway, you can create RESTful APIs and WebSocket APIs that enable near real-time, two-way communication applications. API Gateway supports a variety of backend integrations, enabling containerized, serverless, and traditional instance-based workloads.

API Gateway handles all the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls. This includes traffic management, cross-origin resource sharing (CORS) support, authorization and access control, throttling, monitoring, caching, and API version management. API Gateway has no minimum fees or startup costs. You pay for the API calls you receive, and the amount of data transferred out. With the API Gateway tiered pricing model, your cost per million invocations reduces as your API usage scales.

Benefits of Amazon API Gateway

API Gateway offers a variety of benefits and capabilities:

- **Unified front door** – Use API Gateway to compose unified APIs to a variety of integration types and microservices with advanced routing and transformation features.
- **Security and governance built in** – API Gateway supports authorization using any form of bearer or JSON web tokens (JWTs), integration with AWS Web Application Firewall (AWS WAF) for layer 7 request validation, and integration with AWS CloudTrail and AWS Config to enable auditing, logging, monitoring, and compliance out of the box.
- **Standards built in** – API Gateway supports OpenAPI specification versions 2 and 3 for import and export of APIs, and authorization with native OpenID Connect and OAuth 2.0 token parsing.
- **Regulatory compliance support** – Use API Gateway to build architectures and systems to meet requirements for regulatory compliance attestations including SOC, PCI, ISO, FedRAMP, HIPAA, and more.
- **Observability built in** – Native integrations with Amazon CloudWatch and AWS X-Ray provide extensive Amazon CloudWatch metrics, monitoring and alarming, and end-to-end tracing capabilities.
- **API lifecycle management** – Use API Gateway to run multiple versions of the same API simultaneously, so that applications can continue to call previous API versions even after newer

versions are published. API Gateway also helps you manage multiple release stages for each API version, such as alpha, beta, and production. Each API stage can be configured to interact with different backend endpoints based on your API setup. Stage and version management allow you to test new API versions while ensuring backward-compatibility as user communities transition to adopt the latest release.

- **Streamlined developer experience** – An open-source developer portal enables streamlined API registration and onboarding processes. It issues API keys to authenticated users and enables them to interactively explore and test APIs. Third-party developers of your APIs can download generated client SDKs for a number of platforms. You can use these SDKs to test new APIs from your applications and distribute them to third-party developers. The generated SDKs handle API keys and sign requests using AWS credentials. API Gateway can generate client SDKs for numerous programming languages.
- **Performance at any scale** – API Gateway is an always-on, scalable service that supports practically any load with no warm-up limitations. It provides you with the lowest possible latency for API requests and responses using cached content and by accelerating content delivery with global edge network locations using Amazon CloudFront. It can also handle bursts of traffic for your workloads while throttling and authorizing API calls, to help ensure that backend operations can withstand traffic spikes and not be unnecessarily called.
- **Pay for value pricing** – Cost savings are realized at scale through the API Gateway simple, tiered, price-per-million request pricing. You pay only for the requests made to your API, with no minimum.

API types

API Gateway supports multiple API types and a variety of architectural patterns:

- **HTTP APIs** – Build stateless RESTful APIs optimized for serverless workloads and HTTP backends using HTTP APIs. HTTP APIs are the best choice for building APIs that require only API proxy functionality. If your APIs require API proxy functionality and API management features in a single solution, API Gateway also offers REST APIs.
- **WebSocket APIs** – Build real-time, two-way communication applications, such as chat apps and streaming dashboards, with WebSocket APIs. API Gateway maintains a persistent connection to handle message transfer between your backend service and your clients.

Endpoint types

Amazon API Gateway offers three types of endpoints:

- **Private API endpoints** – Can be accessed only from your Amazon Virtual Private Cloud (Amazon VPC) and approved subnets using an interface VPC endpoint.
- **Regional API endpoints** – Offload transport layer security (TLS) within the API deployment in your chosen AWS Region. This is suggested for use cases where API client calls originate in the same region, or for when you want to custom-manage an Amazon CloudFront distribution with a Regional API Gateway endpoint as your origin for dynamic content. This is the default selection for HTTP and WebSocket API Gateway endpoints.
- **Edge-optimized API endpoints** – Provide API access to geographically distributed clients with managed edge network acceleration built-in. This is the default selection for REST API Gateway endpoints. It should not be used for APIs where clients consist of other services within the same Region, or when you require granular control of CloudFront CDN caching behaviors. Client TLS connections end at the CloudFront edge location where the API request is first routed, and AWS manages TLS communication between CloudFront and API Gateway instances.

Cost for Amazon API Gateway-based applications

With API Gateway, you pay only for invocation requests made to your APIs. There are no minimum fees or upfront commitments. For HTTP APIs and REST APIs, you pay only for the API calls you receive and the amount of data transferred out. There are no data transfer out charges for Private APIs, though AWS PrivateLink charges apply when using Private APIs in API Gateway. API Gateway also provides optional data caching, charged at an hourly rate that varies based on the cache size you select. For WebSocket APIs, you pay only when your APIs are in use based on number of messages sent and received and connection minutes, as well as any data transfer.

