



Developer Guide

AWS Network Firewall



AWS Network Firewall: Developer Guide

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What is Network Firewall?	1
Network Firewall AWS resources	2
Network Firewall concepts	2
Accessing Network Firewall	3
Regions and endpoints for Network Firewall	4
Pricing for Network Firewall	5
Network Firewall quotas	5
Network Firewall additional resources	5
How Network Firewall works	6
Firewall components	7
High-level steps for implementation	7
Firewall behavior	8
Stateless and stateful rules engines	9
How Network Firewall filters network traffic	11
Route table configurations	12
Avoiding asymmetric routing with AWS Network Firewall	13
Architecture and routing examples	14
Single zone internet gateway	15
Multi zone internet gateway	19
Internet gateway and NAT gateway	22
Setting up	24
Sign up for an AWS account	24
Create a user with administrative access	25
Setting up tool access	26
Getting started with Network Firewall	28
Before you begin	28
Step 1: Create rule groups	29
Step 2: Create a firewall policy	30
Step 3: Create a firewall	31
Step 4: Update Amazon VPC route tables	32
Step 5: Remove the firewall and clean up your resources	33
Configuring your VPC	35
VPC subnets	35
VPC route tables	36

Transit gateway attachments	37
Firewalls	38
Firewall settings	38
Managing your firewall	39
Creating a firewall	40
Updating a firewall	42
Deleting a firewall	43
Troubleshooting firewall endpoint failures	43
Firewall policies	48
Firewall policy settings	48
Capacity limitations	50
Stream exception policy options	50
Managing your firewall policy	51
Creating a firewall policy	52
Updating a firewall policy	55
Deleting a firewall policy	56
Rule groups	57
Using AWS managed rule groups	58
Working with managed rule groups in Network Firewall	58
Domain and IP managed rule groups	65
Threat signature managed rule groups	67
Disclaimer for AWS managed rule groups in AWS Network Firewall	78
Managing your own rule groups	78
Common rule group settings	79
Setting rule group capacity	80
Defining rule actions	81
Working with stateful rule groups	84
Working with stateless rule groups	121
Resource groups	129
Resource group settings	129
Creating a resource group	130
Updating a resource group	131
Deleting a resource group	132
TLS inspection configurations	134
Considerations when working with TLS inspection configurations	134
Logging for TLS inspection in AWS Network Firewall	138

Using SSL/TLS certificates with TLS inspection configurations	141
General requirements	141
Server certificates - Inbound SSL/TLS inspection	142
CA certificate - Outbound SSL/TLS inspection	143
Checking certificate revocation status	143
TLS inspection configuration settings	145
Managing your TLS inspection configuration	146
Creating a TLS inspection configuration	147
Updating a TLS inspection configuration	149
Deleting a TLS inspection configuration	150
Sharing AWS Network Firewall firewall policies and rule groups	151
Prerequisites for sharing firewall policies and rule groups	151
Related services	152
Sharing across Availability Zones	152
Sharing a firewall policy or rule group	152
Unsharing a shared firewall policy or rule group	153
Security in Network Firewall	154
Data protection	155
Encryption at rest	156
Identity and Access Management	167
Audience	168
Authenticating with identities	169
Managing access using policies	172
How AWS Network Firewall works with IAM	174
Identity-based policy examples	181
Resource-based policy examples	184
AWS managed policies	185
Troubleshooting	187
Using service-linked roles	189
AWS logging and monitoring tools	195
Compliance validation and security best practices for Network Firewall	196
Resilience	197
Infrastructure security	197
VPC endpoints	197
Considerations	198
Create an interface endpoint	198

Create a VPC endpoint policy	199
Example VPC endpoint policy	199
Logging and monitoring	200
Logging network traffic	201
Contents of a Network Firewall log	202
Timing of log delivery	204
Permissions to configure logging	205
Pricing for logging	205
Firewall logging destinations	206
Logging with server-side encryption and customer-provided keys	214
Updating a firewall's logging configuration	216
Logging calls to the API with AWS CloudTrail	217
AWS Network Firewall information in CloudTrail	217
CloudTrail log file examples	218
Metrics in CloudWatch	223
Metrics	224
Dimensions	228
Troubleshooting	230
General issues	231
Firewall endpoint creation or deletion fails	231
Availability Zone is unsupported	231
How do I check if I have asymmetric routing?	231
I'm using Network Firewall with AWS Transit Gateway and Network Firewall is dropping traffic	235
High latency and intermittent packet drops when traffic passes through Network Firewall	236
Logging	238
My firewall isn't logging all traffic that matches pass action rules	238
I don't see Alert logs that match drop action rules in my rule group	239
I don't see any TLS logs even though I have TLS logging enabled	239
Rules	239
Rules with the HOME_NET variable are not working as expected with managed rule groups	240
I created a rule to allow only outbound traffic from HOME_NET to EXTERNAL_NET, but EXTERNAL_NET was also able to initiate a connection back to HOME_NET. How do I prevent this from happening?	240

I'm using strict ordering, but stateful rules near the bottom of my ruleset appear to be handling traffic before rules near the top of my ruleset	241
I've configured a drop action rule but traffic still goes through the firewall	242
TLS inspection	242
Outbound TLS - Blocked connections to servers with revoked certificates	242
Outbound TLS - Passing traffic for specific target server with revoked certificates by adjusting scope	245
Outbound TLS - Connection issues with service endpoints	247
TLS - Connections dropping or resetting	247
Resource tagging	249
Supported resources in Network Firewall	249
Tag naming and usage conventions	249
Managing tags	250
Using the Network Firewall REST API	251
Making HTTPS requests to Network Firewall	251
Request URI	252
HTTP headers	252
HTTP request body	253
HTTP responses	254
Error responses	254
Authenticating requests	255
Quotas	256
Resources	259
AWS resources	259
Document history	261

What is AWS Network Firewall?

AWS Network Firewall is a stateful, managed, network firewall and intrusion detection and prevention service for your virtual private cloud (VPC) that you create in Amazon Virtual Private Cloud (Amazon VPC). With Network Firewall, you can filter traffic at the perimeter of your VPC. This includes filtering traffic going to and coming from an internet gateway, NAT gateway, or over VPN or AWS Direct Connect.

Network Firewall uses the open source intrusion prevention system (IPS), Suricata, for stateful inspection, and supports Suricata compatible rules. AWS Network Firewall supports Suricata version 7.0. For more information, see [Working with stateful rule groups in AWS Network Firewall](#) and the [Suricata website](#).

You can use Network Firewall to monitor and protect your Amazon VPC traffic in a number of ways, including the following:

- Pass traffic through only from known AWS service domains or IP address endpoints, such as Amazon S3.
- Use custom lists of known bad domains to limit the types of domain names that your applications can access.
- Perform deep packet inspection on traffic entering or leaving your VPC.
- Use stateful protocol detection to filter protocols like HTTPS, independent of the port used.

To enable Network Firewall for your VPC, you perform steps in both Amazon VPC and in Network Firewall. For information about managing your Amazon Virtual Private Cloud VPC, see the [Amazon Virtual Private Cloud User Guide](#). For more information about how Network Firewall works, see [How AWS Network Firewall works](#).

Network Firewall is supported by AWS Firewall Manager. You can use Firewall Manager to centrally configure and manage your firewalls across your accounts and applications in AWS Organizations. You can manage firewalls for multiple accounts using a single account in Firewall Manager. For more information, see [AWS Firewall Manager](#) in the *AWS WAF, AWS Firewall Manager, and AWS Shield Advanced Developer Guide*.

Topics

- [AWS Network Firewall AWS resources](#)
- [AWS Network Firewall concepts](#)

- [Accessing AWS Network Firewall](#)
- [Regions and endpoints for AWS Network Firewall](#)
- [Pricing for AWS Network Firewall](#)
- [AWS Network Firewall quotas](#)
- [AWS Network Firewall additional resources](#)

AWS Network Firewall AWS resources

Network Firewall manages the following AWS resource types:

- **Firewall** – Provides traffic filtering logic for the subnets in a VPC.
- **FirewallPolicy** – Defines rules and other settings for a firewall to use to filter incoming and outgoing traffic in a VPC.
- **RuleGroup** – Defines a set of rules to match against VPC traffic, and the actions to take when Network Firewall finds a match. Network Firewall uses stateless and stateful rule group types, each with its own Amazon Resource Name (ARN).

AWS Network Firewall concepts

AWS Network Firewall is a firewall service for Amazon Virtual Private Cloud (Amazon VPC). For information about managing your Amazon Virtual Private Cloud VPC, see the [Amazon Virtual Private Cloud User Guide](#).

The following are the key concepts for Network Firewall:

- **Virtual private cloud (VPC)** – A virtual network dedicated to your AWS account.
- **Internet gateway** – A gateway that you attach to your VPC to enable communication between resources in your VPC and the internet.
- **Subnet** – A range of IP addresses in your VPC. Network Firewall creates firewall endpoints in subnets inside your VPC, to filter network traffic. In a VPC architecture that uses Network Firewall, the firewall endpoints sit between your protected subnets and locations outside your VPC.
- **Firewall subnet** – A subnet that you've designated for exclusive use by Network Firewall for a firewall endpoint. A firewall endpoint can't filter traffic coming into or going out of the subnet in which it resides, so don't use your firewall subnets for anything other than Network Firewall.

- **Route table** – A set of rules, called routes, that are used to determine where network traffic is directed. You modify your VPC route tables in Amazon VPC to direct traffic through your firewalls for filtering.
- **Network Firewall *firewall*** – An AWS resource that provides traffic filtering logic for the subnets in a VPC.
- **Network Firewall *firewall policy*** – An AWS resource that defines rules and other settings for a firewall to use to filter incoming and outgoing traffic in a VPC.
- **Network Firewall *rule group*** – An AWS resource that defines a set of rules to match against VPC traffic, and the actions to take when Network Firewall finds a match.
- **Stateless rules** – Criteria for inspecting a single network traffic packet, without the context of the other packets in the traffic flow, the direction of flow, or any other information that's not provided by the packet itself.
- **Stateful rules** – Criteria for inspecting network traffic packets in the context of their traffic flow.

Accessing AWS Network Firewall

You can create, access, and manage your firewall, firewall policy, and rule group resources in Network Firewall using any of the following methods:

- **AWS Management Console** – Provides a web interface for managing the service. The procedures throughout this guide explain how to use the AWS Management Console to perform tasks for Network Firewall. You can access the AWS Management Console at <https://aws.amazon.com/console>. To access Network Firewall using the console:

```
https://<region>.console.aws.amazon.com/network-firewall/home
```

- **AWS Command Line Interface (AWS CLI)** – Provides commands for a broad set of AWS services, including Network Firewall. The CLI is supported on Windows, macOS, and Linux. For more information, see the [AWS Command Line Interface User Guide](#). To access Network Firewall using the CLI endpoint:

```
aws network-firewall
```

- **AWS Network Firewall API** – Provides a RESTful API. The REST API requires you to handle connection details, such as calculating signatures, handling request retries, and handling errors.

For more information, see [AWS APIs](#) and the [AWS Network Firewall API Reference](#). To access Network Firewall, use the following REST API endpoint:

```
https://network-firewall.<region>.amazonaws.com
```

- **AWS SDKs** – Provide language-specific APIs. If you're using a programming language that AWS provides an SDK for, you can use the SDK to access AWS Network Firewall. The SDKs handle many of the connection details, such as calculating signatures, handling request retries, and handling errors. They integrate easily with your development environment, and provide easy access to Network Firewall commands. For more information, see [Tools for Amazon Web Services](#).
- **AWS CloudFormation** – Helps you model and set up your Amazon Web Services resources so that you can spend less time managing those resources and more time focusing on your applications that run in AWS. You create a template that describes all the AWS resources that you want and AWS CloudFormation takes care of provisioning and configuring those resources for you. For more information, see [Network Firewall resource type reference](#) in the *AWS CloudFormation User Guide*.
- **AWS Tools for Windows PowerShell** – Let developers and administrators manage their AWS services and resources in the PowerShell scripting environment. For more information, see the [AWS Tools for Windows PowerShell User Guide](#).

Regions and endpoints for AWS Network Firewall

To view the complete list of AWS Regions where Network Firewall is available, see [Service endpoints and quotas](#) in the *AWS General Reference*.

IPv4 endpoints

```
https://network-firewall.<region>.amazonaws.com
```

Dual-stack (IPv4 and IPv6) endpoints

Dual-stack endpoints support both IPv4 and IPv6 traffic. When you make a request to a dual-stack endpoint, the endpoint URL resolves to an IPv6 or IPv4 address, depending on the protocol used by your network and client.

```
https://network-firewall.<region>.api.aws
```

Pricing for AWS Network Firewall

For detailed information about pricing for Network Firewall, see [AWS Network Firewall pricing](#).

Some configurations can incur additional costs, on top of the basic costs for using Network Firewall. For example, if you use a firewall endpoint in one Availability Zone to filter traffic from another zone, you can incur cross-zone traffic charges. If you enable logging, you incur additional charges according to factors such as the logging destination that you use and the amount of traffic that you choose to log.

AWS Network Firewall quotas

AWS Network Firewall defines maximum settings and other quotas on the number of Network Firewall resources that you can use. You can request an increase for some of these quotas. For more information, see [AWS Network Firewall quotas](#).

AWS Network Firewall additional resources

To get a hands-on introduction to AWS Network Firewall, complete [Getting started with AWS Network Firewall](#).

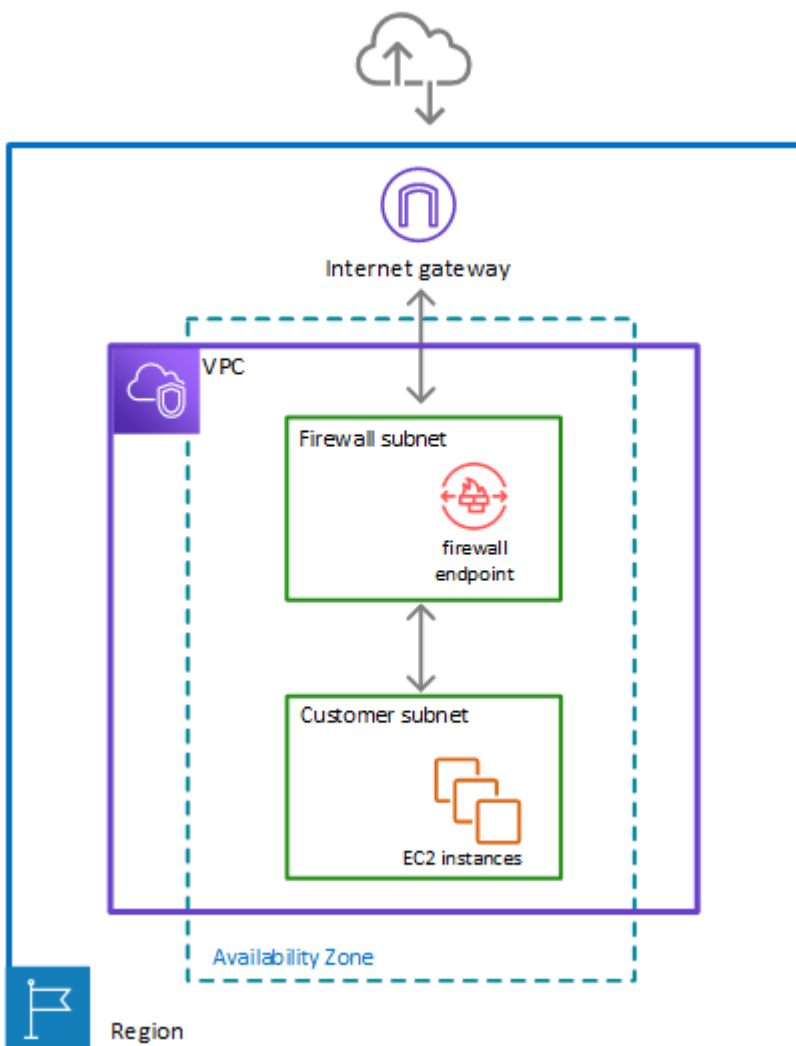
Use the following resources to get additional information and guidance for using AWS Network Firewall.

- [AWS discussion forums](#) – A community-based forum for discussing technical questions related to this and other AWS services.
- [Getting started resource center](#) – Information to help you get started building on AWS.
- [AWS Support center](#) – The home page for AWS Support.
- [Contact Us](#) – A central contact point for inquiries concerning billing, accounts, and events.

How AWS Network Firewall works

AWS Network Firewall is a stateful, managed, network firewall and intrusion detection and prevention service for Amazon Virtual Private Cloud (Amazon VPC). You can combine Network Firewall with services and components that you use with your VPC, for example an internet gateway, a NAT gateway, a VPN, or a transit gateway. For information about managing your Amazon Virtual Private Cloud VPC, see the [Amazon Virtual Private Cloud User Guide](#). You need a VPC to use Network Firewall.

The firewall protects the subnets within your VPC by filtering traffic going between the subnets and locations outside of your VPC. The following example figure depicts the placement of a firewall in a very simple architecture.



To enable the firewall's protection, you modify your Amazon VPC route tables to send your network traffic through the Network Firewall firewall endpoints. For information about managing route tables for your VPC, see [Route tables](#) in the *Amazon Virtual Private Cloud User Guide*.

Firewall components in AWS Network Firewall

The AWS Network Firewall firewall runs stateless and stateful traffic inspection rules engines. The engines use rules and other settings that you configure inside a firewall policy.

You install the firewall endpoints on a per-Availability Zone basis in your VPC. For each Availability Zone where you want an endpoint, you choose a subnet to host it. The firewall endpoint can protect any subnet in your VPC except for the one in which it's located.

You manage Network Firewall firewalls with the following central components.

- **Rule group** – Holds a reusable collection of criteria for inspecting traffic and for handling packets and traffic flows that match the inspection criteria. For example, you can choose to drop or pass a packet or all packets in a traffic flow based on the inspection criteria. Some rule groups fully define the behavior and some use lower-level rules that provide more detail. Rule groups are either stateless or stateful. For more information about rule groups and rules, see [Rule groups in AWS Network Firewall](#).
- **Firewall policy** – Defines a reusable set of stateless and stateful rule groups, along with some policy-level behavior settings. The firewall policy provides the network traffic filtering behavior for a firewall. You can use a single firewall policy in multiple firewalls. For more information about firewall policies, see [Firewall policies in AWS Network Firewall](#).
- **Firewall** – Connects the inspection rules in the firewall policy to the VPC that the rules protect. Each firewall requires one firewall policy. The firewall additionally defines settings like how to log information about your network traffic and the firewall's stateful traffic filtering. For more information about firewalls, see [Firewalls in AWS Network Firewall](#).

High-level steps for implementing AWS Network Firewall

To install and use an AWS Network Firewall firewall in your Amazon Virtual Private Cloud VPC, you configure the firewall components and your VPC's subnets and route tables in the following high-level steps.

- **Configure the VPC subnets for your firewall endpoints** – In your VPC, in each Availability Zone where you want a firewall endpoint, create a subnet specifically for use by Network Firewall. A firewall endpoint can't protect applications that run in the same subnet, so reserve these subnets for exclusive use by the firewall. The subnets that you use for your firewall endpoints must belong to a single AWS Region and must be in different Availability Zones within the Region. Network Firewall is available in the Regions listed at [AWS service endpoints](#).

For information about managing subnets in your VPC, see [VPCs and subnets](#) in the *Amazon Virtual Private Cloud User Guide*.

- **Create the firewall** – Create a Network Firewall firewall and provide it with the specifications for each of your firewall subnets. Network Firewall creates a firewall endpoint in each subnet that you specify, available to monitor and protect the resources for the subnets whose traffic you send through it.
- **Configure the firewall policy** – Define the firewall policy for your firewall by specifying its rule groups and other behavior that you want the firewall to provide.
- **Modify your VPC route tables to include the firewall** – Using Amazon VPC ingress routing enhancements, change your routing tables to route traffic through the Network Firewall firewall. These changes must insert the firewall between the subnets that you want to protect and outside locations. The exact routing that you need to do depends on your architecture and its components.

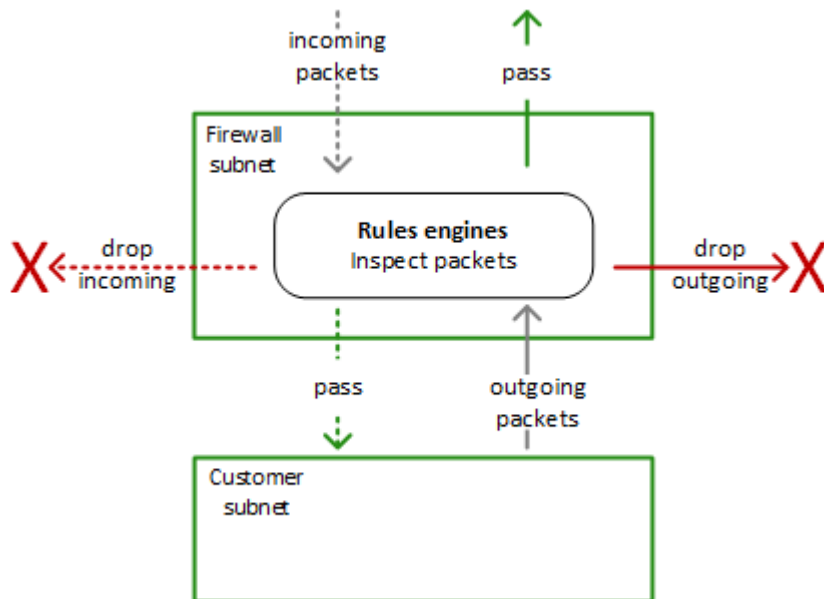
For information about managing route tables for your VPC, see [Route tables](#) in the *Amazon Virtual Private Cloud User Guide*.

Firewall behavior in AWS Network Firewall

AWS Network Firewall provides virtual firewalls dedicated to protecting your VPC from attacks. You define and create a firewall, then use it to monitor and protect your subnets. The firewall monitors incoming and outgoing traffic and allows it to pass or drops it, according to your specifications. The firewall only allows packets to pass that pass inspection.

Network Firewall monitors and controls traffic to and from your protected subnets

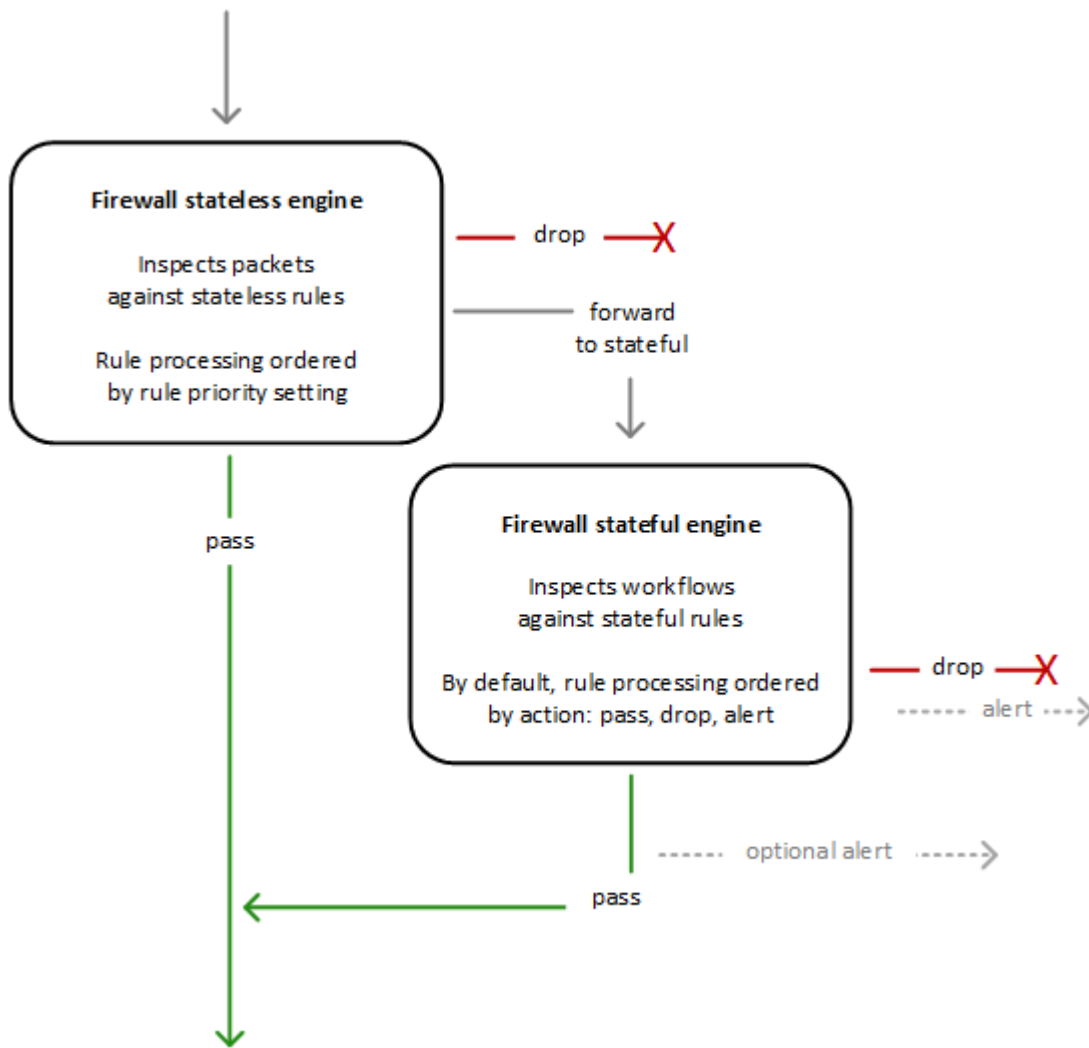
The following figure shows the basic interaction of your firewall with traffic coming into your customer subnet and with traffic going out from your customer subnet.



Network Firewall stateless and stateful rules engines

AWS Network Firewall uses two rules engines to inspect packets. The engines inspect packets according to the rules that you provide in your firewall policy.

The following figure shows the processing flow for packets coming through the firewall. First the stateless engine inspects the packet against the configured stateless rules. Depending on the packet settings, the stateless inspection criteria, and the firewall policy settings, the stateless engine might drop a packet, pass it through to its destination, or forward it to the stateful rules engine. The stateful engine inspects packets in the context of their traffic flow, using the configured stateful rules. The stateful engine either drops packets or passes them to their destination. Stateful engine activities are recorded in the firewall's logs, if logging is configured. The stateful engine sends alerts for dropped packets and can optionally send them for passed packets.



The stateless and stateful rules inspection engines operate in different ways:

- **Stateless rules engine** – Inspects each packet in isolation, without regard to factors such as the direction of traffic, or whether the packet is part of an existing, approved connection. This engine prioritizes the speed of evaluation. It takes rules with standard network connection attributes. The engine processes your rules in the order that you prioritize them and stops processing when it finds a match.

Network Firewall stateless rules are similar in behavior and use to Amazon VPC network access control lists (ACLs).

- **Stateful rules engine** – Inspects packets in the context of their traffic flow, allows you to use more complex rules, and allows you to log network traffic and to log Network Firewall firewall alerts on traffic. Stateful rules consider traffic direction. The stateful rules engine might delay packet delivery in order to group packets for inspection. By default, the stateful rules engine

processes your rules in the order of their action setting, with pass rules processed first, then drop, then alert. The engine stops processing when it finds a match.

The stateful engine takes rules that are compatible with Suricata, an open source intrusion prevention system (IPS). Suricata provides a standard rule-based language for stateful network traffic inspection. For more information about Suricata, see [Working with stateful rule groups in AWS Network Firewall](#) and the [Suricata website](#). AWS Network Firewall supports Suricata version 7.0.

Network Firewall stateful rules are similar in behavior and use to Amazon VPC security groups. By default, the stateful rules engine allows traffic to pass, while the security groups default is to deny traffic.

Whether you use only one of these engines or a combination depends on your specific use case.

How AWS Network Firewall filters network traffic

When AWS Network Firewall inspects a packet, it evaluates the packet against the rules in the policy's stateless rule groups first, using the stateless rules engine. Then, depending on that inspection and on other settings in the policy, it might evaluate the packets against the rules in the policy's stateful rule groups, using the stateful rules engine.

