AWS Whitepaper

Guidelines for Implementing AWS WAF

Guidelines for Implementing AWS WAF: AWS Whitepaper

Table of Contents

Guidelines for Implementing AWS WAF

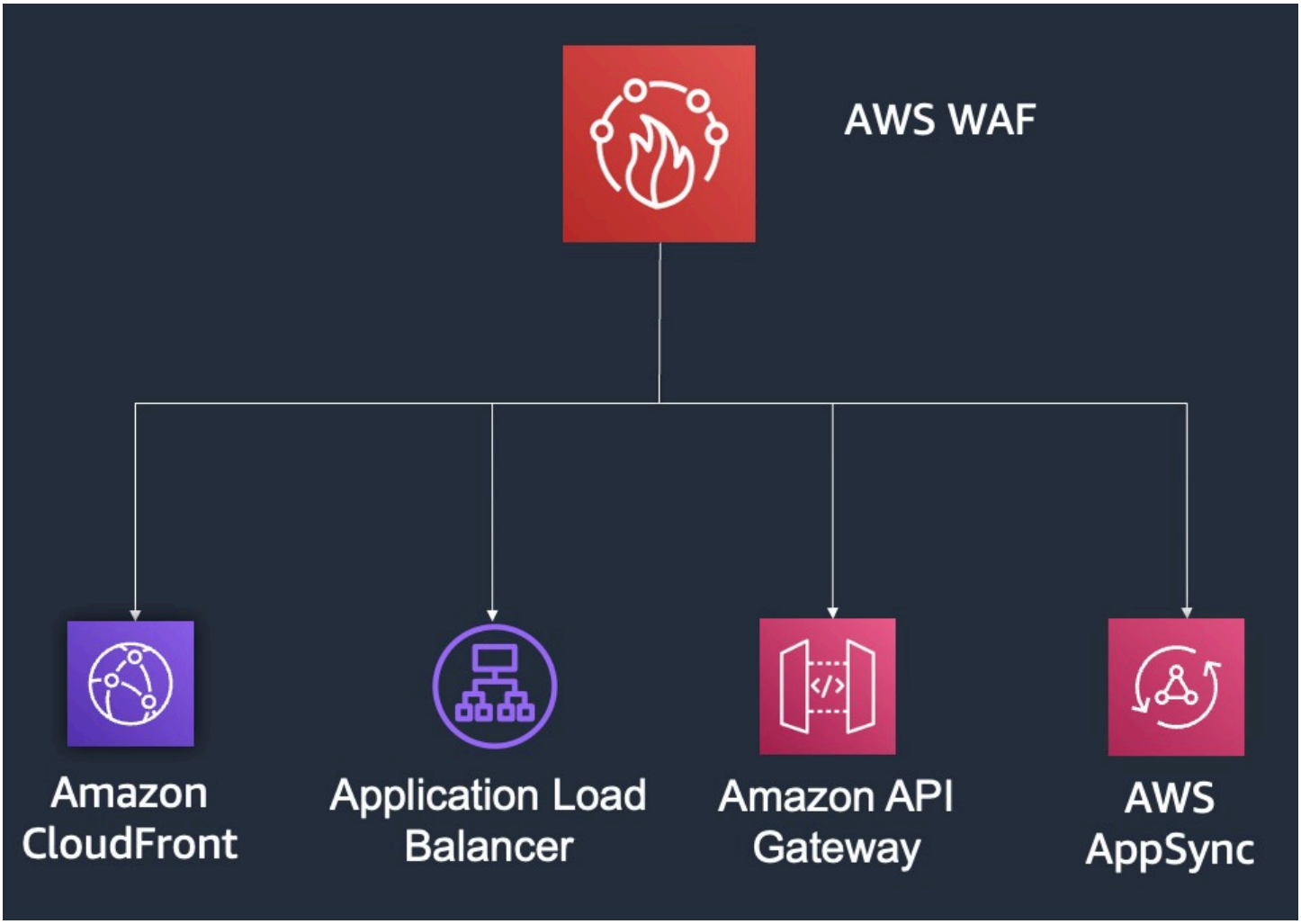
Publication date: **January 19, 2022** (*Document history*)

AWS WAF is a web application firewall (WAF) that helps you protect your websites and web applications against various attack vectors at the application layer (OSI Layer 7). This whitepaper outlines recommendations for implementing AWS WAF to protect existing and new web applications. This whitepaper applies to anyone who is tasked with protecting web applications.

Overview

Security is a shared responsibility between AWS and the customer, with responsibility boundaries that vary depending on factors such as the AWS services used. For example, when you build your web application with AWS services such as Amazon CloudFront, Amazon API Gateway, Application Load Balancer, or AWS AppSync you are responsible of protecting your web application at Layer 7 of the OSI Model. AWS WAF is a tool that helps you protect web applications by filtering and monitoring HTTP(S) traffic, including traffic from the public internet. Web application firewalls (WAFs) protect applications at the application layer from common web exploits that can affect application availability, compromise security, and consume excessive resources. For example, you can use AWS WAF to protect against attacks such as cross-site request forgery, cross-site scripting (XSS), file inclusion, and SQL injection, among other threats in the OWASP Top 10. This layer of security can be used together with a suite of tools to create a holistic defense-in-depth architecture.

AWS WAF is a managed web application firewall that can be used in conjunction with a wide variety of networking and security services such as Amazon Virtual Private Cloud (Amazon VPC), and AWS Shield Advanced.



AWS WAF integrations

AWS WAF can be natively enabled on CloudFront, Amazon API Gateway, Application Load Balancer, or AWS AppSync and is deployed alongside these services. AWS services terminate the TCP/TLS connection, process incoming HTTP requests, and then pass the request to AWS WAF for inspection and filtering. Unlike traditional appliance-based WAFs, there is no need to deploy and manage infrastructure, or plan for capacity. AWS WAF provides flexible options for implementing protections through managed rules, partner-provided rules, and custom rules that you can write yourself.

It's important to understand that with AWS WAF, you are controlling ingress traffic to your application. To control egress traffic, refer to Security best practices for your VPC.

This whitepaper covers recommendations for protecting existing and new applications with AWS WAF, and outlines the following steps and options to consider when deploying AWS WAF:

- Understanding threats and mitigations
- Requirements for AWS WAF
- Implementing AWS WAF
- Deploying AWS WAF to production
- Cost considerations

Note

AWS WAF provides two versions of the service: WAFv2 and WAFClassic. AWS recommends using AWS WAFv2 to stay up to date with the latest features. AWS WAF Classic no longer receives new features. AWSWAFv2 includes features that are not available in WAF classic, including a separate API and Console. This paper focuses on implementation with AWSWAFv2.

Understanding threats and mitigations

Before deciding how to deploy AWS WAF, you need to understand what type of threats your web applications may be facing and the protection options available with AWS WAF. Web applications face different kinds of threats that AWS WAF can help you mitigate.

- **Distributed denial of service (DDoS) attacks** – Try to exhaust your application resources so that they are not available to your customers. At Layer 7, DDoS attacks are typically well-formed HTTP requests that attempt to exhaust your application servers and resources.
- **Web application attacks** – Try to exploit a weakness in your application code or its underlying software to steal web content, gain control over web servers, or alter databases; these can involve HTTP requests with deliberately malformed arguments.
- **Bots** – Generate a large portion of the internet's website traffic. Some *good* bots associated with search engines, crawl websites for indexing. However, *bad* bots may scan applications, looking for vulnerabilities and to scrape content, poison backend systems, or disrupt analytics.

AWS WAF helps you to improve your security posture against these types of threats (refer to *AWS WAF integrations*).

Types of threats at Layer 7

Topics

- DDoS attacks at Layer 7

- Web application attacks
- Bad bots
- Custom request and response

DDoS attacks at Layer 7

For HTTP floods, you can use AWS WAF rate limiting rules to block clients from specific IP addresses that are sending abusive number of requests to your application. AWS WAF also provides the ability to block known malicious IP addresses using the Amazon IP reputation list from the AWS Managed Rules or by subscribing to AWS Partner IP reputation lists from the AWS Marketplace. For more advanced mitigations, you can activate **Scanners and probes protections** and **Reputation list protection** using the AWS WAF Security Automations solution.