The API Gateway free tier includes one million HTTP API calls, one million REST API calls, one million messages, and 750,000 connection minutes per month for up to 12 months.

Security design principles

This whitepaper provides best practice guidance for securing your workloads when using API Gateway. Building on the principles of the Security Pillar of the AWS Well-Architected Framework, the following design principles can help strengthen your security:

- **Understand the AWS security and compliance Shared Responsibility Model** – Security and Compliance is a shared responsibility between AWS and you as a customer. Understanding this shared model can help reduce your operational burden.
- **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.
- **Implement a strong identity and access 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 long-lived credentials through integrated authentication and authorization.
- **Minimize attack surface area** – When architecting your application, examine the connectivity requirements of each component and restrict the options to the minimum exposure possible.
- **Mitigate Distributed Denial of Service (DDoS) attack impacts** – Architect your application for, and prepare teams to deal with, impacts from DDoS attacks.
- **Implement inspection and protection** – For components transacting over HTTP-based protocols, a web application firewall (WAF) can help protect from common attacks inspecting and filtering your traffic.
- **Enable auditing and traceability** – Monitor, alert, and audit actions and changes to your environment in near real-time. Integrate log and metric collection with systems to automatically investigate and take action.
- **Automate security best practices** – Automated software-based security mechanisms help improve your ability to securely scale more rapidly and cost-effectively.
- **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).

We will now explore each of the key design principles individually.

Understand the AWS Security and Compliance shared responsibility model

Security and Compliance is a shared responsibility between AWS and the customer. This shared model can help relieve your operational burden, because AWS manages the security *of* the cloud. This includes operating, managing, and controlling the components from the host operating system and virtualization layer, down to the physical security of the facilities in which the service operates. As a customer, you assume responsibility for security *in* the cloud. This includes management of the guest operating system (including updates and security patches) and other associated application software, and configuration of the AWS-provided security group firewall.

For API Gateway, AWS manages the underlying infrastructure and foundation services, the operating system, and the application platform. You as a customer are responsible for the security of your configuration, including your API definition, identity and access management, and network configuration.

Protect data in-transit and at-rest

Encryption in-transit – API Gateway requires encryption in-transit for all data sent to both control plane operations, such as creating, updating, and deleting your APIs, and data plane operations such as invoking your APIs. Operations must be encrypted in transit using TLS, and require the use of HTTPS endpoints. Unencrypted API Gateway endpoints are not supported. API developers can optionally choose to require a specific TLS version for their custom domain names. You can configure mutual TLS using certificate-based authentication on a custom domain name for client invocations.

Encryption at-rest – All API definitions are deployed in memory and are cached only to encrypted disks. Customer log files are temporarily stored in encrypted form before being sent securely to CloudWatch Logs or Amazon Kinesis, which stores the logs encrypted at-rest. All integration responses selected to be cached with API Gateway are persisted on cache nodes, and are configurable to use encryption at-rest while stored. Logging is not configured or persisted by default unless explicitly configured by the customer.

Implement a strong identity and access foundation

AWS Identity and Access Management (AWS IAM) is an AWS service that helps an administrator securely control access to AWS resources. AWS IAM administrators can control who can be

authenticated (signed in) and authorized (have permissions) to use API Gateway resources. API Gateway integrates with AWS IAM for a number of purposes.

> **Note**
>
> Any policy should follow the principle of least privileges, giving the user, group, or role only the minimum set of permissions needed, and nothing more.

Amazon API Gateway IAM constructs

Identity-based policies

Identity-based policies are attached to a user, group, or role, and let you specify what that identity can do. Some examples of identity-based policies are:

- Allowing the role of "`api-developer`" the ability to create and manage a specific API.
- Allowing the user "Sam" in the group "Finance" to invoke a specific resource and method (for example, `/records/{record#}/GET`) on an API.

Resource policies

API Gateway resource policies are policy documents that you attach to an API that controls whether a specified principal (typically a user or role) can invoke the API. You can use API Gateway resource policies to allow your API to be securely invoked by:

- Users from a specified AWS account
- Specified source IP address ranges or Classless Inter-Domain Routing (CIDR) blocks

Service-linked roles

A service-linked role is a unique type of role that is linked directly to API Gateway for its exclusive use in accessing other AWS resources in your account. Service-linked roles are predefined by API Gateway, and include all the permissions that the service requires to call other AWS services on your behalf.

Tag-based permissions

In API Gateway, resources can have tags, and some actions can include tags. When you create a policy, you can use tag condition keys to control:

- Which users can perform actions on an API Gateway resource, based on tags that the resource has
- Which tags can be passed in an action's request
- Whether specific tag keys can be used in a request

Authentication and authorization

API Gateway supports multiple mechanisms to help you control and manage access to your API. A key capability you can use is the ability to authorize all API requests with API Gateway, and block any unauthorized requests directly at the API Gateway layer before any requests are sent to your backend integrations.