1. Stateless rules engine

Network Firewall evaluates each packet against the firewall policy's stateless rules until it finds a match or exhausts all of the stateless rules. Network Firewall evaluates the rule groups in the order that they are prioritized in the policy, starting from the lowest setting. Within each rule group, Network Firewall evaluates the rules in the order that they are prioritized in the rule group, starting from the lowest setting. When you create a stateless rule group, you set the priority of the rules in the rule group. When you create a firewall policy, you set the priority of the stateless rule groups in the policy. For more information, see [Working with stateless rule groups in AWS Network Firewall](#) and [Firewall policies in AWS Network Firewall](#).

When Network Firewall finds a match, it handles the packet according to the matching rule's configuration. You configure a stateless rule to pass the packet through, drop it, or forward it to your stateful rules. Additionally, you can configure a stateless rule to perform a custom action, for example you can publish metrics for the packet to Amazon CloudWatch. For more information, see [Defining rule actions in AWS Network Firewall](#).

2. Default stateless rule actions

If a packet doesn't match any stateless rule, Network Firewall performs the firewall policy's default stateless rule action for full packet or UDP packet fragment, depending on the packet type. Network Firewall only applies the fragment action setting to UDP packet fragments, and silently drops packet fragments for other protocols. The options for these actions settings are the same as for stateless rules. For more information, see [Defining rule actions in AWS Network Firewall](#).

3. Stateful rules engine

When Network Firewall forwards a packet to the stateful engine for inspection, it inspects each packet against the stateful rule groups, in the context of the packet's traffic flow. You can configure a stateful rule to pass the packet through, with or without an alert, or drop it and send an alert. Alerts require logging to be configured for the firewall.

The Suricata stateful rules engine controls how the stateful rules in your firewall policy are processed. The engine evaluates the packet's traffic flow against the conditions in the policy's stateful rules until it finds a match or exhausts all of the rules. When the engine finds a match, it handles the packet according to the rule's configuration. By default, the Suricata stateful rules engine orders rule processing according to the rule action setting, processing first the rules with pass action, then drop, then alert. For more information, see [Managing evaluation order for Suricata compatible rules in AWS Network Firewall](#) and the [Suricata Action-order documentation](#).

Depending on the Suricata compatible rules that you provide, the stateful engine might perform deep packet inspection of your traffic. Deep packet inspection works on the payload data within your packets, rather than on the header information.

For more information about stateful rules, see [Rule groups in AWS Network Firewall](#).

Route table configurations for AWS Network Firewall

To include the Network Firewall firewall in your Amazon Virtual Private Cloud VPC, you modify the VPC route tables so that the traffic that you want the firewall to filter passes through the firewall endpoints. Exactly how you do this depends on your architecture and the traffic that you want to filter. For example, to filter all traffic between an internet gateway and your customer subnets, you redirect incoming traffic from the internet gateway and outgoing traffic from the customer subnets through the firewall endpoint.

For information about managing route tables for your VPC, see [Route tables](#) in the *Amazon Virtual Private Cloud User Guide*.

For descriptions of common architectures for AWS Network Firewall, with example route table configurations, see [AWS Network Firewall example architectures with routing](#).

Avoiding asymmetric routing with AWS Network Firewall

Network Firewall doesn't support asymmetric routing. Use the guidance that follows to prevent asymmetric routing in your network traffic where you use Network Firewall.

In Network Firewall, asymmetric routing occurs when both request network traffic and its related response network traffic are not routed to the same Network Firewall endpoint. In order for Network Firewall to properly process traffic, the traffic must be routed to the Network Firewall endpoint in both directions.

The following are considerations to keep in mind to prevent asymmetric routing:

- **Centralized deployment model** - If your firewall uses a centralized deployment model:
 - On the Transit Gateway which is on the inspection VPC of the firewall, use the Transit Gateway appliance mode to keep the traffic request and response flows on the same Network Firewall endpoint. For information about configuring the Transit Gateway appliance mode, see [AWS Transit Gateway traffic flow and asymmetric routing](#).
 - Configure your Transit Gateway route tables to route both forward and return direction traffic via your firewall attachment.
- **Decentralized deployment model** - If your firewall is deployed in a decentralized deployment model inspecting internet-bound traffic from an internet gateway, use a route table with an Internet Gateway [edge association](#) to route inbound traffic through the Network Firewall endpoint, in addition to an outbound route in the application subnet.
- **NAT gateway** - If Network Firewall is downstream of your Network address translation (NAT) Gateway, make sure that the NAT gateway's subnet routes traffic through the Network Firewall endpoint. For information about using NAT gateway with Network Firewall, see the following resources:
 - [Architecture with an internet gateway and a NAT gateway using AWS Network Firewall](#)
 - [Using the NAT gateway with AWS Network Firewall for centralized egress](#)
 - [How do I set up AWS Network Firewall with a NAT gateway?](#)
- **Stateless rules** - If your Network Firewall firewall uses stateless rules:
 - Be aware that unidirectional pass rules can create asymmetric forwarding when the policy's stateless default action is **forward to stateful rules**.

- Ensure that your stateless rules forward traffic symmetrically to the stateful engine using the **forward to stateful rule groups** action. Often this means writing pairs of rules to match both forward and return direction traffic. For information about the forward to stateful rule groups option, see [Creating a firewall policy in AWS Network Firewall](#). The following example shows a pair of rules that match both forward and return direction traffic:

Priority	Protocol	Source	Destination	Source port range	Destination port range	Action	Custom action	Masks	Flags
1	TCP	10.10.10.0/24	10.20.20.0/24	0:65535	80	Forward	-	-	-
2	TCP	10.20.20.0/24	10.10.10.0/24	80	0:65535	Forward	-	-	-

AWS Network Firewall example architectures with routing

This section provides a high-level view of simple architectures that you can configure with AWS Network Firewall and shows example route table configurations for each. For additional information and examples, see [Deployment models for AWS Network Firewall](#).

Note

For information about managing route tables for your VPC, see [Route tables](#) in the *Amazon Virtual Private Cloud User Guide*.

Unsupported architectures

The following lists architectures and traffic types that Network Firewall doesn't support:

- VPC peering.
- Inspection of AWS Global Accelerator traffic.
- Inspection of AmazonProvidedDNS traffic for Amazon EC2.

Topics

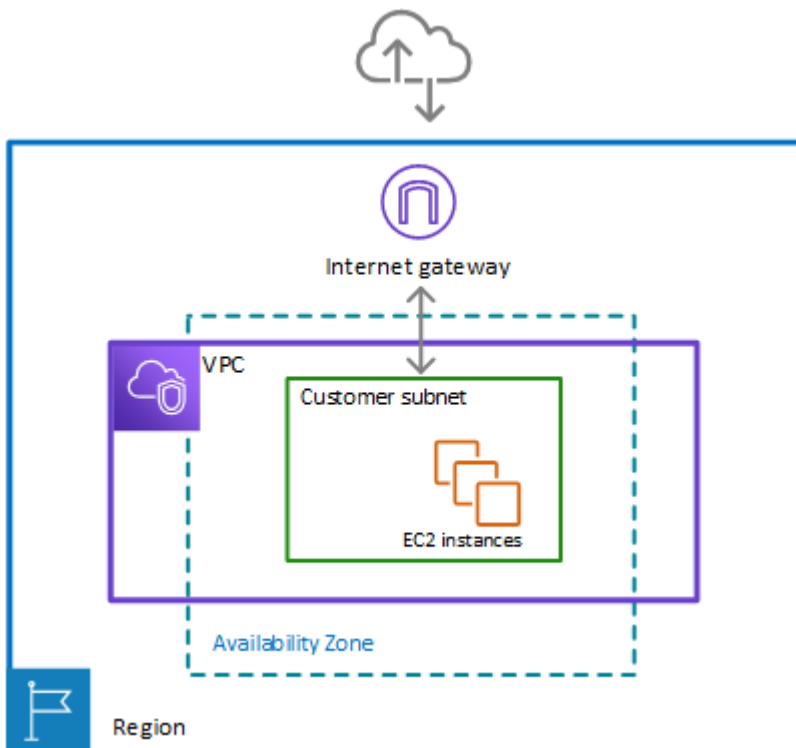
- [Simple single zone architecture with an internet gateway using AWS Network Firewall](#)
- [Multi zone architecture with an internet gateway using AWS Network Firewall](#)
- [Architecture with an internet gateway and a NAT gateway using AWS Network Firewall](#)

Simple single zone architecture with an internet gateway using AWS Network Firewall

This topic provides a high-level view of a simple VPC configuration using an internet gateway and AWS Network Firewall. It describes the basic route table modifications that are required to use the firewall.

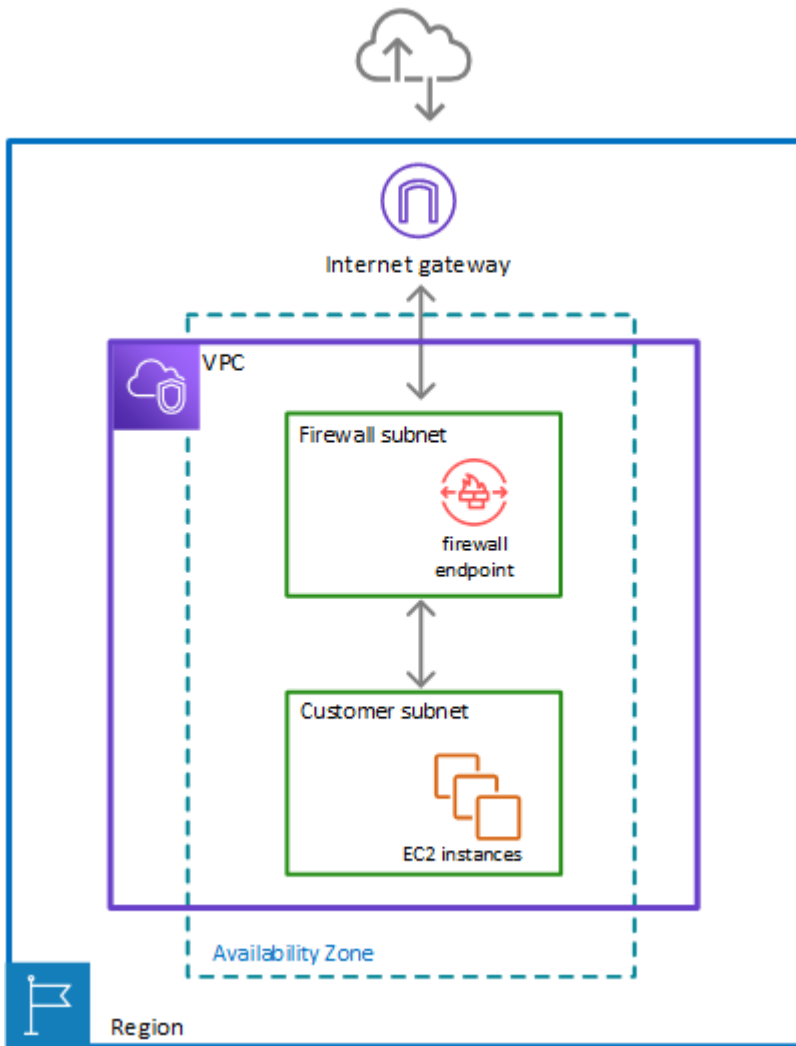
Single zone architecture with internet gateway and no firewall

The following figure depicts a simple VPC configuration with a single customer subnet, and no firewall. The VPC has an internet gateway for internet access. All incoming and outgoing traffic routes through the internet gateway to the subnet.



Single zone architecture with internet gateway and the Network Firewall firewall

The following figure depicts a simple VPC configuration with the firewall and the subnet association in place. The VPC has an internet gateway for internet access. All incoming and outgoing traffic for the VPC routes through the firewall.



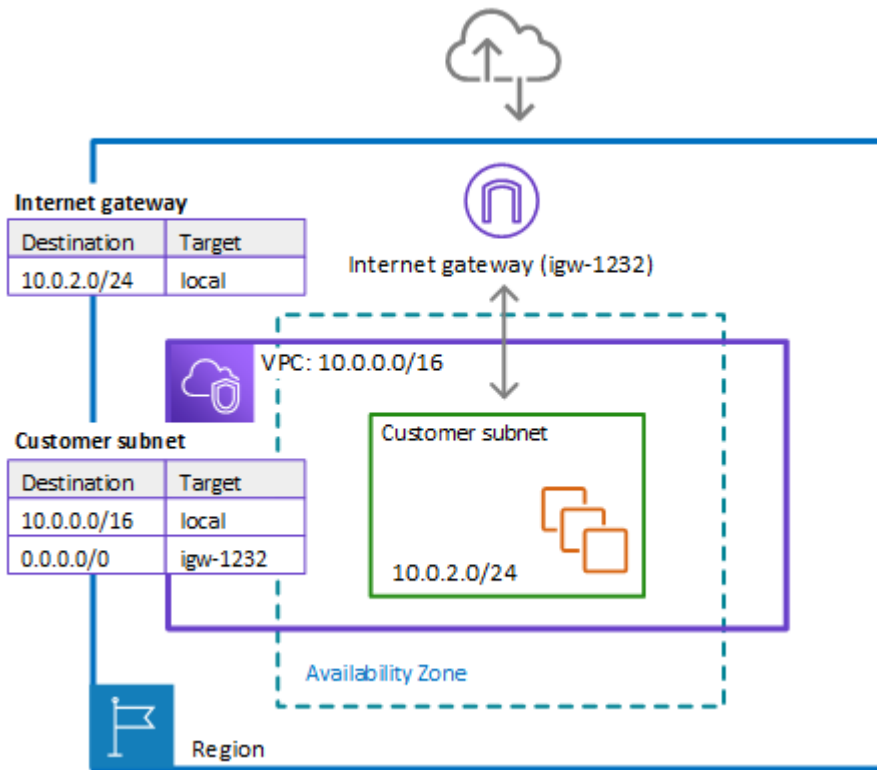
To include the firewall in your Amazon Virtual Private Cloud VPC, you need to modify the VPC route tables so that traffic between the customer subnets and the internet passes through the firewall, for both incoming and outgoing traffic.

Note

For information about managing route tables for your VPC, see [Route tables](#) in the *Amazon Virtual Private Cloud User Guide*.

Example route tables in the single zone architecture with no firewall

The following figure depicts the route tables that provide the correct flow of traffic for a single Availability Zone without a firewall:

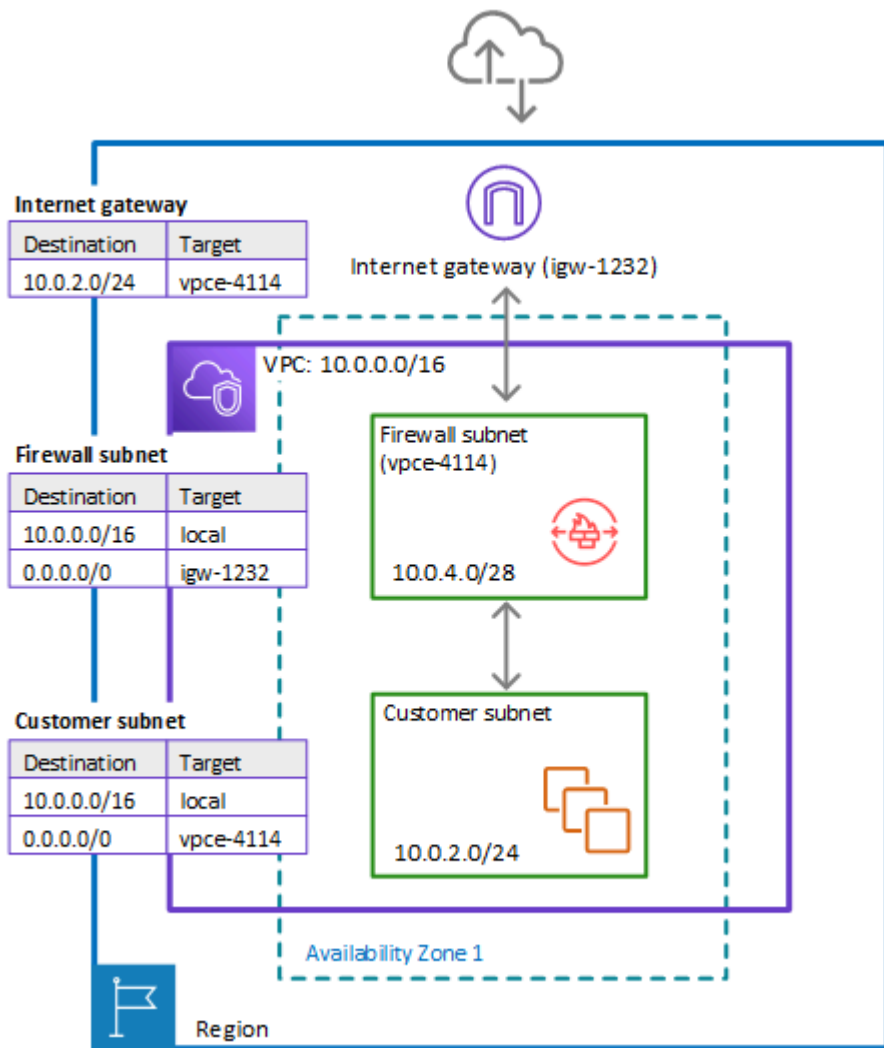


In the preceding figure, the route tables enforce the following traffic flows:

- **Internet gateway route table** – Routes traffic that's destined for the customer subnet (range 10.0.2.0/24) to local. The customer subnet shows the private IP address range behind the publicly assigned address. The subnet has public addresses assigned, which are either auto-generated or assigned via Elastic IP address. Within a VPC, only private IP addresses are used for communication.
- **Customer subnet route table** – Routes traffic that's destined for anywhere inside the VPC (10.0.0.0/16) to the local address. Routes traffic that's destined for anywhere else (0.0.0.0/0) to the internet gateway (igw-1232).

Example route tables in the single zone architecture with the firewall

The following figure depicts the same installation with the Network Firewall firewall added and the route tables changed to include the firewall. The route tables direct traffic between the customer subnet and the internet gateway through the firewall endpoint:



In the preceding figure, the route tables enforce the following traffic flows:

- **Internet gateway route table** – Routes traffic that's destined for the customer subnet (range 10.0.2.0/24) to the firewall subnet (named vpce-4114 in the figure). The customer subnet shows the private IP address range behind the publicly assigned address. The subnet has public addresses assigned, which are either auto-generated or assigned via Elastic IP address. Within a VPC, only private IP addresses are used for communication.
- **Firewall subnet route table** – Routes traffic that's destined for anywhere inside the VPC (10.0.0.0/16) to the local address. Routes traffic that's destined for anywhere else (0.0.0.0/0) to the internet gateway (igw-1232).
- **Customer subnet route table** – Routes traffic that's destined for anywhere inside the VPC (10.0.0.0/16) to the local address. Routes traffic that's destined for anywhere else (0.0.0.0/0) to the firewall subnet (vpce-4114).

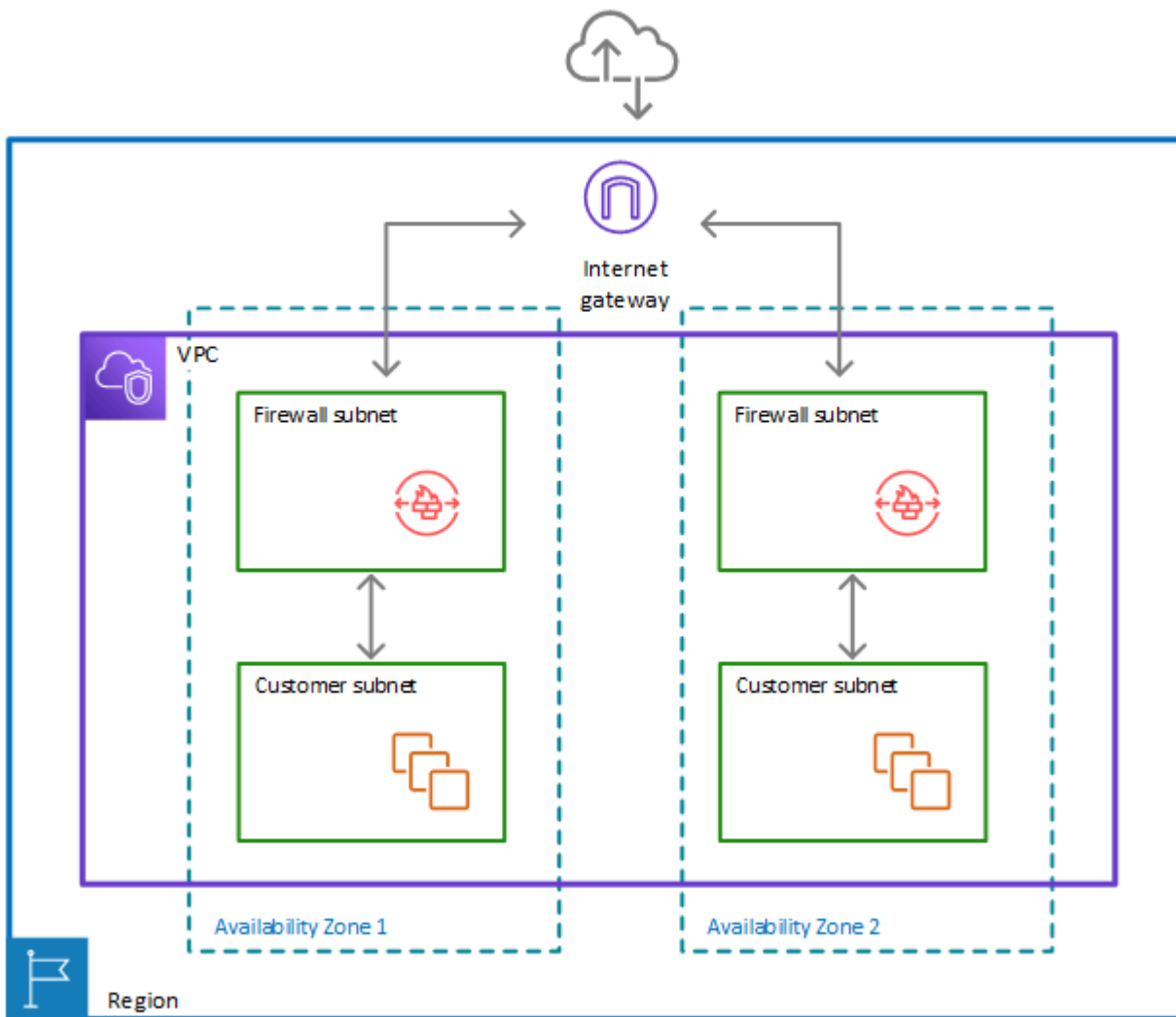
Before the firewall inclusion, the customer subnet route table routed the `0.0.0.0/0` traffic to `igw-1232`.

Multi zone architecture with an internet gateway using AWS Network Firewall

This topic provides a high-level view of a simple two zone VPC configuration using an internet gateway and AWS Network Firewall. It describes the basic route table modifications that are required to use the Network Firewall firewall.

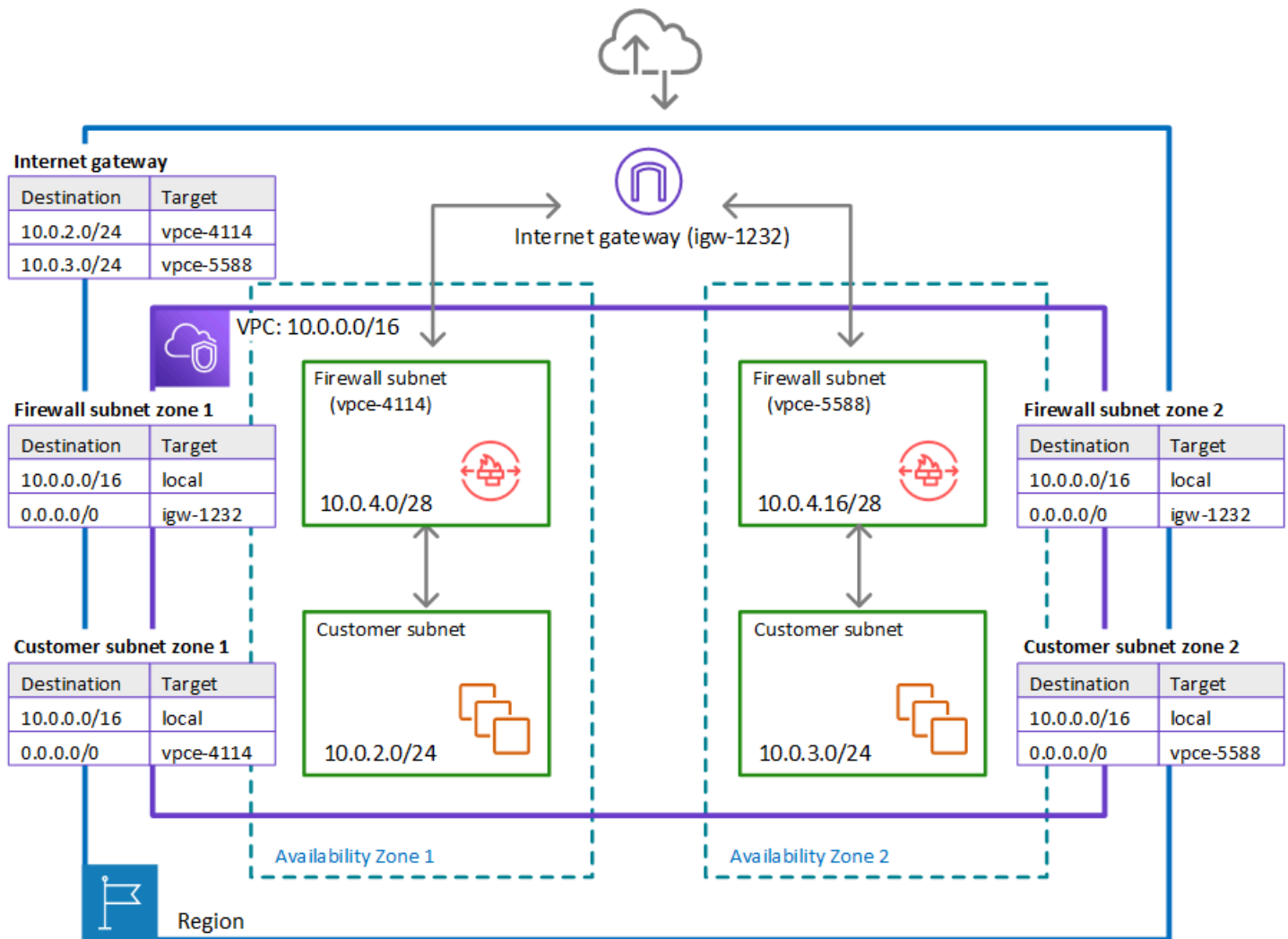
Two zone architecture with internet gateway and the Network Firewall firewall

The following figure depicts a Network Firewall configuration for a VPC that spans multiple Availability Zones. In this case, each Availability Zone that the VPC spans has a firewall subnet and a customer subnet. The VPC has an internet gateway for internet access. All incoming traffic for the VPC routes to the firewall in the same Availability Zone as the destination customer subnet. All outgoing traffic routes through the firewalls.



Route tables in the two zone architecture with the firewall

The following figure depicts a VPC configuration with two Availability Zones. Each zone has its own Network Firewall firewall, which provides monitoring and protection for the subnets in the zone. You can expand this configuration to any number of zones in your VPC.