- **Scanners and probes protections** – Parse application access logs searching for suspicious behavior, such as an abnormal number of errors generated by an origin to block bad actors.
- **Reputation list protection** – Block requests from IP addresses on third-party reputation lists such as DROP and EDROP from Spamhaus, the Tor exit node list, and the Proofpoint Emerging Threats IP list.

In addition to using AWS WAF, AWS recommends reviewing AWS Shield Advanced which detects application layer attacks such as HTTP floods or DNS query floods by baselining traffic on your application and identifying anomalies. With the assistance of the Shield Response Team (SRT), AWS Shield Advanced includes intelligent DDoS attack detection and mitigation for network layer (Layer 3) and transport layer (Layer 4) attacks, but also for application layer (Layer 7) attacks. For further reading, you can refer to the AWS Best Practices for DDoS Resiliency whitepaper when architecting for DDoS resiliency.

Web application attacks

AWS WAF provides the following options for protecting against web application exploits.

AWS WAF rule statements

Rule statements are the part of a rule that tells AWS WAF how to inspect a web request. When AWS WAF finds the inspection criteria in a web request, we say that the web request matches the

statement. Every rule statement specifies what to look for and how, according to the statement type.

Every rule in AWS WAF has a single top-level rule statement, which can contain other statements. Rule statements can be very simple. For example, you could have a statement that checks each web request against a set of originating countries. Rule statements can also be very complex. For example, you could have a statement that combines many other statements with logical *AND*, *OR*, and *NOT* statements.

AWS Managed Rules

AWS Managed Rules for AWS WAF is a set of AWS WAF rules curated and maintained by the AWS Threat Research Team that provides protection against common application vulnerabilities or other unwanted traffic, without having to write your own rules. You can select and add some of the AWS managed rule groups to protect your application from various threats. Managed rule groups include:

- **Baseline rule groups** – Cover some of the common threats and security risks described in the OWASP Top 10 publication.
- **Use-case specific rule groups** – Provide incremental protection based on your application characteristics, such as the application OS or database.
- **IP reputation rule groups** – An IP reputation list derived from the Amazon threat intelligence team blocks known malicious IP addresses.

AWS WAF allows you to select a specific version of a managed rule group within your web access control list (ACL), giving you the ability to test new rule updates safely and roll back to previously tested versions. When using a versioned managed rule group, you control when new rule updates are applied to your traffic. By default, you will continue to automatically receive rule updates to your managed rule group.

You can change this behavior by manually selecting a version, allowing you to pause automatic updates or go back to a previous version. After you select a specific version, you will no longer receive automatic updates but will remain on the selected version until it reaches end of life. You should monitor the end of life of each version you use, by monitoring the Amazon CloudWatch metrics, to ensure you are notified ahead of time when you should start to consider moving to a newer version.

AWS WAF now provides early notifications of upcoming rule updates to your managed rule groups through Amazon Simple Notification Service. By subscribing to the SNS topic in the AWS WAF console, you can be notified when the managed rule group provider stages updates.

Custom rules

In addition to AWS Managed Rules, you can also write custom rules specific to your application to block undesired patterns in parts of the HTTP request, such as headers, method, query string, Uniform Resource Identifier (URI), body, and IP address. You can also inspect up to 10 IP addresses in `X-Forwarded-For` (XFF), `True-Client-IP`, or other custom headers in the incoming request and write custom rules to block undesired values. You can use these rules together with the AWS Managed Rules groups to provide customized protections. You can construct custom rules using the rule builder in the AWS Management Console. Or, you can write custom rules in JSON and configure the rules using the AWS Command Line Interface (AWS CLI) or using automation tools such as AWS CloudFormation. For example, you can use custom rules to block requests that do not respect your expected API URL scheme. For the full list of logical statements that you can express using custom rules, refer to the Rule statements list.

AWS Marketplace rules

On the AWS Marketplace, you can find rules created by security partners that have built their own rule sets on AWS WAF. These rules are available based on subscription and can be used together with AWS Managed Rules and your own custom rules.

Bad bots

To protect against bot traffic, you can use AWS WAF Bot Control. Bot Control allows you to monitor, block, or rate-limit bot traffic activity in real time and gain additional insights such as bot categories, identities, and other bot traffic details. When AWS WAF evaluates a web request against the Bot Control managed rule group, the evaluation adds labels to requests that it detects as bot related. This label information can then be used to create any custom rules. By blocking the bot traffic at the edge, your application costs and performance are unaffected. Bot Control can be added as a managed rule to any new or existing WAF web ACL.

IP reputation rule groups allow you to block requests based on their source. Blocking these IP addresses can help mitigate bots and reduce the risk of a malicious actor discovering a vulnerable application. You can also use reputation rules for bot protection. AWS WAF has two managed reputation lists: Amazon IP reputation list and Anonymous IP list. The Amazon IP reputation list

rule group contains rules that are based on Amazon internal threat intelligence. The Anonymous IP list rule group contains rules to block requests from services that allow the obfuscation of viewer identity, and these include requests from VPNs, proxies, Tor nodes, and hosting providers (including AWS). This rule group is useful if you want to filter out viewers that might be trying to hide their identity from your application. Blocking the IP addresses of these services can help mitigate bots and evasion of geographic restrictions.

You can configure AWS WAF rules to require WAF CAPTCHA challenges to be solved for specific resources that are frequently targeted by bots such as login, search, and form submissions. You can also require WAF CAPTCHA challenges for suspicious requests based on the rate, attributes, or labels generated from AWS Managed Rules, such as AWS WAF Bot Control or the Amazon IP reputation list.

You can use the AWS WAF Security Automations solution to defend against bots by implementing honeypots and behavioral detection with WAF logs. For more sophisticated detections of the most difficult bots involved in application-level attacks (such as bots attempting credential-stuffing), AWS recommends adding a bot management solution to your architecture. You can find third-party solutions on the AWS Marketplace that provide advanced bot mitigation capabilities. Some of these solutions also provide the ability to integrate with CloudFront using Lambda@Edge for inline protection.

You can add a scope-down statement inside some rules. The scope-down statement narrows the scope of the requests that the rule evaluates. If a rule has a scope-down statement, traffic is first evaluated using the scope-down statement. If it matches the scope-down statement criteria, then it's evaluated using the rule's standard criteria. Traffic that doesn't match the scope-down statement is not evaluated further by AWS WAF. You can define a scope-down statement inside the following statement types:

- **Managed rule group statement** – If you add a scope-down statement to a managed rule group statement, any request that doesn't match the scope-down statement results as not matching the rule group. Only requests that match the scope-down statement are evaluated against the rule group. For managed rule groups with pricing that's based on the number of requests evaluated, scope- down statements can help contain costs. For more information about managed rule group statements, refer to the Managed rule group statement.
- **Rate-based rule statement** – A rate-based rule without a scope-down statement controls the rate of all requests that come in to your applications. If you want to only control the rate for a specific category of requests, you add a scope-down statement to the rate-based rule. For example, to only track and control the rate of requests from a specific geographical area, you

specify that geographical areas in a geographic match rule as the scope-down statement. For more information about rate-based rule statements, refer to the Rate-based rule statement.

AWS WAF uses web ACL capacity units to calculate and control the operating resources that are required to run your rules, rule groups, and web ACLs. AWS WAF calculates capacity differently for each rule type, to reflect each rule's relative cost. The maximum capacity for a web ACL is 5000, which is sufficient for most use cases. If you need more capacity, contact the AWS Support Center.

Custom request and response

AWS WAF provides the ability to customize requests and responses. The custom request feature is applicable to allowed and counted requests, while the custom response feature is for blocked requests.

For incoming requests, AWS WAF allows you to add a custom header (`x-amzn-waf-*`) prior to processing. This allows you to route this request differently. You can use this functionality to add additional verification steps like CAPTCHA, or use this additional metadata to respond to this request in a different way, for example, routing to a different backend.