API Gateway provides fine-grained authorization for your APIs, as granular as authorizing decisions per caller at the combination of unique per-path, per-method level. API Gateway supports the parsing and handling of any bearer token, and supports native parsing of standardized OpenID Connect (OIDC) and OAuth 2.0 JWTs. Though API Gateway does not serve as an identity provider and does not issue tokens itself, it supports seamless integration with one or more identity providers (IdPs) of your choice. You can enable these capabilities through a choice of three different authorizers:

- **JWT and Amazon Cognito user pools authorizers** – Enable authenticating a user by validating their tokens through checking the issuer, client ID, timestamp, signature, and authorization scopes when specified. This authorizer provides seamless validation of Amazon Cognito user pools tokens, or any standards-compliant OpenID Connect (OIDC) and OAuth 2.0 tokens without the need to write custom code. This option can be set up quickly, and supports basic user validation. JWT and Amazon Cognito user pools authorizers do not require any custom code.
- **AWS Lambda authorizers** – Provide fine-grained access control by enabling authorizer validation using custom business logic that you write according to your specifications. This authorizer choice provides you with the most flexibility in enabling external lookups, and generating per-user fine-grained policies in response to the first time a user makes a request with their bearer token. Lambda authorizers also provide you with the ability to cache the resulting user's policy, so the Lambda authorizer is not invoked more often than needed.

Additionally, AWS Lambda authorizers optionally allow an API key to be sent along with the user's policy in the response and associated with the calling user's bearer token. There is an implicit mapping for metering/throttling purposes without the end user needing to know about their API key, or send it explicitly in their calls. This is the most flexible option of the three authorizer choices, but does require that you write custom code for your Lambda function, which can be accelerated through use of the approved Lambda authorizer blueprint samples.

- **IAM-based authorization** – Provides you with the ability to enable your service to authorize requests in the same manner all AWS APIs do, which is to validate a unique canonical request signature which is generated and sent by the API client with each request. Such a signature is uniquely generated, and incorporates the time of request, resource requested, and action, so that even if the signature were compromised and re-used later on, the request signature would no longer be valid at a later time. This is the most secure authorizer option, but it requires that API clients understand how to sign their requests. Using an SDK with request signing built in is advisable if you choose AWS IAM-based authorization.

Certificate-based authentication

API Gateway supports certificate-based authentication via mutual TLS (mTLS). API Gateway provides integrated mutual TLS authentication, which helps you minimize the cost or operational overhead required to manage and scale a traditional reverse proxy fleet offloading mutual TLS connections at the API Gateway. You can enable mutual TLS authentication on your custom domains to authenticate regional REST and HTTP APIs while still authorizing requests with bearer or JWTs, or signing requests with IAM-based authorization. You only need to upload a trusted certificate authority (CA) public key certificate bundle to an Amazon Simple Storage Service (Amazon S3) bucket as an object, containing public or private/self-signed CA certificates to be used for validation of issuance of client certificates. All existing authorization options are available for use in conjunction with mutual TLS, while offering additional request context on the calling user's certificate, and identity for granular authorization decisions.

Minimize attack surface area

A best practice in IT for security is to minimize the attack surface of your applications so that bad actors have minimal targets that can be probed for exploits or misconfigurations. API Gateway helps to minimize the attack surface of your applications by presenting a single point of entry for any of your services and functions. Your services and functions can reside in:

- AWS-managed environments, configured to mitigate against external access
- Your own VPCs, which you can configure to mitigate against external access
- Your data centers

Endpoint type selection

Choose the API Gateway endpoint type based on your use case. Private endpoints are recommended when your clients are within a VPC or transit VPC setting, allowing your traffic to and from the endpoint to remain within your VPC. Private endpoints are insulated from public distributed denial of service (DDoS) attacks because they are not exposed to the internet. This can allow for more granular restriction of traffic flows between systems, such as allowing invocations only from clients in a specific VPC that traverse a given VPC endpoint. Public endpoint types should be selected based on the requirements for operations and security.

API Gateway resource policies

API Gateway resource policies are policy documents that you attach to an API to control whether a specified principal (typically a user or role) can invoke the API. Resource policies are optional for API Gateway public endpoints, and are required for private endpoints. Resource policies can be used in conjunction with authorizers. Refer to *Authentication and Authorization* in this document.

Configurations for public endpoints

API Gateway public endpoints offer an optional resource policy capability which you can implement to improve your security posture, and reduce the possibility of an impact to your service via configuration. Resource policies control whether a specified principal (typically a user or role) can invoke the API. Sample use cases that you can implement via resource policies include:

- Users from a specified AWS account
- Specified source IP address ranges or CIDR blocks

