AWS WAF Security Automations

AWS Implementation Guide

Heitor Vital

Lee Atkinson

Ben Potter

Vlad Vlasceanu

Aijun Peng

Chaitanya Deolankar

September 2016

Last updated: July 2020 (see revisions)

Contents

About this guide

This implementation guide discusses architectural considerations and configuration steps for deploying the AWS WAF Security Automations solution on the Amazon Web Services (AWS) Cloud. It includes links to AWS CloudFormation templates that launch, configure, and run the AWS security, compute, storage, and other services required to deploy this solution on AWS, using AWS best practices for security and availability.

The guide is intended for IT Managers, Security Engineers, DevOps Engineers, Developers, Solutions Architects, and Website Administrators.

Overview

AWS WAF is a web application firewall that helps protect web applications from common web exploits that can affect application availability, compromise security, or consume excessive resources. AWS WAF enables customers to define customizable web security rules, giving them control over which traffic to allow or block to web applications and APIs deployed on Amazon CloudFront, an Application Load Balancer, or API Gateway.

Configuring WAF rules can be challenging and burdensome to large and small organizations alike, especially for those who do not have dedicated security teams. To simplify this process, AWS offers the AWS WAF Security Automations solution, which automatically deploys a single web access control list (web ACL) with a set of AWS WAF rules designed to filter common web-based attacks. During initial configuration of this solution's AWS CloudFormation template, users specify which protective features to include, as depicted in the image below. After this solution is deployed, AWS WAF will begin inspecting web requests to their existing CloudFront distributions or Application Load Balancer, and block them as applicable.

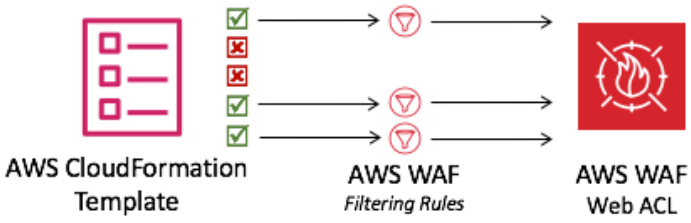

Figure 1: Configuration of the AWS WAF web ACL

The information in this guide assumes working knowledge of AWS services such as AWS WAF, Amazon CloudFront, Application Load Balancers, and AWS Lambda. It also requires basic knowledge of common web-based attacks, and mitigation strategies.

> **Note:** Starting from version 3.0, the AWS WAF Security Automations solution supports the latest version of **AWS WAF** (AWS WAFV2) service API.

Cost

You are responsible for the cost of the AWS services used while running the AWS WAF Security Automations solution. The total cost for running this solution depends on the protection activated and the amount of data ingested, stored, and processed. For full details, see the pricing webpage for each AWS service you will be using in this solution.

Note: If you choose to use the Athena Log Parser on installation, this solution schedules a query to run against the WAF and/or application access logs in your Amazon S3 bucket(s) as configured. You are charged based on the amount of data scanned by each query. Partitioning is applied to logs and queries to keep costs low. By default, application access logs are moved from their original S3 location to a partitioned folder structure. You have the option to keep original logs as well but you will be charged for duplicated log storage. This solution uses Workgroups to segment workloads and these can be configured to manage query access and costs. See Appendix F for a sample cost estimate calculation. For more information, see Amazon Athena Pricing.

Protection Capabilities

Web applications are vulnerable to a variety of attacks. These attacks include specially crafted requests designed to exploit a vulnerability or take control of a server; volumetric attacks designed to take down a website; or bad bots and scrapers programmed to scrape and steal web content.

This solution leverages AWS CloudFormation to quickly and easily configure AWS WAF rules that help block the following common attacks:

- **SQL injection:** Attackers insert malicious SQL code into web requests in an effort to extract data from your database. This solution is designed to block web requests that contain potentially malicious SQL code.
- **Cross-site scripting:** Also known as XSS, attackers use vulnerabilities in a benign website as a vehicle to inject malicious client-site scripts into a legitimate user's web browser. This solution is designed to inspect commonly explored elements of incoming requests to identify and block XSS attacks.
- **HTTP floods**: Web servers and other backend resources are at risk of Distributed Denial of Service (DDoS) attacks, such as HTTP floods. This solution automatically triggers a rate-based rule when web requests from a client exceed a configurable threshold. Alternatively, enforce this threshold by processing AWS WAF logs using an AWS Lambda function or an Amazon Athena query.
- **Scanners and probes**: Malicious sources scan and probe Internet-facing web applications for vulnerabilities. They send a series of requests that generate HTTP 4xx error codes, and you can use this history to help identify and block malicious source IP addresses. This solution creates an AWS Lambda function or an Amazon Athena query that automatically parses Amazon CloudFront or Application Load Balancer access logs, counts the number of bad requests from unique source IP addresses per

minute, and updates AWS WAF to block further scans from addresses with high error rate – the ones that reached the defined-error threshold.

- **Known attacker origins (IP reputation lists)**: A number of organizations maintain reputation lists of IP addresses operated by known attackers, such as spammers, malware distributors, and botnets. This solution leverages the information in these reputation lists to help you block requests from malicious IP addresses.
- **Bots and scrapers**: Operators of publicly accessible web applications have to trust that the clients accessing their content identify themselves accurately, and that they will use services as intended. However, some automated clients, such as content scrapers or bad bots, misrepresent themselves to bypass restrictions. This solution helps you identify and block bad bots and scrapers.

Architecture Overview

Deploying this solution with the **default parameters** builds the following environment in the AWS Cloud.

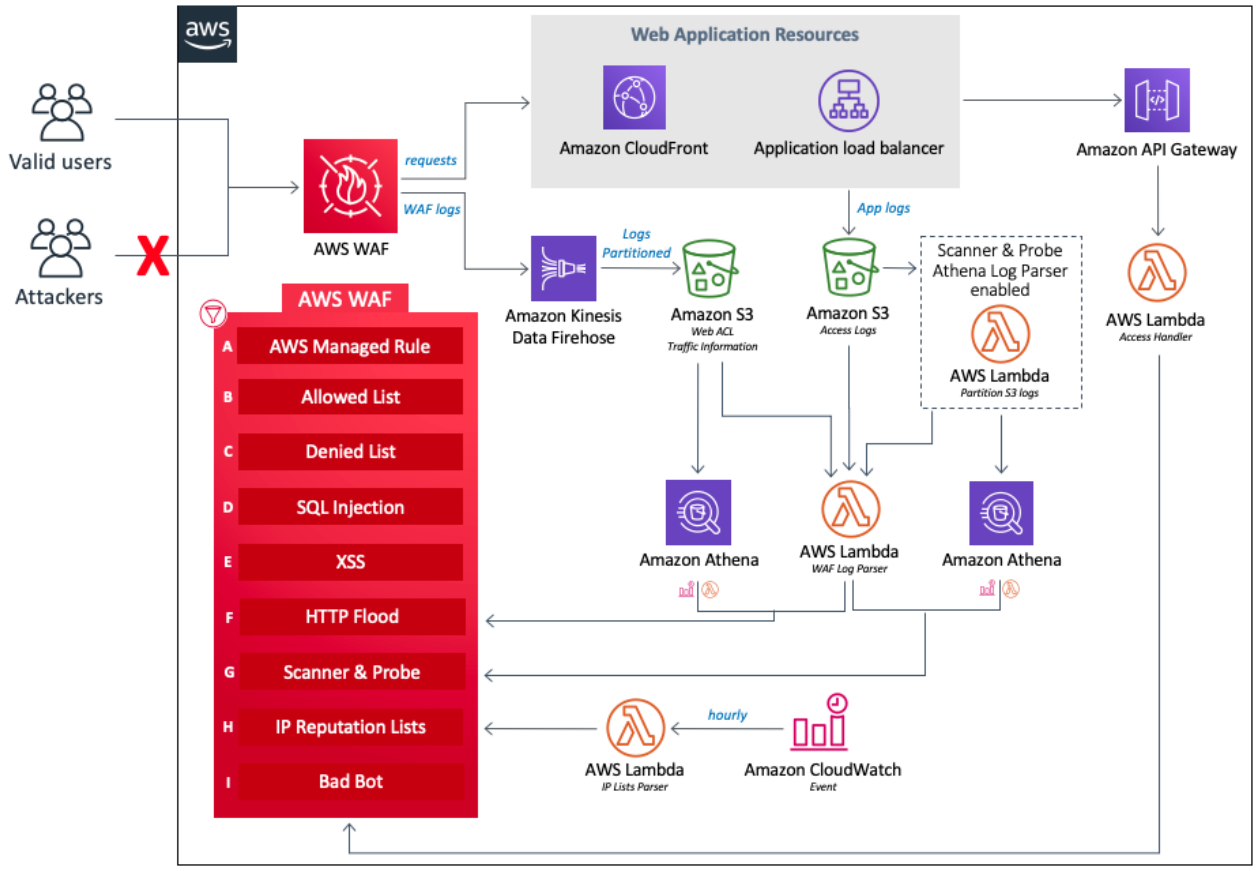

Figure 2: AWS WAF Security Automations architecture on AWS

At the core of the design is an AWS WAF web ACL, which acts as the central inspection and decision point for all incoming requests to a web application. During initial configuration of the AWS CloudFormation stack, the user defines which protective components to activate. Each component operates independently and adds different rules to the web ACL.