With the custom response feature, you can modify the response code and display a custom error page. You can use this feature to provide more descriptive error statements. For example, instead of displaying an *Access Denied* error due to throttling, you can use AWS WAF to send a more descriptive error page such as *Please slow down and try again later*. You can use the custom response code feature to respond with HTTP 2xx, 3xx, 4xx, and 5xx instead of HTTP 403 response codes. The custom responses can also be used to differentiate blocked requests generated by AWS WAF or your server.

Requirements

As a first step towards implementing AWS WAF, AWS recommends that you gather and define the requirements which will make this implementation successful for your business. This section will cover some common WAF requirements.

Protections

After you have identified which threats are applicable for your application, define your baseline criteria for success. These criteria can include passing penetration tests performed by third-party or internal security teams, meeting specific compliance requirements, or simply having coverage for common web vulnerabilities (for example, OWASP Top 10). The sensitivity of the content that your application serves may dictate whether you choose to implement a positive security compared to negative security model (allow compared to deny APIs) when creating your WAF web ACL. If your application does not use a SQL database, you can save WAF capacity units by not adding SQL injection detection rules. AWS recommends that you add WAF rules that are specific to your application's requirements, because adding unnecessary rules can lead to an increase in false positives. False positives are legitimate requests that are considered by WAF as attacks and may be blocked as a consequence.

For existing applications, you may already have visibility into application usage patterns and be looking to block malicious requests identified from previous incidents and observations. Therefore, you may be looking for protections against a specific attack. If you are already using a WAF implementation, you may have a baseline of the average number of requests blocked by the existing WAF rules. In some cases, you may have visibility into the existing rules implemented and you can implement similar rules in AWS WAF.

Comparing AWS managed rules and Custom rules

Depending on your organization's resources and security culture, you must decide how to implement AWS WAF. You can deploy out-of-the-box AWS Managed Rules sets, create your own custom rules, or use a combination of both. For most applications, AWS recommends starting with the baseline rule groups and the Amazon IP reputation list from the AWS Managed Rules, then selecting application specific rule groups that match the application's profile.

For some workloads, advanced protection may be required. In such cases, you might add additional custom rules in addition to existing protection. Managing and implementing your own rules

requires that your security and application teams develop skills in creating and managing WAF rules. To help with these workloads, AWS Professional Services or AWS WAF Partners can help you create these rules, perform periodic reviews, and train your teams to develop this expertise.

Governance

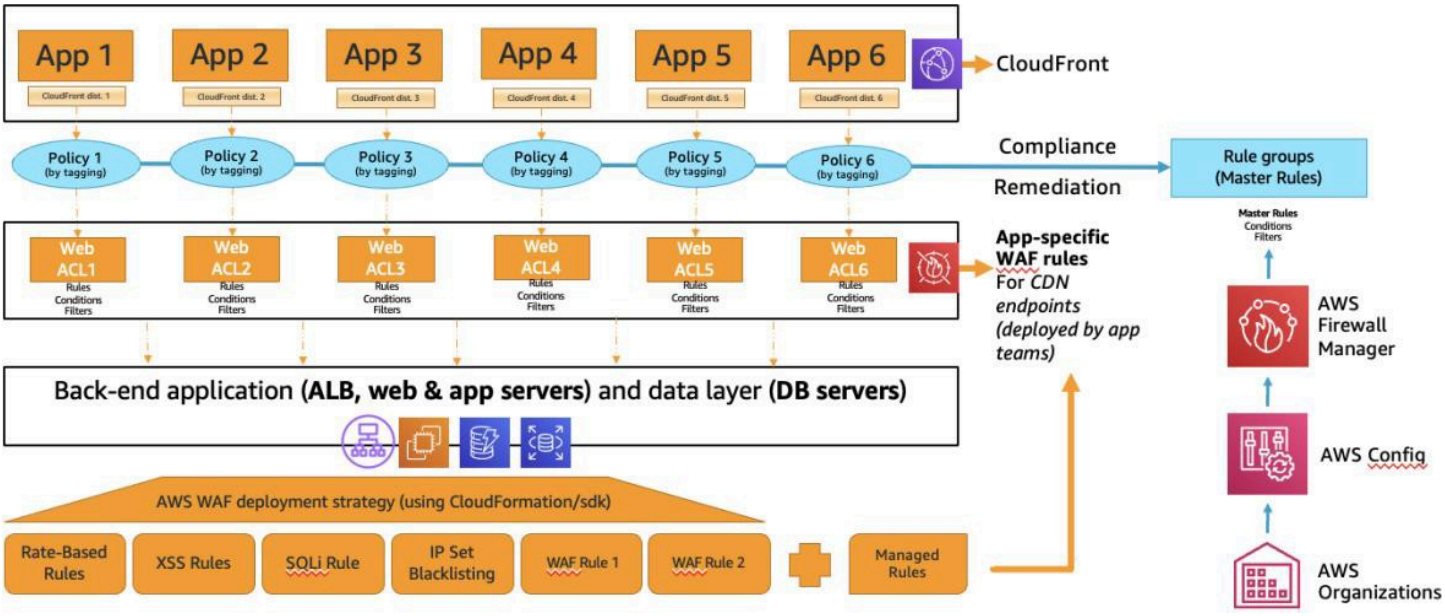
You might also have governance requirements to define how to manage and monitor WAF implementations across your organization. In some organizations, WAF configurations are managed centrally by a security team. In this case, the security team must audit and ensure that WAF is configured correctly across resources managed by application teams. In other organizations, WAF configuration and deployment is managed by the application teams so that the WAF rules deployed can be specific to the protected application.

To simplify centralized management of AWS WAF, AWS Firewall Manager allows you to define security policies that automatically deploy WAF across accounts within your AWS Organization. AWS Firewall Manager provides you with visibility to ensure that resources have the appropriate WAF web ACL associated and are within compliance of the WAF policies. To illustrate the possibilities, review the following governance examples:

Example 1: AWS Firewall Manager implementation

In this example, you have autonomous application teams that own WAF configurations with the supervision of a central security team.

1. The central security team provides and documents generic guidance in the form of best practices for the application teams.
2. The central security team uses Firewall Manager with a WAF policy to deploy a central web ACL (based on AWS managed baseline rule groups) to each team's account without automatic remediation. This policy is configured to deploy a copy of the web ACL but not automatically associate it to application resources (for example, CloudFront, Application Load Balancer, Amazon API Gateway). Although this approach does not force the protection on the application teams, it provides the central security team with visibility of which applications have WAF attached to their endpoints.
3. Application teams can choose to apply the central web ACL as it is, or modify it before application. Their choice is mostly driven by their security requirements and governance.



Example of an AWS Firewall Manager implementation

Example 2: AWS Firewall Manager implementation with two WAF policies

In this example, you have a central security team that manages WAF deployments and rules for applications across your organization.

1. The central security team creates two Firewall Manager WAF policies with automatic remediation.
   a. One policy uses managed rules for WordPress (as an example of a sample application) for all resources tagged as a WordPress application.
   b. One policy uses Amazon IP reputation list and rate limiting rules for all other HTTP(S) applications.
2. Application teams tag resources associated with WordPress applications accordingly.
3. In each AWS account within the organization, Firewall Manager creates two web ACLs, one for each policy. Firewall Manager automatically associates the web ACLs to the appropriate resources as configured by the policy. When this occurs, existing WAF web ACLs associated to those resources are overridden.
4. The security team can monitor WAF compliance through the Firewall Manager in the AWS Management Console. Firewall Manager allows you to identify if resources have the correct WAF web ACL associated as configured by the Firewall Manager policy. You can also integrate AWS

Security Hub with AWS Firewall Manager to detect resources that are not properly protected by WAF rules.

Example 3: AWS Firewall Manager implementation with one centralized WAF policy

In this example, you have a central security team that manages baseline WAF deployments and application teams have the flexibility to create new WAF rules specific to their applications.