In the preceding figure, the route tables enforce similar traffic flows to the single Availability Zone model, with the primary difference being the splitting of incoming traffic by the internet gateway, to accommodate the two different customer subnets:

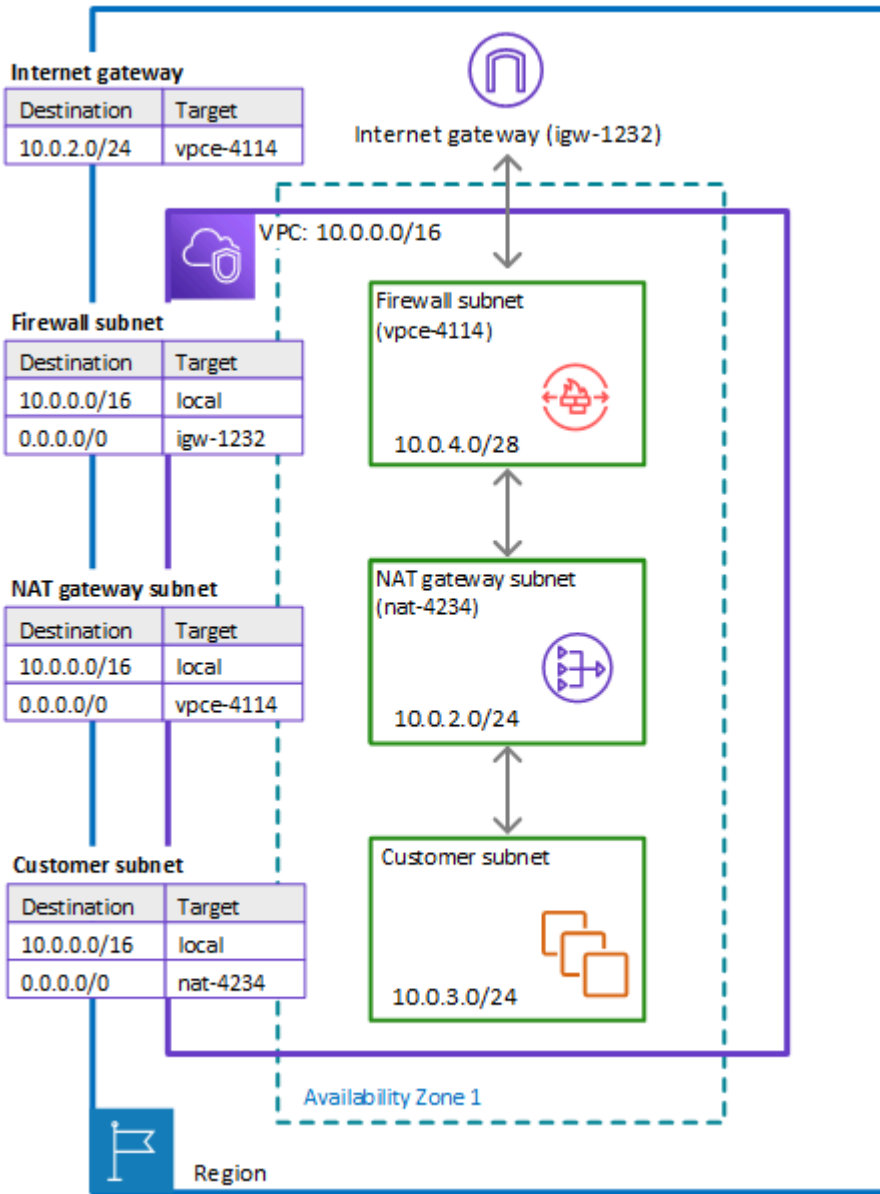
- **Internet gateway route table** – Routes traffic that's destined for each customer subnet (range 10.0.2.0/24 or 10.0.3.0/24) to the firewall subnet in the same Availability Zone (vpce-4114 or vpce-5588, respectively).
- **Firewall subnet route tables** – Route traffic that's destined for anywhere inside the VPC (10.0.0.0/16) to the local address. Route traffic that's destined for anywhere else (0.0.0.0/0) to the internet gateway (igw-1232). These are identical to the route table for the firewall subnet in the single Availability Zone.
- **Customer subnet route tables** – Route traffic that's destined for anywhere inside the VPC (10.0.0.0/16) to the local address. Route traffic that's destined for anywhere else (0.0.0.0/0)

to the firewall subnet in the same Availability Zone (vpce-4114 for zone AZ1 and vpce-5588 for zone AZ2).

Architecture with an internet gateway and a NAT gateway using AWS Network Firewall

You can add a network address translation (NAT) gateway to your AWS Network Firewall architecture, for the areas of your VPC where you need NAT capabilities. AWS provides NAT gateways decoupled from your other cloud services, so you can use it in your architecture only where you need it. This can help you reduce load and load costs. For information about NAT gateways, see [NAT gateways](#) in the *Amazon Virtual Private Cloud User Guide*.

The following figure depicts a VPC configuration for Network Firewall with an internet gateway and a NAT gateway.



Setting up AWS Network Firewall

This topic describes preliminary steps, such as getting an AWS account, to prepare you to use Network Firewall. You aren't charged to set up your account or for the other preliminary items. You are charged only for AWS services that you use.

Note

Network Firewall is a network traffic firewall for your Amazon Virtual Private Cloud VPCs. If you're already working with VPCs, the setup described here shouldn't be necessary.

After you complete these steps, see [Getting started with Network Firewall](#) to continue getting started with Network Firewall.

Topics

- [Sign up for an AWS account](#)
- [Create a user with administrative access](#)
- [Setting up tool access](#)

Sign up for an AWS account

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

1. Open <https://portal.aws.amazon.com/billing/signup>.
2. Follow the online instructions.

Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to a user, and use only the root user to perform [tasks that require root user access](#).

AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to <https://aws.amazon.com/> and choosing **My Account**.

Create a user with administrative access

After you sign up for an AWS account, secure your AWS account root user, enable AWS IAM Identity Center, and create an administrative user so that you don't use the root user for everyday tasks.

Secure your AWS account root user

1. Sign in to the [AWS Management Console](#) as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

For help signing in by using root user, see [Signing in as the root user](#) in the *AWS Sign-In User Guide*.

2. Turn on multi-factor authentication (MFA) for your root user.

For instructions, see [Enable a virtual MFA device for your AWS account root user \(console\)](#) in the *IAM User Guide*.

Create a user with administrative access

1. Enable IAM Identity Center.

For instructions, see [Enabling AWS IAM Identity Center](#) in the *AWS IAM Identity Center User Guide*.

2. In IAM Identity Center, grant administrative access to a user.

For a tutorial about using the IAM Identity Center directory as your identity source, see [Configure user access with the default IAM Identity Center directory](#) in the *AWS IAM Identity Center User Guide*.

Sign in as the user with administrative access

- To sign in with your IAM Identity Center user, use the sign-in URL that was sent to your email address when you created the IAM Identity Center user.

For help signing in using an IAM Identity Center user, see [Signing in to the AWS access portal](#) in the *AWS Sign-In User Guide*.

Assign access to additional users

1. In IAM Identity Center, create a permission set that follows the best practice of applying least-privilege permissions.

For instructions, see [Create a permission set](#) in the *AWS IAM Identity Center User Guide*.

2. Assign users to a group, and then assign single sign-on access to the group.

For instructions, see [Add groups](#) in the *AWS IAM Identity Center User Guide*.

Setting up tool access

The AWS Management Console includes a console for Network Firewall, but if you want to access Network Firewall programmatically or through the command line, the following documentation and tools will help you:

- If you want to call the Network Firewall API without handling low-level details like assembling raw HTTP requests, you can use an AWS SDK. The AWS SDKs provide functions and data types that encapsulate the functionality of Network Firewall and other AWS services. To download an AWS SDK, see the applicable page, which also includes prerequisites and installation instructions:
 - [Java](#)
 - [JavaScript](#)
 - [.NET](#)
 - [Node.js](#)
 - [PHP](#)
 - [Python](#)
 - [Ruby](#)

For a complete list of AWS SDKs, see [Tools for Amazon Web Services](#).

- If you're using a programming language for which AWS doesn't provide an SDK, the [AWS Network Firewall API Reference](#) documents the operations that Network Firewall supports.

- The AWS Command Line Interface (AWS CLI) supports Network Firewall. The AWS CLI lets you control multiple AWS services from the command line and automate them through scripts. For more information, see [AWS Command Line Interface](#).
- AWS Tools for Windows PowerShell supports Network Firewall. For more information, see [AWS Tools for PowerShell Cmdlet Reference](#).

Getting started with AWS Network Firewall

AWS Network Firewall provides network traffic filtering protection for your Amazon Virtual Private Cloud VPCs. This tutorial provides steps for getting started with Network Firewall using the AWS Management Console. You can also use Network Firewall API operations to create and manage your firewalls. For more information about working with Network Firewall API operations, see the [AWS Network Firewall API Reference](#).

Topics

- [Before you begin](#)
- [Step 1: Create rule groups](#)
- [Step 2: Create a firewall policy](#)
- [Step 3: Create a firewall](#)
- [Step 4: Update your Amazon VPC route tables](#)
- [Step 5: Remove the firewall and clean up your resources](#)

Before you begin

This tutorial walks you through configuring and implementing an AWS Network Firewall firewall for a VPC with a basic internet gateway architecture, like the one depicted at [Simple single zone architecture with an internet gateway using AWS Network Firewall](#).

To follow this tutorial, you'll need a test VPC where you want to implement a network firewall. Additionally, you must know how to manage the subnets and route tables in your VPC.

- For information about managing subnets in your VPC, see [VPCs and subnets](#) in the *Amazon Virtual Private Cloud User Guide*.
- For information about managing route tables for your VPC, see [Route tables](#) in the *Amazon Virtual Private Cloud User Guide*.

The test VPC that you use for this tutorial must have the following configuration in one Region:

- An internet gateway.
- A customer subnet.

- Routing configured to send inbound traffic from the internet gateway to the subnet and to send the subnet's outbound traffic to the internet gateway.
- A second subnet to use as the firewall subnet. This subnet must not be used for other purposes and must have at least one available IP address. You'll select the Availability Zone and subnet ID when you create the firewall.

If you have a different architecture that you'd like to add a firewall to, you can adjust the guidance in this tutorial accordingly. Network Firewall doesn't support some VPC architectures. For information, see [AWS Network Firewall example architectures with routing](#).

Step 1: Create rule groups

Rule groups are reusable collections of network filtering rules that you use to configure firewall behavior. In this step, you create a stateless rule group and a stateful rule group. For information about rule groups, see [Rule groups](#).

To create a stateless rule group

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, under **Network Firewall**, choose **Network Firewall rule groups**.
3. Choose **Create rule group**.
4. In the **Create rule group** page, for the **Rule group type**, choose **Stateless rule group**.
5. Enter the name that you want for the rule group. You'll use the name to identify the rule group when you add it to your firewall policy later in the tutorial. You can't change the name of a rule group after you create it.
6. For **Capacity**, enter **10**.
7. Enter the following rule specifications to create a stateless rule that blocks all packets coming from the source IP address CIDR range 192.0.2.0/24:
 - a. Set the priority to **10**.
 - b. Leave the protocol setting at **All**.
 - c. For the source address, specify **192.0.2.0/24**.
 - d. Leave the source port at **Any**.
 - e. Set the destination address to **Any**.

- f. For the action, choose **Drop**.
 - g. Choose **Add rule**. Your rule is added to the **Rules** list.
8. Review the settings for the rule group, then choose **Create rule group**.

Your new rule group is added to the list in the **Rule groups** page.

To create a stateful rule group

1. From the **Rule groups** page, choose **Create rule group**.
2. In the **Create rule group** page, for the **Rule group type**, choose **Stateful rule group**.
3. Enter a name for the stateful rule group.
4. For **Capacity**, enter **10**.
5. Choose the stateful rule group configuration option **Import Suricata compatible rules**. The entry form for Suricata compatible rule string appears. Copy and paste the following Suricata rule into the text box. This rule drops TLS traffic for a specific target domain:

```
drop tls $HOME_NET any -> $EXTERNAL_NET any (ssl_state:client_hello; tls.sni;
content:"evil.com"; startswith; nocase; endswith; msg:"matching TLS denylisted
FQDNs"; priority:1; flow:to_server, established; sid:1; rev:1;)
```

6. Choose **Add rule**. Your rule is added to the **Rules** list for the rule group.
7. Review the settings for the rule group, then choose **Create rule group**.

Your stateless rule group and your stateful rule group are listed in the **Rule groups** page. You can now use these rule groups in your firewall policies.

Step 2: Create a firewall policy

Firewall policies use rule groups and other settings to define the traffic filtering behavior for a firewall. In this procedure, you'll create a policy using the rule groups that you created in the previous step. For information about firewall policies, see [Firewall policies in AWS Network Firewall](#).

To configure a firewall policy

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.

2. In the navigation pane, under **Network Firewall**, choose **Firewall policies**.
3. In the **Firewall policies** page, choose **Create firewall policy**.
4. Enter the name that you want to use for the firewall policy. You'll use the name to identify the policy when you associate it with your firewall later in the tutorial. You can't change the name of a firewall policy after you create it.
5. Choose **Next** to go to the firewall policy's **Add rule groups** page.
6. In the **Stateless rule groups** section, choose **Add rule groups**, then select the check box for the stateless rule group that you created in the prior procedure. Choose **Add rule groups**. At the bottom of the page, the firewall policy's capacity counter shows the capacity consumed by adding this rule group next to the maximum capacity allowed for a firewall policy.
7. Your stateless rule group blocks some incoming traffic. In the stateless default actions, you choose what to do with the rest of the traffic. For this tutorial, we'll forward it to the stateful engine. Use the same default action for packets and packet fragments. Network Firewall only manages UDP packet fragments and silently drops packet fragments for other protocols. Set the action to **Forward to stateful rules**.
8. In the **Stateful rule groups** section, choose **Add rule groups**, then select the check box for the stateful rule group that you created in the prior procedure. Choose **Add rule groups**.
9. Choose **Next** then **Next** again to proceed through the tagging option and to the **Review and create** page. From this page, you can choose **Edit** for any area to return to the corresponding page in the firewall policy creation wizard.
10. Choose **Create firewall policy**.

Your new firewall policy is added to the list in the **Firewall policies** page. You can now use your firewall policy in your firewalls.

Step 3: Create a firewall

Firewalls associate the traffic filtering behavior of a firewall policy with the VPC where you want to filter traffic. In this procedure, you'll create a firewall using the firewall policy that you created in the previous step. For information about firewalls, see [Firewalls in AWS Network Firewall](#).

To create a firewall

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.

2. In the navigation pane, under **Network Firewall**, choose **Firewalls**.
3. Choose **Create firewall**.
4. For **Name**, enter the name that you want to use to identify this firewall. You can't change the name of a firewall after you create it.
5. For **VPC**, select your VPC from the dropdown.
6. For **Availability Zone** and **Subnet**, select the zone and firewall subnet that you identified in [Before you begin](#).
7. For **Associated firewall policy**, choose **Associate an existing firewall policy**, then select the firewall policy that you created in the prior procedure.
8. Choose **Create firewall**.

Your new firewall is listed in the **Firewalls** page. You've configured the firewall's behavior with the firewall policy and rule groups, and your firewall has an endpoint that's running in your VPC, ready to filter network traffic.

 **Note**

Network Firewall supports up to 100 Gbps of network traffic per firewall endpoint. If you require more traffic bandwidth, you can split your resources into subnets and create a Network Firewall firewall in each subnet.

The next step is to route the VPC's network traffic through the firewall endpoint. You'll insert it into the traffic flow between the internet gateway and your customer subnet.

Step 4: Update your Amazon VPC route tables

After you create your firewall, you insert its firewall endpoint into your Amazon Virtual Private Cloud network traffic flow, in between your internet gateway and your customer subnet. You create routing for the firewall endpoint so that it forwards traffic between the internet gateway and your subnet. Then, you update the route tables for your internet gateway and your subnet, to send traffic to the firewall endpoint instead of to each other.

This procedure covers the high-level steps for route table management. For information about managing route tables for your VPC, see [Route tables](#) in the *Amazon Virtual Private Cloud User Guide*.

To modify your route tables to insert a firewall endpoint between your internet gateway and your subnet

1. Review your routing for the internet gateway and for your customer subnet, to determine the components used to route traffic between the two.

Record the current settings. You'll use them to reverse your changes at the end of the tutorial.

- The internet gateway's route table typically has an entry with a destination set to your customer subnet's CIDR block and a target of `local`.
 - The subnet's route table typically has an entry with a destination set to `0.0.0.0/0` and a target set to the internet gateway ID.
2. Create a route table configuration for the firewall endpoint with the following two routes:
 - An entry that matches the internet gateway's route specification for traffic going to the customer subnet's CIDR block.
 - An entry that matches the subnet's route specification for traffic going to the internet gateway.

The firewall endpoint is now ready to filter and forward traffic between the internet gateway and the customer subnet. The endpoint only forwards traffic to its intended destination if it passes the inspection criteria that you defined in the rule groups and firewall policy.

3. Update the internet gateway's routing to modify the entry with a destination set to your customer subnet's CIDR block. Change the target to the firewall endpoint ID.
4. Update the customer subnet routing to modify the entry with a destination set to the internet gateway ID. Change the target to the firewall endpoint ID.

The firewall endpoint is now filtering all traffic between your internet gateway and customer subnet.

Step 5: Remove the firewall and clean up your resources

You've now successfully completed the tutorial. To remove the firewall endpoint from your VPC and prevent your account from accruing AWS Network Firewall charges for the tutorial resources, revert your route table changes and clean up the Network Firewall resources that you created.

To modify your route tables to remove the firewall

1. Return the internet gateway and subnet route tables to the configurations they had at the start of the prior procedure. This stops traffic from routing to the firewall endpoint.
2. Remove the route table configuration for the firewall endpoint.

To remove the Network Firewall resources

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, under **Network Firewall**, choose **Firewalls**.
3. In the **Firewalls** page, select the firewall that you created for the tutorial.
4. Choose **Delete**, and then confirm your request.
5. In the navigation pane, under **Network Firewall**, choose **Firewall policies**.
6. In the **Firewall policies** page, select the firewall policy that you created for the tutorial.
7. Choose **Delete**, and confirm your request.
8. In the navigation pane, under **Network Firewall**, choose **Network Firewall rule groups**.
9. In the **Rule group** page, select the name of the rule groups that you created for the tutorial, and then choose **Delete**.

You've successfully removed the firewall from your VPC traffic flow and removed all of the Network Firewall resources that you created for this tutorial.

Configuring your VPC and other components for AWS Network Firewall

This section describes the changes that you must make in your VPC configuration and other components to use AWS Network Firewall. For information about managing your Amazon Virtual Private Cloud VPC, see the [Amazon Virtual Private Cloud User Guide](#).

For examples of architectures that are supported by Network Firewall, see [Architecture and routing examples](#).

Unsupported architectures

The following lists architectures and traffic types that Network Firewall doesn't support:

- VPC peering.
- Inspection of AWS Global Accelerator traffic.
- Inspection of AmazonProvidedDNS traffic for Amazon EC2.

Topics

- [VPC subnet configuration for AWS Network Firewall](#)
- [VPC route table configuration for AWS Network Firewall](#)
- [Transit gateway attachment configuration for AWS Network Firewall](#)

VPC subnet configuration for AWS Network Firewall

When you associate a firewall to your VPC, you must provide a subnet for each Availability Zone where you want to place a firewall endpoint to filter traffic. A common configuration is to have a firewall endpoint in each zone where you have customer subnets that you want to protect, but you can also have a firewall endpoint filter traffic from multiple zones. When you create the firewall, Network Firewall adds a firewall endpoint to each of the designated subnets. Each firewall endpoint uses the firewall's associated firewall policy configuration to filter traffic that you route through it.

To prepare your VPC for your Network Firewall firewall, in each Availability Zone where you want a firewall endpoint, create a subnet for the endpoint. Each subnet must have at least one IP address available. You can't change the IP address type after you create the subnet.

Network Firewall supports up to 100 Gbps of network traffic per firewall endpoint. If you require more traffic bandwidth, you can split your resources into subnets and create a Network Firewall firewall in each subnet.

Note

Reserve these firewall subnets for the exclusive use of Network Firewall. A firewall endpoint can't filter traffic coming into or going out of the subnet in which it resides, so don't place other applications in the firewall endpoint subnets.

For information about managing subnets in your VPC, see [VPCs and subnets](#) in the *Amazon Virtual Private Cloud User Guide*.

When you create your Network Firewall firewall, you must provide at least one zone and subnet for the firewall configuration. You can add and remove subnets after you create a firewall.

VPC route table configuration for AWS Network Firewall

After you create your firewall, you reroute your VPC network traffic through the firewall endpoints so they can start filtering traffic. Perform the following steps:

1. Review the route table configurations in your VPC Availability Zones for the subnets that you want to protect and for any location that sends traffic to the subnets or receives traffic from them.
2. Determine which traffic you want the firewall to filter and insert your firewall endpoints into the traffic flow. Network Firewall supports up to 100 Gbps of network traffic per firewall endpoint. Update the route tables for both directions of traffic flow, if you want to filter incoming and outgoing traffic.

For example, suppose you wanted to filter traffic that's currently routed between a customer subnet and an internet gateway. You would update your route table configuration as follows to insert a firewall endpoint into the traffic flow:

1. Change the customer subnet route table so that it directs internet-bound traffic to the firewall endpoint.
2. Change the internet gateway route table so that it directs traffic that's bound for the customer subnet to the firewall endpoint.

3. Create a route table for the firewall endpoint so that it directs internet-bound traffic to the internet gateway and directs traffic that's bound for any destination inside the VPC to the destination specification `local`.

In this way, the firewall endpoint sits between the customer subnet and the internet gateway and can filter all incoming and outgoing traffic for the customer subnet.

For an overview of common Network Firewall architectures, with example route table configurations, see [Architecture and routing examples](#).

For information about managing route tables for your VPC, see [Route tables](#) in the *Amazon Virtual Private Cloud User Guide*.

Transit gateway attachment configuration for AWS Network Firewall

This section applies to the use of Network Firewall with a transit gateway in multiple Availability Zones where the firewall endpoints might reside in different Availability Zones than the subnets whose traffic they're filtering.

Note

To use this configuration, you must enable appliance mode on the transit gateway VPC attachment for any VPC where Network Firewall endpoints reside.

A Network Firewall endpoint is a stateful network appliance. Enabling appliance mode ensures that the transit gateway continues to use the same Availability Zone for the VPC attachment over the lifetime of a flow of traffic between source and destination.

For information about VPC transit gateways, see the guide [Amazon Virtual Private Cloud Transit Gateways](#).

For information about appliance mode and how to enable it in your attachments, see [Availability Zones](#) and [Example: Appliance in a shared services VPC](#).

Firewalls in AWS Network Firewall

An AWS Network Firewall *firewall* connects a firewall policy, which defines network traffic monitoring and filtering behavior, to the VPC that you want to protect. The firewall configuration includes specifications for the Availability Zones and subnets where the firewall endpoints are placed. It also defines high-level settings like the firewall logging configuration and tagging on the AWS firewall resource.

Topics

- [Firewall settings in AWS Network Firewall](#)
- [Managing your firewall in AWS Network Firewall](#)
- [Troubleshooting firewall endpoint failures in AWS Network Firewall](#)

Firewall settings in AWS Network Firewall

A firewall in Network Firewall has the following configuration settings, which you define when you create or update the firewall. All settings except for the firewall name are mutable.

- **Name** – The identifier for the firewall. You assign a unique name to every firewall. You can't change the name of a firewall after you create it.
- **Description** – Optional additional information about the firewall. Fill in any information that might help you remember the purpose of the firewall and how you want to use it. The description is included in firewall lists in the console and through the APIs.
- **VPC** – The VPC that's associated with the firewall. This is the VPC that the firewall provides protection for.
- **Subnets** – The subnets to use for your firewall endpoints. You can specify up to one subnet for each Availability Zone that your VPC spans. See [Configuring your VPC and other components for AWS Network Firewall](#).
- **Firewall policy** – The firewall policy that's associated with the firewall. The firewall policy provides the monitoring and protection behavior for the firewall. You can use the same firewall policy for more than one firewall. For more information about firewall policies, see [Firewall policies in AWS Network Firewall](#).
- **Logging** – The type and location of the logs that Network Firewall provides for the firewall's stateful rules engine. You can enable flow logging for the network traffic that passes through the stateful rules engine. You can enable alert logging for traffic that matches the stateful rules that

have an action setting of `Alert`, `Drop`, or `Reject`. You can enable TLS logging for TLS errors and for errors in server certificate revocation checks on outbound traffic. For more information about logging, see [Logging network traffic from AWS Network Firewall](#).

- **Encryption options** (Optional) – Network Firewall encrypts and decrypts Network Firewall resources, to protect against unauthorized access. By default, Network Firewall uses AWS owned keys for this. If you want to use your own keys, you can configure customer managed keys from AWS Key Management Service and provide them to Network Firewall. For information about this option, see [Encryption at rest with AWS Key Management Service](#).
- **Tags** – Zero or more key-value tag pairs. A tag is a label that you assign to an AWS resource. You can use tags to search and filter your resources and to track your AWS costs. For more information, see [Tagging AWS Network Firewall resources](#).
- **Delete protection** – A Boolean setting that is enabled when you create a firewall, and protects against accidental deletion of the firewall. The setting isn't shown in the console because the firewall deletion process disables this protection. Through the API, you must explicitly disable delete protection before you can delete the firewall.

Managing your firewall in AWS Network Firewall

This section describes how to create, update, and delete your firewall in AWS Network Firewall.

How Network Firewall propagates your changes

When you make any changes to a firewall, including changes to any of the firewall's components, like rule groups, TLS inspection configurations, and firewall policies, Network Firewall propagates the changes everywhere that the firewall is used. Your changes are applied within seconds, but there might be a brief period of inconsistency when the changes have arrived in some places and not in others. For example, if you modify a rule group so that it drops an additional type of packet, for a firewall that uses the rule group, the new packet type might briefly be dropped by one firewall endpoint while still being allowed by another.

This temporary inconsistency can occur when you first create a firewall and when you make changes to an existing firewall. Generally, any inconsistencies of this type last only a few seconds.

When you add a TLS inspection configuration to an existing firewall, Network Firewall interrupts traffic flows that match the criteria defined by the TLS inspection configuration scope configuration. Network Firewall will begin SSL/TLS decryption and inspection for new connections to the firewall.

Changes to stateful rules are applied only to new traffic flows. Other firewall changes, including changes to stateless rules, are applied to all network packets.

Topics

- [Creating a firewall in AWS Network Firewall](#)
- [Updating a firewall in AWS Network Firewall](#)
- [Deleting a firewall in AWS Network Firewall](#)

Creating a firewall in AWS Network Firewall

Create a firewall in Network Firewall to start using the protections you've defined in a firewall policy to protect a VPC.

To follow this procedure, the VPC that you want to protect must have at least one subnet available to host a firewall endpoint. For information, see [VPC subnets](#).

To create a firewall through the console

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, under **Network Firewall**, choose **Firewalls**.
3. Choose **Create firewall**.
4. Enter a **Name** to identify this firewall.

Note

You can't change the name after you create the firewall.

5. (Optional) Enter a **Description** for the firewall to help you identify it among your other resources.
6. Choose **Next**.
7. Choose your **VPC** from the dropdown list.

Note

You can't change the VPC after you create the firewall.

8. For **Firewall subnets**, choose the Availability Zones and subnets that you want to use for your firewall endpoints. You can choose up to one subnet for each Availability Zone that your VPC spans. The subnets should be dedicated for Network Firewall firewall use. For more information, see [VPC subnets](#).
9. Choose **Next**.
10. (Optional) Under **Protection against changes**, optionally enable **Deletion protection** and **Subnet change protection** to protect your firewall against accidental changes.
11. (Optional) Under **Customer managed key**, optionally toggle **Customize encryption settings** to use a AWS Key Management Service customer managed key to encrypt your resources. For more information about this option, see [Encryption at rest with AWS Key Management Service](#).
12. Choose **Next**.
13. For the **Associate firewall policy** section, choose the firewall policy that you want to associate with the firewall. If you already have a firewall policy defined, you can select it. Otherwise, you can associate an empty policy, which you must name permanently here. If you associate an empty policy, Network Firewall creates the policy and you can define its rules and other settings using the procedure at [Creating a firewall policy](#).
14. Choose **Next**.
15. (Optional) For the **Add tags - optional** section, assign key-value tags to your firewall. For information about tagging your AWS resources, see [Tagging AWS Network Firewall resources](#).
16. Choose **Create firewall**.

Your new firewall is added to the list in the **Firewalls** page.

Perform the following additional steps to finish configuring your new firewall and start using it to filter your network traffic.