The components of this solution can be grouped into the following areas of protection:

- **AWS Managed Rules (A):** This component contains a set of AWS managed core rules that are generally applicable to web applications. It provides protection against exploitation of a wide range of common application vulnerabilities or other unwanted traffic, including those described in OWASP publications, without having to write your own rules.

- **Manual IP lists (B and C):** This component creates two specific AWS WAF rules that allow you to manually insert IP addresses that you want to allow or deny.
- **SQL Injection (D) and XSS (E):** This solution configures two native AWS WAF rules that are designed to protect against common SQL injection or cross-site scripting (XSS) patterns in the URI, query string, or body of a request.
- **HTTP flood (F):** This component protects against attacks that consist of a large number of requests from a particular IP address, such as a web-layer DDoS attack or a brute-force login attempt. With this rule, you set a threshold that defines the maximum number of incoming requests allowed from a single IP address within a five-minute period. Once this threshold is breached, additional requests from the IP address are temporarily blocked. You can implement this rule by using an AWS WAF rate-based rule, or by processing AWS WAF logs using an AWS Lambda function or an Amazon Athena query. For more information about the tradeoffs related to HTTP flood mitigation options, see Appendix A.
- **Scanners and Probes (G):** This component parses application access logs searching for suspicious behavior, such as an abnormal amount of errors generated by an origin. It then blocks those suspicious source IP addresses for a customer-defined period of time. You can implement this rule using an AWS Lambda function or an Amazon Athena query. For more information about the tradeoffs related to Scanners and Probes mitigation options, see Appendix A.
- **IP Reputation Lists (H):** This component is the `IP Lists Parser` AWS Lambda function which checks third-party IP reputation lists hourly for new ranges to block. These lists include the Spamhaus Don't Route Or Peer (DROP) and Extended DROP (EDROP) lists, the Proofpoint Emerging Threats IP list, and the Tor exit node list.
- **Bad Bots (I):** This component automatically sets up a honeypot, which is a security mechanism intended to lure and deflect an attempted attack. This solution's honeypot is a trap endpoint that you can insert in your website to detect inbound requests from content scrapers and bad bots. If a source accesses the honeypot, the `Access Handler` AWS Lambda function will intercept and inspect the request to extract its IP address, and then add it to an AWS WAF block list.

Each of the three custom AWS Lambda functions in this solution publish execution metrics to Amazon CloudWatch. For more information on these Lambda functions, see Appendix B.

Deployment Considerations

The AWS WAF Security Automations solution is designed to protect web applications and APIs deployed with Amazon CloudFront, an Application Load Balancer, or an Amazon API

Gateway. The following sections provide other constraints and considerations for implementing this solution.

> **Note**: The included AWS CloudFormation template should be used as a starting point for implementing AWS WAF rules. We recommend adding custom rules, applying log analysis, and leveraging AWS WAF managed rules, based on your company's needs.

AWS WAF

Web ACL Rules

The web ACL that this solution generates is designed to offer comprehensive protection for web applications. The default configuration adds nine WS WAF rules to this solution's web ACL. You can manually modify the web ACL to add further rules, but note that there is a 10-rule limit for individual web ACLs. This solution also allows the option to include AWS Managed Rules as the first priority before all additional eight custom AWS WAF rules. To include AWS Managed Rules, choose `yes` for the relevant boxin the parameter list when launching the CloudFormation stack.

IP Match Conditions

AWS WAF can block a maximum of 10,000 IP address ranges in Classless Inter-Domain Routing (CIDR) notation per IP match condition. Each list that this solution creates is subject to this limit. See AWS WAF quotas (formerly called limits) in the AWS WAF Developer Guide for more information. Starting from version 3.0, this solution creates two IP sets to attach to each rule, one for IPv4 and one for IPv6.

Web ACL Traffic Logging

If you create the stack outside US East (N. Virginia) and set the Endpoint Type as CloudFront, you must set **Activate HTTP Flood Protection** to `no` or `yes - AWS WAF rate based rule`.

The other two options (`yes - AWS Lambda log parser` and `yes - Amazon Athena log parser`) require activating AWS WAF Logs on a Web ACL that runs in all AWS Edge Locations and this is not supported outside US East (N. Virginia). For more information about logging Web ACL traffic, see the AWS WAF developer guide.

Endpoint Type Update

You must use blue-green deployment to update the Endpoint Type after creating the stack. Do not manually change the Endpoint Type.

Solution Updates

Starting from version 3.0, this solution supports AWS WAFV2 (AWS WAF). All the AWS WAF classic API calls have been replaced with AWS WAFV2 API calls. It removes dependencies on Node.js and uses the most up-to-date Python runtime. To continue using this solution with the latest features and improvements, you must deploy version 3.0 as a new stack.

AWS Regions and Multiple Deployments

This solution includes an AWS CloudFormation template for web applications. The template contains two nested templates: one that deploys the Web ACL and a separate template that includes resources related to AWS Glue, Amazon Athena, and Amazon Kinesis Data Firehose.

	AWS WAF Web ACL	AWS Glue	Amazon Athena	Amazon Kinesis Data Firehose
Endpoint Type				
CloudFront	✓			
Application Load Balancer (ALB)	✓			
Activate HTTP Flood Protection				
yes - AWS Lambda log parser				✓
yes - Amazon Athena log parser		✓	✓	✓
Activate Scanner & Probe Protection				
yes - Amazon Athena log parser		✓	✓	

This solution chooses which nested template to deploy based on the user selected input template parameters. See the parameters table under Step 1 for details about services dependencies.

Customers can deploy the AWS WAF Security Automations solution in different AWS Regions, or deploy it multiple times in the same AWS Region. Note that each unique deployment will incur additional charges.

> **Note:** If you plan to configure multiple instances of this solution in the same Region, you must use a unique AWS CloudFormation stack name and Amazon S3 bucket name for each deployment.

Depending on the input parameters values you define, this solution requires different resources. These resources are listed in the table below, and are currently available in specific AWS Regions only.

Cross-Site Scripting False Positives

This solution configures a native AWS WAF rule that inspects commonly explored elements of incoming requests to identify and block cross-site scripting (XSS) attacks. This detection

pattern is less effective if your workload legitimately allows users to compose and submit HTML, for example a rich text editor in a content management system. In this scenario, consider creating an exception rule that bypasses the default XSS rule for specific URL patterns that accept rich text input, and implement alternate mechanisms to protect those excluded URLs.

Additionally, some image or custom data formats can trigger false positives because they contain patterns indicating a potential XSS attack in HTML content. For example, an SVG file might contain a `<script>` tag. If you expect this type of content from legitimate users, narrowly tailor your XSS rules to allow HTML requests that include these other data formats.

Complete the following steps to update XSS rule in order to exclude URLs that accept HTML as input. See the Amazon WAF Developer Guide for detailed instructions.

1. Sign in to the AWS Management Console and open the AWS WAF console.
2. Create a string match or regex condition.
3. Configure the filter settings to inspect URI and list values that you want to accept against the XSS rule.
4. Edit this solution's **XSS Rule** and add the new condition that you created.