1. The central security team creates one baseline AWS Firewall Manager WAF policy with automatic remediation.
2. Central policy uses a baseline rule for example, a managed IP reputation rule to deny access to IP addresses with a bad reputation score.
3. Application teams can then configure application specific policies for WordPress (as an example of a sample application) for all resources tagged as a WordPress application.
4. In each account within the organization, Firewall Manager creates one web ACL. Firewall Manager automatically associates the web ACLs to the appropriate resources as configured by the policy. When this occurs, existing WAF web ACLs associated to those resources are overridden.
5. The security team can monitor WAF compliance through the Firewall Manager in the AWS Management Console. Firewall Manager allows you to identify if resources have the correct WAF web ACL associated as configured by the Firewall Manager policy. You can also integrate AWS Security Hub with AWS Firewall Manager to detect resources that are not properly protected by WAF rules.

Logging

WAF logging is a common requirement for security teams to meet their compliance and auditing needs. AWS WAF provides near-real-time logs through Amazon CloudWatch Logs log group, an Amazon Simple Storage Service (Amazon S3) bucket, or an Amazon Data Firehose.

For each inspected request by AWS WAF, a corresponding log entry is written that contains request information such as timestamp, header details, and the action for the rule that matched. AWS WAF does not currently log the request body.

In the logging configuration for your web ACL, you can customize what AWS WAF sends to the logs as follows:

- **Log filtering** – You can add filtering to specify which web requests are kept in the logs and which are dropped. You can filter on the rule action and on the web request labels that were applied during the request evaluation. For information about rule action settings, refer to AWS WAF rule action. For information about labels, refer to AWS WAF labels on web requests.
- **Field redaction** – You can redact some fields from the log records. Redacted fields appear as *XXX* in the logs. For example, if you redact the **URI** field, the **URI** field in the logs will be *XXX*. For a list of the log fields, refer to Log Fields.

The Log filtering feature allows you to reduce processing of unwanted logs and only store the logs that you want to analyze. This helps save costs on log delivery and storage. With the field redaction feature, you can remove sensitive fields such as PII data or the session ID from the log files.

You can use logs for debugging and additional forensics by integrating with your Security Information and Event Management (SIEM) or other log analysis tools. By default, logging is not enabled when you create a web ACL. To automate log enabling, you can use AWS Config to configure logging whenever a new WAF web ACL is created.

Automation of logging activation

Implementation

Select a starting point

After you have identified your requirements, you must decide which application to start with. If you are new to AWS WAF, AWS recommends starting with a non-critical application when possible. This approach allows you to familiarize yourself with the new platform, and tune your configuration, while limiting the risks of misconfiguration to your business. AWS also recommends starting with an application that you have a good understanding of the application traffic patterns. This approach allows you to quickly identify the impact of deploying WAF on your application traffic so that you can tune accordingly.

AWS WAF integration design

Depending on your application's requirements, you must decide where to deploy AWS WAF. As mentioned previously, you can configure AWS WAF on Amazon CloudFront, Amazon API Gateway, Application Load Balancer, and AWS AppSync. For all public facing web applications, AWS recommends deploying AWS WAF with CloudFront for the best security posture, unless you have constraints that require otherwise.

CloudFront can be used for both dynamic and static content. By default, CloudFront blocks non-HTTP(S) traffic and malformed HTTP requests, and provides inline DDoS protection for attacks at network Layers 3 and 4 with sub-second time-to-mitigation. CloudFront employs advanced DDoS protections, such as stateless SYN Flood mitigation and automated traffic engineering systems that can disperse or isolate the impact of large volumetric attacks on the CloudFront Global Edge Network, most effectively when deployed in conjunction with Amazon Route 53. If your application is hosted outside of AWS, CloudFront provides a seamless way to use the AWS global network to stop threats before they reach your data centers.

For applications with an additional level of requirements, you might choose to implement a layered WAF model by using AWS WAF in conjunction with another WAF offering at the origin providing or load-balancing your service.

For example, you might want to inspect responses returned by the origin. In this case, you can use AWS WAF to inspect incoming requests to CloudFront at the edge, and use an appliance-based WAF to inspect incoming requests and outgoing responses from your origin. Some appliance-based

WAFs have the ability, when they detect attack traffic, to synthesis rules for the AWS WAF and push them into your AWS WAF rule set, if configured appropriately and given an IAM role with appropriate allowed actions.

Another example, is protecting multiple applications on a single domain served by CloudFront. You can use AWS WAF on CloudFront for common IP and IP geolocation- based blocking at the edge, and deploy additional AWS WAF capability on each of your Application Load Balancers for application-specific rules.

Validation in staging environment

After you choose an application, AWS recommends setting up a staging environment. This approach allows you to experiment with AWS WAF without negatively impacting production traffic. There are two approaches to staging (according to practices in your organization):

- Replicate your entire application stack to a staging environment including AWS WAF.
- Create a new endpoint for your production environment. Your staging environment is based on this new endpoint with AWS WAF deployed. For example, you can create a new CloudFront distribution with WAF web ACL attached and set the origin to your existing Application Load Balancer.

> **Note**
>
> If you are already using CloudFront, you can still create a new CloudFront distribution but you can't reuse the same domain attached to the existing distribution.

With your WAF staging environment set up, AWS recommends that you restrict access to the entirety of this environment to the authorized developer team. There are multiple options for achieving this:

- Use the AWS WAF to block requests that do not come from your organization's public IP address range. However, this approach doesn't offer authentication and can be difficult to manage if developers work remotely, unless they are required to VPN into the corporate network environment.
- Implement an authorization mechanism in your application, and forward the authorization header in CloudFront's cache behavior configuration.

- If you do not want to make a change to your application, you can offload authorization to your endpoint. For example, if you are using CloudFront, you can add a Lambda@Edge function that provides access control, or use CloudFront native signed cookies for the same purpose.

Deployment in staging

When deploying a web ACL, AWS recommends starting with the following setup:

1. **Add rules based on your defined requirement.**

   If you are new to AWS WAF, or do not have specific requirements, you can start with coverage for common web vulnerabilities offered by AWS Managed Rules. It is important to think about the order of rules in your web ACL, because AWS WAF processes rules in order of priority, and stops the web ACL evaluation when there is a match to implement the action of the matching rule.

   > **Note**
   >
   > Deploying rules in block mode allows you to see how rules impact test traffic in your staging environment. However, consider implementing your production deployment procedures in staging so that your operators understand how AWS WAF behaves before moving to production. In many cases, it is common to start new rules in count mode before switching to block mode when deploying to production. This way, you avoid compromising the availability of your application because of a misconfigured AWS WAF rule that could block legitimate traffic.

2. **Enable rate-based rules to protect yourself against DDoS types of attack (for example, HTTP flood).**

   The rate-based rule keeps track of the number of requests seen per IP address based on a sliding time window of five minutes. The sliding window is updated every 30 seconds, and after the rate limit is reached, the rule immediately takes action pertinent to the IP address. It does not wait until the five minutes has passed before taking the action. The rate-based rule keeps blocking requests from the offending IP address until the address lowers the rate of requests being sent from it.

> **Note**
>
> You are able to set the limit as low as 100 requests per five minutes; however, AWS recommends that you start with 2,000 requests and gradually reduce this number as needed.