- Configure the associated firewall policy, if it's not configured already. For information, see [Firewall policies in AWS Network Firewall](#).
- Optionally update your firewall to configure logging for your firewall. For information, see [Logging network traffic from AWS Network Firewall](#).
- Configure your VPC route tables to send traffic through the firewall endpoints. For information, see [VPC route table configuration for AWS Network Firewall](#).

Updating a firewall in AWS Network Firewall

To make changes to your firewall settings through the console, use the following procedure:

Warning

If your firewall update changes your stateful rule evaluation order type, you will experience an interruption of in-flight traffic through the firewall for a few seconds during the reset. This is the only type of update that has this effect. For more information about stateful rule evaluation order types, see [Managing evaluation order for Suricata compatible rules in AWS Network Firewall](#).

To update a firewall

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, under **Network Firewall**, choose **Firewalls**.
3. In the **Firewalls** page, choose the name of the firewall that you want to edit. This takes you to the firewall's details page.
4. Choose the tab **Firewall details**, then, in each section where you want to make changes, choose **Edit** and follow the console guidance to make your changes.
 - In the **Details** section, you can change the firewall description. The name is fixed after creation.
 - In the **Associated policy and VPC** section, you can add and remove Availability Zones and subnets and you can associate a different firewall policy. The VPC is fixed after creation.
 - In the **Logging** section, you can configure logging for alert, flow, and TLS logs. For information about your logging options and costs, see [Logging network traffic from AWS Network Firewall](#).
 - In the **Firewall tags** section, you can change the tags assigned to the AWS firewall resource. For information about tagging, see [Tagging AWS Network Firewall resources](#).
5. Choose **Save** to save your changes and return to the firewall's detail page.

Deleting a firewall in AWS Network Firewall

To delete a Network Firewall firewall, you first disassociate all AWS resources from the firewall and remove the firewall from any route tables that mention it. You must also disable the firewall's logging configuration. For information about updating a firewall's logging configuration, see [Updating a AWS Network Firewall logging configuration](#). To delete a firewall in the console, perform the following procedure.

To delete a firewall

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, under **Network Firewall**, choose **Firewalls**.
3. In the **Firewalls** page, select the firewall that you want to delete.
4. Choose **Delete**, and then confirm your request.

Your firewall is removed from the list in the **Firewalls** page. The removal can take a few minutes to complete.

Troubleshooting firewall endpoint failures in AWS Network Firewall

If Network Firewall can't create or delete a firewall endpoint in the subnet because of an error, the service displays a *status message* describing how to resolve the issues. Use the status message in the console, API, or CLI to troubleshoot the issues causing the endpoint failure. Depending on the issue, it can take as many as 15 minutes for Network Firewall to display the status message.

Console

To view the status message

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, under **Network Firewall**, choose **Firewalls**.
3. In the **Firewall details** tab, in the **Firewall endpoints** section, hover over the **firewall endpoint status** to view the status message.

API

The [DescribeFirewall](#) response includes StatusMessage.

CLI

The [describe-firewall](#) response includes StatusMessage.

The following table lists the possible causes of the error or failure as indicated in the Network Firewall console or the StatusMessage parameter in the API or CLI. Errors indicate an error that you can take actions to fix. Failures indicate a non-recoverable failed state. For errors, after you apply any of the remedial steps, Network Firewall automatically attempts to complete creation or deletion of the firewall.

Firewall endpoint status	Reason for error or failure	Cause	Solution
Error	AWS Key Management Service encryption key misconfigured	The specified AWS KMS encryption key either doesn't exist in the Region, or you aren't allowed to access it.	Either update the encryption configuration with a new key or delete the firewall. For information about using encryption keys with Network Firewall, see Encryption at rest with AWS Key Management Service .
Error	Endpoint tag removed	Network Firewall can't access the endpoint because the AWSNetworkFirewallManaged:true tag was removed from the VPC endpoint. Network	Add the AWSNetworkFirewallManaged:true tag back to the firewall endpoint, and try your request again. For information about using tags, see Tagging AWS

Firewall endpoint status	Reason for error or failure	Cause	Solution
		Firewall automatically adds this tag to the endpoint when the service creates the firewall.	Network Firewall resources .
Error	Invalid chain of trust	The firewall's TLS inspection configuration contains a certificate with an invalid chain of trust.	Replace the certificate with a valid certificate.
Error	Invalid root certificate	The firewall's TLS inspection configuration contains a certificate that Network Firewall can't validate. Network Firewall can't validate cross-signed root certificates, such as Let's Encrypt certificates. For more information, see Using SSL/TLS certificates with TLS inspection configurations in AWS Network Firewall .	Replace the certificate with a valid certificate.

Firewall endpoint status	Reason for error or failure	Cause	Solution
Error	IP limit exceeded	You've reached the quota of IPv4 or IPv6 CIDR blocks per VPC. For information about CIDR block limits per VPC, see Amazon VPC quotas in the <i>Amazon VPC User Guide</i> .	Either choose a different VPC or reduce the number of CIDR blocks associated with the VPC, and try again. For information about disassociating CIDR blocks, see Work with VPCs in the <i>Amazon VPC User Guide</i> .
Error	Subnet deleted	The specified subnet has been deleted. Your firewall must refer to an existing subnet.	Enter an existing subnet and try again.
Error	Subnet invalid IP address type	Network Firewall can't create a VPC endpoint containing the specified subnet because the subnet is associated with an IPv6 CIDR block that was removed from the firewall.	Do one of the following actions: <ul style="list-style-type: none"> • Add a new IPv6 CIDR block back to the subnet • Delete the firewall • Associate a different subnet to the firewall

Firewall endpoint status	Reason for error or failure	Cause	Solution
Failed	VPC deleted	The firewall is associated with a VPC that's been deleted.	Delete this firewall, and then create a new firewall using an existing VPC. For information about how to delete a firewall, see Deleting a firewall in AWS Network Firewall .
Error	VPCE limit exceeded	You've reached the quota of VPC endpoints you can have per VPC. For information about the VPC endpoint limits, see Amazon VPC quotas in the <i>Amazon VPC User Guide</i> .	Either delete the VPC endpoint, or delete the firewall and then create the endpoint using another VPC. For information about creating or deleting endpoints, see Work with VPCs in the <i>Amazon VPC User Guide</i> .
Error	VPCE reference exists	You can't delete the firewall because the specified firewall endpoint is associated to a VPC route table.	Remove the firewall endpoint from your route table and try again. For information about route tables, see Configure route tables in the <i>Amazon VPC User Guide</i> .

Firewall policies in AWS Network Firewall

An AWS Network Firewall *firewall policy* defines the monitoring and protection behavior for a firewall. The details of the behavior are defined in the rule groups that you add to your policy, and in some policy default settings. To use a firewall policy, you associate it with one or more firewalls.

Topics

- [Firewall policy settings in AWS Network Firewall](#)
- [Stream exception policy options in your AWS Network Firewall firewall policy](#)
- [Managing your firewall policy in AWS Network Firewall](#)

Firewall policy settings in AWS Network Firewall

A firewall policy in Network Firewall has the following configuration settings, which you define when you create or update the firewall policy. All settings except for the firewall policy name are mutable.

- **Name** – The identifier for the firewall policy. You assign a unique name to every firewall policy. You can't change the name of a firewall policy after you create it.
- **Description** – Optional additional information about the firewall policy. Fill in any information that might help you remember the purpose of the firewall policy and how you want to use it. The description is included in firewall policy lists in the console and through the APIs.
- **Stream exception policy** – The stream exception policy determines how Network Firewall handles traffic when a network connection breaks midstream. Network connections can break due to disruptions in external networks or within the firewall itself. For more information, see [Stream exception policy options in your AWS Network Firewall firewall policy](#).
- **Stateless rule groups** – Zero or more collections of stateless rules, with priority settings that define their processing order within the policy. For information about creating and managing rule groups for use in your policies, see [Rule groups in AWS Network Firewall](#).
- **Stateless default actions** – Define how Network Firewall handles a packet or UDP packet fragment that doesn't match any of the rules in the stateless rule groups. You can specify different default settings for full packets and for UDP packet fragments. Network Firewall silently drops packet fragments for other protocols.

You provide this configuration regardless of whether you define stateless rule groups for the policy.

The options for the firewall policy's default settings are the same as for stateless rules. For information about the options, see [Defining rule actions in AWS Network Firewall](#).

- **Stateful engine options** – The structure that holds stateful rule order settings. Note that you can only configure RuleOrder settings when you first create the policy. RuleOrder can't be edited later.
- **Stateful rule groups** – Zero or more collections of stateful rules, provided in Suricata compatible format. For information about creating and managing rule groups for use in your policies, see [Rule groups in AWS Network Firewall](#).
- **Stateful default actions** – Define how Network Firewall handles a packet that doesn't match any of the rules in the stateful rule groups.

These settings apply when you use strict ordering for stateful rule evaluation, and you can provide them even if you don't define stateful rule groups for the policy.

For more information about the options, see [Strict evaluation order](#).

- **Customer-managed key** (Optional) – Network Firewall encrypts and decrypts Network Firewall resources, to protect against unauthorized access. By default, Network Firewall uses AWS owned keys for this. If you want to use your own keys, you can configure customer managed keys from AWS Key Management Service and provide them to Network Firewall. For information about this option, see [Encryption at rest with AWS Key Management Service](#).
- **Policy variables** (Optional) – You can configure one or more IPv4 or IPv6 addresses in CIDR notation to override the default value of Suricata HOME_NET. If your firewall is deployed using a centralized deployment model, you might want to override HOME_NET with the CIDRs of your home network. Otherwise, Network Firewall uses the CIDR of your inspection VPC.

The firewall policy EXTERNAL_NET setting is the negation of its HOME_NET setting. For example, if the HOME_NET is 11.0.0.0, then EXTERNAL_NET is set to !11.0.0.0.

- **TCP idle timeouts** (Optional) – Defines the number of seconds that can pass without any traffic sent through the firewall before the firewall determines that the TCP connection is idle. Existing TCP connections and flows are not impacted when you update this value. Only new connections after you update this value are impacted.

You can define the value to be between 60 and 6000 seconds. If no value is provided, it defaults to 350 seconds.

- **TLS inspection configuration** (Optional) – Contains settings to turn on decryption and re-encryption of the Secure Socket Layer (SSL)/Transport Layer Security (TLS) traffic going to your firewall so that the traffic can be inspected according to the policy's stateful rules. For more information, see [Inspecting SSL/TLS traffic with TLS inspection configurations in AWS Network Firewall](#).
- **Tags** (Optional) – Zero or more key-value tag pairs. A tag is a label that you assign to an AWS resource. You can use tags to search and filter your resources and to track your AWS costs. For more information about tags, see [Tagging AWS Network Firewall resources](#).

AWS Network Firewall firewall policy capacity limitations

Network Firewall uses capacity calculations and limiting to control the operating resources that are required to process your rule groups and firewall policies. Each rule group has a capacity setting that's reserved for it in the firewall policy when you add it. Additionally, the firewall policy has limits on the count of rule groups that you can add. For information about limits, see [AWS Network Firewall quotas](#) for information about rule group capacity, see [Setting rule group capacity in AWS Network Firewall](#).

Stream exception policy options in your AWS Network Firewall firewall policy

The firewall policy's stream exception policy setting determines how Network Firewall handles traffic when a network connection breaks midstream. Network connections can break due to disruptions in external networks or within the firewall itself. A stream exception policy presents the following options:

- **Drop** - Network Firewall fails closed and drops all subsequent traffic going to the firewall. This is the default behavior.
- **Continue** - Network Firewall continues to apply rules to the subsequent traffic without context from traffic before the break. This impacts the behavior of rules that depend on this context. For example, if you have a stateful rule to drop httptraffic, Network Firewall won't match the traffic for this rule because the service won't have the context from session initialization defining the application layer protocol as HTTP. However, this behavior is rule dependent—a TCP-layer

rule using a `flow:stateless` rule would still match, as would the `aws:drop_strict` default action.

- **Reject** - Network Firewall fails closed and drops all subsequent traffic going to the firewall. Network Firewall also sends a TCP reject packet back to your client so that the client can immediately establish a new session. Network Firewall will have context about the new session and will apply rules to the subsequent traffic.

Managing your firewall policy in AWS Network Firewall

This section describes how to create, update, and delete your firewall policy in Network Firewall.

How Network Firewall propagates your changes

When you make any changes to a firewall, including changes to any of the firewall's components, like rule groups, TLS inspection configurations, and firewall policies, Network Firewall propagates the changes everywhere that the firewall is used. Your changes are applied within seconds, but there might be a brief period of inconsistency when the changes have arrived in some places and not in others. For example, if you modify a rule group so that it drops an additional type of packet, for a firewall that uses the rule group, the new packet type might briefly be dropped by one firewall endpoint while still being allowed by another.

This temporary inconsistency can occur when you first create a firewall and when you make changes to an existing firewall. Generally, any inconsistencies of this type last only a few seconds.

When you add a TLS inspection configuration to an existing firewall, Network Firewall interrupts traffic flows that match the criteria defined by the TLS inspection configuration scope configuration. Network Firewall will begin SSL/TLS decryption and inspection for new connections to the firewall.

Changes to stateful rules are applied only to new traffic flows. Other firewall changes, including changes to stateless rules, are applied to all network packets.

Topics

- [Creating a firewall policy in AWS Network Firewall](#)
- [Updating a firewall policy in AWS Network Firewall](#)
- [Deleting a firewall policy in AWS Network Firewall](#)

Creating a firewall policy in AWS Network Firewall

To create a firewall policy in Network Firewall, you need rule groups that you've already defined to use in the policy. You can create new rule groups and reuse existing ones. For information about creating and managing rule groups, see [Rule groups in AWS Network Firewall](#).

If you want to use TLS inspection, you need to first create a TLS inspection configuration to use in the policy. For information about working with TLS inspection configurations, see [Inspecting SSL/TLS traffic with TLS inspection configurations in AWS Network Firewall](#).

To create a firewall policy

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, under **Network Firewall**, choose **Firewall policies**.
3. Choose **Create firewall policy**.
4. Enter a **Name** to identify this firewall policy.

Note

You can't change the name after you create the firewall policy.

5. (Optional) Enter a **Description** for the policy to help you identify it among your other resources.
6. For **Stream exception policy**, choose how Network Firewall handles traffic when a network connection breaks midstream. Network connections can break due to disruptions in external networks or within the firewall itself. Choose from the following options:
 - **Drop** - Network Firewall fails closed and drops all subsequent traffic going to the firewall. This is the default behavior.
 - **Continue** - Network Firewall continues to apply rules to the subsequent traffic without context from traffic before the break. This impacts the behavior of rules that depend on this context. For example, if you have a stateful rule to drop http traffic, Network Firewall won't match the traffic for this rule because the service won't have the context from session initialization defining the application layer protocol as HTTP. However, this behavior is rule dependent—a TCP-layer rule using a `flow:stateless` rule would still match, as would the `aws:drop_strict` default action.

- **Reject** - Network Firewall fails closed and drops all subsequent traffic going to the firewall. Network Firewall also sends a TCP reject packet back to your client so that the client can immediately establish a new session. Network Firewall will have context about the new session and will apply rules to the subsequent traffic.
7. Choose **Next** to go to the firewall policy's **Add rule groups** page.
 8. To choose the actions to take on packets that don't match any stateless rules, in the **Stateless default actions** section, first choose how to treat fragmented packets. You can choose **Use the same actions for all packets** or **Use different actions for full packets and fragmented packets**. You can then choose **Pass**, **Drop**, or **Forward to stateful rule groups** for all packets, or choose individually for full and fragmented packets. You also have the option to enable a custom action that lets you publish custom Amazon CloudWatch metrics to monitor the usage of stateless rules in your rule group.
 9. To choose the way that your stateful rules are ordered for evaluation, and the actions to take on packets that don't match any stateful rules, in the **Stateful rule evaluation order and default action** section, first choose a rule evaluation order:
 - Choose **Strict order** (recommended) to provide your rules in the order that you want them to be evaluated. You can then choose one or more default actions for packets that don't match any rules.
 - Choose **Action order** to have the stateful rules engine determine the evaluation order of your rules. The default action for this rule order is **Pass**, followed by **Drop**, **Reject**, and **Alert** actions. This option was previously named **Default** order.

For more information about stateful default actions for rule groups, see [Action order](#).

10. To add stateless rule groups, in the **Stateless rule groups** section, choose **Add rule groups**, then select the check boxes for the rule groups that you want to add and choose **Add rule groups**.
11. If your firewall policy has multiple stateless rule groups, in the **Stateless rule group** section, update the processing order as needed. Network Firewall processes stateless rule groups by order of priority, starting from the lowest. To move a rule group in the list, select the check box next to its name and then move it up or down. For more information, see [How AWS Network Firewall filters network traffic](#).
12. Choose the stateless default actions for the firewall policy to take if a full packet or UDP packet fragment doesn't match any of the stateless rule groups. Network Firewall silently

drops packet fragments for other protocols. For information about the action options, see [Defining rule actions in AWS Network Firewall](#).

Network Firewall doesn't automatically forward packets to stateful rule groups. It forwards only for the following situations:

- The packet matches a stateless rule whose action specifies forward to stateful rule groups.
 - The packet doesn't match any stateless rule and the applicable default action setting specifies forward to stateful rule groups.
13. To add stateful rule groups, in the **Stateful rule groups** section, choose **Add rule groups**, then select the check boxes for the rule groups that you want to add and choose **Add rule groups**.
 14. Choose **Next**.
 15. On the **Configure advanced settings** page, optionally customize encryption and policy variables, and set the stream exception policy.
 16. (Optional) Under **Customer managed key**, toggle the **Customize encryption settings** option to use a AWS Key Management Service customer managed key to encrypt your resources. For more information about this option, see [Encryption at rest with AWS Key Management Service](#).
 17. (Optional) For **Policy variables** enter one or more IPv4 or IPv6 addresses in CIDR notation to override the default value of Suricata HOME_NET. If your firewall is deployed using a centralized deployment model, you might want to override HOME_NET with the CIDRs of your home network. Otherwise, Network Firewall uses the CIDR of your inspection VPC.
 18. Choose **Next**.
 19. (Optional) Under **Idle Timeouts**, toggle the **Customize TCP idle timeout settings** option. This lets you define the number of seconds a TCP connection can remain idle before Network Firewall drops the traffic. For information about the idle timeout setting, see [Firewall policy settings in AWS Network Firewall](#).
 20. (Optional) On the **Add TLS inspection configuration** page, choose **Add TLS inspection configuration** to turn on decryption and re-encryption of incoming SSL/TLS traffic for the firewalls associated with this policy. You can't add or remove a TLS inspection configuration after firewall policy creation. For information about TLS inspection configurations, see [Inspecting SSL/TLS traffic with TLS inspection configurations in AWS Network Firewall](#).
 21. Choose **Next**.
 22. (Optional) On the **Add tags** page, enter a key and optional value for any tag that you want added to this firewall policy. Tags help you organize and manage your AWS resources. For more information about tagging your resources, see [Tagging AWS Network Firewall resources](#).

23. Choose **Next**.
24. In the **Review and create** page, check over your firewall policy settings. If you want to change any section, choose **Edit** for the section. This returns you to the page in the firewall policy wizard. Make your changes, then choose **Next** on each page until you come back to the review and create page.
25. Choose **Create firewall policy**.

Your new firewall policy is added to the list in the **Firewall policies** page.

Updating a firewall policy in AWS Network Firewall

To change your Network Firewall firewall policy settings, use the following procedure:

To update a firewall policy

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, under **Network Firewall**, choose **Firewall policies**.
3. In the **Firewall policies** page, select the name of the firewall policy you want to update.
4. In the firewall policy's page, make your changes. Note the following constraints:
 - You can't change the name of the firewall policy.
 - You can't add or remove a TLS inspection configuration. However, you can replace an existing TLS inspection configuration with another TLS inspection configuration.
 - You can change other policy details, including rule groups.
5. Choose **Save** to save your changes.

How Network Firewall propagates your changes

When you make any changes to a firewall, including changes to any of the firewall's components, like rule groups, TLS inspection configurations, and firewall policies, Network Firewall propagates the changes everywhere that the firewall is used. Your changes are applied within seconds, but there might be a brief period of inconsistency when the changes have arrived in some places and not in others. For example, if you modify a rule group so that it drops an additional type of packet, for a firewall that uses the rule group, the new packet type might briefly be dropped by one firewall endpoint while still being allowed by another.

This temporary inconsistency can occur when you first create a firewall and when you make changes to an existing firewall. Generally, any inconsistencies of this type last only a few seconds.

When you add a TLS inspection configuration to an existing firewall, Network Firewall interrupts traffic flows that match the criteria defined by the TLS inspection configuration scope configuration. Network Firewall will begin SSL/TLS decryption and inspection for new connections to the firewall.

Changes to stateful rules are applied only to new traffic flows. Other firewall changes, including changes to stateless rules, are applied to all network packets.

Deleting a firewall policy in AWS Network Firewall

To delete a firewall policy, perform the following procedure.

Deleting a rule group, TLS inspection configuration, or firewall policy

When you delete a rule group, TLS inspection configuration, or a firewall policy, AWS Network Firewall checks to see if it's currently being referenced. A rule group and TLS inspection configuration can be referenced by a firewall policy, and a firewall policy can be referenced by a firewall. If Network Firewall determines that the resource is being referenced, it warns you. Network Firewall is almost always able to determine whether a resource is being referenced. However, in rare cases, it might not be able to do so. If you need to be sure that the resource that you want to delete isn't in use, check all of your firewalls or firewall policies before deleting it. Note that policies that have associations can't be deleted.

To delete a firewall policy

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, under **Network Firewall**, choose **Firewall policies**.
3. In the **Firewall policies** page, select firewall policy that you want to delete.
4. Choose **Delete**, and confirm your request.

Your firewall policy is removed from the list in the **Firewall policies** page.

Rule groups in AWS Network Firewall

An AWS Network Firewall *rule group* is a reusable set of criteria for inspecting and handling network traffic. You add one or more rule groups to a firewall policy as part of policy configuration. For more information about firewall policies and firewalls, see [Firewall policies in AWS Network Firewall](#) and [Firewalls in AWS Network Firewall](#).

You can use your own rule groups and you can use rule groups that are managed for you by AWS.

Network Firewall rule groups are either *stateless* or *stateful*. Stateless rule groups evaluate packets in isolation, while stateful rule groups evaluate them in the context of their traffic flow. You can create and manage the following categories of rule groups in Network Firewall:

- **Stateless** – Defines standard network connection attributes for examining a packet on its own, with no additional context.
- **Stateful** – Defines criteria for examining a packet in the context of traffic flow and of other traffic that's related to the packet.

Network Firewall uses a Suricata rules engine to process all stateful rules. You can write any of your stateful rules in Suricata compatible format. Alternately, for domain list rules and for very basic rules, you can use an easy entry form provided by Network Firewall.

Stateful rule groups are available in the following categories:

- **Suricata compatible rule strings** – Provides match and action settings, in Suricata compatible format. You can provide all of your stateful rules through this method if you want to.
- **Domain list** – Defines a list of domain names and specifies the protocol type to inspect.
- **Standard stateful rules** – Defines standard network connection attributes for examining a packet within the context of a traffic flow.

Depending on the type of rule group, you might also define rules inside the rule group. Rules provide detailed criteria for packet inspection and specify what to do when a packet matches the criteria. When Network Firewall finds a match between the criteria and a packet, we say that the packet matches the rule group.

This section provides guidance for creating and managing your own rule groups.

Topics

- [Using AWS managed rule groups in AWS Network Firewall](#)
- [Managing your own rule groups in AWS Network Firewall](#)

Using AWS managed rule groups in AWS Network Firewall

Managed rule groups are collections of predefined, ready-to-use rules that AWS writes and maintains for you. AWS managed rule groups are available for free to Network Firewall customers.

Automatic updates

Keeping up to date on the constantly changing threat landscape can be time consuming and expensive. Managed rule groups can save you time when you implement and use Network Firewall. AWS automatically updates managed rule groups when new vulnerabilities and threats emerge. AWS updates managed rule groups once a day to once a week.

In some cases, AWS is notified of new vulnerabilities before public disclosure due to its participation in a number of private disclosure communities. In those cases, AWS Network Firewall can update rule groups and deploy them to your environment before a new threat is widely known.

Topics

- [Working with managed rule groups in Network Firewall](#)
- [AWS domain and IP managed rule groups for AWS Network Firewall](#)
- [AWS threat signature managed rule groups for AWS Network Firewall](#)
- [Disclaimer for AWS managed rule groups in AWS Network Firewall](#)

Working with managed rule groups in Network Firewall

This section provides guidance for accessing and using AWS managed rule groups in Network Firewall.

You can select one or more rules or rule groups to use in your Network Firewall policies. Each set of managed rule groups counts as a single rule group toward the maximum number of stateful rule groups per firewall policy.

Through the console, you access managed rule group information when you add and edit rules in your firewall policies. Through the APIs and the command line interface (CLI), you can directly request managed rule group information.

When you use a managed rule group in your firewall policy, you can edit the following setting:

- **Set rule actions to alert** – Managed rule groups are designed to block traffic with drop rules. This setting in the API matches the **Run in alert mode** setting in the console. This overrides all rule actions in the rule group to alert instead. This is useful for testing a rule group before using it to control traffic.

To edit the managed rule group alert settings in your firewall policy:

- **Console**
 - After you add the managed rule group to your firewall policy, from the **Policies** page, choose the firewall policy you just created. This takes you to the policy detail page where you can edit aspects of the policy, and view details about the policy.

In the **Network Firewall rule groups** tab, in the **Stateful rule groups** section, choose the rule group that you'd like to run in alert mode, then from the **Actions** drop-down menu, choose **Rule group details**. For the **Run in alert mode** setting, toggle to **Enabled** to run the rule group in alert mode.

- **API** – StatefulRuleGroupOverride. [StatefulRuleGroupOverride](#).

Topics

- [Adding managed rule groups to your AWS Network Firewall policy using the console](#)
- [Viewing managed rule groups in AWS Network Firewall](#)
- [Copying threat signature rules into your own AWS Network Firewall rule group](#)
- [Getting notified of updates to a threat signature rule group in AWS Network Firewall](#)
- [Mitigating false-positive scenarios in AWS Network Firewall](#)

Adding managed rule groups to your AWS Network Firewall policy using the console

The following procedure describes how to add one or more managed rule groups to your Network Firewall policy, to implement the managed rule group protections. You can either add the managed rule group to your policy via the Network Firewall rule groups page, or from your policy's detail page.

To add a managed rule group to a policy via the Network Firewall rule group page

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, under **Network Firewall**, choose **Network Firewall rule groups**.
3. In the **AWS managed rule groups** tab, choose **Add rule groups to policy**.
4. In the **Choose a firewall policy** section, select the firewall policy to add your AWS managed rule groups to.
5. Choose **Next**.
6. In the **Choose rule groups** section, choose one or more rule groups to add to your policy. You can add your own rule groups, or AWS managed rule groups.
7. Choose **Next**.
8. (Optional) On the **Add tags** page, enter a key and optional value for any tag that you want to add to this firewall policy. Tags help you organize and manage your AWS resources. For more information about tagging your resources, see [Tagging AWS Network Firewall resources](#).
9. Choose **Next**.
10. On the **Review and confirm** page, check the rule group settings for your policy. If you want to change any section, choose **Edit** for the section. This returns you to the corresponding step in the add rule group to policy wizard. Make your changes, then choose **Next** on each page until you come back to the review and confirm page.
11. Choose **Add rule groups to policy**.

To add one or more managed rule groups to your policy via the Network Firewall policy detail page

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, under **Network Firewall**, choose **Firewall policies**.
3. Select the policy that you'd like to add one or more AWS managed rule groups to.
4. In the **Stateful rule groups** section, in the **Actions** drop-down menu, select **Add managed stateful rule groups**.
5. Select the AWS managed rule groups to add to your policy.
6. Choose **Add to policy**.

Viewing managed rule groups in AWS Network Firewall

You can view the managed rule groups that are available for your use in your Network Firewall policy.