   To exclude all URLs in the list, match the highlighted section in green below:

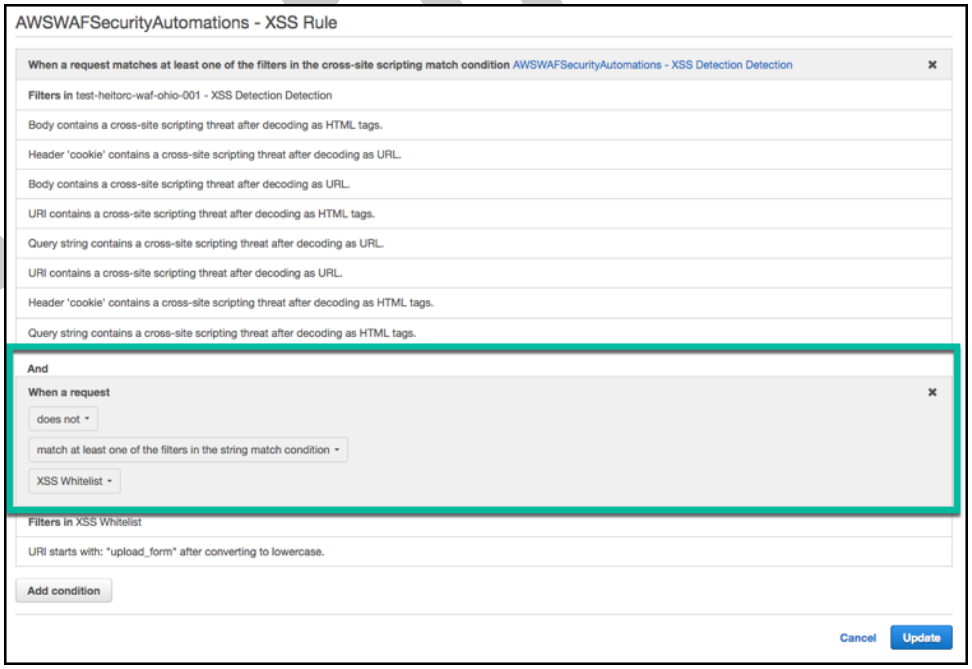

Figure 3: XSS extra condition to accept services with high false positive rate

AWS CloudFormation Template

This solution uses AWS CloudFormation to bootstrap AWS infrastructure and automate the deployment of the AWS WAF Security Automations solution on the AWS Cloud. It includes the following AWS CloudFormation template which contains two nested templates: one that deploys Amazon CloudFront and one that deploys an Application Load Balancer.

View template

aws-waf-security-automations.template: Use this template to launch the AWS WAF Security Automations solution for web applications. The default configuration deploys an AWS WAF web ACL with eight preconfigured rules, but you can also customize the template based on your specific needs.

Automated Deployment

Before you launch the AWS CloudFormation template, review the architectural and configuration considerations discussed in this guide. Follow the step-by-step instructions in this section to configure and deploy the AWS WAF Security Automations solution into your account.

Time to deploy: Approximately 15 minutes.

Prerequisites

This solution is designed to work with web applications deployed with Amazon CloudFront or an Application Load Balancer. If you don't already have one of these resources configured, complete the applicable task before you launch this solution.

Configure a CloudFront Distribution

Complete the following steps to configure a CloudFront Distribution for your web application's static and dynamic content. See the Amazon CloudFront Developer Guide for detailed instructions.

1. Create a CloudFront web application distribution. See Creating or Updating a Web Distribution Using the CloudFront Console.
2. Configure static and dynamic origins. See Using Amazon S3 Origins and Custom Origins for Web Distributions.
3. Specify your distribution's behavior. See Values that You Specify When You Create or Update a Web Distribution.

> **Note:** If you choose `CLOUDFRONT` as your endpoint, you must create your WAFV2 resources in the US East (N. Virginia) Region, `us-east-1`.

Configure an Application Load Balancer

Complete the following steps to configure an Application Load Balancer to distribute incoming traffic to your web application. See the Application Load Balancer Guide for detailed instructions.

1. Configure a Load Balancer and a Listener
2. Configure Security Settings for an HTTPS Listener
3. Configure a Security Group
4. Configure a Target Group
5. Configure Targets for the Target Group
6. Create the Load Balancer

What We'll Cover

The procedure for deploying this architecture on AWS consists of the following steps. For detailed instructions, follow the links for each step.

Step 1. Launch the Stack

- Launch the AWS CloudFormation template into your AWS account.
- Enter values for required parameters: **Stack Name**, **Application Access Log Bucket Name**.
- Review the other template parameters, and adjust if necessary.

Step 2. Modify the Accepted and Denied Sets (Optional)

- Manually add applicable IP addresses to the AWS WAF accept list and deny list.

Step 3. Embed the Honeypot Link in Your Web Application (Optional)

- Embed the hidden trap endpoint in your application.
- Explicitly disallow access to the endpoint using the robots exclusion standard (CloudFront only).

Step 4. Associate the Web ACL with Your Web Application

- Associate your Amazon CloudFront web distribution(s) or Application Load Balancers with the web ACL that this solution generates. You can associate as many distributions or load balancers as you want.

Step 5. Configure Web Access Logging

- Enable web access logging for your Amazon CloudFront web distribution or Application Load Balancer, and send log files to the appropriate Amazon S3 bucket. Remember to save logs in a folder named `AWSLogs` (log prefix `AWSLogs/`).

Step 1. Launch the Stack

This automated AWS CloudFormation template deploys the AWS WAF Security Automations solution on the AWS Cloud.

> **Note**: You are responsible for the cost of the AWS services used while running this solution. For full details, see the pricing webpage for each AWS service you will be using in this solution.

1. Log in to the AWS Management Console and click the button to launch the `aws-waf-security-automations` AWS CloudFormation template.
   You can also download the template as a starting point for your own implementation.

2. The template launches in the US East (N. Virginia) Region by default. To launch this solution in a different AWS Region, use the Region selector in the console navigation bar.

> **Note**: Depending on the input parameters values you define, this solution requires different resources. These resources are currently available in specific AWS Regions only. Therefore, you must launch this solution in an AWS Region where these services are available. For more information, see AWS Regions and Multiple Deployments.

3. On the **Specify template** page, verify that you selected the correct template and choose **Next**.
4. On the **Specify stack details** page, assign a name to your AWS WAF configuration in the **Stack name** field. This will also be the name of the web ACL that the template creates.
5. Under **Parameters**, review the parameters for the template and modify them as necessary. To opt out of a particular feature, choose `none` or `no` as applicable. This solution uses the following default values.

Parameter	Default	Description
Stack Name	*<Requires input>*	The stack name cannot contain spaces and must be unique within your AWS account. This will also be the name of the web ACL that the template creates.

Parameter	Default	Description
Activate AWS Managed Rules	no	Choose yes to enable the component designed to add AWS Managed Rules to the top of the Web ACL priority list.
Activate SQL Injection Protection	yes	Choose yes to enable the component designed to block common SQL injection attacks.
Activate Cross-site Scripting Protection	yes	Choose yes to enable the component designed to block common XSS attacks.
Activate HTTP Flood Protection	yes - AWS WAF rate-based rule	Select the component used to block HTTP flood attacks. See Appendix A for more information about the tradeoffs related to the mitigation options.
Activate Scanner & Probe Protection	yes - AWS Lambda log parser	Select the component used to block scanners and probes. See Appendix A for more information about the tradeoffs related to the mitigation options.
Activate Reputation List Protection	yes	Choose yes to block requests from IP addresses on third-party reputation lists (supported lists: spamhaus, torproject, and emergingthreats).
Activate Bad Bot Protection	yes	Choose yes to enable the component designed to block bad bots and content scrapers.
Application Access Log Bucket Name	*<Requires input>*	If you select yes for the **Activate Scanner & Probe Protection** parameter, enter the name of the Amazon S3 bucket where you want to store access logs for your CloudFront distribution or Application Load Balancer. To deactivate this protection, ignore this parameter. If you use an existing S3 bucket for this parameter, it must be located in the same AWS Region where you are deploying the AWS CloudFormation template. You cannot use the same S3 bucket for multiple deployments in the same AWS Region. **Note:** Enable web access logging for your Amazon CloudFront web distribution or Application Load Balancer to send log files to this Amazon S3 bucket and remember to save logs in a folder named AWSLogs (log prefix AWSLogs/).
Endpoint Type	CloudFront	Select the type of resource being used.
Request Threshold	100	If you chose yes for the **Activate HTTP Flood Protection** parameter, enter the maximum acceptable requests per five (5) minutes per IP address. The minimum acceptable value is 100 for the rate-based rule. If you are using Athena or a Lambda log parser, it can be any value. To deactivate this protection, ignore this parameter.
Error Threshold	50	If you chose yes for the **Activate Scanner & Probe Protection** parameter, enter the maximum acceptable bad requests per minute per IP address. If you chose to deactivate this protection, ignore this parameter.