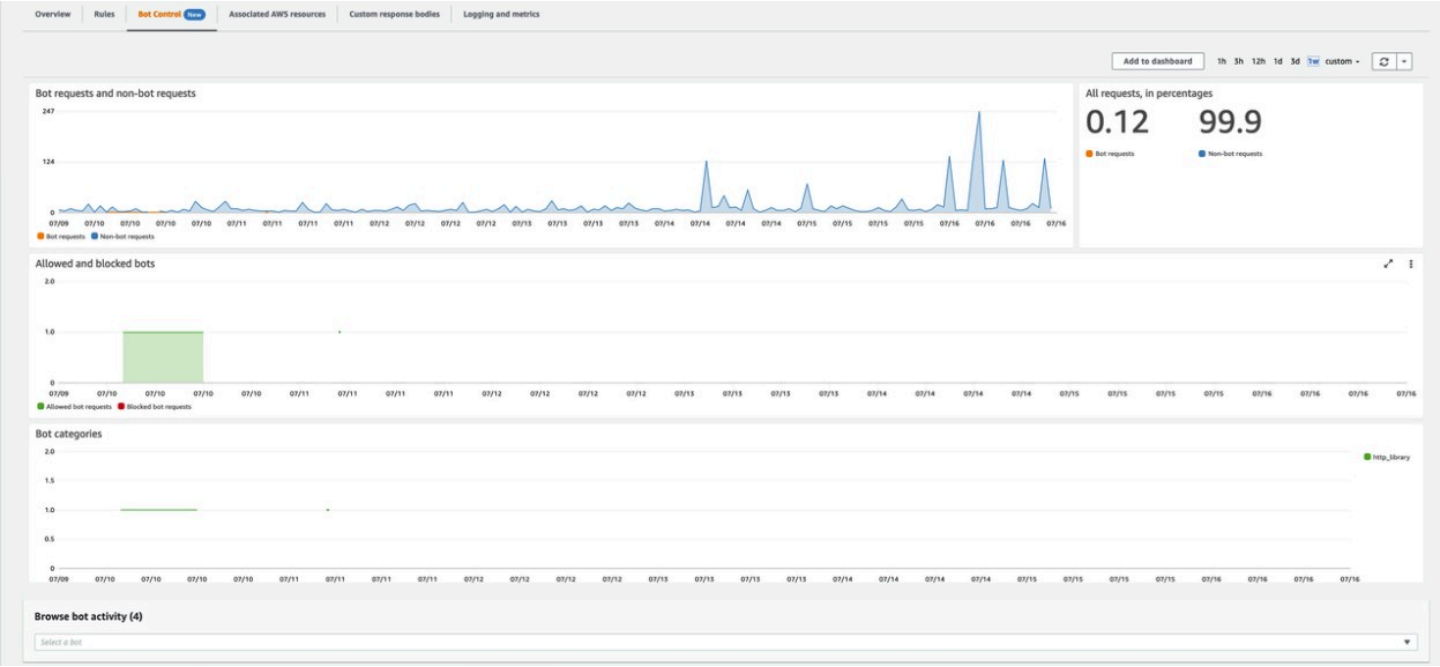
After you are satisfied with the rules you created within the staging environment, you can duplicate them to your production environment or another account by simply copying the rules from the web ACL. In the web ACL overview page, there is an option to download the entire web ACL configuration, including all rules, in a JSON file. After downloading the configuration, you can either manually copy the rules and recreate the web ACL or convert the JSON to YAML and use it in a CloudFront template to deploy the web ACL

Monitoring and visibility

It is important for operating your WAF implementation to have good visibility of what is being blocked by your web ACL. This visibility is useful for threat intelligence, hardening rules, troubleshooting false positives, and responding to an incident. There are multiple monitoring options available with AWS WAF.

Monitoring bot traffic with AWS Bot Control dashboard

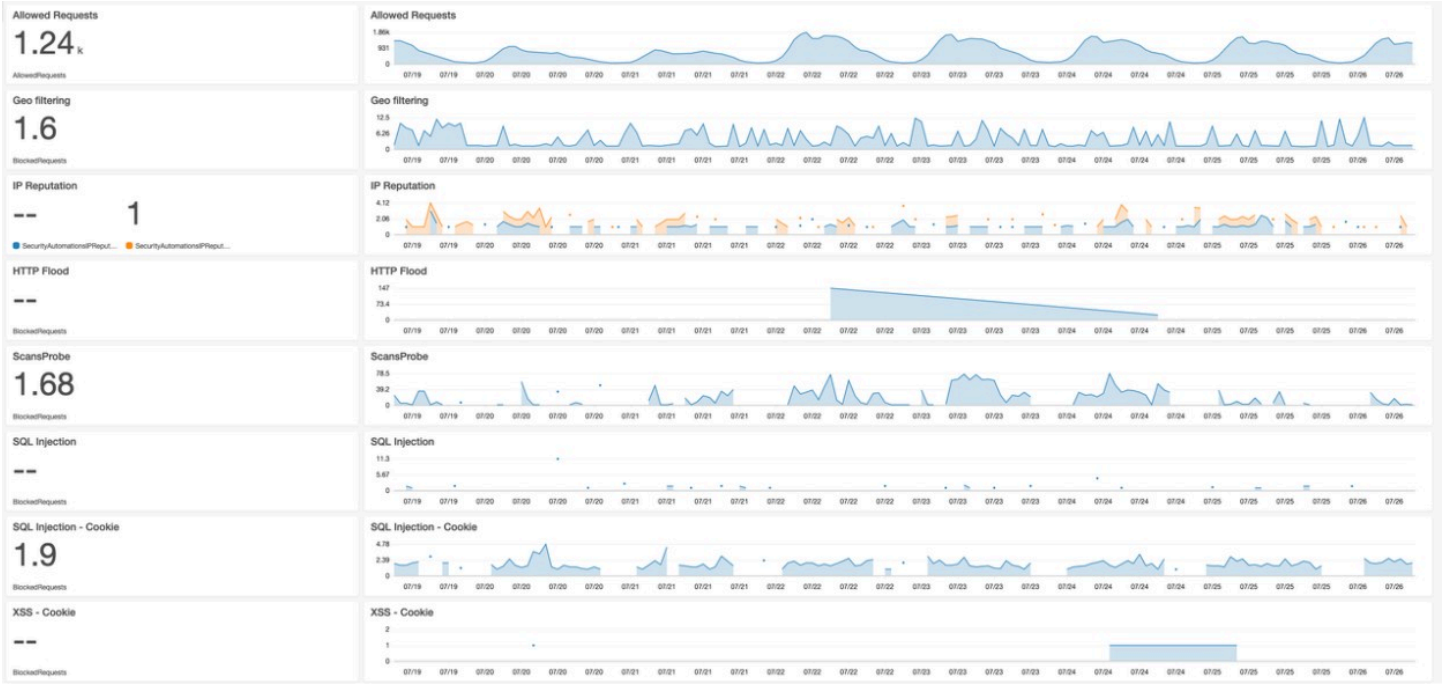
AWS WAF Bot Control provides a dashboard to view all bot-related details in a single view. This includes information about the number of bot and non-bot requests, bot categorization, and whether the bot traffic was allowed or blocked. This dashboard also provides the ability to query bot activity.



Bot Control dashboard

Monitoring using Amazon CloudWatch

You can set up a dashboard for AWS WAF to display information about the activity of rules in your web ACL. For each rule, CloudWatch emits near-real-time metrics such as `AllowedRequests`, `BlockedRequests`, and `PassedRequests`, which are recorded for a period of two weeks. The following image is an example of what you can easily set up with CloudWatch to display real-time and historical information about how your web ACL is protecting your application. You can set up alarms on CloudWatch metrics to receive notifications when a certain WAF rule is abnormally triggered based on predefined thresholds.



Security dashboard using CloudWatch

However, CloudWatch doesn't provide you with information about the processed requests themselves. If you need to get more details more about inspected requests, you have two options:

- **View a sample of the WAF log in the WAF console –** For each sampled request, you can view detailed data about the request, such as the originating IP address and the headers included in the request. With this approach, you can quickly debug false positives in a staging environment. The sampled request works by randomly fetching 5,000 requests that the web ACL has processed over the time period that you chose (up to the previous three hours).
- **Enable and process AWS WAF logs for full and detailed information** – This approach is more suitable, for deeper troubleshooting in a production environment. For each request, AWS WAF logs provide raw HTTP(S) headers along with information on which rules were triggered. AWS WAF logs provide the exact patterns that initiated SQLi and XSS rules in the `terminatingRuleMatchDetails` field. AWS WAF logs are ingested using Amazon Data Firehose and can be delivered in JSON format to multiple destinations, including Amazon Simple Storage Service (Amazon S3). AWS recommends this approach for all production workloads for the best visibility and troubleshooting.

It's common to build custom dashboards based on AWS WAF logs, to provide a near- real-time global view of your application security, and deep dive into request details when needed. With

AWS WAF logs, you can build your own dashboard using AWS or third-party services. If you are already using third-party monitoring services such as Splunk, Datadog, or Sumo Logic, you can export WAF logs to these services. For example, Sumo Logic has a template to create a dashboard for examining WAF logs.