To view the list of managed rule groups

- **Console** – You can view the list of managed rule groups either in the **Network Firewall rule groups** page in the **AWS managed rule groups** tab, or in the policy details page. When you add managed rule groups to a policy, you'll see only the managed rule groups that fit your policy type. For example, if your policy type is default ordered, you'll see only the managed rule groups that have a type of default ordered.
- **API** – [ListRuleGroups](#) with the parameter `Scope`.
- **CLI** – `aws network-firewall list-rule-groups --scope MANAGED`. To filter by managed rule group type, you can include the parameter `managed-type` and filter by `AWS_MANAGED_THREAT_SIGNATURES` and `AWS_MANAGED_DOMAIN_LISTS`.

Copying threat signature rules into your own AWS Network Firewall rule group

Network Firewall provides full visibility into the threat signature rule content of its AWS managed rules. This enables you to choose between using the rule group as-is in your firewall policy or copying the rule group's rules into your own rule group and customizing them for your specific needs.

Important

Copied rules don't automatically inherit rule updates that AWS makes to managed rule group rules. We recommend that you subscribe to Amazon SNS topics for updates made to the originating rule group. For more information, see [Getting notified of updates to a threat signature rule group](#). You're responsible for validating rule changes and making sure that your own rules are up-to-date.

To copy a managed threat signature rule group's rules, create a local copy of the rule group rules, make your modifications, then create your own rule group. The following procedure explains how to copy a threat signature rule group's rules, and then create your own rule group.

Console

To copy a managed threat signature rule group's rules using the console

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, under **Network Firewall**, choose **Network Firewall rule groups**.
3. In the **AWS managed rule groups** tab, under **Threat signature rule groups**, select a rule group to view its details.
4. Choose **Duplicate rule group** to copy the rules into your own rule group. You can modify the rule group details, and then choose **Create rule group**.

Alternatively, you can choose **Copy** to copy the rules to your clipboard. You can then modify them in a text editor, or create a new rule group and paste the rules into your own stateful rule group. For information about how to create your own stateful rule group, see [Creating a stateful rule group](#).

CLI

To copy a managed threat signature rule group's rules using the AWS CLI

1. Run `aws network-firewall list-rule-groups --scope MANAGED --managed-type AWS_MANAGED_THREAT_SIGNATURES` to filter the AWS managed threat signature rule groups.
2. In the following command, replace *rulegroup-arn* with the Amazon Resource Name (ARN) of the threat signature managed rule group that you'd like to copy:

```
aws network-firewall describe-rule-group --rule-group-arn rulegroup-arn.
```

Network Firewall returns the rule group details in the response, which you can parse and modify in your text editor. Then, you can use the modified rule group details to create your own rule group using the command [create-rule-group](#).

Getting notified of updates to a threat signature rule group in AWS Network Firewall

You can subscribe to Amazon Simple Notification Service (Amazon SNS) notifications for updates to a managed threat signature rule group, such as updates made for urgent security updates. AWS updates managed threat signature rule groups for Network Firewall as often as once a day to once a week.

The AWS threat signature managed rule groups use a single SNS subscription topic ARN, so you subscribe once for all the rule groups.

How to subscribe

To subscribe to notifications for a rule group, create an Amazon SNS subscription for the rule group's Amazon SNS topic ARN.

For information about how to subscribe to an Amazon SNS topic, see [Configuring Amazon Simple Notification Service](#) in the [Amazon Simple Notification Service Developer Guide](#).

Where to find the Amazon SNS topic ARN for a threat signature managed rule group

The AWS managed rule groups use a single SNS topic ARN, so you can retrieve the topic ARN from one of the rule groups and subscribe to it to get notifications for all of the managed rule groups.

- **Console**
 - On the Network Firewall rule groups page, in the **AWS managed rule group** tab, in the **Threat signature rule groups** section, select a rule group to view the rule group's details. The details include the rule group's Amazon SNS topic ARN.
 - (Option) After you've added the managed rule group into your firewall policy, choose **Edit** on the firewall policy, and then select and edit the rule group rule to view the rule group's Amazon SNS topic ARN.
- **API** – The [DescribeRuleGroup](#) response includes `SnsTopic`. The value for `SnsTopic` is the Amazon SNS topic ARN.
- **CLI** – The [describe-rule-group](#) response includes `SnsTopic`. The value for `SnsTopic` is the Amazon SNS topic ARN.

The notification format for AWS managed rule group

The Amazon SNS notifications for AWS managed rule groups always contain the fields `Subject`, `Message`, and `MessageAttributes`. Other fields are included according to the type of message and which managed rule group the notification is for.

The following shows an example notification listing for the `AWS-Managed-Threat-Signatures`.

```
{
  "Type" : "Notification",
  "MessageId" : "82a03348-5419-5945-9a82-699adada25e3",
  "TopicArn" : "arn:aws:sns:us-west-2:696851677263:AWS-Managed-Threat-Signatures",
  "Subject" : "New version available for: StatefulRG2",
  "Message" : "The following Network Firewall managed resource has a new version:
arn:aws:network-firewall:us-west-2:aws-managed:stateful-rulegroup/StatefulRG2. To view
the new version, either call DescribeRuleGroup or view the resource in the Network
Firewall console.",
  "Timestamp" : "2022-04-14T21:05:07.002Z",
  "SignatureVersion" : "1",
  "Signature" : "ZoDQM5iIhp6E7u84qnip14RTQo/5Vi+fpQ7/tYuqwk28o+7uXuHz9TygI6otycw6Dz5Pw
+VOLu0PDuIK4xrGwFYrJypbsaZ1cbNRnM9upkzwGH8w/VORCDZ1QwKYKNP4Ep/mSKVyigh9qe+CHSW/
jD2HNE9LY96li5D0h7a2594A12MH5koAXucnYUcHkclBAzwwxbca2fCkI4PaT24SYyHem1C0w86hLt1mDZYE8o7crIX70U
+/1xEv33RYPushZIIYjf+H3EW7jX6fAc7+Dz/KLCX5Jeft2pheVMomQ==",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/
SimpleNotificationService-7ff5318490ec183fbaddaa2a969abfda.pem",
  "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-west-2:123456789012:AWS-Managed-
Threat-Signatures:f2b28278-6d26-4d05-8332-1a96687c850f",
  "MessageAttributes" : {
    "source_revision_token" : {"Type":"String","Value":"14a7e0f5-e050-40d0-
a0b1-001f690d44b9"},
    "managed_arn" : {"Type":"String","Value":"arn:aws:network-firewall:us-west-2:aws-
managed:stateful-rulegroup/StatefulRG2"}
  }
}
```

The notification contains `source_revision_token`. The value for `source_revision_token` is the `UpdateToken` that you can view when you call [DescribeRuleGroup](#) in the *AWS Network Firewall API Reference*.

For general information about Amazon SNS notification formats and how to filter the notifications that you receive, see [Parsing message formats](#) and [Amazon SNS subscription filter policies](#) in the *Amazon Simple Notification Service Developer Guide*.

Mitigating false-positive scenarios in AWS Network Firewall

As a best practice, before using a rule group in production, with logging enabled, run the managed rule group in **alert mode** if you're using an intrusion detection system (IDS), or in **drop mode** if you use an intrusion prevention system (IPS) in a non-production environment. Either mode sends alert messages to the logs for traffic that doesn't pass inspection. For more information, see [Logging network traffic from AWS Network Firewall](#).

Running a managed rule group in either alert mode or drop mode allows you to do a dry run with alert logs that show you what the resulting behavior would be before you commit to making changes to your traffic. Evaluate the rule group using Network Firewall logs. When you're satisfied that the rule group does what you want it to do, disable test mode on the group.

Mitigating false-positive scenarios

If you are encountering false-positive scenarios with AWS managed rule groups, perform the following steps:

1. In the firewall policy's AWS managed rule group settings in the Network Firewall console, override the actions in the rules of the rule groups by enabling **Run in alert mode**. This stops them from blocking legitimate traffic.
2. Use [Network Firewall logs](#) to identify which AWS managed rule group is triggering the false positive.
3. In the AWS Network Firewall console, edit the firewall policy, and locate the AWS managed rule group that you've identified. Then, disable **Run in alert mode** for the rules that aren't causing the false positive, and leave the rule group that is causing the false positive in alert mode.

For more information about a rule in an AWS managed rule group, contact the [AWS Support Center](#).

AWS domain and IP managed rule groups for AWS Network Firewall

This section describes the AWS managed rule groups that inspect domain and IP information for Network Firewall. You see these in the console in the list of AWS managed rule groups, or when you add rule groups to your firewall policy. Through the API, you can retrieve the list of AWS managed rule groups by calling [ListRuleGroups](#).

Domain and IP rule groups block HTTP/HTTPS traffic to domains identified as low-reputation, or that are known or suspected to be associated with malware or botnets. Choose one or more of these rule groups to establish domain list protection for your resources.

Each rule name in the table below is appended by either `StrictOrder` or `ActionOrder`. A firewall policy's *rule evaluation order* determines whether you can add `StrictOrder` or `ActionOrder` managed rule groups to the policy. For example, you can only add a rule group appended with `StrictOrder` if the policy uses strict order for its rule evaluation order. In the console, Network Firewall automatically filters the managed rule groups available for you to add to your policy. For information about rule evaluation order, see [Managing evaluation order for Suricata compatible rules in AWS Network Firewall](#).

Rule name	Description and label
<code>AbusedLegitMalwareDomainsStrictOrder</code> , <code>AbusedLegitMalwareDomainsActionOrder</code>	Rules that allow you to block requests to a class of domains, which are generally legitimate but are compromised and may host malware. This can help reduce the risk of receiving malware or viruses originating from these sources with poor reputation.
<code>MalwareDomainsStrictOrder</code> , <code>MalwareDomainsActionOrder</code>	Rules that allow you to block requests to domains that are known for hosting malware. This can help reduce the risk of receiving malware or viruses originating from these known sources.
<code>AbusedLegitBotNetCommandAndControlDomainsStrictOrder</code> , <code>AbusedLegitBotNetCommandAndControlDomainsActionOrder</code>	Rules that allow you to block requests to a class of domains, which are generally legitimate but are compromised and may host botnets. This can help reduce the risk of resources accessing botnets originating from these sources with poor reputation.
<code>BotNetCommandAndControlDomainsStrictOrder</code> , <code>BotNetCommandAndControlDomainsActionOrder</code>	Rules that allow you to block requests to domains that are known for hosting botnets. This can help reduce the risk of resources

Rule name	Description and label
	accessing botnets originating from these known sources.

AWS threat signature managed rule groups for AWS Network Firewall

This section describes the AWS managed rule groups that inspect for threat signatures for Network Firewall. You see these in the console in the list of AWS managed rule groups, or when you add rule groups to your firewall policy. Through the API, you can retrieve the list of AWS managed rule groups by calling [ListRuleGroups](#).

Network Firewall managed threat signature rule groups support several categories of threat signatures to protect against various types of malware and exploits, denial of service attempts, botnets, web attacks, credential phishing, scanning tools, and mail or messaging attacks. There are also signatures for intrusion detection and to enforce fair use policies as well as guard against emerging threats. Currently, Network Firewall supports only Suricata-compatible stateful managed rule groups.

Each rule name in the table below is appended by either `StrictOrder` or `ActionOrder`. A firewall policy's *rule evaluation order* determines whether you can add `StrictOrder` or `ActionOrder` managed rule groups to the policy. For example, you can only add a rule group appended with `StrictOrder` if the policy uses strict order for its rule evaluation order. In the console, Network Firewall automatically filters the managed rule groups available for you to add to your policy. For information about rule evaluation order, see [Managing evaluation order for Suricata compatible rules in AWS Network Firewall](#).

Category	Rule name	Description and label
Botnet	ThreatSignaturesBotnetStrictOrder , ThreatSignaturesBotnetActionOrder	Signatures that are autogenerated from several sources of known and confirmed active botnet and other Command and Control (C2) hosts.
Botnet Web	ThreatSignaturesBotnetWebStrictOrder	Signatures that detects HTTP botnets.

Category	Rule name	Description and label
	, ThreatSignaturesBotnetWebActionOrder	
Botnet Windows	ThreatSignaturesBotnetWindowsStrictOrder , ThreatSignaturesBotnetWindowsActionOrder	Detects Windows botnets.
Compromised	ThreatSignaturesIOCStraightOrder , ThreatSignaturesIOCActionOrder	<p>Attack Response - Signatures to identify responses indicative of intrusion—examples included but not limited to LMHost file download, presence of certain web banners and the detection of Metasploit Meterpreter kill command. These are designed to catch the results of a successful attack. Things like “id=root”, or error messages that indicate a compromise may have happened.</p> <p>Exploit Kit - Signatures to detect activity related to Exploit Kits, their infrastructure, and delivery.</p>
DoS	ThreatSignaturesDoSStraightOrder , ThreatSignaturesDoSActionOrder	<p>Signatures that detect Denial of Service (DoS) attempts. These rules are intended to catch inbound DoS activity, and provide indication of outbound DoS activity.</p>

Category	Rule name	Description and label
Emerging Threats	ThreatSignaturesEmergingEventsStrictOrder ,ThreatSignaturesEmergingEventsActionOrder	Current Events - Signatures with rules developed in response to active and short-lived campaigns and high-profile items that are expected to be temporary. The rules in this category are ones that are not intended to be kept in the ruleset for long, or that need to be further tested before they are considered for inclusion. Most often these will be simple sigs for the Storm binary URL of the day, sigs to catch CLSID's of newly found vulnerable apps where we don't have any detail on the exploit.

Category	Rule name	Description and label
Exploits	ThreatSignaturesExploitsStrictOrder , ThreatSignaturesExploitsActionOrder	<p>Exploits - Signatures that protect against direct exploits not otherwise covered in a specific service category. This is the category where you'll find specific attacks against vulnerabilities such as against Microsoft Windows. Attacks with their own category such as SQL injection have their own category.</p> <p>ActiveX - Signatures that protect against attacks against Microsoft ActiveX controls and exploits targeting vulnerabilities in ActiveX controls.</p> <p>FTP - Signatures that protect against attacks, exploits, and vulnerabilities regarding File Transfer Protocol (FTP). This category also includes rules that detect non-malicious FTP activity such as logins for logging purposes.</p> <p>ICMP - Signatures that protect against attacks and vulnerabilities regarding Internet Control Message Protocol (ICMP).</p> <p>NetBIOS - Signatures that protect against attacks,</p>

Category	Rule name	Description and label
		<p>exploits, and vulnerabilities regarding NetBIOS. This category also includes rules that detect non-malicious NetBIOS activity for logging purposes.</p> <p>RPC - Signatures that protect against attacks, exploits, and vulnerabilities regarding Remote Procedure Call (RPC). This category also includes rules that detect non-malicious RPC activity for logging purposes.</p> <p>ShellCode - For remote shellcode detection. Remote shellcode is used when an attacker wants to target a vulnerable process running on another machine on a local network or intranet. If successfully executed, the shellcode can provide the attacker access to the target machine across the network. Remote shellcodes normally use standard TCP/IP socket connections to allow the attacker access to the shell on the target machine.</p> <p>SNMP - Signatures that protect against attacks, exploits, and vulnerabilities</p>

Category	Rule name	Description and label
		<p>regarding Simple Network Management Protocol (SNMP). This category also includes rules that detect non-malicious SNMP activity for logging purposes.</p> <p>Telnet - Signatures that protect against attacks, exploits, and vulnerabilities regarding TELNET. This category also includes rules that detect non-malicious TELNET activity for logging purposes.</p> <p>TFTP - Signatures that protect against attacks, exploits, and vulnerabilities regarding Trivial File Transport Protocol (TFTP). This category also includes rules that detect non-malicious TFTP activity for logging purposes.</p> <p>VOIP - Signatures that protect against attacks and vulnerabilities regarding Voice over IP (VOIP) including SIP, H.323 and RTP among others.</p> <p>SQL - Signatures that protect against attacks, exploits, and vulnerabilities regarding Structured Query Language</p>

Category	Rule name	Description and label
		(SQL). This category also includes rules that detect non-malicious SQL activity for logging purposes.
FUP	ThreatSignaturesFU PStrictOrder , ThreatSignaturesFU PActionOrder	Signatures to detect gaming traffic, potentially inappropriate websites, and P2P traffic as well as signatures that may indicate violations to an organization's policy.

Category	Rule name	Description and label
Malware	ThreatSignaturesMalwareStrictOrder , ThreatSignaturesMalwareActionOrder	<p>Signatures that detect malware (TCP, UDP, SMTP, ICMP, SMB, IP) and WORM.</p> <p>Malware - Detects malicious software. Rules in this category detect activity related to malicious software that is detected on the network including malware in transit, active malware, malware infections, malware attacks, and updating of malware.</p> <p>Worm - Detects malicious activity that automatically attempts to spread across the internet or within a network by exploiting a vulnerability. While the exploit itself is typically identified in the exploit or given protocol category, an additional entry in this category might be made if the actual malware engaging in worm-like propagation can be identified.</p>

Category	Rule name	Description and label
Malware Coin Mining	ThreatSignaturesMalwareCoinminingStrictOrder , ThreatSignaturesMalwareCoinminingActionOrder	Signatures with rules that detect malware that performs coin mining. These signatures can also detect some legitimate (though often undesirable) coin mining software.
Malware Mobile	ThreatSignaturesMalwareMobileStrictOrder , ThreatSignaturesMalwareMobileActionOrder	Signatures that indicate malware that's associated with mobile and tablet operating systems such as Google Android, Apple iOS, and others. Malware that's detected and is associated with mobile operating systems is generally placed in this category rather than the standard categories such as Malware.
Malware Web	ThreatSignaturesMalwareWebStrictOrder , ThreatSignaturesMalwareWebActionOrder	Signatures that detect malicious code in HTTP and TLS protocols.
Phishing	ThreatSignaturesPhishingStrictOrder , ThreatSignaturesPhishingActionOrder	Signatures that detect credential phishing activity. This includes landing pages exhibiting credential phishing as well as successful submission of credentials into credential phishing sites.

Category	Rule name	Description and label
Scanners	ThreatSignaturesScannersStrictOrder , ThreatSignaturesScannersActionOrder	Signatures that detect reconnaissance and probing from tools such as Nessus, Nikto, and other port scanning tools. This category can be useful for detecting early breach activity and post-infection lateral movement within an organization.

Category	Rule name	Description and label
Suspect	ThreatSignaturesSuspectStrictOrder , ThreatSignaturesSuspectActionOrder	<p>JA3 - Fingerprints malicious SSL certificates using JA3 hashes. These rules are based on parameters that are in the SSL handshake negotiation by both clients and servers. These rules can have a high false positive rate but can be very useful for threat hunting or malware detonation environments.</p> <p>Chat - Signatures that identify traffic related to numerous chat clients such as Internet Relay Chat (IRC). Chat traffic can be indicative of possible check-in activity by threat actors.</p> <p>User Agents - Signatures that detect suspicious and anomalous user agents. Known malicious user agents are generally placed in the Malware category.</p>

Category	Rule name	Description and label
Web Attacks	ThreatSignaturesWebAttacksStrictOrder , ThreatSignaturesWebAttacksActionOrder	<p>Web Client - Signatures that detect attacks and vulnerabilities regarding web clients such as web browsers as well as client-side applications like CURL, WGET and others.</p> <p>Web Server - Signatures that detect attacks against web server infrastructure such as APACHE, TOMCAT, NGINX, Microsoft Internet Information Services (IIS) and other web server software.</p> <p>Web Specific Apps - Signatures that detect attacks and vulnerabilities in specific web applications.</p>

Disclaimer for AWS managed rule groups in AWS Network Firewall

AWS managed rule groups are designed to protect you from common web threats. When used in accordance with the documentation, AWS managed rule groups add another layer of security for your applications. However, AWS managed rule groups aren't intended as a replacement for your security responsibilities, which are determined by the AWS resources that you select. Refer to the [Shared Responsibility Model](#) to ensure that your resources in AWS are properly protected.

Managing your own rule groups in AWS Network Firewall

Follow the guidance in this section to manage your AWS Network Firewall rule groups.

How Network Firewall propagates your changes

When you make any changes to a firewall, including changes to any of the firewall's components, like rule groups, TLS inspection configurations, and firewall policies, Network Firewall propagates

the changes everywhere that the firewall is used. Your changes are applied within seconds, but there might be a brief period of inconsistency when the changes have arrived in some places and not in others. For example, if you modify a rule group so that it drops an additional type of packet, for a firewall that uses the rule group, the new packet type might briefly be dropped by one firewall endpoint while still being allowed by another.

This temporary inconsistency can occur when you first create a firewall and when you make changes to an existing firewall. Generally, any inconsistencies of this type last only a few seconds.

When you add a TLS inspection configuration to an existing firewall, Network Firewall interrupts traffic flows that match the criteria defined by the TLS inspection configuration scope configuration. Network Firewall will begin SSL/TLS decryption and inspection for new connections to the firewall.

Changes to stateful rules are applied only to new traffic flows. Other firewall changes, including changes to stateless rules, are applied to all network packets.

Topics

- [Common rule group settings in AWS Network Firewall](#)
- [Setting rule group capacity in AWS Network Firewall](#)
- [Defining rule actions in AWS Network Firewall](#)
- [Working with stateful rule groups in AWS Network Firewall](#)
- [Working with stateless rule groups in AWS Network Firewall](#)

Common rule group settings in AWS Network Firewall

Every Network Firewall rule group has the following top-level settings:

- **Type** – Whether the rule group is stateless or stateful.
- **Name** – Identifier for the rule group. You assign a unique name to every rule group. You can't change the name of a rule group after you create it.
- **Description** – Optional additional information about the rule group. Fill in any information that might help you remember the purpose of the rule group and how you want to use it. The description is included in rule group lists in the console and through the APIs.
- **Capacity** – Limit on the processing requirements for the rule group. You can't change this setting after you create the rule group. For more information, including how to estimate your required capacity for a rule group, see [Setting rule group capacity in AWS Network Firewall](#).

- **Rules** – Set of packet inspection criteria used in the rule group. Rules in a rule group are either stateless or stateful, depending on the rule group type.
- **Encryption options** (Optional) – Network Firewall encrypts and decrypts Network Firewall resources, to protect against unauthorized access. By default, Network Firewall uses AWS owned keys for this. If you want to use your own keys, you can configure customer managed keys from AWS Key Management Service and provide them to Network Firewall. For information about this option, see [Encryption at rest with AWS Key Management Service](#).
- **Tags** – Zero or more key-value tag pairs. A tag is a label that you assign to an AWS resource. You can use tags to search and filter your resources and to track your AWS costs. For more information, see [Tagging AWS Network Firewall resources](#).

Setting rule group capacity in AWS Network Firewall

AWS Network Firewall uses capacity settings to calculate and manage the processing requirements for its rules groups and firewall policies. Each rule group must have a capacity setting that's fixed at creation. When you reference a rule group from a firewall policy, Network Firewall reserves the rule group's capacity in the policy, increasing the total capacity that's used by the policy.

Using the **consumed capacity** fields in the console, you can also describe a rule group or a policy to find out how much of the rule group or policy capacity is currently in use.

For information about the maximum capacity settings for rule groups and firewall policies, see [AWS Network Firewall quotas](#).

You can't change or exceed a rule group's capacity when you make changes to it, so when you set the rule group's capacity, leave room for it to grow.

Stateless rule group capacity

Estimate a stateless rule group's capacity as the sum of the capacities of the rules that you expect to have in it.

The capacity required for a single rule is the product of the complexity values of all of its match settings.

- A match setting with no criteria specified has a complexity value of 1. Through the console, the **All** and **Any** settings are equivalent to providing no criteria, and they have a complexity value of 1.

- A match setting with criteria specifications has a complexity value equal to the number of specifications in the setting. For example, a protocol specification set to UDP and a source specification set to `10.0.0.0/24` each have a value of 1. A protocol set to UDP, TCP has a value of 2 and a source set to `10.0.0.0/24`, `10.0.1.0/24`, `10.0.2.0/24` has a value of 3.

The following lists example calculations of stateless rule capacity requirements.

- A rule with protocol that specifies the two settings UDP, TCP and source with the three settings `10.0.0.0/24`, `10.0.1.0/24`, `10.0.2.0/24` and single or no specifications for the other match settings has a capacity requirement of 6.
- A rule with a protocol that specifies 30 different protocols, a source with 3 settings, and single or no specifications for the other match settings has a capacity requirement of 90.
- A rule with a protocol that specifies 30 different protocols, a source with 3 settings, a destination with 5 settings, and single or no specifications for the other match settings has a capacity requirement of $(30*3*5) = 450$.

To calculate the capacity of a rule group, add the capacity requirements of all rules that you expect to have in the rule group during its lifetime. You can't change this setting after you create the rule group.

The maximum capacity setting for a stateless rule group is 30,000.

Stateful rule group capacity

Estimate a stateful rule group's capacity as the number of rules that you expect to have in it during its lifetime. You can't change this setting after you create the rule group.

The maximum capacity setting for a stateful rule group is 30,000.

Defining rule actions in AWS Network Firewall

The rule action setting tells AWS Network Firewall how to handle a packet that matches the rule's match criteria.

Actions for stateless rules

The action options for stateless rules are the same as for the firewall policy's default stateless rule actions.

You are required to specify one of the following options:

- **Pass** – Discontinue all inspection of the packet and permit it to go to its intended destination.
- **Drop** – Discontinue all inspection of the packet and block it from going to its intended destination.
- **Forward to stateful rules** – Discontinue stateless inspection of the packet and forward it to the stateful rule engine for inspection.

Additionally, you can optionally specify a named custom action to apply. For this action, Network Firewall assigns a dimension to Amazon CloudWatch metrics with the name set to `CustomAction` and a value that you specify. For more information, see [AWS Network Firewall metrics in Amazon CloudWatch](#).

After you define a named custom action, you can use it by name in the same context as where you defined it. You can reuse a custom action setting among the rules in a rule group and you can reuse a custom action setting between the two default stateless custom action settings for a firewall policy.

Stateful actions

The actions that you specify for your stateful rules help determine the order in which the Suricata stateful rules engine processes them. Network Firewall supports the Suricata rule actions `pass`, `drop`, `reject`, and `alert`. By default, the engine processes rules in the order of `pass` action, `drop` action, `reject` action, and then finally `alert` action. Within each action, you can set a priority to indicate processing order. For more information, see [Managing evaluation order for Suricata compatible rules in AWS Network Firewall](#).

Stateful rules can send alerts to the firewall's logs, if you have logging configured. To see the alerts, you must enable logging for the firewalls that use the rules. Logging incurs additional costs. For more information, see [Logging network traffic from AWS Network Firewall](#).

The options for stateful action settings vary by rule type.

Standard rules and Suricata compatible strings

You specify one of the following action options for both the rules that you provide in Suricata compatible strings and the rules that you specify using the standard stateless rules interface in Network Firewall. These options are a subset of the action options that are defined by Suricata. For more information, see [Working with stateful rule groups in AWS Network Firewall](#).

- **Pass** – Discontinue inspection of the matching packet and permit it to go to its intended destination. Rules with pass action are evaluated before rules with other action settings.
- **Drop or Alert**– Evaluate the packet against all rules with drop or alert action settings. If the firewall has alert logging configured, send a message to the firewall's alert logs for each matching rule. The first log entry for the packet will be for the first rule that matched the packet.

After all rules have been evaluated, handle the packet according to the the action setting in the first rule that matched the packet. If the first rule has a drop action, block the packet. If it has an alert action, continue evaluation.

- **Reject** – Drop traffic that matches the conditions of the stateful rule and send a TCP reset packet back to sender of the packet. A TCP reset packet is a packet with no payload and a RST bit contained in the TCP header flags. Reject is available only for TCP traffic. This option doesn't support FTP and IMAP protocols.

Note

Matching a drop or alert rule for a packet doesn't necessarily mean the end of rule processing for that packet. The engine continues evaluating other rules for matches. For example, if there's a drop match that drops a packet, the packet can still go on to match an alert rule that generates alert logs. Matching an alert rule also doesn't imply a pass. The packet can go on to match a drop rule, and drop the packet after it's previously matched an alert rule.

For information about what you can do to manage the evaluation order of your stateful rules, see [Managing evaluation order for Suricata compatible rules in AWS Network Firewall](#).