Parameter	Default	Description
WAF Block Period	240	If you chose `yes Athena or Lambda log parser` for the **Activate Scanner & Probe Protection** or **Activate HTTP Flood Protection** parameters, enter the period (in minutes) to block applicable IP addresses. To deactivate log parsing, ignore this parameter.
Keep Data in Original s3 location	No	If you chose `Amazon Athena log parser` for the **Activate Scanners & Probes Protection parameter**, partitioning will be applied to application access log files and Athena queries. By default, log files will be moved from their original location to a partitioned folder structure in s3. Choose `yes` if you also want to keep a copy of the logs in their original location. Choosing `yes` will duplicate your log storage. If you did not choose to activate Athena log parsing, ignore this parameter.

6. Choose **Next.**
7. On the **Configure stack options** page, you can specify tags (key-value pairs) for resources in your stack and set additional options, and then choose **Next**.
8. On the **Review** page, review and confirm the settings. Check the boxes acknowledging that the template will create AWS Identity and Access Management (IAM) resources and any additional capabilities required.
9. Choose **Create** to deploy the stack.
5. View the status of the stack in the AWS CloudFormation Console in the **Status** column. You should see a status of CREATE_COMPLETE in approximately 15 minutes.

> **Note:** In addition to the Log Parser, IP Lists Parser, Access Handler AWS Lambda functions, this solution includes the `helper` and `custom-resource` Lambda functions, which run only during initial configuration or when resources are updated or deleted.
>
> When running this solution, you will see all functions in the AWS Lambda console, but only the three primary solution functions are regularly active. However, do not delete the other two functions, they are necessary to manage associated resources.

10. To see details for the stack resources, choose the **Outputs** tab. This will include the **BadBotHoneypotEndpoint** value, which is the API Gateway honeypot endpoint. Note this value because you will use it in Step 3.

Step 2. Modify the Allowed and Denied Sets (Optional)

After deploying this solution's AWS CloudFormation stack, you can manually modify the allowed and denied sets to add or remove IP addresses as necessary.

1. Open the AWS WAF console, and in the left navigation pane, choose **IP addresses**.
2. Choose **Whitelist Set** and add IP addresses from trusted sources.
3. Choose **Manual Blacklist Set** and add IP addresses you want to block.

Step 3. Embed the Honeypot Link in Your Web Application (Optional)

If you chose to activate scanner and probe protection in Step 1, the AWS CloudFormation template creates a trap endpoint to a low-interaction production honeypot. It is intended to detect and divert inbound requests from content scrapers and bad bots. Valid users will not attempt to access this endpoint. However, content scrapers and bots, such as malware that scans for security vulnerabilities and scrapes email addresses might attempt to access the trap endpoint. In this scenario, the Access Handler AWS Lambda function will inspect the request in order to extract its origin, and then update the associated AWS WAF rule to block subsequent requests from that IP address.

Use the applicable procedure to embed the honeypot link for requests from either a CloudFront distribution or an Application Load Balancer.

Create a CloudFront Origin for the Honeypot Endpoint

Use this procedure for web applications that are deployed with a CloudFront distribution. With CloudFront, you can include a `robots.txt` file to help identify content scrapers and bots that ignore the robots exclusion standard. Complete the following steps to embed the hidden link and then explicitly disallow it in your `robots.txt` file.

1. Open the AWS CloudFormation console, choose the stack that you built in Step 1, and then choose the **Outputs** tab.
2. From the **BadBotHoneypotEndpoint** key, copy the endpoint URL. It contains two components that you will need to complete this procedure: the endpoint host name (e.g., `xxxxxxxxxx.execute-api.region.amazonaws.com`) and the request URI (`/ProdStage`).
3. Open the Amazon CloudFront console and choose the distribution that you want to use.
4. Choose **Distribution Settings**, and on the **Origins** tab, choose **Create Origin**.

5. In the **Origin Domain Name** field, paste the host name component of the endpoint URL that you copied in Step 2. In **Origin Path**, paste the request URL that you also copied in Step 2. Accept the default values for the other fields and choose **Create**.
6. On the **Behaviors** tab, choose **Create Behavior**.
7. Create a new cache behavior and point it to the new origin. You can use a custom domain, such as a fake product name that is similar to other content in your web application.
8. Embed this endpoint link in your content pointing to the honeypot. You should hide this link from your human users. As an example, review the following code sample:

```
<a href="/behavior_path" rel="nofollow" style="display: none" aria-
hidden="true">honeypot link</a>
```

Note: It is your responsibility to verify what tag values work in your website environment. Do not use rel="nofollow" if your environment doesn't observe it. For more information about robots meta tags configuration, see the Google developer's guide.

9. Modify the `robots.txt` file in the root of your website to explicitly disallow the honeypot link, as follows:

```
User-agent: *
Disallow: /behavior_path
```

Embed the Honeypot Endpoint as an External Link

Use this procedure for web applications that are deployed with an Application Load Balancer.

1. Open the AWS CloudFormation console, choose the stack that you built in Step 1, and then choose the **Outputs** tab.
2. From the **BadBotHoneypotEndpoint** key, copy the endpoint URL.
3. Embed this endpoint link in your web content. Use the full URL that you copied in Step 2. Hide this link from your human users. As an example, review the following code sample:

```
<a href="BadBotHoneypotEndpoint value" rel="nofollow" style="display:
none" aria-hidden="true">honeypot link</a>
```

Note: This example uses `nofollow` to instruct robots to not access the honeypot URL. However, because the link is embedded externally, you cannot include a `robots.txt` file to explicitly disallow the link. It is your responsibility to verify what

tags work in your website environment. Do not use rel="nofollow" if your environment doesn't observe it.

Step 4. Associate the Web ACL with Your Web Application

Update your Amazon CloudFront distribution(s) or Application Load Balancer(s) to activate AWS WAF and logging using the resources you generated in Step 1.

Note: You can associate only one web ACL with a CloudFront distribution or an Application Load Balancer. Therefore, you cannot use this solution's web ACL in addition to an existing association.

1. Open the AWS WAF console and choose the web ACL that you want to use.
2. On the **Rules** tab, choose **Add association**.
3. For **AWS resources using this web ACL**, choose the CloudFront distribution or Application Load Balancer.
4. Choose **Add** to save your changes.

Step 5. Configure Web Access Logging

Configure Amazon CloudFront or your Application Load Balancer to send web access logs to the appropriate Amazon S3 bucket so that this data is available for the Log Parser AWS Lambda function.

Store Web Access Logs from a CloudFront Distribution

1. Open the Amazon CloudFront console.
2. Select your web application's distribution, and choose **Distribution Settings**.
3. On the **General** tab, choose **Edit**.
4. For **AWS WAF Web ACL**, choose the web ACL solution created (the same name you assigned to the stack during initial configuration).
5. For **Logging**, choose **On**.
6. For **Bucket for Logs**, choose the Amazon S3 bucket that you want use to store web access logs (defined in Step 1). The drop-down list enumerates the buckets associated with the current AWS account.
7. Set the log prefix as `AWSLogs/`. If you enter `AWSLogs` as the prefix but get a message saying **prefix cannot start with 'AWSLogs'**, then remove the prefix. The Application Load Balancer will use `AWSLogs` as the default prefix.

8. Choose **Yes, edit** to save your changes.

6. For more information, see Access Logs in the *Amazon CloudFront Developer Guide*.

Store Web Access Logs from an Application Load Balancer

1. Open the Amazon Elastic Compute Cloud (Amazon EC2) console.
2. In the navigation pane, choose **Load Balancers**.
3. Select your web application's Application Load Balancer.
4. On the Description tab, choose Edit attributes.
5. Choose Enable access logs.
6. For **S3 location**, type the name of the Amazon S3 bucket that you want use to store web access logs (defined in Step 1).
7. Set the log prefix as `AWSLogs/`. If you enter `AWSLogs` as the prefix but get a message saying **prefix cannot start with 'AWSLogs'**, then remove the prefix. Application Load Balancer will use `AWSLogs` as the default prefix.
8. Choose **Save**.