Testing and tuning

After your initial implementation of WAF, there is normally a phase of tuning to mitigate potential false negatives and false positives. False negatives are attacks that were not caught by your WAF and require you to harden your rules. False positives represent legitimate requests that were considered by WAF wrongly as attacks and blocked as a consequence.

False negatives

To identify false negatives based on your security requirements, you can use penetration testing providers, automated vulnerabilities scanners from third parties, or open source web application security scanners. Although automated vulnerability scanners are great tools to test your architecture quickly against known vulnerabilities and predefined attack vectors, they only cover known cases and may falsely flag an issue when there isn't one. Therefore, keep in mind that testing your WAF rules using a vulnerability scanner is not a guarantee that your application is fully protected. However, you can use a vulnerability scanner for sanity-checking. Make sure to regularly update your scanner and set a regular cadence for running it so that you are aware of new attacks and can update the WAF rules whenever necessary.

False positives

False positives are generally identified by quality assurance teams by testing the application after changes are introduced to the application's code or WAF configuration. In many cases, this testing requires a deep understanding of the application to tell the difference between suspicious request signatures and a malicious attack. In some cases, false positives may only appear in production owing to a shortcoming in quality assurance test coverage. To help identify these cases, consider the following:

- Set up alarming on selected WAF rules in CloudWatch to be notified when a rule is triggered when predefined thresholds are exceeded.
- Update your application experience to allow real users to report unexpected unauthorized access to your application. For example, when WAF is deployed with CloudFront, you can use custom

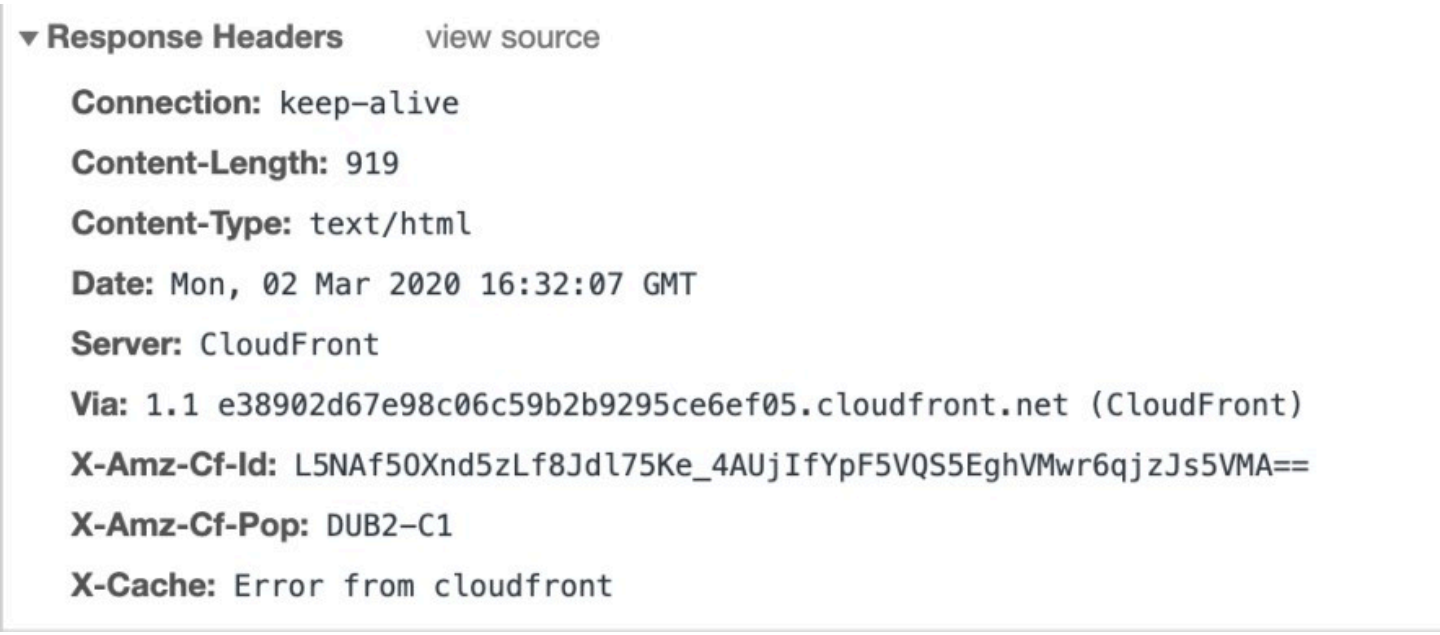
error pages to catch 403 error codes and serve a friendly response. This page can prompt the user to provide information on the issue they've encountered.

- Enable WAF logging. Have security and application teams review and baseline blocked traffic on a regular cadence to identify threat patterns and anomalies in blocked traffic.

When you detect a false positive, you must understand what has initiated it. A quick way to identify the trigger is by using the AWS Management Console to check sample logs that corresponds to the rule being initiated. A more robust way to identify the false positive is to check the logs generated by AWS WAF and filter on the blocked request by URL, IP, and timestamp. If you are using AWS WAF with CloudFront when a request is blocked, CloudFront returns a `Request ID` within the response headers and body.

You can use this `Request ID` to find the corresponding record in WAF logs.

Error page displayed by CloudFront



Error page response headers

With the log record, you have the information about which rule was initiated—for example, a specific header doesn't respect your size constraint rule. In some cases, the information is not sufficient to identify the issue. For example, a SQLi rule triggers on a cookie in the request, but it's not obvious which part of the cookie is causing this trigger. For SQLi and XSS rules, the `terminatingRuleMatchDetails` field in the log record provides more details on the match.

Resolving false positives

After you understand the issue, you can resolve it. The best approach is to change the application code that is generating requests that look similar to attacks, but that may take some time and effort. For a quick fix, you can create an exception to the rule in WAF, but note that creating exceptions in WAF exposes your application to potential attacks.

Example 1: Override rules using AWS CLI

Suppose that you have a legitimate URL pattern *xxxx* that is blocked by an XSS rule on the path of the URI. You can override this block by adding an additional rule to include an exception on the *xxxx* URL pattern:

```
BLOCK [XSS condition] AND NOT[String Match Condition on Path]
```

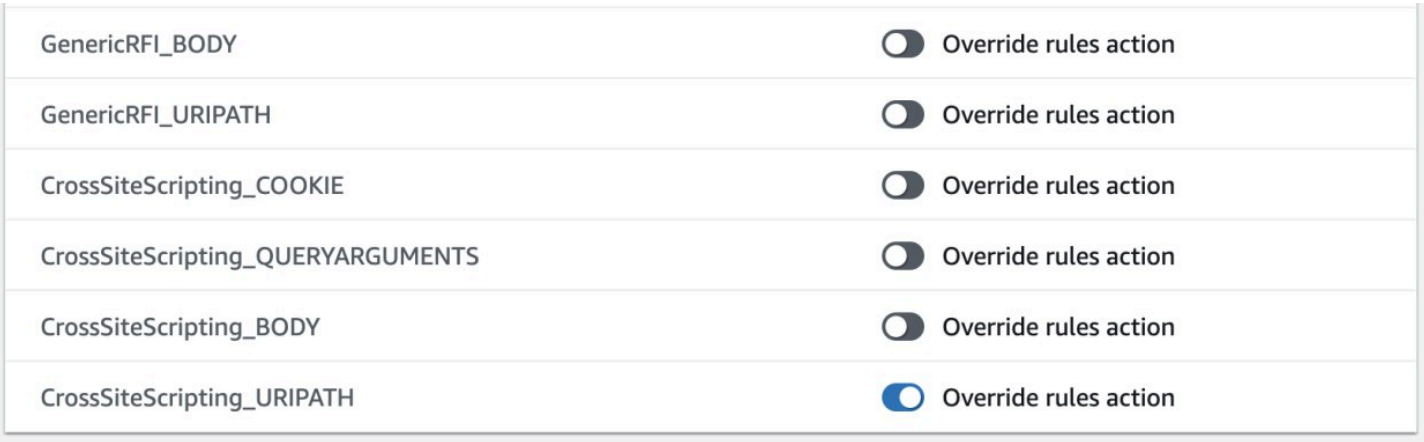
The following WAF rule statement illustrates this example. This solution works for SQLi and XSS rules, and for any custom rules.

```
{
    "Name": "XSSprotection",
    "Priority": 0,
    "Action":  {
        "Block":  {}
    },
    "VisibilityConfig":  {
        "SampledRequestsEnabled": true,
        "CloudWatchMetricsEnabled": true,
        "MetricName": "XSSprotection"
    },
    "Statement":  {
        "AndStatement":  {
            "Statements":  [
                {
                    "XssMatchStatement":  {
                        "FieldToMatch":  {
                            "UriPath":  {}
                        },
                        "TextTransformations":  [
                            {
                                "Type": "URL_DECODE",
                                "Priority": 0
                            }
                        ]
                    }
                },
                {
                    "NotStatement":  {
                        "Statement":  {
                            "ByteMatchStatement":  {
                                "FieldToMatch":  {
                                    "UriPath":  {}
                                },
                                "PositionalConstraint": "CONTAINS",
                                "SearchString": "xxxx",
                                "TextTransformations":  [
                                    {
                                        "Type": "LOWERCASE",
                                        "Priority": 0
```

```
                                    }
                                ]
                            }
                        }
                    }
                }
            ]
        }
    }
}
```