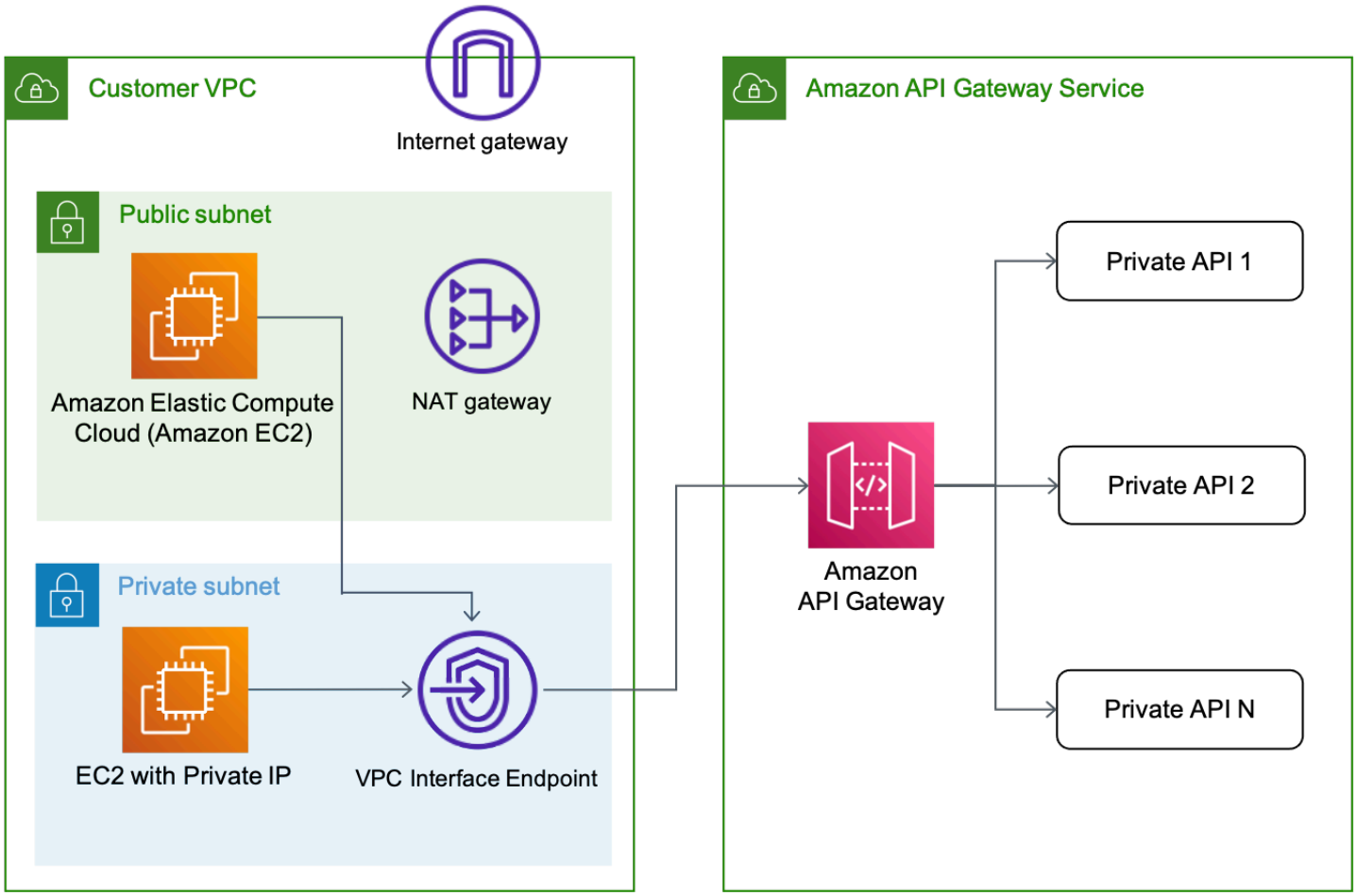
Configurations for Private Endpoints

API Gateway resource policies are also offered for API Gateway private endpoints, and are required on the API prior to deploying it. Resource policies on endpoints for private APIs enable you to control whether instances and services in VPCs and VPC endpoints can invoke your API, in addition

to all the same controls that are offered for public endpoints. Sample use cases that you can implement via resource policies include:

- Restricting calls to production API Gateway deployments to only services in production VPCs
- Restricting calls to pre-production API Gateway deployments to services with an assumed role

The following figure illustrates how you access private APIs through interface VPC endpoints for API Gateway.



How to access private APIs through interface VPC endpoints for API Gateway

API integration security

API Gateway also offers security for API integrations to back-end resources. API integrations enable you to invoke applications, functions, or services to respond to API requests. These security mechanisms allow API Gateway to securely integrate and access AWS services and other HTTP

endpoints to respond to requests to your API. The AWS IAM permission policies you assign to the back-end service determine which resources the back-end service can or cannot access.

AWS Lambda integrations

AWS Lambda integrations allow you to map a single resource/method on your API to a Lambda function. This integration works directly with the AWS Lambda service endpoints. You can use an AWS Lambda function resource policy to allow only Amazon API Gateway to invoke the specified AWS Lambda function to respond to an API request.

AWS service first-class integrations

AWS service first-class integrations allow you to directly integrate your API with AWS services, such as Amazon Kinesis Data Streams, AWS Step Functions, Amazon EventBridge, AWS AppConfig, or Amazon Simple Queue Service (Amazon SQS). This integration requires that you create an IAM execution role with a trust policy, where API Gateway is the Principal, and a permissions policy to allow the action required for the AWS service you are integrating with.