   For more information, see Access Logs for Your Application Load Balancer in the *Elastic Load Balancing User Guide*.

Uninstall Solution

To uninstall the solution, delete the CloudFormation stacks:

1. Sign in to the AWS CloudFormation console.
2. Select the solution's parent stack. All other solution stacks will be deleted automatically.
3. Choose **Delete**.

Note: Uninstalling the solution deletes all the AWS resources used by the solution except for the Amazon S3 buckets. If some IP sets fail to delete due to rate exceeded throttling issue caused by the AWA WAF API limits, manually delete those IP sets and then delete the stack.

Additional Resources

AWS service documentation

- AWS CloudFormation
- AWS WAF
- Amazon CloudFront
- Elastic Load Balancing
- Amazon API Gateway
- AWS Lambda
- Amazon Kinesis Data Firehose
- Amazon S3
- AWS Glue
- Amazon Athena
- Amazon CloudWatch

Associated AWS whitepapers

- AWS Best Practices for DDoS Resiliency

Associated AWS Security Blog posts

- How to Reduce Security Threats and Operating Costs Using AWS WAF and Amazon CloudFront
- How to Configure Rate-Based Blacklisting with AWS WAF and AWS Lambda
- How to Use AWS WAF to Block IP Addresses That Generate Bad Requests
- How to Import IP Address Reputation Lists to Automatically Update AWS WAF IP Blacklists
- How to Use AWS CloudFormation to Automate Your AWS WAF Configuration with Example Rules and Match Conditions
- How to Prevent Hotlinking by Using AWS WAF, Amazon CloudFront, and Referer Checking

Third-Party IP Reputation Lists

- Spamhaus DROP List website
- Proofpoint Emerging Threats IP list
- Tor exit node list

Appendix A: Log Parser Options

As described in the Architecture Overview, there are three options to handle HTTP flood and scanner and probe protections. The following sections explain each of these options in more detail.

AWS WAF Rate-based Rule

Rate-based rules are available for HTTP flood protection and can be configured in AWS WAF. This feature allows you to specify the maximum number of web requests to allow from any single IP address in a trailing, continuously updated five-minute period. If an IP address breaches the configured limit, new requests will be blocked until the request rate falls below the configured threshold. For details, refer to AWS CloudFormation service role in the *AWS CloudFormation User Guide*.

Amazon Athena Log Parser

Both HTTP Flood and Scanner & Probe protection template parameters provide the Amazon Athena Log Parser option. When activated, AWS CloudFormation provisions an Amazon Athena query and a scheduled AWS Lambda function responsible for orchestrating Athena to execute, process result output, and update AWS WAF. This Lambda function is triggered by an Amazon CloudWatch event configured to trigger every five minutes. You can configure their run schedules by changing `QueryScheduledRunTime` in `aws-waf-security-automations.template`.

We recommend selecting this option when AWS WAF rate-based rules cannot be used and if you have familiarity with SQL language to implement customizations. For more information about how to change the default query, see Appendix D.

HTTP flood protection is based on AWS WAF access log processing and uses Amazon CloudFront/ALB log files. The WAF access log type has a lower lag time, which can be used to identify HTTP flood origins more quickly when compared to CloudFront/ALB log delivery time. However, you must select the CloudFront/ALB log type in the **Activate Scanner & Probe Protection** template parameter to receive response status codes.

AWS Lambda Log Parser

The HTTP Flood and Scanner & Probe template parameters provides the `AWS Lambda Log Parser` option. Use the Lambda log parser only when the previous two options are not available. A known limitation of this option is that information is processed within the context of the file being processed. For example, an IP may generate more requests/errors than the defined threshold but because this info is split into different files, each file doesn't store enough data to exceed the threshold.

Appendix B: Component Details

As described in the Architecture Overview, four of this solution's components use automations to inspect IP addresses and add them to the AWS WAF block list. The following sections explain each of these functions in more detail.

Log Parser - Application

The Application Log Parser helps protect against Scanners and Probes.

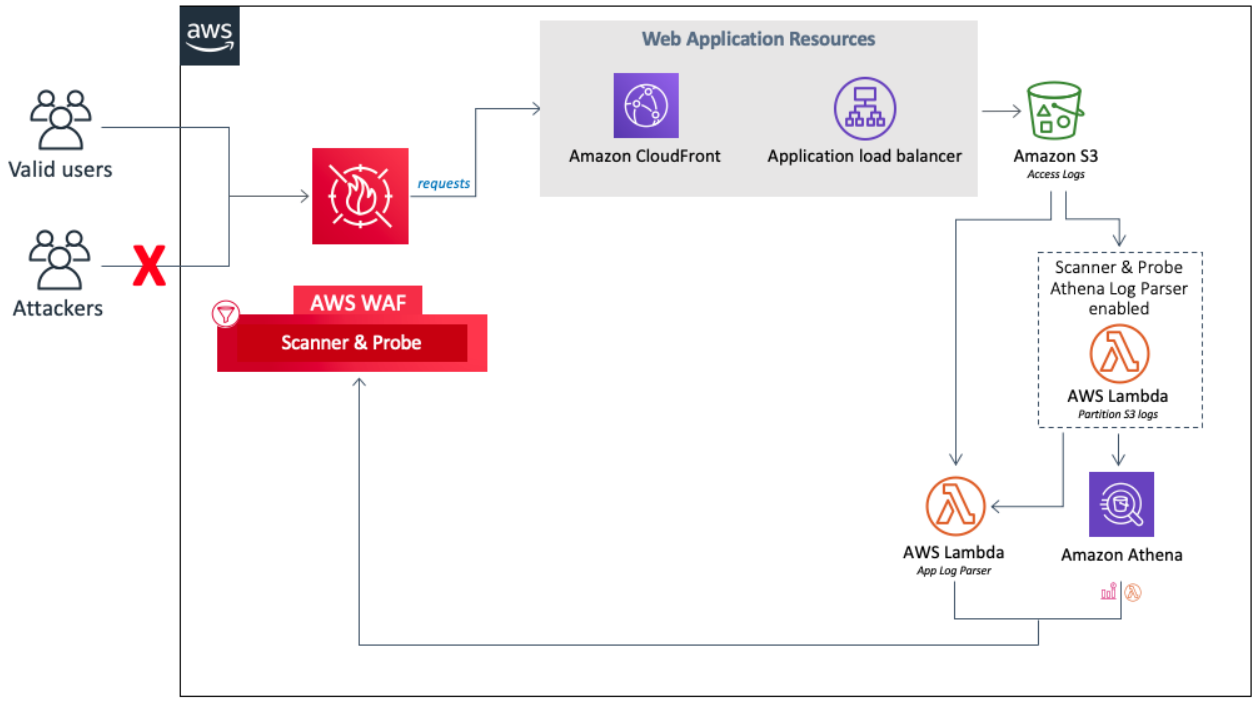

Figure 4: App Log Parser flow

1. Once Amazon CloudFront or an Application Load Balancer receives requests on behalf of your web application, it sends access logs to an Amazon S3 bucket.

   (Optional) If you select `Yes - Amazon Athena log parser` for the template parameters **Activate HTTP Flood Protection** and **Activate Scanner & Probe Protection,** a Lambda moves access logs from their original folder `customer-bucket/AWSLogs` to a newly partitioned folder `customer-bucket/AWSLogs-partitioned/optional-prefix/year=YYYY/month=MM/day=DD/hour=HH/`, upon their arrival in S3. If you select `yes` for the **Keep Data in Original S3 location** template parameter, logs will be kept in their original location as well as being copied to their partitioned folder, and this will duplicate your log storage.

> **Note:** For Athena Log Parser, this solution only partitions new logs that arrive in your Amazon S3 bucket after you deploy this solution. If you have existing logs that you would like to be partitioned, you must manually upload those logs to S3 after you deploy this solution.