Example 2: Override rules using AWS Managed Rules

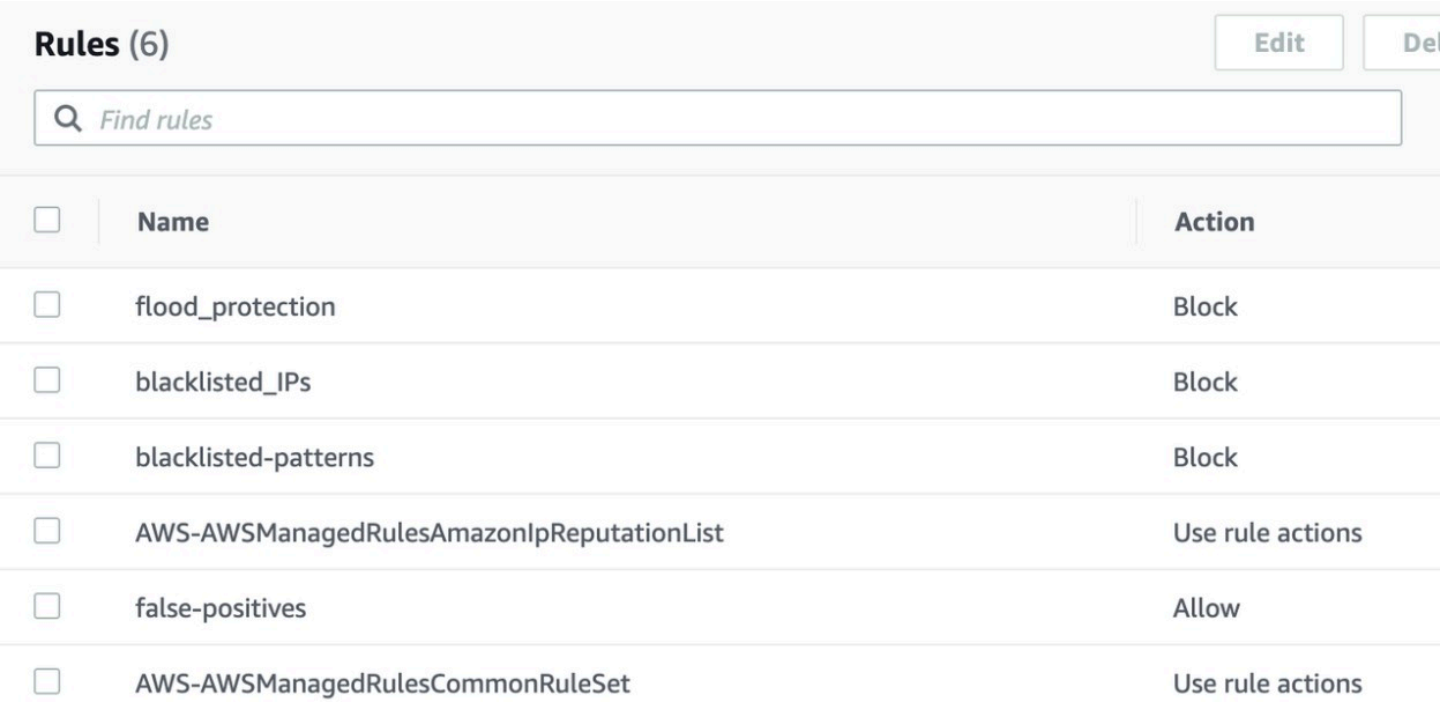
Suppose you have a legitimate URL pattern *xxxx* that is blocked by a managed rule such as a Core rule set by AWS. You can override the action of the blocking sub rule by setting it to count mode to mitigate false positives temporarily. However, you open your environment to attacks previously stopped by this sub rule.



Overriding the action of a sub rule in AWS Managed Rules

Another approach is to create an allowed list rule before the managed rule, which allows traffic when it matches the false positive *xxxx* pattern and stops evaluating the rest of rules. With this approach, you risk exposing your application to the attacks normally stopped by the rules coming after your false positive allowed list rule when it matches.

One way to reduce this risk is by setting the allowed list rule to the lowest priority in the web ACL rules order.



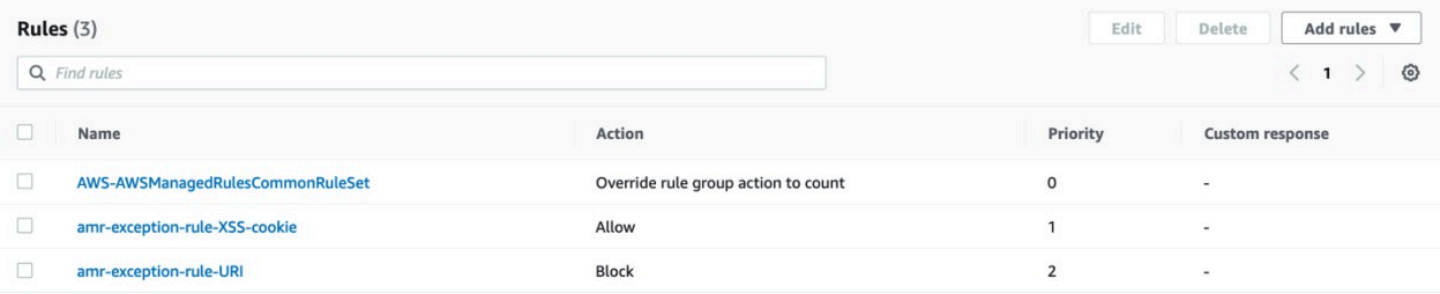
Adding a rule to handle exceptions

After you deal with false positives, either by adding exceptions to WAF rules or by changing your application code, you can validate your remediation by replaying the requests that caused the false positive by using tools such as cURL or Postman. If the false positive is mitigated properly, the request should not be blocked.

Example 3: Using labels for creating exceptions for known false positives

Suppose you have a legitimate traffic (for example, XSS matching in cookie header) that is blocked by a managed rule such as a Core rule set by AWS. You can use labels with AWS Managed Rules groups to create exception rules. When a web request matches a rule, AWS WAF adds the rule's labels to the request. The labels remain available on the request as long as AWS WAF is evaluating it against the web ACL. This means that you can use Amazon Managed Core rule set in count mode to create labels and create custom rules to run after the rule group for known false positives.

For example, you can create Amazon Managed Core set rule group in count mode and then create custom rules for allowing known XSS patterns in the cookie header and blocking known XSS patterns in URI path as shown in the following figure.