HTTP integrations with public endpoints

Use HTTP integrations to integrate your API with HTTP/S services with public endpoints. You can create API Gateway-generated client certificates to secure requests made to HTTP endpoints. These certificates can enable you to verify the requester's authenticity.

HTTP integrations with private endpoints

HTTP integrations to private resources within a VPC are performed through a VPC link. The VPC link manages integrations between API Gateway and private VPC resources through a Network Load Balancer (NLB), Application Load Balancer (ALB), or the AWS Cloud Map service. VPC links require a subnet and a security group, both of which you can use to limit what VPC link can access in your private VPC. For integration with private endpoints on-premises or in different AWS accounts or AWS Regions, you can use traditional VPC networking techniques to provide end-to-end private connectivity.

Mitigate Distributed Denial of Service (DDoS) attack impacts

Amazon API Gateway rate limiting

Rate limiting helps you prevent your API from being overwhelmed by too many requests. API Gateway throttles requests to your API using the token bucket algorithm, where a token counts for

a request and the maximum bucket size is the burst. API Gateway sets a limit on a steady-state rate and a burst of request submissions against all APIs.

There are a number of ways to implement rate limiting on your APIs. The various types of rate limits are processed in sequential order, as shown in Table 1. If any of the limits are exceeded for the rate limit, Amazon API Gateway blocks the request and returns a "429 Too Many Requests" error response to the client. Client logic or SDKs should be configured to retry such errors, although with increasing back off intervals upon repeat failures of the same type.

Table 1 – Types of rate limits

Type of rate limit	Applies to:	Set using:	Enforced by default?
Per-client, per method	API stage and specific resource/method	Usage plan with API key	No
Per-client	API stage	Usage plan with API key	No
Per-method overall	API stage and specific resource/method	API setting on the resource/method	No
Account-level throttling	All APIs in an account per AWS Region	AWS service quota	Yes

Amazon CloudFront integration

Amazon CloudFront distributes traffic across multiple edge locations, and filters requests to help ensure that only valid requests will be forwarded to your API Gateway deployments. There are two ways to use CloudFront with API Gateway:

- With an edge-optimized endpoint API Gateway instance which delivers your API via an AWS-managed CloudFront distribution which is controlled by AWS
- With a Regional endpoint API Gateway instance that you can integrate with your own self-managed CloudFront distribution

When integrating CloudFront with Regional API endpoints, CloudFront supports geo-blocking, which you can use to help prevent requests from particular geographic locations from being served.

API Gateway can be configured to accept requests only from CloudFront, using a few approaches. This can help prevent anyone from accessing your API Gateway deployment directly.

Methods include:

- Requiring an API key to be validated for requests on API Gateway, which CloudFront can insert into the `x-api-key` header before forwarding the request to the origin, in this case API Gateway.
- Requiring validation of a customized header (not `x-api-key`) with a known valid value for requests on API Gateway. CloudFront inserts the header and value on the request. A Lambda request authorizer can validate the presence of the expected header and return a "403 unauthorized" error if it is not present.
- Authenticating the user with AWS Lambda@Edge, then signing all requests with AWS request signing before sending the request to API Gateway. API Gateway uses AWS IAM-based authorization to validate the signature.

AWS Shield and AWS Shield Advanced

AWS Shield Standard defends against most common, frequently occurring network and transport layer DDoS attacks that target your website or applications. All customers benefit from AWS Shield Standard.

AWS Shield Advanced can be added to protect Amazon CloudFront distributions and Amazon Route 53 hosted zones, providing additional protections against DDoS attacks. During a DDoS attack, your instances can mitigate the attack up to the throughput of the instance. AWS Shield Advanced provides expanded DDoS attack protection for web applications running on the resources. AWS Shield Advanced manages mitigation of Layers 3, 4, and 7 attacks. Additionally, with the appropriate AWS support level, AWS Shield Advanced provides access for customers to the AWS DDoS Response Team.

Implement inspection and protection

Inspecting and filtering your traffic at your API layer allows you to validate the requests and identify and stop invalid requests before they reach your backend services. The result of these

actions can help improve your data and application security by not allowing requests that do not meet data standards, or that include items such as SQL injection attacks. Inspection and protection can also improve performance and availability of backend services, because bad requests are discarded in advance of reaching the backend service. Inspection and protection may also assist with cost controls.

Request validation

API Gateway includes features for validating and transforming API requests before sending the request to backend integrations. You can validate the format of a request against your defined API models to ensure the expected data is included in the request before sending it to the backend service. If the request properties do not match the API model's schema, API Gateway will respond with a "400 Bad Request" message, and will not invoke the backend service.