2. Based on your selection for the template parameters **Activate HTTP Flood Protection** and **Activate Scanner & Probe Protection**, this solution processes logs using one of the following:
   a. **AWS Lambda**: each time a new access log is stored in the Amazon S3 bucket, the Log Parser Lambda function is triggered.
   b. **Amazon Athena:** by default, every five minutes the scanner and probe Athena query is executed and the output is pushed to AWS WAF. This process is triggered by an Amazon CloudWatch event, that then triggers the Lambda function responsible for executing the Amazon Athena query, and pushes the result into AWS WAF.
3. The log data is analyzed in order to identify IP addresses that have generated more errors than the defined threshold, it then updates an AWS WAF IP Set condition to block those IP addresses for a customer-defined period of time.

Log Parser - AWS WAF

If you select `yes - AWS Lambda log parser` or `yes - Amazon Athena log parser` for HTTP flood protection, this solution will provision the following components, which will be responsible for parsing AWS WAF logs in order to identify and block origins that flood the endpoint with a request rate above the threshold you defined.

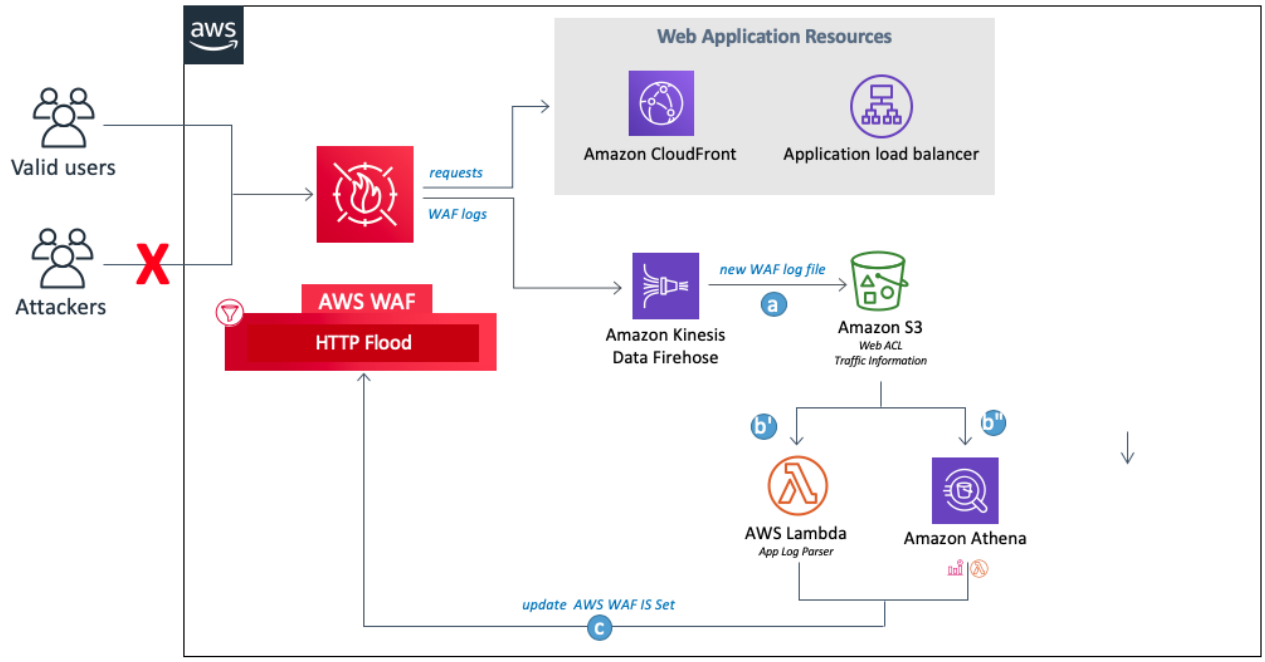

Figure 5: AWS WAF Log Parser flow

1. As AWS WAF receives access logs, it sends the logs to an Amazon Kinesis Data Firehose endpoint. Firehose then delivers the logs to a partitioned folder in S3: `customer-bucket/AWSLogs/optional-prefix/year=YYYY/month=MM/day=DD/hour=HH/`.
2. Based on your selection for the template parameters **Activate HTTP Flood Protection** and **Activate Scanner & Probe Protection**, this solution processes logs using one of the following:
   a. **AWS Lambda**: each time a new access log is stored in the Amazon S3 bucket, the Log Parser Lambda function is triggered.
   b. **Amazon Athena:** by default, every five minutes the scanner and probe Athena query is executed and the output is pushed to AWS WAF. This process is triggered by an Amazon CloudWatch event, that then triggers the Lambda function responsible for executing the Amazon Athena query, and pushes the result into AWS WAF.
3. The log data is analyzed in order to identify IP addresses that have sent more requests than the defined threshold, it then updates an AWS WAF IP Set condition to block those IP addresses for a customer-defined period of time.

IP Lists Parser

The IP Lists Parser AWS Lambda function helps protect against known attackers identified in third-party IP reputation lists.

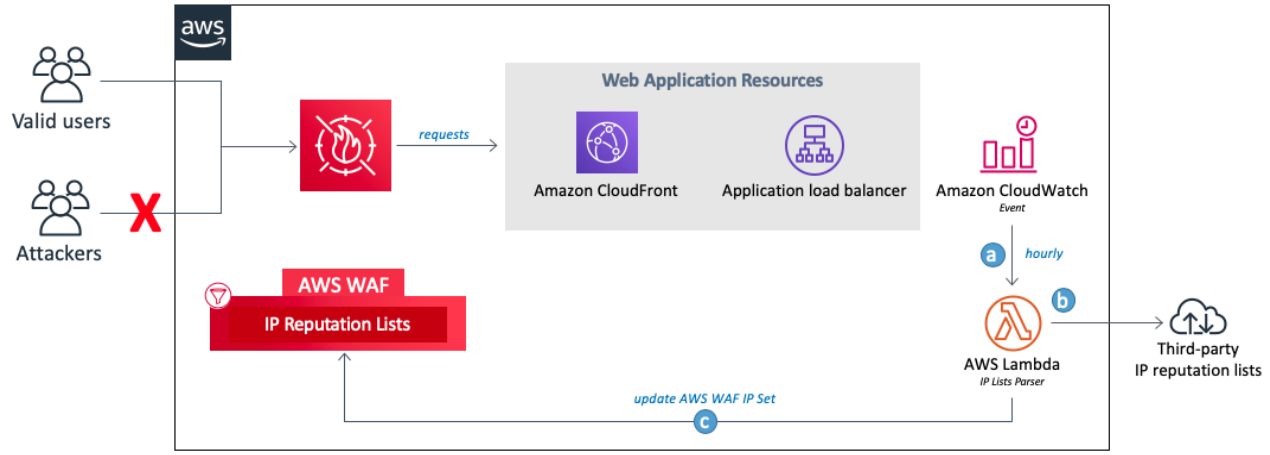

Figure 6: IP Reputation Lists Parser flow

1. An hourly Amazon CloudWatch event triggers the IP Lists Parser Lambda function.
2. The Lambda function gathers and parses data from three sources:
    - Spamhaus Don't Route or Peer (DROP) and Extended DROP (EDROP) lists
    - Proofpoint Emerging Threats IP list
    - Tor exit node list
3. The Lambda function updates the AWS block list with the most current IP addresses.

Access Handler

The Access Handler AWS Lambda function inspects requests to the honeypot endpoint in order to extract their source IP address.