Domain lists

The domain list rule group has one action setting at the rule group level. You specify one of the following options:

- **Allow** – Indicates that the domain name list is to be used as an allow list for all traffic that matches the specified protocols. For matching packets, discontinue inspection of the packet and permit it to pass to its intended destination. For non-matching packets, discontinue inspection of the packet, block it from going to its intended destination, and send a message to the firewall's alert logs if the firewall has alert logging configured.

- **Deny** – Indicates that the domain name list is to be used as a deny list for traffic that matches the specified protocols. For matching packets, discontinue inspection of the packet, block it from going to its intended destination, and send a message to the firewall's alert logs if the firewall has alert logging configured. For non-matching packets, take no action.

Working with stateful rule groups in AWS Network Firewall

A stateful rule group is a rule group that uses Suricata compatible intrusion prevention system (IPS) specifications. Suricata is an open source network IPS that includes a standard rule-based language for stateful network traffic inspection. AWS Network Firewall supports Suricata version 7.0.

Stateful rule groups have a configurable top-level setting called `StatefulRuleOptions`, which contains the `RuleOrder` attribute. You can set this in the console when you create a rule group, or in the API under `StatefulRuleOptions`. You can't change the `RuleOrder` after the rule group is created.

You can enter any stateful rule in Suricata compatible strings. For standard Suricata rules specifications and for domain list inspection, you can alternately provide specifications to Network Firewall and have Network Firewall create the Suricata compatible strings for you.

As needed, depending on the rules that you provide, the stateful engine performs deep packet inspection (DPI) of your traffic flows. DPI inspects and processes the payload data within your packets, rather than just the header information.

The rest of this section provides requirements and additional information for using Suricata compatible rules with Network Firewall. For full information about Suricata, see the Suricata website at [Suricata](#) and the [Suricata User Guide](#). AWS Network Firewall supports Suricata version 7.0.

Suricata version upgrade to 7.0

AWS Network Firewall supports Suricata version 7.0. Network Firewall upgraded from Suricata version 6.0.9 to 7.0 in November of 2024.

Note

For full information about the upgrade from version 6.0.9, see [Upgrading 6.0 to 7.0](#).

The following are examples of the changes in this upgrade:

- PCRE 1 rule format is no longer supported, and has been replaced with PCRE2.
- When you specify a sticky buffer in a rule, it needs to be immediately followed by the payload keywords. For example, keywords such as `dns.query` and `tls.sni` must be followed by a content modifier.
- Keywords that use ranges, such as `ip` now require the range to be specified with the format *min:max*.

Topics

- [Limitations and caveats for stateful rules in AWS Network Firewall](#)
- [Best practices for writing Suricata compatible rules for AWS Network Firewall](#)
- [Managing evaluation order for Suricata compatible rules in AWS Network Firewall](#)
- [IP set references in Suricata compatible AWS Network Firewall rule groups](#)
- [Geographic IP filtering in Suricata compatible AWS Network Firewall rule groups](#)
- [Options for providing stateful rules to AWS Network Firewall](#)
- [Creating a stateful rule group](#)
- [Updating a stateful rule group](#)
- [Deleting a stateful rule group](#)
- [Examples of stateful rules for Network Firewall](#)

Limitations and caveats for stateful rules in AWS Network Firewall

AWS Network Firewall stateful rules are Suricata compatible. Most Suricata rules work out of the box with Network Firewall. Your use of Suricata rules with Network Firewall has the restrictions and caveats listed in this section.

Suricata features that Network Firewall doesn't support

The following Suricata features are not supported by Network Firewall:

- Datasets. The keywords `dataset` and `datarep` aren't allowed.
- ENIP/CIP keywords.
- File extraction. File keywords aren't allowed.
- FTP-data protocol detection.
- IP reputation. The `iprep` keyword is not allowed.

- Lua scripting.
- Rules actions except for pass, drop, reject, and alert. Pass, drop, reject, and alert are supported. For additional information about stateful rule actions, see [Stateful actions](#).
- SCTP protocol.
- Thresholding.
- IKEv2 protocol.

Suricata features that Network Firewall supports with caveats

The following Suricata features have caveats for use with Network Firewall:

- If you want a rule group to use settings for HOME_NET and EXTERNAL_NET that are different from those that are set for the firewall policy, you must explicitly set both of these variables.
 - In a firewall policy's variables, you can set a custom value for HOME_NET. The default HOME_NET setting is the CIDR of the inspection VPC. The policy's EXTERNAL_NET setting is always the negation of the policy's HOME_NET setting. For example, if the HOME_NET is 11.0.0.0, the EXTERNAL_NET is set to !11.0.0.0.
 - In a rule group's variables, you can set custom values for both HOME_NET and EXTERNAL_NET. If you explicitly set rule group variables, those are used. Otherwise, rule group variables inherit their settings from the corresponding policy variables.

This means that, if you don't specify the rule group's EXTERNAL_NET, it inherits the setting from the policy's EXTERNAL_NET setting, regardless of the value of the rule group's HOME_NET setting.

For example, say you set the rule group's HOME_NET to 10.0.0.0, and the firewall policy's HOME_NET to 11.0.0.0. If you don't set the rule group's EXTERNAL_NET, then Network Firewall sets it to !11.0.0.0, based on the policy's HOME_NET setting.

- The AWS Network Firewall stateful inspection engine supports inspecting inner packets for tunneling protocols such as Generic Routing Encapsulation (GRE). If you want to block the tunneled traffic, you can write rules against the tunnel layer itself or against the inner packet. Due to the service inspecting the different layers, you might see flows and alerts for the packets within the tunnel.
- To create a rule that requires a variable, you must specify the variable in the rule group. Without the required variables, the rule group isn't valid. For an example of a rule group that's configured with variables, see [Stateful rules examples: rule variables](#).

- In payload keywords, the `pcr` keyword is only allowed with `content`, `tls.sni`, `http.host`, and `dns.query` keywords.
- The `priority` keyword is not supported for rule groups that evaluate rules using strict evaluation order.
- When you use a stateful rule with a layer 3 or 4 protocol such as IP or TCP, and you don't include any flow state context, for example `flow: not_established`, then Suricata treats this rule as an IP-only rule. Suricata only evaluates IP-only rules for the first packet in each direction of the flow. For example, Suricata will process the following rule as an IP-only rule:

```
pass tcp $HOME_NET any -> [10.0.0.0/8] $HTTPS_PORTS (sid: 44444; rev:2;)
```

However, if the destination IP contains a `!`, then Suricata treats this as per the protocol specified in the rule. Suricata will process the following rule as a TCP rule.

```
pass tcp $HOME_NET any -> [!10.0.0.0/16] $HTTPS_PORTS (sid: 44444; rev:2;)
```

Best practices for writing Suricata compatible rules for AWS Network Firewall

A stateful rule group is a rule group that uses Suricata compatible intrusion prevention system (IPS) specifications. Suricata is an open source network IPS that includes a standard rule-based language for stateful network traffic inspection. AWS Network Firewall supports Suricata version 7.0. You can enter any stateful rule in Suricata compatible strings.

When you write your stateful rules, verify the configuration of the firewall policy where you intend to use them, to make sure that all of your rules evaluate as you want them to. For information about how AWS Network Firewall handles network traffic and when it sends it to the stateful engine, see [How AWS Network Firewall filters network traffic](#).

For example, many stateful rules rely on seeing a complete bidirectional traffic flow for correct evaluation, such as rules with options like `flow: established`. To use rules like this, you must configure your stateless rules and default actions to ensure forwarding of traffic for both directions to the stateful engine. For information about these action options, see [Defining rule actions in AWS Network Firewall](#).

Managing evaluation order for Suricata compatible rules in AWS Network Firewall

You can configure and manage the evaluation order of the rules in your Suricata compatible stateful rule groups.

All of your stateful rule groups are provided to the rule engine as Suricata compatible strings. Suricata can evaluate stateful rule groups by using the default rule group ordering method, or you can set an exact order using the *strict* ordering method. We recommend that you use strict order because it lets you specify the exact order that you'd like the stateful engine to evaluate your rules. The settings for your rule groups must match the settings for the firewall policy that they belong to.

Action order

If your firewall policy is set up to use action order rule group ordering, the action order by which Suricata evaluates stateful rules is determined by the following settings, listed in order of precedence:

1. The Suricata action specification. This takes highest precedence.

Actions are processed in the following order:

- a. pass
- b. drop
- c. reject
- d. alert

Note

If a packet within a flow matches a rule containing pass action, then Suricata doesn't scan the other packets in that flow and it passes the unscanned packets.

For more information about the action specification, see [Suricata.yaml: Action-order](#) in the [Suricata User Guide](#).

2. The Suricata `priority` keyword. Within a specific action group, you can use the `priority` setting to indicate the processing order. By default, Suricata processes from the lowest numbered `priority` setting on up. The `priority` keyword has a mandatory numeric value ranging from 1 to 255. Note that the `priority` keyword is only valid using the default action order.

For more information about priority, see [Suricata.yaml: Action-order](#) in the [Suricata User Guide](#).

For example, Suricata evaluates all pass rules before evaluating any drop, reject, or alert rules by default, regardless of the value of priority settings. Within all pass rules, if priority keywords are present, Suricata orders the processing according to them.

The protocol layer does not impact the rule evaluation order by default. If you want to avoid matching against lower-level protocol packets before higher-level application protocols can be identified, consider using the `flow` keyword in your rules. This is needed because, for example, a TCP rule might match on the first packet of a TCP handshake before the stateful engine can identify the application protocol. For information about the `flow` keyword, see [Flow Keywords](#).

For examples of default rule order management, see [Stateful rules examples: manage rule evaluation order](#).

For additional information about evaluation order for stateful rules, see the following topics in the [Suricata User Guide](#):

- [Suricata.yaml: Action-order](#)
- [Meta Keywords: priority](#)

Strict evaluation order

If your firewall policy is set up to use strict ordering, Network Firewall allows you the option to manually set a *strict* rule group order. With strict ordering, the rule groups are evaluated by order of priority, starting from the lowest number, and the rules in each rule group are processed in the order in which they're defined.

When you choose **Strict** for your rule order, you can choose one or more **Default actions**. Note that this does not refer to default action rule ordering, but rather, to the default actions that Network Firewall takes when following your strict, or exact, rule ordering. The default actions are as follows:

Drop actions

If you have rules that match application layer data, such as those that evaluate HTTP headers, a default drop action might trigger earlier than you want. This can happen when the data that your rules match against spans multiple packets, because a default drop action can apply to a single

packet. For this case, don't choose any default drop action and instead use drop rules that are specific to the application layer.

Choose none or one. You can't choose both.

- **Drop all** – Drops all packets.
- **Drop established** – Drops only the packets that are in established connections. This allows the layer 3 and 4 connection establishment packets that are needed for the upper-layer connections to be established, while dropping the packets for connections that are already established. This allows application-layer *pass* rules to be written in a default-deny setup without the need to write additional rules to allow the lower-layer handshaking parts of the underlying protocols.

Choose this option when using strict order for your own domain list rule groups because Network Firewall requires an established connection in order to evaluate whether to pass or drop the packets for domain lists.

For other protocols, such as UDP, Suricata considers the connection established only after seeing traffic from both sides of the connection.

Alert actions

Choose none, one, or both.

- **Alert all** - Logs an ALERT message on all packets. This does not drop packets, but alerts you to what would be dropped if you were to choose **Drop all**.
- **Alert established** - Logs an ALERT message on only the packets that are in established connections. This does not drop packets, but alerts you to what would be dropped if you were to choose **Drop established**.

For more information about logging network traffic, see [Logging network traffic from AWS Network Firewall](#).

IP set references in Suricata compatible AWS Network Firewall rule groups

An *IP set reference* is a Network Firewall rule group variable that references a set of IP addresses or CIDR blocks contained in an AWS resource, such as an Amazon Virtual Private Cloud prefix list. IP set references enable you to dynamically use IP addresses or CIDRs from another AWS service in your Suricata compatible rules. When you create, update, or delete the IP sets that you

reference in your rules, Network Firewall automatically updates the rules with the changes. For example, if you add five CIDRs to an IP set resource that you're referencing in a rule, then the rule will automatically include the five CIDRs that you added to the resource.

Network Firewall currently supports the following AWS resources as IP set references:

- **Amazon VPC prefix lists.** For information about referencing Amazon VPC prefix lists in your rule groups, see the following section [Referencing Amazon VPC prefix lists](#).
- **Resource groups.** For information about referencing resource groups in your rule groups, see following section [Referencing resource groups](#).

For an example of a rule that uses an IP set reference, see [Stateful rules examples: IP set reference](#).

For more information about adding IP sets to your Suricata compatible rule groups via the console, see the [the section called “Creating a stateful rule group”](#) procedure.

Limits for IP set references

The following limits apply to IP set references:

- Maximum of five IP set references per rule group. You can use IP set references in addition to IP set variables or port variables in a rule group. Only IP set references count against this limit.
- Maximum of 1,000,000 CIDRs - You can use a maximum of 1,000,000 CIDRs in all of the IP set references used in a single firewall. If you exceed this limit, then Network Firewall includes only the first 1,000,000 CIDRs from your referenced IP set resources. Network Firewall calculates CIDRs differently for prefix lists and resource groups:
 - Prefix lists – Network Firewall takes an aggregated account of the CIDRs in each referenced IP set.
 - Resource groups – Network Firewall calculates the number of IP addresses associated with all of the resources in the group, such as all of the IP addresses associated with an Amazon EC2 instance, both public and private.

Referencing Amazon VPC prefix lists

A *prefix list* is a set of one or more CIDR block entries that you can use to configure security groups, routing tables, and transit gateways in Amazon VPC. A reference to a prefix list helps you to simplify the management of the CIDR blocks in your rules. If you frequently use the same CIDRs across multiple rules, you can manage those CIDRs in a single prefix list, instead of repeatedly

referencing the same CIDRs in each rule. If you need to remove a CIDR block, you can remove its entry from the prefix list instead of removing the CIDR from every affected rule.

For more information about Amazon VPC prefix lists, see [Group CIDR blocks using managed prefix lists](#) in the *Amazon VPC User Guide*.

Referencing resource groups

A *tag-based resource group* is a collection of AWS resources whose membership in a resource group is based on tags. Tags are key value metadata that you associated with a resource type, such as an Amazon EC2 instance. Similar to prefix lists, a reference to a resource group helps you to simplify the management of the IP addresses in your rules. If you frequently want to reference the IP addresses of the same set of resources, you can manage those IPs in a single resource group, instead of repeatedly referencing the same IPs in each rule. Network Firewall constantly checks for resources that match the resource group grouping criteria in your account, and then resolves IPs of the matching resources in the rule. If you need to remove a set of IP addresses, you can remove the tagged resource type from the resources group instead of removing the IP from every affected rule.

For more information about using resource groups in Network Firewall, see [Using tag-based resource groups in Network Firewall](#).

Geographic IP filtering in Suricata compatible AWS Network Firewall rule groups

Use Geographic IP filtering to filter on country codes for the IP addresses of network traffic. You can use Geographic IP filtering with Suricata compatible rule strings and with the standard Network Firewall stateful rule groups.

Suricata supports filtering for source and destination IPs. You can filter on either of these types by itself, by specifying `dst` or `src`. You can filter on the two types together with AND or OR logic, by specifying both or any. For more information see the Suricata `geoip` keyword documentation at [IP Keywords: geoip](#).

To use a Geographic IP filter, you provide the `geoip` keyword, the filter type, and the country codes for the countries that you want to filter for, for example `geoip:dst,CN,US;`. For additional examples, see [the section called "Geographic IP filter"](#).

For more information about adding Geographic IP filtering to your Suricata compatible rule groups via the console, see the [the section called "Creating a stateful rule group"](#) procedure.

IPv4 and IPv6 support

The Network Firewall implementation of Suricata `geoip` provides support for IPv6 and IPv4. This is an expansion of the support provided by Suricata.

MaxMind IP geolocation

Suricata determines the location of requests using MaxMind GeoIP databases. MaxMind reports very high accuracy of their data at the country level, although accuracy varies according to factors such as country and type of IP. For more information about MaxMind, see [MaxMind IP Geolocation](#).

If you think any of the geographic IP data is incorrect, you can submit a correction request to Maxmind at [MaxMind Correct GeoIP Data](#).

Country codes

IP geolocation uses the alpha-2 country codes from the International Organization for Standardization (ISO) 3166 standard.

- You can search country codes at the ISO website, at [ISO Online Browsing Platform \(OBP\)](#).
- You can also find them listed on Wikipedia, at [ISO 3166-2](#).

Network Firewall only allows you to save Geographic IP filter rules that have valid country codes.

Options for providing stateful rules to AWS Network Firewall

When you create a Network Firewall stateful rule group from Suricata compatible rules, you can provide the rules to the rule group creation operation in one of the following ways:

- **Rule strings that are written in Suricata compatible syntax** – When you use this option, Network Firewall passes your rule strings to Suricata for processing.
- **Domain list rule specification** – With this option, Network Firewall translates your rule specification into Suricata compatible rules and then passes the resulting rule strings to Suricata for processing.
- **Standard, simple rule group specification** – With this option, Network Firewall translates your specification into Suricata compatible rules and then passes the resulting rule strings to Suricata for processing.

The sections that follow provide details for each of these options.

Suricata compatible rule strings in AWS Network Firewall

When you use this rule group type, you provide match and action settings in a string, in a Suricata compatible specification. Your specification fully defines what the stateful rules engine looks for in a traffic flow and the action to take on the packets in a flow that matches the inspection criteria.

All rule groups have the common settings that are defined at [the section called “Common rule group settings”](#).

You can provide your Suricata compatible specification to Network Firewall in rules strings or files, depending on how you're accessing Network Firewall.

- **Console** – In the AWS Management Console, provide the rules string in the text box that appears for the stateful rule group option **Import Suricata compatible rules**. For information about using the console to manage your rule group, see [Creating a stateful rule group](#).
- **API** – Through the API, you can provide either the rules or the name of the file that contains the rules. In a file, Suricata compatible rules are usually written one rule per line.

You provide either the file or the rules string in the `RulesString` field within the `RuleGroup` structure when you create or update the rule group. For information, see [CreateRuleGroup](#) in the *AWS Network Firewall API Reference*.

- **CLI** – Through the CLI, you can provide the rules, the name of a file that contains the rules, or the name of a file that contains the rule group structure in JSON format, with the rules defined in that.

The following listing shows the syntax for providing the rules in a file. To use a command like this, substitute in your new rule group name, its calculated capacity, and the JSON rules file name.

```
aws network-firewall create-rule-group --rule-group-name <ruleGroupName> --capacity
<capacityCalculation> --type STATEFUL --rules <rules file name>
```

Stateful domain list rule groups in AWS Network Firewall

AWS Network Firewall supports domain name stateful network traffic inspection. You can create allow lists and deny lists with domain names that the stateful rules engine looks for in network traffic.

All rule groups have the common settings that are defined at [the section called “Common rule group settings”](#).

General settings

A domain list rule group has the following general settings.

- **Action** – Defines whether Network Firewall allows traffic that matches the rule match settings. Valid values for domain rules are Allow and Deny. For Allow, traffic of the specified protocol type that doesn't match the domain specifications is denied. For more information about actions, see [Defining rule actions in AWS Network Firewall](#).
- **(Optional) HOME_NET rule group variable** – Used to expand the local network definition beyond the CIDR range of the VPC where you deploy Network Firewall. For additional information about this setting, see [Domain list inspection for traffic from outside the deployment VPC](#).

See the caveats for the HOME_NET and EXTERNAL_NET settings at [Suricata features that Network Firewall supports with caveats](#).

Note

The console doesn't currently allow entry of all rule group variables. To specify other rule group variables, use one of the APIs or AWS CloudFormation. For information, see [StatefulRule](#) in the *AWS Network Firewall API Reference* and [AWS::NetworkFirewall::RuleGroup StatefulRule](#) in the *AWS CloudFormation User Guide*.

Match settings

A domain list rule group has the following match settings. These specify what the Network Firewall stateful rules engine looks for in a packet. A packet must satisfy all match settings to be a match.

- **Domain list** – List of strings specifying the domain names that you want to match. A packet must match one of the domain specifications in the list to be a match for the rule group. Valid domain name specifications are the following:
 - Explicit names. For example, `abc.example.com` matches only the domain `abc.example.com`.
 - Names that use a domain wildcard, which you indicate with an initial `'.'`. For example, `.example.com` matches `example.com` and matches all subdomains of `example.com`, such as `abc.example.com` and `www.example.com`.

- **Protocols** – You can inspect HTTP or HTTPS protocols, or both.

For HTTPS traffic, Network Firewall uses the Server Name Indication (SNI) extension in the TLS handshake to determine the hostname, or domain name, that the client is trying to connect to. For HTTP traffic, Network Firewall uses the HTTP host header to get the name. In both cases, Network Firewall doesn't pause connections to do out-of-band DNS lookups. It uses the SNI or host header, not the IP addresses, when evaluating domain list rule groups. If you want to inspect IP addresses, to mitigate situations where the SNI or host headers have been manipulated, write separate rules for that and use them in conjunction with or in place of your domain list rules.

For examples of domain list specifications and the Suricata compatible rules that Network Firewall generates from them, see [Stateful rules examples: domain list rules](#).

Domain list inspection for traffic from outside the deployment VPC

To use domain name filtering for traffic from outside the VPC where you've deployed Network Firewall, you must manually set the HOME_NET variable for the rule group. The most common use case for this is a central firewall VPC with traffic coming from other VPCs through a transit gateway.

By default, domain list inspection uses a HOME_NET that is set to the CIDR range of the VPC where Network Firewall is deployed. Only traffic from that range is passed through the domain list filtering. To filter traffic from outside the deployment VPC, you must provide a HOME_NET setting that includes the other CIDR ranges that you want to inspect, along with the CIDR range of the VPC where Network Firewall is deployed.

For example, say that the VPC where you deploy Network Firewall has the CIDR range 192.0.2.0/24. In addition to the traffic for that VPC, you want to filter traffic for two other VPCs that have CIDR ranges 10.0.0.0/16 and 10.1.0.0/16. You're using a domain list rule group named domains.

The following command line call retrieves the JSON listing for the rule group:

```
aws network-firewall describe-rule-group --type STATEFUL \  
--rule-group-name domains --region us-west-2
```

The following shows the example JSON response. This rule group has only RulesSource defined, which contains the domain list inspection specifications.

```
{
```

```

"UpdateToken": "a4648a25-e315-4d17-8553-283c2eb33118",
"RuleGroup": {
  "RulesSource": {
    "RulesSourceList": {
      "Targets": [
        ".example.com",
        "www.example.org"
      ],
      "TargetTypes": [
        "HTTP_HOST",
        "TLS_SNI"
      ],
      "GeneratedRulesType": "DENYLIST"
    }
  }
},
"RuleGroupResponse": {
  "RuleGroupArn": "arn:aws:network-firewall:us-west-2:111122223333:stateful-
rulegroup/domains",
  "RuleGroupName": "domains",
  "RuleGroupId": "f3333333-fb99-11c1-bbe3-1d1caf1d1111",
  "Type": "STATEFUL",
  "Capacity": 100,
  "RuleGroupStatus": "ACTIVE",
  "Tags": []
}
}

```

Variable settings are defined for a rule group in a `RuleVariables` setting. This rule group currently has no `HOME_NET` variable declaration, so we know that `HOME_NET` is set to the default. In our example case, it's `192.0.2.0/24`.

To add CIDR ranges to the `HOME_NET` setting, we update the rule group with our variable declaration. The following shows a file named `variables.json` that contains the rule group JSON with the added variables settings:

```

{
  "RuleVariables": {
    "IPSets": {
      "HOME_NET": {
        "Definition": [
          "10.0.0.0/16",
          "10.1.0.0/16",

```



```

        "192.0.2.0/24"
      ]
    }
  },
  "RulesSource": {
    "RulesSourceList": {
      "Targets": [
        ".example.com",
        "www.example.org"
      ],
      "TargetTypes": [
        "HTTP_HOST",
        "TLS_SNI"
      ],
      "GeneratedRulesType": "DENYLIST"
    }
  }
}

```

The following command uses the `variables.json` file to update the rule group definition with the correct `HOME_NET` settings:

```

aws network-firewall update-rule-group \
--rule-group-arn arn:aws:network-firewall:us-west-2:111122223333:stateful-rulegroup/
domains \
--update-token a4648a25-e315-4d17-8553-283c2eb33118 \
--rule-group file://variables.json \
--region us-west-2

```

The following shows an example response to the call:

```

{
  "UpdateToken": "32ebfb82-40a2-4896-b34d-91dada978f67",
  "RuleGroupResponse": {
    "RuleGroupArn": "arn:aws:network-firewall:us-west-2:111122223333:stateful-
rulegroup/domains",
    "RuleGroupName": "domains",
    "RuleGroupId": "f3333333-fb99-11c1-bbe3-1d1caf1d1111",
    "Type": "STATEFUL",
    "Capacity": 100,
    "RuleGroupStatus": "ACTIVE",
    "Tags": []
  }
}

```

```
}  
}
```

If we retrieve the domains rule group again, we see that the rule group has the added variable definition:

```
aws network-firewall describe-rule-group --type STATEFUL \  
--rule-group-name domains --region us-west-2
```

The response JSON contains the added variable:

```
{  
  "UpdateToken": "42ffac91-20b5-5512-a24c-85cbca797e23",  
  "RuleGroup": {  
    "RuleVariables": {  
      "IPSets": {  
        "HOME_NET": {  
          "Definition": [  
            "10.0.0.0/16",  
            "10.1.0.0/16",  
            "192.0.2.0/24"  
          ]  
        }  
      }  
    },  
    "RulesSource": {  
      "RulesSourceList": {  
        "Targets": [  
          ".example.com",  
          "www.example.org"  
        ],  
        "TargetTypes": [  
          "HTTP_HOST",  
          "TLS_SNI"  
        ],  
        "GeneratedRulesType": "DENYLIST"  
      }  
    }  
  },  
  "RuleGroupResponse": {  
    "RuleGroupArn": "arn:aws:network-firewall:us-west-2:111122223333:stateful-  
rulegroup/domains",  
    "RuleGroupName": "domains",
```

```
    "RuleGroupId": "f3333333-fb99-11c1-bbe3-1d1caf1d1111",
    "Type": "STATEFUL",
    "Capacity": 100,
    "RuleGroupStatus": "ACTIVE",
    "Tags": []
  }
}
```

Standard stateful rule groups in AWS Network Firewall

AWS Network Firewall supports easy entry for standard stateful rules for network traffic inspection. The match criteria for this stateful rule type is similar to the Network Firewall stateless rule.

All rule groups have the common settings that are defined at [the section called “Common rule group settings”](#).

General settings

A stateful basic rule has the following general settings.

- **Action** – Defines how Network Firewall handles a packet that matches the rule match settings. Valid values are pass, drop, reject, and alert. For more information about actions, see [Defining rule actions in AWS Network Firewall](#).

Match settings

A basic stateful rule has the following match settings. These specify what the Network Firewall stateful rules engine looks for in a packet. To be a match, a packet must satisfy all of the match settings in the rule.

- **Protocol** – Transport protocol. Choose the protocol that you want to inspect. For all protocols, you can use IP, because all traffic on AWS and on the internet is IP.
- **Source IP** – Source IP addresses and ranges. If specified, a packet must come from a source address that's included in this list in order to match.
- **Source port** – Source ports and port ranges. If specified, a packet must have a source port that's included in this list in order to match.
- **Destination IP** – Destination IP addresses and ranges. If specified, a packet must have a destination address that's included in this list in order to match.

- **Destination port** – Destination ports and port ranges. If specified, a packet must have a destination port that's included in this list in order to match.
- **Traffic direction** – Direction of traffic flow. Valid settings are Any and Forward. Forward matches packets whose origination matches the rule's source settings and whose destination matches the rule's destination settings. Any matches the forward match, and also matches packets whose origination matches the rule's destination settings, and whose destination matches the rule's source settings.
- **Rule options** – Define the specifics of the rule, in keyword, settings pairs.