AWS WAF rules for creating false-positive exceptions

The JSON representation of the `amr-exception-rule-XSS-cookie` is shown in the following figure.

```
{
    "Name": "amr-exception-rule-XSS-cookie",
    "Priority": 1,
    "Statement":  {
        "LabelMatchStatement":  {
            "Scope": "LABEL",
            "Key": "awswaf:managed:aws:core-ruleset:CrossSiteScripting_Cookie"
        }
    },
    "Action":  {
        "Allow":  {
            "CustomRequestHandling":  {
                "InsertHeaders":  [
                    {
                        "Name": "XSS",
                        "Value": "false-positive"
                    }
                ]
            }
        }
    },
    "VisibilityConfig":  {
        "SampledRequestsEnabled": true,
        "CloudWatchMetricsEnabled": true,
        "MetricName": "amr-exception-rule"
    }
}
```

The JSON representation of the `amr-exception-rule-URI-cookie` is shown in the following figure.

```
{
    "Name": "amr-exception-rule-URI",
    "Priority": 2,
    "Statement":  {
        "LabelMatchStatement":  {
            "Scope": "LABEL",
            "Key": "awswaf:managed:aws:core-ruleset:CrossSiteScripting_URIPath"
        }
    },
    "Action":  {
        "Block":  {}
    },
    "VisibilityConfig":  {
        "SampledRequestsEnabled": true,
        "CloudWatchMetricsEnabled": true,
        "MetricName": "amr-exception-rule-URI"
    }
}
```

Deployment to production

Operational readiness

After you have tested and validated your WAF implementation in your staging environment, decide when you should deploy it to production. Select a date and time when you expect to have lowest user traffic. Before deployment, make sure your application and security teams review operational readiness, discuss how to rollback changes, and review dashboards to ensure all metrics and alarms required are configured. You can create runbooks so that team members understand how to implement rollback and other mitigating operations. In the event of a security threat, your teams should know how to update your configuration, deploy in a different account, troubleshoot problems, and respond. These runbooks outline steps to take when an event occurs. For example, your runbook may include the following instructions:

- How to deploy the setup
- How to mitigate false positives
- How to troubleshoot issues
- Details on which metrics to review for understanding applications health, such as: CloudFront response types (HTTP 200, 4xx, 5xx), application and database servers CPU/memory, and WAF metrics
- Common queries to analyze and query log data to filter request data for deep inspection
- Steps for engaging security teams who configure and deploy AWS WAF
- Steps for engaging AWS

If you are subscribed to AWS Shield Advanced and are under a DDoS attack, you can engage the AWS Shield Response Team (SRT). The SRT helps you analyze the suspicious activity and assists you in mitigating the issue. This mitigation often involves creating or updating AWS WAF rules and web ACLs in your account. The SRT can inspect your AWS WAF configuration and create or update AWS WAF rules and web ACLs for you. AWS recommends that as part of setting up AWS Shield Advanced, you proactively provide the SRT with the needed authorization to complete these tasks. Providing authorization ahead of time helps prevent mitigation delays in the event of an actual attack.

Also, ahead of deployment, be sure to create a common messaging channel with all stakeholders, including network engineers, security engineers, application teams, and any account teams supporting you. Make sure you communicate updates through this channel.

Deployment

After you are ready, enable AWS WAF for your production endpoints, and decide if you want to first trial the rules in count mode to catch any potential false positives that were not found during staging. If you are deploying a WAF rule for the first time, this technique can help avoid blocking legitimate traffic. However, placing rules in count mode makes your application vulnerable to attacks mitigated by the rule until you place the rule in block mode. Trialing rules in count mode may be undesirable if you have a high degree of confidence that the rule will not generate false positives. After you review metrics and dashboards and you are comfortable with how your rules match, you can change the rules from count mode to block mode.

If you are already using another WAF offering, for example a content delivery network– based WAF, AWS recommends that you shift traffic progressively from the existing offering to AWS WAF. For example, you can use the Amazon Route 53 weighted routing policy to shift traffic progressively to a newly created CloudFront endpoint with WAF activated on it.

AWS does not recommend that you stack AWS WAF with other WAF offerings for evaluation because this can result in conflicts in how rules are matched. For example, one of the capabilities of a WAF is the ability to block IP addresses. Many WAF solutions rely on the IP address from the TCP connection to determine the client IP address. When you layer WAF solutions, the upstream WAF solution loses visibility into the client IP address which prevents IP rules from working effectively. In some cases, you may accidentally block legitimate traffic from one WAF to the other.

If you choose to use AWS Firewall Manager to deploy AWS WAF, AWS recommends starting with a policy that deploys to a single endpoint first. After you test and validate your WAF policy with one endpoint, you can update your policy to match all endpoints you want to protect. You can tag similar endpoints (CloudFront distributions, Application Load Balancer, and API Gateway) with a common tag and create a Firewall Manager policy to apply the rule group to all the endpoints with that tag. It is also important to tag production endpoints with the correct tag when you create a new production endpoint, so the correct rule group is attached to the endpoint upon deployment.

Post deployment

After deploying AWS WAF to production, it is critical to regularly review and monitor your application. Application and security teams should review dashboards to develop a baseline understanding of application traffic patterns. With AWS WAF logs, you can use Amazon OpenSearch Service, Amazon Athena, or third-party SIEM tools to analyze application traffic patterns in detail and identify past trends in application traffic and changes in behavior. With this information you can dive deeper into anomalies to identify new threats or false alarms, and iterate on your WAF rules to defend accordingly. Your runbooks should be reviewed, practiced, and updated on a regular basis. AWS recommends exercising runbooks on a regular basis to ensure that your teams are comfortable responding to security events. For example, simulate security incidents so that operators can practice these procedures.

Consider scheduling regular penetration tests to ensure that you stay on top of the latest threats to address vulnerabilities. It is important to keep WAF rules up to date to defend against newly identified threats. To reduce this effort, you can use managed rules instead of custom rules. Managed rules are updated by the WAF vendor (AWS or partner) based on the evolving threat landscape. However, it is also important that you update your application-specific custom WAF rules to match changes made to your applications.

Cost considerations

AWS WAF offers standalone pricing that is charged based on your usage of web ACLs, rules, and the number of requests that are inspected. For logging configurations, you will be charged based on your usage of Amazon Data Firehose. AWS Bot Control rulesets are paid AWS Managed Rules that can be added to your web ACL and you are charged a monthly charge (prorated hourly) for Bot Control rule group and an additional charge for the number of web requests processed by Bot Control. If you choose to use Managed rules for AWS Web Application Firewall you can subscribe to managed rules and pay only for what you use. There are no contracts or subscription commitments.

For workloads with high volumes of requests, consider evaluating AWS Shield Advanced to reduce the per request charges. When AWS WAF is used with resources protected by AWS Shield Advanced, there is no additional charges for using AWS WAF and AWS Firewall Manager. You simply pay for the charges associated with AWS Shield Advanced. This approach can help optimize cost for request-heavy workloads. For more details on pricing, refer to AWS Shield, AWS Firewall Manager, and AWS WAF pricing pages.

Scope-down statements are an efficient way to save costs. Using scope-down statements, you can limit what rules get analyzed. For example, you can assign the Bot Control managed rule for login page only, whereas all other pages of your website will not be analyzed for bot traffic. This will help reduce the cost of running AWS Bot Control with your AWS WAF.

Conclusion

This whitepaper described the various threats that AWS WAF can address, the different requirements that should be considered when implementing a WAF, and how to deploy AWS WAF to production environments with minimal disruptions. Anyone tasked with protecting web applications can use this whitepaper to better understand how to implement AWS WAF as a part of their protection tools.

Contributors

Contributors to this document include:

- Achraf Souk, Principal Solutions Architect, Edge Services
- Tino Tran, Principal Solutions Architect, Edge Services
- Damindra Bandara, Senior Security Consultant, Professional Services
- Anuj Butail, Senior Solutions Architect
- Kaustubh Phatak, Senior Solutions Architect

Further reading

For additional information, refer to:

- AWS Architecture Center
- AWS WAF Documentation
- AWS Security Incident Response Guide
- AWS WAF Security Automations
- AWS Best Practices for DDoS Resiliency

Document history

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

Change	Description	Date
Whitepaper updated	Second publication.	January 19, 2022
Initial publication	Whitepaper first published.	May 1, 2020

Note

To subscribe to RSS updates, you must have an RSS plug-in enabled for the browser that you are using.

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.

AWS Glossary

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