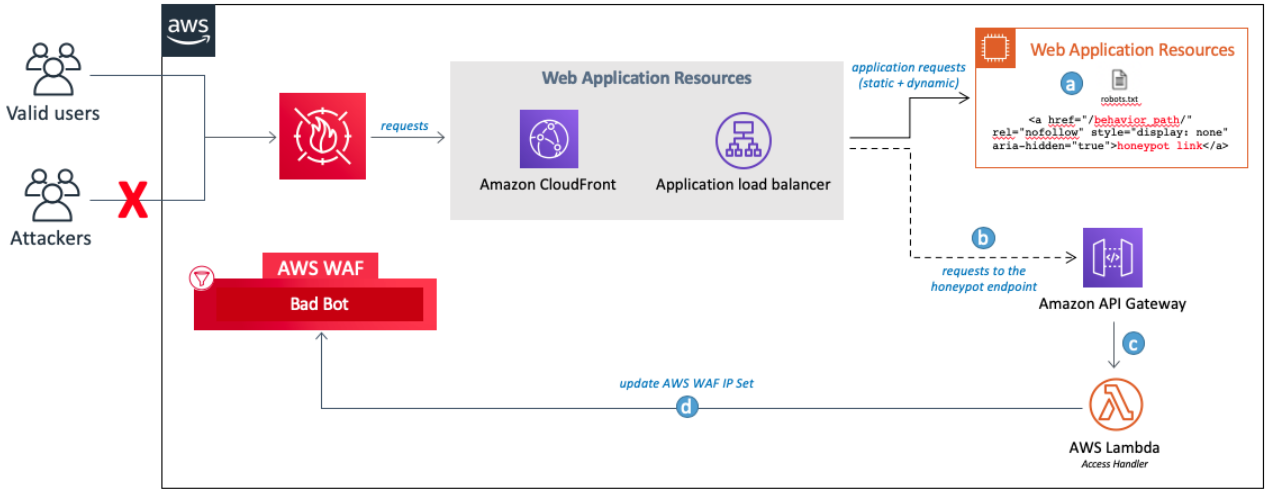

Figure 7: Access Handler and the honeypot endpoint

1. Embed the honeypot endpoint in your website and update your robots exclusion standard, as described in [Step 3. Embed the Honeypot Link in Your Web Application (Optional)].
2. When a content scraper or bad bot accesses the honeypot endpoint, it triggers the `Access Handler` Lambda function.
3. The Lambda function intercepts and inspects the request headers to extract the IP address of the source that accessed the trap endpoint.
4. The Lambda function updates an AWS WAF IP Set condition to block those IP addresses.

Appendix C: Log Parser JSON file

If you selected `Yes - AWS Lambda log parser` for the **Activate HTTP Flood Protection** template parameter, this solution creates a configuration file `<stack_name>-waf_log_conf.json` and uploads it to the Amazon Simple Storage Service (Amazon S3) bucket used to store the AWS WAF log files. To find the bucket name, see the **WafLogBucket** variable in the AWS CloudFormation output.

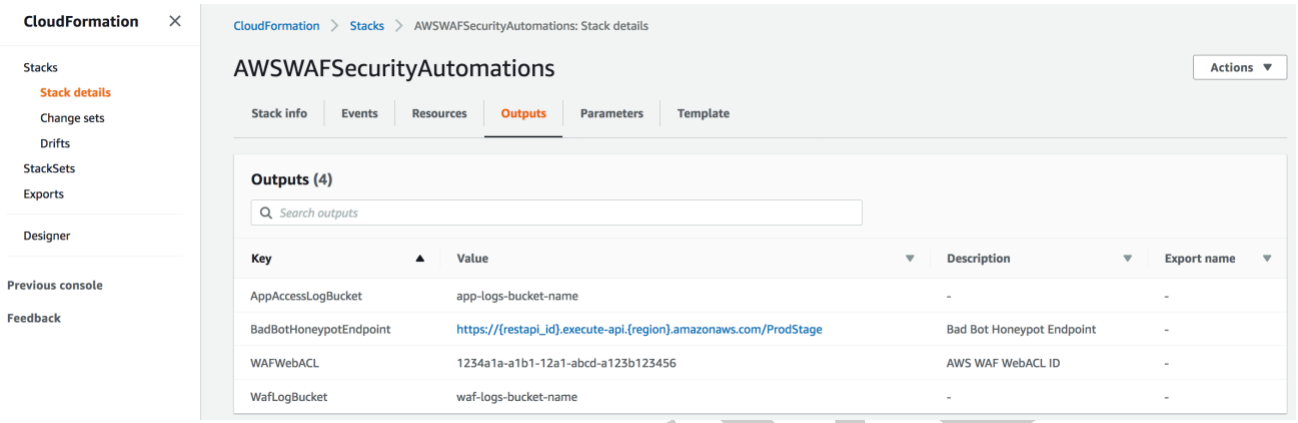

Figure 8: Stack Outputs

If you edit and overwrite the `<stack_name>-waf_log_conf.json` file on Amazon S3, the `Log Parser` Lambda function will consider the new values when processing new AWS WAF log files. Below is a sample configuration file:

```
{
    "general": {
        "requestThreshold": 2000,
        "blockPeriod": 240,
        "ignoredSufixes": [".css", ".js", ".jpg", "png", ".gif"]
    },
    "uriList": {
        "/search": {
            "errorThreshold": 500,
            "blockPeriod": 600
        }
    }
}
```

Figure 9: HTTP flood configuration file

Parameters:

- General
  - Request Threshold **[required]**: the maximum acceptable requests per five minutes per IP address. This solution uses the value you define when provisioning/updating the CloudFormation stack.
  - Block Period **[required]**: the period (in minutes) to block applicable IP addresses. This solution uses the value you define when provisioning/updating the CloudFormation stack.
  - Ignored Suffixes: requests accessing this type of resource will not count to request threshold. By default, this list is empty.
- URI List: use this to define a custom request threshold and block period for specifics URLs. By default, this list is empty.

If you selected `Yes - AWS Lambda log parser` for the **Activate Scanner & Probe Protection** template parameter, this solution creates a configuration file `<stack_name>-app_log_conf.json` and uploads it to the defined S3 bucket used to store CloudFront or Application Load Balancer log files.

If you edit and overwrite on the `<stack_name>-app_log_conf.json` on Amazon S3, the `Log Parser` Lambda function will consider the new values when processing new AWS WAF log files. Below is a sample configuration file:

```
{
    "general": {
        "errorThreshold": 50,
        "blockPeriod": 240,
        "errorCodes": ["400", "401", "403", "404", "405"]
    },
    "uriList": {
        "/login": {
            "errorThreshold": 5,
            "blockPeriod": 600
        },
        "/api/feedback": {
            "errorThreshold": 10,
            "blockPeriod": 240
        }
    }
}
```

Figure 10: Scanners and Probes configuration file

Parameters:

- General
    - Error Threshold **[required]**: the maximum acceptable bad requests per minute per IP address. This solution uses the value you defined when provisioning/updating the CloudFormation stack.
    - Block Period **[required]**: the period (in minutes) to block applicable IP addresses. This solution uses the value you defined when provisioning/updating the CloudFormation stack.
    - Error Codes: return status code considered errors. By default, the list considers the following HTTP status codes as errors: 400 (Bad Request), 401 (Unauthorized), 403 (Forbidden), 404 (Not Found), and 405 (Method Not Allowed).
- URI List: use this to define a custom request threshold and block period for specifics URLs. By default, this list is empty.

Appendix D: Amazon Athena Queries

If you selected `Yes - Amazon Athena log parser` for the **Activate HTTP Flood Protection** and/or the **Activate Scanner & Probe Protection** template parameters, this solution creates and executes Athena queries for CloudFront/ALB (ScannersProbesLogParser) or WAF logs (HTTPFloodLogParser), parses the output, and updates AWS WAF accordingly.

In order to improve performance and keep costs low, logs are partitioned based on timestamps in the file names. Athena queries are dynamically generated to use partition keys (year, month, day, and hour). By default, queries run every five minutes. You can configure their run schedules by changing `QueryScheduledRunTime in aws-waf-security-automations.template`. Each query run scans the last four to five hours of data by default. You can configure the amount of data that a query scans by changing the value for the **WAF Block Period** template parameter. Queries are also placed in separate workgroups to manage query access and costs.

> **Note**: Verify that Amazon Athena is configured to access the AWS Glue Data Catalog. This solution creates the access logs data catalog in AWS Glue and configures an Athena query to process the data. If Athena is not configured correctly, the query will fail to execute. For more information, see Upgrading to the latest AWS Glue Data Catalog Step-by-Step.

Use the following procedure to view these queries:

View WAF log queries:

1. Navigate to the Amazon Athena console, select the **Workgroup** tab.
2. Select **WAFLogAthenaQueryWorkGroup** from the list, then click **Switch workgroup**. This workgroup exists only if you selected **Yes - Amazon Athena log parser** for the **Activate HTTP Flood Protection** template parameter.