Note

The console doesn't currently allow entry of rule options. Rule options are usually required for complete specification of this rule type. If you need to specify rule options, use one of the APIs or AWS CloudFormation. For information, see [StatefulRule](#) in the *AWS Network Firewall API Reference* and [AWS::NetworkFirewall::RuleGroup StatefulRule](#) in the *AWS CloudFormation User Guide*.

For an example rule specification and the Suricata compatible rule that Network Firewall generates from it, see [Stateful rules examples: standard stateful rule groups](#).

Creating a stateful rule group

This section provides guidance for creating a stateful rule group.

To create a stateful rule group

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, under **Network Firewall**, choose **Network Firewall rule groups**.
3. Choose **Create Network Firewall rule group**.
4. Under **Choose rule group type**, for the **Rule group format**, choose **Stateful rule group**.

For **Rule evaluation order**, choose the way that your stateful rules are ordered for evaluation:


- Choose **Strict order** (recommended) to provide your rules in the order that you want them to be evaluated. You can then choose one or more default actions for packets that don't match any rules.

- Choose **Action order** to have the stateful rules engine determine the evaluation order of your rules. The default action for this rule order is **Pass**, followed by **Drop**, **Reject**, and **Alert** actions. This option was previously named **Default** order.

For more information about stateful default actions for rule groups, see [Action order](#).

For more information about stateful rule groups, see [Working with stateful rule groups in AWS Network Firewall](#).

5. Choose **Next**.
6. Enter a **Name** to identify this rule group.

 **Note**

You can't change the name after you create the rule group.

7. (Optional) Enter a **Description** for the rule group to help you identify it among your other resources.
8. For **Capacity**, set the maximum capacity you want to allow for the stateful rule group, up to the maximum of 30,000. You can't change this setting after you create the rule group. For information about how to calculate this, see [Setting rule group capacity in AWS Network Firewall](#). For information about the maximum setting, see [AWS Network Firewall quotas](#).
9. Choose **Next**.
10. Select the type of rule group that you want to add, from the **Stateful rule group options**. The rest of your rule group specifications depend on the option you choose.


 **Note**

If you need to specify options that aren't available through the console, you can use one of the APIs or AWS CloudFormation. For information, see [StatefulRule](#) in the *AWS Network Firewall API Reference* and [AWS::NetworkFirewall::RuleGroup StatefulRule](#) in the *AWS CloudFormation User Guide*.

- (Option) **Standard stateful rule** – Entry form for a basic Suricata rule.

For each rule that you want in your rule group, specify the following information and then choose **Add rule**. Your added rules are listed in the **Rules** list.

- Choose the protocol and source and destination settings for your rule.
- For **Traffic direction**, choose whether to apply the rule to any direction or only for traffic that flows forward, from the specified source to the specified destination.

 **Note**

Network Firewall doesn't automatically add the direction keyword to `_server`, and will inspect all the packets in the flow, irrespective of the flow state.

- For **Action**, select the action that you want Network Firewall to take when a packet matches the rule settings. For information on these options, see [Stateful actions](#).

To define IP sets and ports as variables that you can reference in your rules:

- In the **Rule variables** section, enter variables and values for **IP set variables** and **Port variables**.

To add one or more references to IP set resources, such as Amazon VPC prefix lists, that you can use as variables in your rules:

- In the **IP set reference** section, enter a **IP set variable name** and select an **IP set reference ID**. The **IP set reference ID** corresponds to the resource ID of the IP set Amazon Resource Name (ARN) that you want to reference. Network Firewall currently supports Amazon VPC prefix lists and resource groups as IP set references. For more information about working with IP set references in Network Firewall, see [Referencing Amazon VPC prefix lists](#).

For information about these rules, see [Standard stateful rule groups in AWS Network Firewall](#).

- (Option) **Domain list** – Specify the following information.
 - For **Domain name source**, enter the domain names that you want to inspect for, one name specification per line. Valid domain name specifications are the following:
 - Explicit names. For example, `abc.example.com` matches only the domain `abc.example.com`.

- Names that use a domain wildcard, which you indicate with an initial '.'. For example, `.example.com` matches `example.com` and matches all subdomains of `example.com`, such as `abc.example.com` and `www.example.com`.
- For **Protocols**, choose the protocols you want to inspect.
- For **Action**, select the list type that you are creating, either **Allow** or **Deny**. For information on these options, see [Stateful actions](#).

For information about stateful domain name rules, see [Stateful domain list rule groups in AWS Network Firewall](#).

- (Option) **Suricata compatible rule string**

To define IP sets and ports as variables that you can reference in your rules:

- In the **Rule variables** section, enter variables and values for **IP set variables** and **Port variables**.

To add one or more references to IP set resources, such as Amazon VPC prefix lists, that you can use as variables in your rules:

- In the **IP set reference** section, enter a **IP set variable name** and select an **IP set reference ID**. The **IP set reference ID** corresponds to the resource ID of the IP set Amazon Resource Name (ARN) that you want to reference. Network Firewall currently supports Amazon VPC prefix lists and resource groups as IP set references. For more information about working with IP set references in Network Firewall, see [Referencing Amazon VPC prefix lists](#).

Paste your rules into the text box.

11. Choose **Next**.
12. (Optional) On the **Configure advanced settings** page, under **Customer managed key**, toggle the **Customize encryption settings** option to configure your customer managed key. For more information about this option, see [Encryption at rest with AWS Key Management Service](#).
13. Choose **Next**.
14. (Optional) On the **Add tags** page, enter a key and optional value for any tag that you want added to this firewall policy. Tags help you organize and manage your AWS resources. For more information about tagging your resources, see [Tagging AWS Network Firewall resources](#).
15. Choose **Next**.
16. Review the settings that you've provided for the rule group, then choose **Create stateful rule group**.

Your new rule group is added to the list in the **Network Firewall rule groups** page.

To use your rule group in a firewall policy, follow the procedures at [Managing your firewall policy](#).

Updating a stateful rule group

To change your stateful rule group settings, use the following procedure.

To update a stateful rule group

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, under **Network Firewall**, choose **Network Firewall rule groups**.
3. In the **Network Firewall rule groups** page, choose the name of the rule group that you want to update. The rule group's details page appears.
4. In your rule group's details page, in the area that you want to change, choose **Edit**. Follow the prompts to make your updates. The interface varies according to the rule group type. When you're done editing an area, choose **Save** to save your changes in the rule group.

How Network Firewall propagates your changes

When you make any changes to a firewall, including changes to any of the firewall's components, like rule groups, TLS inspection configurations, and firewall policies, Network Firewall propagates the changes everywhere that the firewall is used. Your changes are applied within seconds, but there might be a brief period of inconsistency when the changes have arrived in some places and not in others. For example, if you modify a rule group so that it drops an additional type of packet, for a firewall that uses the rule group, the new packet type might briefly be dropped by one firewall endpoint while still being allowed by another.

This temporary inconsistency can occur when you first create a firewall and when you make changes to an existing firewall. Generally, any inconsistencies of this type last only a few seconds.

When you add a TLS inspection configuration to an existing firewall, Network Firewall interrupts traffic flows that match the criteria defined by the TLS inspection configuration scope configuration. Network Firewall will begin SSL/TLS decryption and inspection for new connections to the firewall.

Changes to stateful rules are applied only to new traffic flows. Other firewall changes, including changes to stateless rules, are applied to all network packets.

Deleting a stateful rule group

To delete a rule group, use the guidance in this section.

Deleting a rule group, TLS inspection configuration, or firewall policy

When you delete a rule group, TLS inspection configuration, or a firewall policy, AWS Network Firewall checks to see if it's currently being referenced. A rule group and TLS inspection configuration can be referenced by a firewall policy, and a firewall policy can be referenced by a firewall. If Network Firewall determines that the resource is being referenced, it warns you. Network Firewall is almost always able to determine whether a resource is being referenced. However, in rare cases, it might not be able to do so. If you need to be sure that the resource that you want to delete isn't in use, check all of your firewalls or firewall policies before deleting it. Note that policies that have associations can't be deleted.

To delete a stateful rule group

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, under **Network Firewall**, choose **Network Firewall rule groups**.
3. In the **Network Firewall rule groups** page, select the name of the rule group that you want to delete, and then choose **Delete**.

How Network Firewall propagates your changes

When you make any changes to a firewall, including changes to any of the firewall's components, like rule groups, TLS inspection configurations, and firewall policies, Network Firewall propagates the changes everywhere that the firewall is used. Your changes are applied within seconds, but there might be a brief period of inconsistency when the changes have arrived in some places and not in others. For example, if you modify a rule group so that it drops an additional type of packet, for a firewall that uses the rule group, the new packet type might briefly be dropped by one firewall endpoint while still being allowed by another.

This temporary inconsistency can occur when you first create a firewall and when you make changes to an existing firewall. Generally, any inconsistencies of this type last only a few seconds.

Changes to stateful rules are applied only to new traffic flows. Other firewall changes, including changes to stateless rules, are applied to all network packets.

Examples of stateful rules for Network Firewall

This section lists examples of Suricata compatible rules that could be used with AWS Network Firewall.

Note

Examples are not intended to be used in your Network Firewall configuration exactly as they are listed.

The examples provide general information and sample rule specifications for common use cases. Before using any rule from these examples or elsewhere, test and adjust it carefully to be sure that it fits your needs. It's your responsibility to ensure that each rule that you use is suited to your specific use case and functioning the way that you want it to.

Stateful rules examples: allow traffic

Note

Before using any example rule, test and adapt it to your needs.

The examples in this section contain examples that allow specified traffic.

Allow access to any ssm. Server Name Indication (SNI) ending with .amazonaws.com

Allows access to any domain that begins with ssm. and ends with .amazonaws.com (<http://amazonaws.com/>).

```
pass tls $HOME_NET any -> $EXTERNAL_NET any (ssl_state:client_hello; tls.sni;
content:"ssm."; startswith; content:".amazonaws.com"; endswith; nocase; flow:
to_server; sid:202308311;)
```

JA3 hash

This rule allows outbound access using a specific JA3 hash

```
pass tls $HOME_NET any -> $EXTERNAL_NET any (msg:"Only allow Curl 7.79.1 JA3";
ja3.hash; content:"27e9c7cc45ae47dc50f51400db8a4099"; sid:12820009;)
```

Outbound requests to checkip.amazonaws.com

These rules only allow outbound requests to the SNI checkip.amazonaws.com (http://checkip.amazonaws.com/) if the server certificate issuer is also Amazon. Requires that your firewall policy uses strict order rule evaluation order.

```
alert tls $HOME_NET any -> $EXTERNAL_NET 443 (ssl_state:client_hello;
  tls.sni; content:"checkip.amazonaws.com"; endswith; nocase; xbits:set,
  allowed_sni_destination_ips, track ip_dst, expire 3600; noalert; sid:238745;)
pass tcp $HOME_NET any -> $EXTERNAL_NET 443 (xbits:isset, allowed_sni_destination_ips,
  track ip_dst; flow: stateless; sid:89207006;)
pass tls $EXTERNAL_NET 443 -> $HOME_NET any (tls.cert_issuer; content:"Amazon";
  msg:"Pass rules do not alert"; xbits:isset, allowed_sni_destination_ips, track ip_src;
  sid:29822;)
reject tls $EXTERNAL_NET 443 -> $HOME_NET any (tls.cert_issuer; content:"="; nocase;
  msg:"Block all other cert issuers not allowed by sid:29822"; sid:897972;)
```

Outbound SSH/SFTP servers with AWS_SFTP banner

These rules only allow outbound access to SSH/SFTP servers that have a banner that includes AWS_SFTP, which is the banner for AWS Transfer Family servers. To check for a different banner, replace AWS_SFTP with the banner you want to check for.

```
pass tcp $HOME_NET any -> $EXTERNAL_NET 22 (flow:stateless; sid:2221382;)
pass ssh $EXTERNAL_NET 22 -> $HOME_NET any (ssh.software; content:"AWS_SFTP";
  flow:from_server; sid:217872;)
drop ssh $EXTERNAL_NET 22 -> $HOME_NET any (ssh.software; content:"!@"; pcre:"/[a-z]/
  i"; msg:"Block unauthorized SFTP/SSH."; flow: from_server; sid:999217872;)
```

Send DNS query including .amazonaws.com to external DNS servers

This rule allows any DNS query for domain names ending in .amazonaws.com (http://amazonaws.com/) to be sent to external DNS servers.

```
pass dns $HOME_NET any -> $EXTERNAL_NET any (dns.query; dotprefix;
  content:".amazonaws.com"; endswith; nocase; msg:"Pass rules do not alert";
  sid:118947;)
```

Stateful rules examples: block traffic

Note

Before using any example rule, test and adapt it to your needs.

The examples in this section contain examples that block specified traffic.

Connections using TLS versions 1.0 or 1.1

This rule blocks connections using TLS version 1.0 or 1.1.

```
reject tls any any -> any any (msg:"TLS 1.0 or 1.1"; ssl_version:tls1.0,tls1.1;
sid:2023070518;)
```

Multiple CIDR ranges

This rule blocks outbound access to multiple CIDR ranges in a single rule.

```
drop ip $HOME_NET any-> [10.10.0.0/16,10.11.0.0/16,10.12.0.0/16] (msg:"Block traffic to
multiple CIDRs"; sid:278970;)
```

Multiple SNIs

This rule blocks multiple SNIs with a single rule.

```
reject tls $HOME_NET any -> $EXTERNAL_NET any (ssl_state:client_hello; tls.sni; pcre:"/
(example1\.com|example2\.com)$/i"; flow: to_server; msg:"Domain blocked"; sid:1457;)
```

Multiple high-risk destination outbound ports

This rule blocks multiple high-risk destination outbound ports in a single rule.

```
drop ip $HOME_NET any -> $EXTERNAL_NET [1389,53,4444,445,135,139,389,3389] (msg:"Deny
List High Risk Destination Ports"; sid:278670;)
```

Outbound HTTP HOST

This rule blocks outbound HTTP connections that have an IP address in the HTTP HOST header.

```
reject http $HOME_NET any -> $EXTERNAL_NET any (http.host; content:"."; pcre:"/^(?:[0-9]{1,3}\.){3}[0-9]{1,3}$/"; msg:"IP in HTTP HOST Header (direct to IP, likely no DNS resolution first)"; flow:to_server; sid:1239847;)
```

Outbound TLS with IP in SNI

This rule blocks outbound TLS connections with an IP address in the SNI.

```
reject tls $HOME_NET any -> $EXTERNAL_NET any (ssl_state:client_hello; tls.sni; content:"."; pcre:"/^(?:[0-9]{1,3}\.){3}[0-9]{1,3}$/"; msg:"IP in TLS SNI (direct to IP, likely no DNS resolution first)"; flow:to_server; sid:1239848;)
```

Any IP protocols other than TCP, UDP, and ICMP

This rule silently blocks any IP protocols other than TCP, UDP, and ICMP.

```
drop ip any any-> any any (noalert; ip_proto:!TCP; ip_proto:!UDP; ip_proto:!ICMP; sid:21801620;)
```

SSH non-standard ports

This rule blocks the use of the SSH protocol on non-standard ports.

```
reject ssh $HOME_NET any -> $EXTERNAL_NET !22 (msg:"Block use of SSH protocol on non-standard port"; flow: to_server; sid:2171010;)
```

TCP/22 servers non-SSH

This rule blocks the use of TCP/22 servers that aren't using the SSH protocol.

```
reject tcp $HOME_NET any -> $EXTERNAL_NET 22 (msg:"Block TCP/22 servers that are not SSH protocol"; flow: to_server; app-layer-protocol:!ssh; sid:2171009;)
```

Stateful rules examples: log traffic

Note

Before using any example rule, test and adapt it to your needs.

The examples in this section demonstrate ways to log traffic. To log traffic, you must configure logging for your firewall. For information about logging Network Firewall traffic, see [Logging and monitoring in AWS Network Firewall](#).

Log traffic direction in default-deny policy

Can be used at the end of a default-deny policy to accurately log the direction of denied traffic. These rules help you to make it clear in the logs who the client is and who the server is in the connection.

```
reject tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"Default Egress TCP block to server"; flow:to_server; sid:202308171;)
drop udp $HOME_NET any -> $EXTERNAL_NET any (msg:"Default Egress UDP block";sid:202308172;)
drop icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"Default Egress ICMP block";sid:202308177;)
drop tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"Default Ingress block to server"; flow:to_server; sid:20230813;)
drop udp $EXTERNAL_NET any -> $HOME_NET any (msg:"Default Ingress UDP block"; sid:202308174;)
drop icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"Default Ingress ICMP block"; sid:202308179;)
```

Log traffic to an allowed SNI.

These rules log traffic to an allowed SNI. Requires your policy to use strict order rule evaluation order.

```
alert tls $HOME_NET any -> $EXTERNAL_NET any (ssl_state:client_hello; tls.sni; content:".example.com"; dotprefix; endswith; nocase; msg:"example.com (http://example.com/) allowed by rule 202307052"; sid:202307051;)
pass tls $HOME_NET any -> $EXTERNAL_NET any (ssl_state:client_hello; tls.sni; content:".example.com"; dotprefix; endswith; nocase; sid:202307052;)
```

Stateful rules examples: rule variables

Note

Before using any example rule, test and adapt it to your needs.

The following JSON defines an example Suricata compatible rule group that uses the variables HTTP_SERVERS and HTTP_PORTS, with the variable definitions provided in the rule group declaration.

```
{
  "RuleVariables": {
    "IPSets": {
      "HTTP_SERVERS": {
        "Definition": [
          "10.0.2.0/24",
          "10.0.1.19"
        ]
      }
    },
    "PortSets": {
      "HTTP_PORTS": {
        "Definition": ["80", "8080"]
      }
    }
  },
  "RulesSource": {
    "RulesString": "alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:
  \".htpasswd access attempt\"; flow:to_server,established; content:\".htpasswd\";
  nocase; sid:210503; rev:1;)"
  }
}
```

The variable EXTERNAL_NET is a Suricata standard variable that represents the traffic destination. For more information, see <https://docs.suricata.io/en/suricata-7.0.0/rules/intro.html#ports-source-and-destination>.

Stateful rules examples: IP set reference

Note

Before using any example rule, test and adapt it to your needs.

To reference a prefix list in your rule group, specify a IP set variable name and associate it with the prefix list's Amazon Resource Name (ARN). Then, specify the variable in one or more of your rules, prefacing the variable with @, such as @IP_Set_Variable. The variable represents the IPv4 prefix

list that you are referencing. For more information about using IP set references, see [Referencing Amazon VPC prefix lists](#).

The following example shows a Suricata compatible rule that uses an IP set reference variable @BETA as the source port in RulesString. To use an IP set reference in your rule, you must use an @ in front of the IP set variable name, such as @My_IP_set_variable_name.

```
{
  "RuleVariables":{
    "IPSets":{
      "HTTP_SERVERS":{
        "Definition":[
          "10.0.2.0/24",
          "10.0.1.19"
        ]
      }
    },
    "PortSets":{
      "HTTP_PORTS":{
        "Definition":[
          "80",
          "8080"
        ]
      }
    }
  },
  "ReferenceSets":{
    "IPSetReferences":{
      "BETA":{
        "ReferenceArn":"arn:aws:ec2:us-east-1:555555555555:prefix-list/pl-11111111111111111111_beta"
      }
    }
  },
  "RulesSource":{
    "RulesString":"drop tcp @BETA any -> any any (sid:1;)"
  }
}
```


Stateful rules examples: Geographic IP filter

Note

Before using any example rule, test and adapt it to your needs.

For information about Geographic IP filtering in Network Firewall, see [the section called “Geographic IP filtering”](#).

The following shows an example Suricata rule string that generates an alert for traffic to or from Russia:

```
alert ip any any -> any any (msg:"Geographic IP is from RU, Russia"; geoip:any,RU;
sid:55555555; rev:1;)
```

The following shows an example standard stateful rule group that drops traffic unless it originates from the United States or the United Kingdom:

```
{
  "RulesSource": {
    "StatefulRules": [
      {
        "Action": "DROP",
        "Header": {
          "DestinationPort": "ANY",
          "Direction": "FORWARD",
          "Destination": "ANY",
          "Source": "ANY",
          "SourcePort": "ANY",
          "Protocol": "IP"
        },
      },
      "RuleOptions": [
        {
          "Settings": [
            "1"
          ],
          "Keyword": "sid"
        },
        {
          "Settings": [
```

```

        "src,!US,UK"
      ],
      "Keyword": "geoip"
    }
  ]
}
],
"StatefulRuleOptions": {
  "RuleOrder": "STRICT_ORDER"
}
}

```

Stateful rules examples: manage rule evaluation order

Note

Before using any example rule, test and adapt it to your needs.

The examples in this section demonstrate ways to modify evaluation behavior by modifying rule evaluation order in Suricata compatible rules. Network Firewall recommends using strict order so that you can have control over the way your rules are processed for evaluation. For information about managing rule evaluation order, see [Managing evaluation order for Suricata compatible rules in AWS Network Firewall](#).

Allow HTTP traffic to specific domains:

Default action order

```

drop tcp $HOME_NET any -> $EXTERNAL_NET 80 (msg:"Drop established TCP:80"; flow:
  from_client,established; sid:172190; priority:5; rev:1;)
pass http $HOME_NET any -> $EXTERNAL_NET 80 (http.host; dotprefix;
  content:".example.com"; endswith; msg:"Allowed HTTP domain"; priority:10; sid:172191;
  rev:1;)
pass tcp $HOME_NET any -> $EXTERNAL_NET 22 (msg:"Allow TCP 22"; sid:172192; rev:1;)
drop tcp $HOME_NET any -> $EXTERNAL_NET !80 (msg:"Drop All non-TCP:80"; sid:172193;
  priority:2; rev:1;)

```

Strict order

```
pass http $HOME_NET any -> $EXTERNAL_NET 80 (http.host; dotprefix;
  content:".example.com"; endswith; msg:"Allowed HTTP domain"; sid:172191; rev:1;)
pass tcp $HOME_NET any -> $EXTERNAL_NET 22 (msg:"Allow TCP 22"; sid:172192; rev:1;)
```

Allow HTTP traffic to specific domains only:

Default action order

```
pass http $HOME_NET any -> $EXTERNAL_NET 80 (http.host; dotprefix;
  content:".example.com"; endswith; msg:"Allowed HTTP domain"; priority:1; sid:102120;
  rev:1;)
pass http $HOME_NET any -> $EXTERNAL_NET 80 (http.host; dotprefix;
  content:".mydomain.test"; endswith; msg:"Allowed HTTP domain"; priority:1; sid:102121;
  rev:1;)
drop http $HOME_NET any -> $EXTERNAL_NET 80 (msg:"Drop HTTP traffic"; priority:1;
  sid:102122; rev:1;)
```

Strict order

```
pass http $HOME_NET any -> $EXTERNAL_NET 80 (http.host; dotprefix;
  content:".example.com"; endswith; msg:"Allowed HTTP domain"; sid:102120; rev:1;)
pass http $HOME_NET any -> $EXTERNAL_NET 80 (http.host; dotprefix;
  content:".mydomain.test"; endswith; msg:"Allowed HTTP domain"; sid:102121; rev:1;)
```

Allow HTTP traffic to specific domains only and deny all other IP traffic:

Default action order

```
pass http $HOME_NET any -> $EXTERNAL_NET 80 (http.host; dotprefix;
  content:".example.com"; endswith; msg:"Allowed HTTP domain"; priority:1; sid:892120;
  rev:1;)
drop tcp $HOME_NET any -> $EXTERNAL_NET 80 (msg:"Drop established non-HTTP to TCP:80";
  flow: from_client,established; sid:892191; priority:5; rev:1;)
drop ip $HOME_NET any <> $EXTERNAL_NET any (msg: "Drop non-TCP traffic."; ip_proto:!
  TCP;sid:892192; rev:1;)
drop tcp $HOME_NET any -> $EXTERNAL_NET !80 (msg:"Drop All non-TCP:80"; sid:892193;
  priority:2; rev:1;)
```

Strict order

```
pass http $HOME_NET any -> $EXTERNAL_NET 80 (http.host; dotprefix;
  content:".example.com"; endswith; msg:"Allowed HTTP domain"; sid:892120; rev:1;)
pass tcp $HOME_NET any <> $EXTERNAL_NET 80 (flow:not_established; sid:892191; rev:1;)
```

Stateful rules examples: domain list rules

Note

Before using any example rule, test and adapt it to your needs.

Deny list example JSON, rule group creation, and generated Suricata rules

The following JSON shows an example rule definition for a Network Firewall domain list rule group that specifies a deny list.

```
{
  "RulesSource": {
    "RulesSourceList": {
      "Targets": [
        "evil.com"
      ],
      "TargetTypes": [
        "TLS_SNI",
        "HTTP_HOST"
      ],
      "GeneratedRulesType": "DENYLIST"
    }
  }
}
```

To use the Network Firewall rule specification, we save the JSON to a local file `domainblock.example.json`, and then create the rule group in the following CLI command:

```
aws network-firewall create-rule-group --rule-group-name "RuleGroupName" --type
  STATEFUL --rule-group file://domainblock.example.json --capacity 1000
```

The following Suricata rules listing shows the rules that Network Firewall creates for the above deny list specification.

```
drop tls $HOME_NET any -> $EXTERNAL_NET any (ssl_state:client_hello; tls.sni;
content:"evil.com"; startswith; nocase; endswith; msg:"matching TLS denylisted FQDNs";
priority:1; flow:to_server, established; sid:1; rev:1;)
drop http $HOME_NET any -> $EXTERNAL_NET any (http.host; content:"evil.com";
startswith; endswith; msg:"matching HTTP denylisted FQDNs"; priority:1;
flow:to_server, established; sid:2; rev:1;)
```

HTTP allow list example JSON and generated Suricata rules

The following JSON shows an example rule definition for a Network Firewall domain list rule group that specifies an HTTP allow list. The `.` before the domain name in `.amazon.com` is the wildcard indicator in Suricata.

```
{
  "RulesSource": {
    "RulesSourceList": {
      "Targets": [
        ".amazon.com",
        "example.com"
      ],
      "TargetTypes": [
        "HTTP_HOST"
      ],
      "GeneratedRulesType": "ALLOWLIST"
    }
  }
}
```

The following Suricata rules listing shows the rules that Network Firewall creates for the above allow list specification.

```
pass http $HOME_NET any -> $EXTERNAL_NET any (http.host; dotprefix;
content:".amazon.com"; endswith; msg:"matching HTTP allowlisted FQDNs"; priority:1;
flow:to_server, established; sid:1; rev:1;)
pass http $HOME_NET any -> $EXTERNAL_NET any (http.host; content:"example.com";
startswith; endswith; msg:"matching HTTP allowlisted FQDNs"; priority:1;
flow:to_server, established; sid:2; rev:1;)
drop http $HOME_NET any -> $EXTERNAL_NET any (http.header_names; content:"|0d 0a|";
startswith; msg:"not matching any HTTP allowlisted FQDNs"; priority:1; flow:to_server,
established; sid:3; rev:1;)
```

TLS allow list example JSON and generated Suricata rules

The following JSON shows an example rule definition for a Network Firewall domain list rule group that specifies a TLS allow list.

```
{
  "RulesSource": {
    "RulesSourceList": {
      "Targets": [
        ".amazon.com",
        "example.com"
      ],
      "TargetTypes": [
        "TLS_SNI"
      ],
      "GeneratedRulesType": "ALLOWLIST"
    }
  }
}
```

The following Suricata rules listing shows the rules that Network Firewall creates for the above allow list specification.

```
pass tls $HOME_NET any -> $EXTERNAL_NET any (ssl_state:client_hello; tls.sni;
dotprefix; content:".amazon.com"; nocase; endswith; msg:"matching TLS allowlisted
FQDNs"; priority:1; flow:to_server, established; sid:1; rev:1;)
pass tls $HOME_NET any -> $EXTERNAL_NET any (ssl_state:client_hello; tls.sni;
content:"example.com"; startswith; nocase; endswith; msg:"matching TLS allowlisted
FQDNs"; priority:1; flow:to_server, established; sid:2; rev:1;)
drop tls $HOME_NET any -> $EXTERNAL_NET any (msg:"not matching any TLS allowlisted
FQDNs"; priority:1; ssl_state:client_hello; flow:to_server, established; sid:3;
rev:1;)
```

Block traffic from \$EXTERNAL_NET to \$HOME_NET, allow outbound domain filtering

These rules block all unsolicited traffic from \$EXTERNAL_NET to \$HOME_NET while still allowing outbound domain filtering.

```
reject tls any any -> any any (msg:"Vulnerable versions of TLS";
ssl_version:tls1.0,tls1.1; sid:2023070518;)
```

Stateful rules examples: standard stateful rule groups

The following JSON shows an example rule definition for a Network Firewall basic stateful rule group.

```
{
  "RulesSource": {
    "StatefulRules": [
      {
        "Action": "DROP",
        "Header": {
          "Protocol": "HTTP",
          "Source": "$HOME_NET",
          "SourcePort": "ANY",
          "Direction": "ANY",
          "Destination": "$EXTERNAL_NET",
          "DestinationPort": "ANY"
        },
        "RuleOptions": [ {
          "Keyword": "msg",
          "Settings": [ "\"this is a stateful drop rule\""
          ]
        },
        {
          "Keyword": "sid",
          "Settings": [ "1234"
          ]
        }
      ]
    ]
  }
}
```

The following Suricata rules listing shows the rules that Network Firewall generates for the above deny list specification.

```
drop http $HOME_NET ANY <> $EXTERNAL_NET ANY (msg:this is a stateful drop rule;
sid:1234;)
```

Working with stateless rule groups in AWS Network Firewall

For stateless rule groups, the AWS Network Firewall stateless rules engine examines each packet in isolation. Network Firewall doesn't consider context such as traffic direction or other related packets.