Request transformation

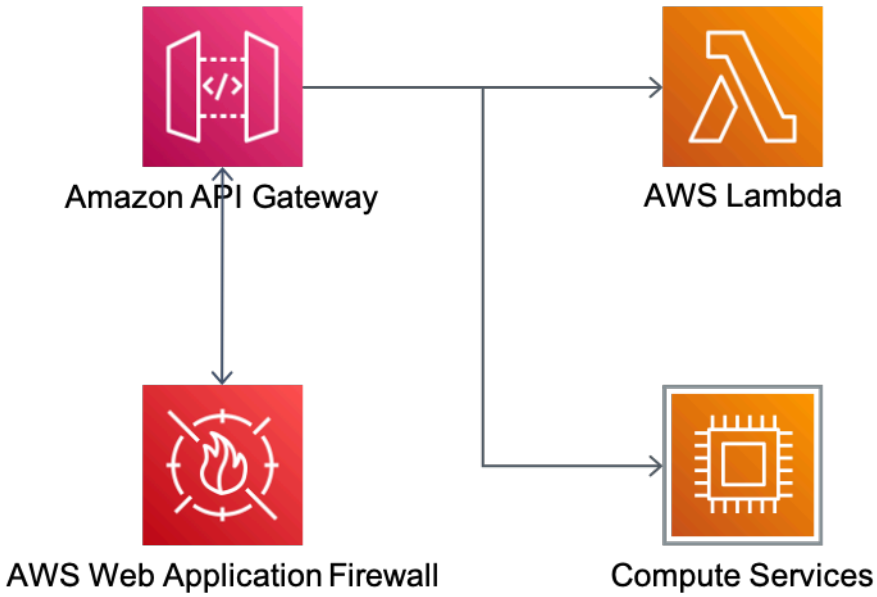
You can transform requests by enriching, filtering, and re-structuring request data prior to invoking downstream integrations. API Gateway can enrich a request with data returned from Lambda authorizers. This provides more data to backend services regarding the requestor, so the backend service can take appropriate actions such as allowing or denying the transaction. Additionally, for sensitive requests, data such as authorization or security details, headers, and other request data can be filtered from downstream integrations after successful authorization of the client with API Gateway.

Cross-Origin Resource Sharing (CORS) configuration

CORS headers can be configured on API Gateway to direct API clients to only invoke API calls from allowed origins. CORS enables clients from one domain or origin to invoke API methods from another domain, and prevents cross-origin requests to domains which do not explicitly allow the originating domain or origin.

AWS WAF integration

AWS WAF is a managed web applications firewall (WAF) that can be used in conjunction with API Gateway regional endpoints, with or without customer-managed CloudFront distributions, as seen in the following figure.



Amazon WAF in conjunction with API Gateway Regional endpoints

AWS WAF provides flexible options for implementing protections via AWS managed rules, partner provided rules, and custom rules that you can write yourself. Many of these rules are focused on protections against the Open Web Application Security Project (OWASP) Top 10 application vulnerabilities. AWS WAF has a number of rules that enable you to combine many types of rules in your Web access control list (ACL), to provide effective security for your API Gateway instances, including the following:

- Block or Allow based on IP-address or country of origin for the request
- Block or Allow based on request components, such as query string, body, and HTTP method

Enable auditing and traceability

You can monitor and audit API Gateway using many AWS capabilities and services.

Amazon CloudWatch

Amazon CloudWatch is the foundational monitoring and observability service for AWS. CloudWatch collects monitoring and operational data in the form of logs, metrics, and events, providing you with a unified view of AWS resources, applications, and services. You can use CloudWatch to detect anomalous behavior in your environments, set alarms, visualize logs and metrics side by side, take automated actions, troubleshoot issues, and discover insights to help keep your APIs running smoothly.

Amazon CloudWatch Metrics

API Gateway automatically monitors APIs on your behalf. CloudWatch reports a number of default metrics, such as the number of requests, 4xx errors, 5xx errors, latency, and integration latency. If caching is enabled, cache hit and miss counts are reported. It is possible for you to filter API Gateway metrics. For REST APIs, these filters are based on API Name and Stage. For HTTP APIs, it's API ID and Stage. If detailed CloudWatch metrics are enabled, it's possible for you to filter these metrics by method and resource.

For a full list of metrics exposed by API Gateway, refer to AWS Amazon API Gateway dimensions and metrics.

Amazon CloudWatch Alarms

You can choose a CloudWatch metric and monitor when a threshold is crossed. Alarms can be metric alarms or composite alarms. A metric alarm watches a single CloudWatch metric, or the result of a math expression based on CloudWatch metrics. A composite alarm watches a rule expression that takes into account the alarm states of other alarms you've created.

A metric alarm for API Gateway might monitor the average of 5xx errors over a given period. A composite alarm might be triggered when both latency and 5xx errors exceed a threshold for a given period.

Amazon CloudWatch Logs

There are two types of logging available in CloudWatch for API Gateway: execution and access logs. CloudWatch Logs are disabled by default. You must grant API Gateway permission to write logs to CloudWatch for your account.

In execution logging, API Gateway manages the format of the CloudWatch logs. API Gateway creates CloudWatch log groups and log streams, recording any caller's requests and responses. Execution logs can include errors, the full request and response payloads (up to 1 MB), data used by Lambda authorizers, whether API keys are required, whether usage plans are enabled, and more.