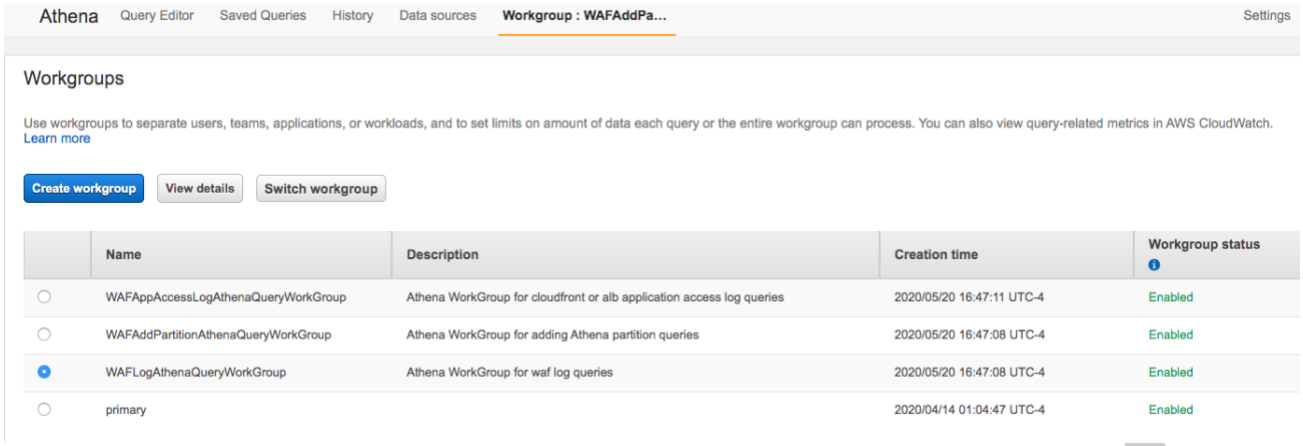

Figure 11: Amazon Athena WAF workgroups

3. Select the **History** tab.
4. Select and open `SELECT` queries from the list.

View application access log queries:

1. Navigate to the Amazon Athena console, select **Workgroup** tab.
2. Select WAFAppAccessLogAthenaQueryWorkGroup from the list, then click Switch workgroup. This workgroup only exists if you selected Yes - Amazon Athena log parser for the Activate Scanner & Probe Protection template parameter.
3. Select the **History** tab.
4. Select and open `SELECT` queries from the list.

View adding Athena partition queries:

1. Navigate to the Amazon Athena console, select **Workgroup** tab.
2. Select WAFAddPartitionAthenaQueryWorkGroup from the list, then click Switch workgroup. This workgroup only exists if you selected Yes - Amazon Athena log parser for the Activate HTTP Flood Protection and/or the Activate Scanner & Probe Protection template parameters.
3. Select the **History** tab.
4. Select and open `ALTER TABLE` queries from the list. These queries run every hour to add a new hourly partition to the Glue/Athena table.

Appendix E: Monitoring Dashboard

AWS recommends that customers configure a custom baseline monitoring system for each critical endpoint. For information on creating and using Customized Metric Views, see CloudWatch Dashboards.

The dashboard below shows an example of a custom baseline monitoring system:

Figure 12: Monitoring Dashboard

The dashboard displays the following metrics:

- **Allowed vs Blocked Requests**: Shows if you receive a surge in allowed access (2 times normal peak access), or blocked access (any period that identifies more than 1K blocked requests). Amazon CloudWatch sends an alert to a Slack channel. This metric can be used to track known DDoS (when blocked requests increase), or a new version of an attack (when the requests are allowed to access the system). Note that this metric is provided by this solution.
- **BytesDownloaded vs Uploaded**: Helps identify when a DDoS attack targets a service that normally doesn't receive a large amount of access in order to exhaust resources (e.g. search engine component sending MBs of information for one specific request parameters set).

- **ELB Spillover and Queue length**: Helps verify if an attack is causing damage to the infrastructure and the attacker is bypassing Amazon CloudFront or the AWS WAF layer, and attacking directly unprotected resources.
- **ELB Request Count**: Helps identify damage to the infrastructure. This metric shows if the attacker is bypassing the protection layer, or if an Amazon CloudFront cache rule should be reviewed to increase the cache hit rate.
- **ELB Healthy Host**: Can be used as another system health check metric.
- **ASG CPU Utilization**: Helps identify if the attacker is bypassing the Amazon CloudFront and AWS WAF, and Elastic Load Balancing. This metric can also be used to identify the damage of an attack.

Appendix F: Cost Estimate of Amazon Athena

If you use the Amazon Athena log parser option while running the HTTP Flood Protection and/or Scanner & Probe Protection rules, you will be charged for Athena usage. By default, each Athena query runs every five minutes and scans the past four hours of data. Partitioning is applied to logs and Athena queries to keep costs low. You can configure the number of hours of data that a query scans by changing the value for the **WAF Block Period** template parameter. However, increasing the amount of data scanned will likely increase Athena cost.

Note: Example CloudFront logs cost calculation:

By average, each Amazon CloudFront hit might generate around 500 bytes of data.

If there are 1.2M CloudFront objects hits per day, then there will be 200k (1.2M/6) hits per four hours assuming that data comes in at a consistent rate. You will need to consider your actual traffic patterns when calculate your cost.

An average 100 MB (0.0001TB) data scanned per query = 500 * 200K

Athena charges $5.00 per TB of data scanned

Cost/query scan = 0.0001* $5/TB = $0.0005

Athena query runs every five minutes:

60 minutes / 5 = 12 runs per hour

12 * 24 = 288 runs a day

Cost estimate/month = $0.0005 * 288 * 30 = $4.32

Actual costs will vary depending on your application's traffic patterns. For more information, see Amazon Athena pricing.

Appendix G: Collection of Operational Metrics

This solution includes an option to send operational metrics to AWS. We use this data to better understand how customers use this solution and related services and products. When enabled, the following information is collected and sent to AWS during initial deployment of this solution's AWS CloudFormation template:

- **Solution ID:** The AWS solution identifier
- **Unique ID (UUID):** Randomly generated, unique identifier for each deployment of this solution
- **Timestamp:** Data-collection timestamp
- **Solution configuration:** Features enabled and parameters set during initial launch
- **Lifecycle:** How long the customer used this solution (based on stack delete)
- **Log Parser data:**
    - The number of IP addresses in the Scanners and Probes set and the HTTP flood set to block
    - The number of requests processed and blocked
- **IP Lists Parser data:**
    - The number of IP addresses in the Reputation Lists set
    - The number of requests processed and blocked
- **Access Handler data:**
    - The number of IP addresses in the Bad Bot set
    - The number of requests processed and blocked

AWS owns the data gathered via this survey. Data collection will be subject to the AWS Privacy Policy. To opt out of this feature, modify the AWS CloudFormation template mapping section as follows:

```
Solution:
  Data:
   SendAnonymousUsageData: "Yes"
```

to

```
Solution:
  Data:
```

```
    SendAnonymousUsageData: "No"
```

Source Code

You can visit our GitHub repository to download the templates and scripts for this solution, and to share your customizations with others.

Document Revisions

Date	Change
September 2016	Initial release
January 2017	Clarification on IP address limits in this solution
March 2017	Additional guidance on creating a cache behavior; updated URLs for AWS Security Blog posts
June 2017	Added ALB support and updated product limits
November 2017	Added rate-based rule support for HTTP flood protection; additional links for storing resource access logs
January 2018	Updated content on regional availability of AWS WAF for Application Load Balancers
December 2018	Added IPv6 Support, expanded CIDR ranges, and added a monitoring dashboard
April 2019	AWS WAF logs integration, Amazon Athena integration, and added a configurable log parser
December 2019	Added information on support for Node.js update
February 2020	Bug fixes and update to the RequestThreshold parameter
June 2020	Added Athena cost optimization using partitioning; updated README instruction; fixed a potential DoS issue within Bad Bots X-Forward-For header
July 2020	Upgrade from AWS WAF Classic to AWS WAFV2 service API

Notices

This document is provided for informational purposes only. It represents AWS current product offerings and practices as of the date of issue of this document, which are subject to change without notice. Customers are responsible for making their own independent assessment of the information in this document and any use of AWS products or services, each of which is provided "as is" without warranty of any kind, whether express or implied. This document does not create any warranties, representations, contractual commitments, conditions or assurances from AWS, its affiliates, suppliers or licensors. 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.