Access logs capture who has accessed your API, and how the caller accessed the API. You can create your own log group, or choose an existing log group that can be managed by API Gateway. Logs can be formatted using Common Log Format (CLF), JSON, XML, or CSV. It's also possible to configure access logging to direct events to Amazon Data Firehose, bypassing CloudWatch Logs.

AWS X-Ray

Using AWS X-Ray, you can analyze and debug distributed API Gateway-based applications. This helps you understand the performance of your application and its underlying services, so you can eventually identify and troubleshoot the root cause of performance issues and errors. The X-Ray end-to-end view of requests shows a map of the application's underlying components as the requests travel through your application, so you can analyze the application performance during development and in production.

AWS CloudTrail

Using AWS CloudTrail, you can implement governance, compliance operational auditing, and risk auditing of your AWS account activity, including API Gateway. CloudTrail enables you to log, continuously monitor, and retain account activity, providing a complete event history of API Gateway management actions taken. Management actions include creating, deploying, or updating API operations invoked through the AWS Management Console, AWS SDKs, and command line tools. CloudTrail users can search for the event source of "apigateway.amazonaws.com," giving them the ability to audit actions taken against their API Gateway instances. Using CloudTrail, you can optionally encrypt the log files using AWS Key Management Service (AWS KMS) and also use the CloudTrail log file integrity validation for positive assertion.

AWS Config

With AWS Config, you can track specific configuration changes to your API Gateway resources, and send notifications based on resource changes. These include:

- Changes to API configurations:
  - endpoint configuration
  - version
  - protocol
  - tags

- Changes to deployments and stages:
  - cache cluster settings
  - throttle settings
  - access log settings

- active deployment set on the stage

For a full list of API Gateway configurations tracked by AWS Config, refer to Monitoring API Gateway API configuration with AWS Config. You can use AWS Config rules to represent ideal configuration settings, and will detect when any changes violate the desired settings. There are a number of AWS Config managed rules for API Gateway, but you have the ability to create custom rules as well.

Automate security best practices

Automated software-based security mechanisms improve your ability to securely scale more rapidly and cost-effectively. The following services can enable such automation for API Gateway.

AWS WAF security automations

AWS WAF provides you with a set of AWS managed rules to provide protection against common application vulnerabilities or other unwanted traffic, without having to write your own rules. In addition, you can automate updates to AWS WAF rules using Lambda, which can analyze web logs and identify malicious requests, and automatically update security rules. For example, AWS WAF rules can be updated to automatically block IP addresses that are sending requests over a specified threshold.

AWS Config rules

AWS Config provides you with a set of AWS Config managed rules to evaluate whether your AWS resources comply with common best practices. You can write your own custom rules to identify whether a resource is compliant or not. You can manually or automatically remediate non-compliant resources. For example, it's possible to enforce that APIs defined in API Gateway must be private. Any attempt to change to a regional or edge API can trigger a function to update the API back to a private endpoint.

AWS CloudTrail and Amazon EventBridge

AWS CloudTrail can detect when management API calls are made against API Gateway, and send notifications to Amazon EventBridge. EventBridge is a serverless event bus service that enables you to connect your applications with data from a variety of sources. EventBridge has the ability to direct messages to a number of available targets based on matched rules. For example, you can create a rule to trigger a Lambda function when an API is created, updated, or deleted.

Amazon CloudWatch Alarms

Amazon CloudWatch Alarms have the capability to send notifications to an Amazon Simple Notification Service (Amazon SNS) topic. When a target threshold for error rate is breached, subscribers can receive email notification, a Lambda function could be triggered, or a message can be published to a HTTP/S target.

Regulatory compliance

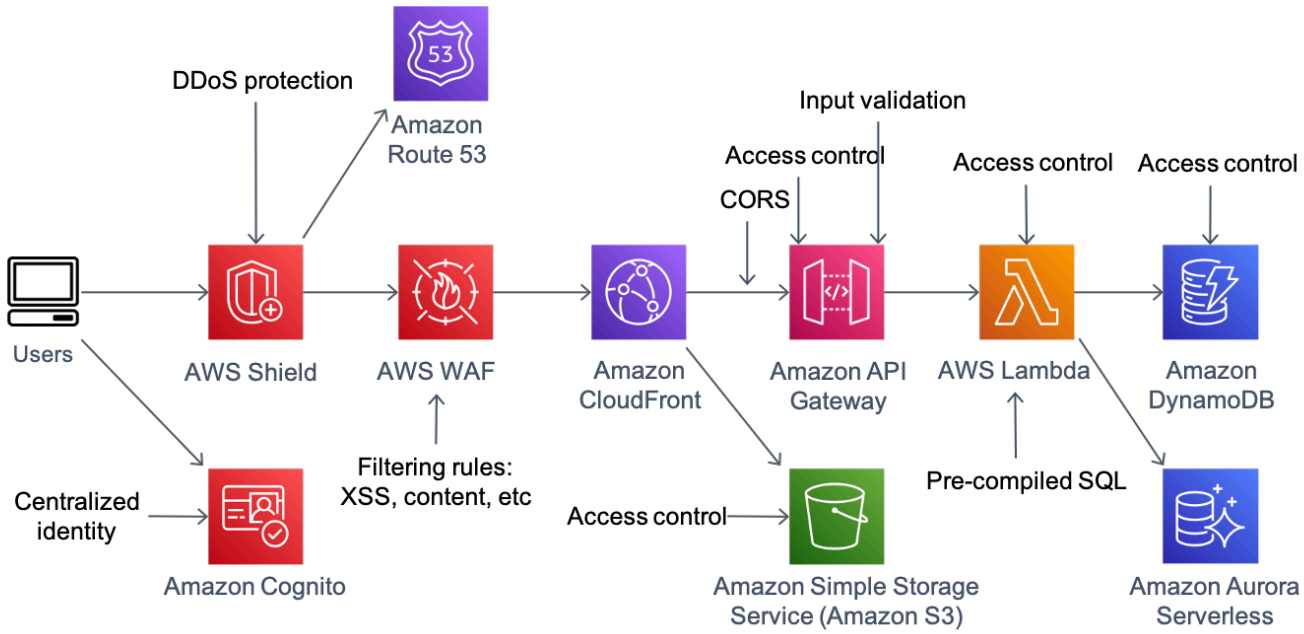
You are responsible for determining the compliance needs of your application. After these have been determined, you can use the various API Gateway features to match those controls. You can contact AWS experts such as Solutions Architects, Technical Account Managers, and other domain experts for assistance. However, AWS cannot advise you on which compliance regimes are applicable to a particular use case.

As of July 2020, API Gateway is compliant with standards including but not limited to SOC 1, SOC 2, SOC 3, PCI DSS, and U.S. Health Insurance Portability and Accountability Act (HIPAA). For a full list of compliance programs, refer to AWS Services in Scope by Compliance Program.

Because of the sensitive nature of some compliance reports, they cannot be shared publicly. For access to these reports, you can sign in to your AWS console and use AWS Artifact, a no-cost, self-service portal, for on-demand access to AWS compliance reports.

Apply security at all layers

It is important to apply security at all layers to enable a defense in-depth strategy. For a Serverless application, holistic security can include the following:



Holistic security layers for a serverless application

- Application identity is managed with a secure identity provider such as Amazon Cognito, enabling secure sign-up, sign-in, and federation.
- DDoS protection is implemented with AWS Shield and AWS WAF to mitigate both network and application layer attacks. AWS WAF is configured to block cross-site scripting, SQL injection, bad bots and user agents, and more.
- Amazon Route 53 DNS is protected with AWS Shield and anycast striping and shuffle sharding to ensure increased availability. Refer to Reduce DDoS Risks Using Amazon Route 53 and AWS Shield.
- Amazon CloudFront enables further DDoS mitigation by splitting any DDoS traffic across 100+ edge locations, and accelerating and caching content. It accelerates delivery of both static content such as HTML, CSS, and JavaScript (JS) via S3, and dynamic content served via API Gateway.
- API Gateway implements CORS protection, restricts requests to only valid clients and sources, and authorizes all requests based on your configured authorizers. It validates requests against defined resource policies, and inputs against defined API models, to block any requests which don't conform to expected schema before invoking your respective integrations.
- Once requests are authorized and your backend Lambda integration is invoked, the Lambda execution environment runs only with a least-privileged IAM permissions. The role grants the request access exclusively to the Amazon DynamoDB table needed, with the minimum

permission set possible. For relational databases, Lambda can authenticate with Amazon Aurora using AWS IAM, and not use static credentials while pre-compiling SQL statements to prevent any SQL injection attacks.

Conclusion

Security is an ongoing effort. Protecting data in-transit and at-rest, implementing a strong identity foundation, minimizing attack surface area, mitigating DDoS attack impacts, implementing inspection and protection techniques, and automating security best practices all enable an in-depth defense strategy that every organization should implement. This effort is easier thanks to the AWS features and services discussed in this paper. Amazon API Gateway strives to help you build and operate architectures that protect information, systems, and assets, while delivering business value and pay-for-value pricing.

Contributors

Contributors to this document include:

- Brian McNamara, Senior Serverless Solutions Architect, Amazon Web Services
- Justin Pirtle, Principal Serverless Solutions Architect, Amazon Web Services
- Tim Bruce, Senior Serverless Solutions Architect, Amazon Web Services

Further reading

For additional information, refer to:

- Amazon API Gateway Pricing
- AWS PrivateLink Pricing
- Amazon API Gateway Resource Policies
- AWS Well-Architected
- AWS Well-Architected Framework
- *AWS Serverless Application Lens* (whitepaper)
- *AWS Best Practices for DDoS Resiliency* (whitepaper)
- Protecting your API using Amazon API Gateway and AWS WAF (blog)
- *Guidelines for Implementing AWS WAF* (whitepaper)
- OWASP Top Ten Web Application Security Risks web application security risks
- AWS Artifact

Document revisions

Date	Description
November 2020	First publication
November 18, 2022	Minor update – Changed links to outdated whitepapers and included correct links for PrivateLink pricing

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.

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.