Network Firewall supports standard network connection identifiers (source IP address, source port, destination IP address, destination port, and protocol) for network traffic inspection. When Network Firewall finds a match between a rule's inspection criteria and a packet, we say that the packet matches the rule and its rule group, and Network Firewall applies the rule's specified action to the packet.

You can add multiple stateless rules to your stateless rule group.

All rule groups have the common settings that are defined at [the section called “Common rule group settings”](#).

General settings

A stateless rule has the following general settings.

- **Priority** – Number that indicates the processing order of the stateless rule within the rule group. This must be unique within the stateless rule group and it must be a positive integer. Network Firewall processes the rules starting from the lowest numbered priority setting. When you plan the rules in your rule group, provide priority settings with space in between, to leave yourself room to add rules later. For example, you might start by using priority settings that are multiples of 100.
- **Actions** – Defines how Network Firewall handles a packet that matches the rule match settings. You assign one standard setting, from among pass, drop, and forward to stateful. You can optionally add a custom setting, for example, to send metrics for the rule match to Amazon CloudWatch metrics. For more information about actions, see [Defining rule actions in AWS Network Firewall](#).

Match settings

A stateless rule has the following match settings. These specify what the Network Firewall stateless rules engine looks for in a packet. To be a match, a packet must satisfy all of the match settings in the rule.

- **Protocol** – Valid settings include ALL and specific protocol settings, like UDP and TCP. You can choose more than one specific setting.
- **Source IP** – Source IP addresses and ranges. If specified, a packet must come from a source address that's included in this list in order to match.
- **Source port range** – Source ports and port ranges. If specified, a packet must have a source port that's included in this list in order to match.
- **Destination IP** – Destination IP addresses and ranges. If specified, a packet must have a destination address that's included in this list in order to match.
- **Destination port range** – Destination ports and port ranges. If specified, a packet must have a destination port that's included in this list in order to match.
- **Optional TCP flags** – Optional, standard TCP flag settings, which indicate which flags to inspect and the values to inspect for. Each flag can be either enabled or disabled. You indicate the flags that you want to inspect in a masks setting, and then you indicate which of those flags must be enabled in the flags setting in order to match. The flags that you specify in the masks setting and don't specify in the flags setting must be unset in order to match.

Example

To create a very simple stateless rule group that passes all traffic from two CIDR blocks, you could provide the following stateless rule settings in a single rule:

- **Priority** – 100
- **Action** – PASS
- **Protocol** – ALL
- **Source** – 192.0.2.0/8, 198.51.100.0/16

To block all other traffic, you would set the firewall policy's stateless default actions to Drop. For more information, see [Firewall policy settings in AWS Network Firewall](#).

Creating a stateless rule group

Follow the guidance in this section to create a stateless rule group through the Network Firewall console.

To create a stateless rule group

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, under **Network Firewall**, choose **Network Firewall rule groups**.
3. Choose **Create Network Firewall rule group**.
4. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
5. In the navigation pane, under **Network Firewall**, choose **Network Firewall rule groups**.
6. Choose **Create Network Firewall rule group**.
7. Under **Choose rule group type**, for the **Rule group format**, choose **Stateless rule group**.
8. Choose **Next**.
9. Enter a name and description for the rule group. You'll use these to identify the rule group when you manage it and use it.

Note

You can't change the name after you create the rule group.

10. For **Capacity**, set the maximum capacity you want to allow for the stateless rule group, up to the maximum of 30,000. You can't change this setting after you create the rule group. For information about how to calculate this, see [Setting rule group capacity in AWS Network Firewall](#). For information about the maximum setting, see [AWS Network Firewall quotas](#).
11. Choose **Next**.
12. Review the rules that you want to add to the stateless rule group. Determine roughly what order you want Network Firewall to process them within the rule group. You need to provide unique, positive integer priority settings for your rules to indicate the processing order. Network Firewall processes from the lowest number up. We recommend using numbers with room in between, to allow for future insertions within the list of rules. For example, you might start with rule priorities numbered 100, 200, and so on.
13. Add each rule to the rule group as follows:
 - a. For **Priority**, provide the priority to set the processing order of your rule.
 - b. Choose the protocol and the source and destination settings for your rule.

- c. (Optional) For **TCP flags** provide the masks and flags that you want to inspect for. In **Masks**, indicate the flags that you want to inspect. In **Flags**, indicate which of the flags that you selected in **Masks** must be set. The other flags that you selected in **Masks** must be unset.
- d. For **Actions**, do the following:
 - i. For **Action**, select the standard action that you want Network Firewall to take when a packet matches the rule settings.
 - ii. (Optional) For **Publish metrics**, add a new named custom action or select one that you've already created in the rule group. This option sends an Amazon CloudWatch metric dimension named CustomAction with a value that you specify.

For additional information on these options, see [Actions for stateless rules](#).

- e. Choose **Add rule**. Your rule is added to the **Rules** list for the rule group, ordered by priority.
14. Choose **Next**.
 15. (Optional) On the **Configure advanced settings** page, configure a customer managed AWS Key Management Service customer managed key to encrypt and decrypt your resources instead of the default key.
 16. Under **Customer managed key**, toggle the **Customize encryption settings** option to configure your customer managed key. For more information about this option, see [Encryption at rest with AWS Key Management Service](#).
 17. Choose **Next**.
 18. (Optional) On the **Add tags** page, enter a key and optional value for any tag that you want added to this firewall policy. Tags help you organize and manage your AWS resources. For more information about tagging your resources, see [Tagging AWS Network Firewall resources](#).
 19. Choose **Next**.
 20. Review the settings for the rule group, then choose **Create stateless rule group**.

Your new rule group is added to the list in the **Network Firewall rule groups** page.

To use your rule group in a firewall policy, follow the procedures at [Managing your firewall policy](#).

Updating a stateless rule group

Follow the guidance in this section to change your rule group settings through the Network Firewall console.

To update a stateless rule group

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, under **Network Firewall**, choose **Network Firewall rule groups**.
3. In the **Network Firewall rule groups** page, choose the name of the rule group that you want to update. The rule group's details page appears.
4. In your rule group's details page, in the area that you want to change, choose **Edit**. Follow the prompts to make your updates. The interface varies according to the rule group type. When you're done editing an area, choose **Save** to save your changes in the rule group.

How Network Firewall propagates your changes

When you make any changes to a firewall, including changes to any of the firewall's components, like rule groups, TLS inspection configurations, and firewall policies, Network Firewall propagates the changes everywhere that the firewall is used. Your changes are applied within seconds, but there might be a brief period of inconsistency when the changes have arrived in some places and not in others. For example, if you modify a rule group so that it drops an additional type of packet, for a firewall that uses the rule group, the new packet type might briefly be dropped by one firewall endpoint while still being allowed by another.

This temporary inconsistency can occur when you first create a firewall and when you make changes to an existing firewall. Generally, any inconsistencies of this type last only a few seconds.

When you add a TLS inspection configuration to an existing firewall, Network Firewall interrupts traffic flows that match the criteria defined by the TLS inspection configuration scope configuration. Network Firewall will begin SSL/TLS decryption and inspection for new connections to the firewall.

Changes to stateful rules are applied only to new traffic flows. Other firewall changes, including changes to stateless rules, are applied to all network packets.

Deleting a stateless rule group

Follow the guidance in this section to delete a rule group through the Network Firewall console.

Deleting a rule group, TLS inspection configuration, or firewall policy

When you delete a rule group, TLS inspection configuration, or a firewall policy, AWS Network Firewall checks to see if it's currently being referenced. A rule group and TLS inspection configuration can be referenced by a firewall policy, and a firewall policy can be referenced by a firewall. If Network Firewall determines that the resource is being referenced, it warns you. Network Firewall is almost always able to determine whether a resource is being referenced. However, in rare cases, it might not be able to do so. If you need to be sure that the resource that you want to delete isn't in use, check all of your firewalls or firewall policies before deleting it. Note that policies that have associations can't be deleted.

To delete a rule group

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, under **Network Firewall**, choose **Network Firewall rule groups**.
3. In the **Network Firewall rule groups** page, select the name of the rule group that you want to delete, and then choose **Delete**.

How Network Firewall propagates your changes

When you make any changes to a firewall, including changes to any of the firewall's components, like rule groups, TLS inspection configurations, and firewall policies, Network Firewall propagates the changes everywhere that the firewall is used. Your changes are applied within seconds, but there might be a brief period of inconsistency when the changes have arrived in some places and not in others. For example, if you modify a rule group so that it drops an additional type of packet, for a firewall that uses the rule group, the new packet type might briefly be dropped by one firewall endpoint while still being allowed by another.

This temporary inconsistency can occur when you first create a firewall and when you make changes to an existing firewall. Generally, any inconsistencies of this type last only a few seconds.

Changes to stateful rules are applied only to new traffic flows. Other firewall changes, including changes to stateless rules, are applied to all network packets.

Analyzing stateless rule groups in AWS Network Firewall

Network Firewall can analyze stateless rule groups for rules that might adversely affect your firewall's functionality. For example, Network Firewall can identify rules that route traffic asymmetrically, which can impact the service's ability to properly process traffic. During analysis, the service includes any identified rules in a list of analysis results. You can analyze your stateless rule groups and view the analysis results using the console or API.

Console

To analyze a stateless rule group

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, under **Network Firewall**, choose **Network Firewall rule groups**.
3. During stateless rule group creation, after you add one or more rules to the rule group, if you select **Analyze**, Network Firewall analyzes the rules in the rule group. If the service determines that any of the rules have the behavior outlined in the following section, Network Firewall displays the identified rule's priority number and the type of identified behavior.

API

Include `AnalyzeRuleGroup` in your [CreateRuleGroupRequest](#), [DescribeRuleGroup](#), or [UpdateRuleGroupRequest](#) request. Network Firewall lists the results in `AnalysisResults` in the response.

To analyze the rule group without creating, describing, or updating the rule group, use the `DryRun` parameter.

CLI

Include `--analyze-rule-group` in your [create-rule-group](#), [describe-rule-group](#), or [update-rule-group](#) request. Network Firewall lists the results in `AnalysisResults` in the response.

To analyze the rule group without creating, describing, or updating the rule group, use the `--dry-run` parameter.

The following table lists the types of rule behavior that Network Firewall analyzes your rule groups for, as well as the details about the cause and solution.

Rule behavior	Cause	Mitigation
Forwarding asymmetrically	One or more stateless rules with the action <code>pass</code> or <code>forward</code> are forwarding traffic asymmetrically. Specifically, the rule's set of source IP addresses or their associated port numbers, don't match the set of destination IP addresses or their associated port numbers.	Make sure that there's an existing return path. For example, if the rule allows traffic from source 10.1.0.0/24 to destination 20.1.0.0/24, you should allow return traffic from source 20.1.0.0/24 to destination 10.1.0.0/24.
Contains TCP flags	At least one stateless rule with the action <code>pass</code> or <code>forward</code> contains TCP flags that are inconsistent in the forward and return directions.	Prevent asymmetric routing issues caused by TCP flags by following these actions: <ul style="list-style-type: none"> • Remove unnecessary TCP flag inspections from the rules. • If you need to inspect TCP flags, check that the rules correctly account for changes in TCP flags throughout the TCP connection cycle, for example SYN and ACK flags used in a 3-way TCP handshake.

Using tag-based resource groups in Network Firewall

Use *tag-based resource groups* to ensure that your rules stay in sync as your AWS resources change. A tag-based resource group is a collection of AWS resources, grouped by tags, that you can reference in an Network Firewall stateful rule group. A tag is a label that you assign to an AWS resource. As you add, delete, and modify your resources belonging to the resource group, Network Firewall automatically updates your rules with the IPs of the resources in the resource group. For information about referencing resource groups in rule groups, see [IP set references in Suricata compatible AWS Network Firewall rule groups](#).

Topics

- [Settings for the resource groups that you use in AWS Network Firewall](#)
- [Creating a resource group in AWS Network Firewall](#)
- [Updating a resource group in AWS Network Firewall](#)
- [Deleting a resource group in AWS Network Firewall](#)

Settings for the resource groups that you use in AWS Network Firewall

The tag-based resource groups that you configure for Network Firewall help ensure that your rules stay in sync as your AWS resources change. You can reference a resource group in your Network Firewall stateful rule groups.

The following settings apply to resource groups.

- **Name** – The identifier for the resource group. You assign a unique name to every resource group. You can't change the name of a resource group after you create it.
- **Description** – Optional additional information about the resource group. Fill in any information that might help you remember the purpose of the resource group and how you want to use it. The description is included in resource group lists in the console and through the APIs.
- **Grouping criteria** – Tag the resource types to add to the resource group. A tag consists of a key and a value, both of which you define. A resource type is a type of AWS resource, such as an Amazon EC2 instance. Network Firewall adds to the resource group all the resource types within your account matching the tags. As you create, update, or delete resource types that match

the tags, Network Firewall automatically updates the resource group to include the resources. Network Firewall constantly checks your account for resources that match the grouping criteria.

- **Preview resources** – A list of all of the resources within your account in the current Region that match the grouping criteria.
- **Tags** – Optional key-value tag pairs. These tags apply to the resource group itself, not the individual resources within it. You can use tags to search and filter your resources and to track your AWS costs. For more information about tags, see [Tagging AWS Network Firewall resources](#).

Creating a resource group in AWS Network Firewall

Use tag-based resource groups to identify collections of AWS resources. You can track the resource groups in your Network Firewall stateful rule groups, to ensure that your rules stay in sync as your resources change.

You can create a resource group for use in AWS Network Firewall either in the Network Firewall console or with the AWS Resource Groups [CreateGroup](#) API. Use the following procedure to create a resource group in the Network Firewall console. For information about creating a resource group in Resource Groups, see [AWS::NetworkFirewall::RuleGroup](#) in the *Resource Groups User Guide*.

To create a resource group in the console

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, under **Network Firewall**, choose **Network Firewall resource groups**.
3. Choose **Create resource group**.
4. Enter a **Name** to identify the resource group.

Note

You can't change the name after you create the resource group.

5. (Optional) Enter a **Description** for the resource group.
6. For **Grouping criteria**, choose the resource types and tags to associate to this resource group. As you create, update, or delete resources in your account in the current Region that match the grouping criteria, Network Firewall automatically updates this resource group with those changes.

- **Resource types** - Choose the resource types to include in the resource group. Network Firewall supports the following resource types:
 - Amazon EC2 instances
 - Amazon EC2 network interfaces
 - **Tags** - Add key value tags to the resource types. A tag consists of a key and a value, both of which you define. For example, if you have two Amazon EC2 instances, you might assign both a tag key of "Stack." But the value of "Stack" might be "Testing" for one and "Production" for the other. You can choose from the existing tags in your account, or add your own tags. For information about best practices for tagging resources, see [Best practices for tag names](#) in the *AWS Tag Editor User Guide*.
7. Select the **Preview resources** button to preview the resources in your account in the current Region that match the grouping criteria. The resources that match the grouping criteria display in the **Preview resources** section. If there aren't resources that match the grouping criteria, no resources will display. However, if at any time you create resources that match the grouping criteria, Network Firewall will automatically display these resources in the **Preview resources** pane.
 8. (Optional) In the **Add tags** section, enter a key and optional value for any tag that you want added to this resource group. Tags help you organize and manage your AWS resources. For more information about tagging your resources, see [Tagging AWS Network Firewall resources](#).
 9. Choose **Create resource group**.

Your new resource group is added to the list in the **Resource group** page.

To use a resource group, include the resource group's Amazon Resource Name (ARN) in a stateful rule's IP set reference. For information about using resource groups in IP set references, see [Referencing resource groups](#).

Updating a resource group in AWS Network Firewall

You can change your resource group settings in the Network Firewall console or the AWS Resource Groups [UpdateGroup](#) API. To change your resource group settings in the Network Firewall console, use the following procedure:

To update a resource group

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, under **Network Firewall**, choose **Network Firewall resource groups**.
3. In the **Resource group** page, select the name of the resource group you want to update.
4. Make your changes. You can't change the name of a resource group after creation, but you can change other details and you can change the rule groups.
5. Choose **Save** to save your changes.

How Network Firewall propagates your changes

When you make any changes to a firewall, including changes to any of the firewall's components, like rule groups, TLS inspection configurations, and firewall policies, Network Firewall propagates the changes everywhere that the firewall is used. Your changes are applied within seconds, but there might be a brief period of inconsistency when the changes have arrived in some places and not in others. For example, if you modify a rule group so that it drops an additional type of packet, for a firewall that uses the rule group, the new packet type might briefly be dropped by one firewall endpoint while still being allowed by another.

This temporary inconsistency can occur when you first create a firewall and when you make changes to an existing firewall. Generally, any inconsistencies of this type last only a few seconds.

When you add a TLS inspection configuration to an existing firewall, Network Firewall interrupts traffic flows that match the criteria defined by the TLS inspection configuration scope configuration. Network Firewall will begin SSL/TLS decryption and inspection for new connections to the firewall.

Changes to stateful rules are applied only to new traffic flows. Other firewall changes, including changes to stateless rules, are applied to all network packets.

Deleting a resource group in AWS Network Firewall

You can delete your resource group in the Network Firewall console or the AWS Resource Groups [DeleteGroup](#) API. To delete a resource group in the Network Firewall console, perform the following procedure.

Deleting a resource group

You can't delete a resource group that's referenced in a rule group. When you try to delete a resource group, AWS Network Firewall checks to see if it's currently being referenced. A resource group can be referenced by a rule group. If Network Firewall determines that the resource is being referenced, it warns you. Network Firewall is almost always able to determine whether a resource is being referenced. However, in rare cases, it might not be able to do so. If you need to be sure that the resource that you want to delete isn't in use, check all of your rule groups before deleting it. Note that policies that have associations can't be deleted.

To delete a resource group

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, under **Network Firewall**, choose **Network Firewall resource groups**.
3. In the **Resource group** page, select the resource group that you want to delete.
4. Choose **Delete**, and confirm your request.

Your resource group is removed from the list in the **Resource group** page.

Inspecting SSL/TLS traffic with TLS inspection configurations in AWS Network Firewall

AWS Network Firewall uses *TLS inspection configurations* to decrypt your firewall's inbound and outbound SSL/TLS traffic. After decryption, Network Firewall inspects the traffic according to your firewall policy's stateful rules, and then re-encrypts it before sending it to its destination. You can enable inspection of your firewall's inbound traffic, outbound traffic, or both. To use TLS inspection with your firewall, you must import or provision certificates to AWS Certificate Manager, create a TLS inspection configuration, add that configuration to a new firewall policy, and then associate that policy with your firewall.

Pricing for using TLS inspection configurations is based on the amount of traffic that Network Firewall inspects—which appears on your bill as advanced inspection—and the number of deployed firewall endpoints. For information about TLS inspection configuration pricing, see [Network Firewall pricing](#). To use the AWS pricing calculator to check Network Firewall costs, see [Network Firewall pricing calculator](#).

Topics

- [Considerations when working with TLS inspection configurations in AWS Network Firewall](#)
- [Logging for TLS inspection in AWS Network Firewall](#)
- [Using SSL/TLS certificates with TLS inspection configurations in AWS Network Firewall](#)
- [TLS inspection configuration settings in AWS Network Firewall](#)
- [Managing your TLS inspection configuration in Network Firewall](#)

Considerations when working with TLS inspection configurations in AWS Network Firewall

Before you implement TLS inspection configuration in Network Firewall, understand the following considerations and limitations.

Conditions for dropping traffic

Network Firewall drops non-TLS traffic that matches the conditions of the TLS inspection configuration scope configuration. For example, if the TLS inspection configuration scope

configuration includes `port 80` as `plain HTTP`, Network Firewall drops this traffic because the service can't identify it as TLS traffic. Network Firewall also drops TLS traffic if the client hello message in the TLS handshake doesn't include a server name, or if the server name that's presented in the client hello doesn't match the server name indication (SNI) that's presented in the downstream server certificate.

Decrypted payload inspection for HTTP2

Network Firewall does not support decrypted payload inspection for HTTP2.

Impact on existing flows

When you add a TLS inspection configuration to an existing firewall, Network Firewall interrupts traffic flows that match the criteria defined by the TLS inspection configuration scope configuration. New connections to the firewall begin SSL/TLS decryption and inspection. When you add new TLS inspection configuration to a firewall and there's ongoing TLS traffic that matches the scope criteria, the firewall drops the traffic.

Initial latency

Some latency is expected during the initial connection due to the TCP and TLS handshakes that occur before data can flow to the firewall. If you enable certificate revocation checking for outbound TLS inspection, you can expect additional latency when you initially connect to a new domain over a firewall. We recommend that you conduct your own testing using your rulesets to ensure that TLS inspection configuration meets your performance expectations.

Inspection capabilities

TLS inspection is available for the request and the response for HTTPS traffic and other TCP-based application protocols that use TLS, such as Secure Mail Transfer Protocol Secure (SMTPS), and Post Office Protocol 3 Secure (POP3s). At the network layer, TLS inspection supports both IPv4 and IPv6 traffic.

Managed rules

You can use AWS managed rules with TLS inspection. However, due to the TLS decryption, TLS keywords don't initiate inspection within the stateful engine. You can benefit from non-TLS rules within managed rule groups and find increased visibility of those rules because the decrypted traffic is visible to the inspection engine. You can also create custom rules based on inner protocols, which are available for inspection. For example, you can match with an HTTP header within the

decrypted HTTPS stream. For more information about using managed rules with Network Firewall, see [Using AWS managed rule groups in AWS Network Firewall](#).

Stateful rules

Network Firewall ends the TLS connection that's initiated by the client and decrypts traffic before it reaches the stateful inspection engine. As a result, the traffic doesn't match any TLS-based keywords except for TLS . SNI. You can still use all TLS keywords in the stateful rule for traffic that doesn't match the TLS inspection scope. Application rules that are based on decrypted payloads are applied, for example, rules that are based on HTTP keywords.

Note

TLS . SNI keyword rules are still matched for traffic that's in TLS scope, even though the traffic is decrypted before it reaches the stateful inspection engine.

If you configure a drop or reject action for a stateful rule that matches the traffic scope that's defined in the TLS inspection configuration, Network Firewall closes the connections to clients as soon as it detects a payload bearing packet that matches the drop or reject rule.

Your traffic is re-encrypted before leaving the Network Firewall host.

Supported TLS versions

TLS versions 1.1, 1.2, and 1.3 are supported.

TLS cipher suites and SNI

Network Firewall terminates the TLS connection that's initiated by the client, and the TLS Server Name Identifier (SNI) must match a configured certificate. Network Firewall uses TLS 1.3 to initiate the forward connection to the server.

The client cipher suites must include one or more of the following:

```
TLS 1.1 & TLS 1.2
AES128-GCM-SHA256
AES128-SHA
AES128-SHA256
AES256-GCM-SHA384
AES256-SHA
ECDHE-RSA-AES128-GCM-SHA256
```

```
ECDHE-RSA-AES128-SHA
ECDHE-RSA-AES128-SHA256
ECDHE-RSA-AES256-GCM-SHA384
ECDHE-RSA-AES256-SHA
```

```
TLS 1.3
TLS_AES_256_GCM_SHA384
TLS_CHACHA20_POLY1305_SHA256
TLS_AES_128_GCM_SHA256
```

Limitations

The following limitations apply to TLS inspection configurations:

- Cross-signed root certificates aren't supported. For more information, see [Using SSL/TLS certificates with TLS inspection configurations in AWS Network Firewall](#).
- Decryption of TLS protocols that rely upon StartTLS aren't supported.
- Network Firewall publishes separate CloudWatch metrics for traffic that's associated with TLS inspection configurations. Existing stateful traffic metrics don't reflect data for traffic that's decrypted and re-encrypted by your firewall policy's TLS inspection configuration.
- Self-signed certificates aren't supported for inbound inspection. For more information about the certificates that Network Firewall supports, see [Using SSL/TLS certificates with TLS inspection configurations in AWS Network Firewall](#).
- TLS inspection of UDP-based transport protocols such as Quick UDP Internet Connections (QUIC) isn't supported. You can configure your applications so that they don't switch from TCP to UDP-based transport protocols. If you want to prevent TLS traffic from using the QUIC protocol, modify your application to deny traffic using QUIC or create a firewall stateful rule to block QUIC traffic.
- TLS version 1.0 and prior SSL versions aren't supported.
- Traffic encrypted using TLS v1.3 Encrypted SNI and Encrypted Client Hello extensions aren't supported. When Network Firewall can't find an SNI in the client hello, the service closes the connection with a RST packet. If this is an issue for your workloads, you can use standard stateless rules to bypass TLS decryption.
- WebSockets inspection over HTTP2 isn't supported; Network Firewall drops this traffic. However, WebSockets over HTTP1.1 is supported.
- If Network Firewall is deployed behind an Application Load Balancer, its TLS inspection can't inspect the inbound TLS traffic that's terminated at the Application Load Balancer.

- You can have the TLS1.3 hybridized Kyber support setting enabled in your browser, but TLS inspection doesn't use Kyber keys. Instead, inspection will use other keys that the browser has enabled in its settings.

Logging for TLS inspection in AWS Network Firewall

You can enable TLS logging for your firewall's stateful engine, to log some categories of events related to TLS inspection. TLS logs report TLS errors and certificate revocation check failures for outbound traffic. For information about enabling logging, see [Logging network traffic from AWS Network Firewall](#).

Log entries for TLS errors

TLS errors currently report connection resets that are due to SNI mismatches and naming errors. These are typically caused by problems with customer traffic or with the customer's client or server. For example, when the client hello SNI is NULL or doesn't match the subject name in the server certificate.

For this type of error, the TLS logs include the source and destination IPs and ports, the SNI, and the TLS error message.

In the following example TLS log entry, the error is in the client hello.

```
{
  "firewall_name": "firewall-tls-test",
  "availability_zone": "us-east-1d",
  "event_timestamp": 1719943304,
  "event": {
    "timestamp": "2024-07-02T18:01:44.412778Z",
    "src_ip": "10.0.2.53",
    "src_port": "59844",
    "dest_ip": "10.0.1.27",
    "dest_port": "443",
    "sni": "-",
    "tls_error": {
      "error_message": "Server name is not found in client hello."
    }
  }
}
```

The following example TLS log entry indicates a server name mismatch.

```
{
  "firewall_name": "test-firewall",
  "availability_zone": "us-east-1d",
  "event_timestamp": 1721849698,
  "event": {
    "timestamp": "2024-07-24T19:34:58.337042Z",
    "src_ip": "10.0.2.22",
    "src_port": "44798",
    "dest_ip": "91.212.12.80",
    "dest_port": "444",
    "sni": "test.com",
    "tls_error": {
      "error_message": "SNI: test.com Match Failed to server certificate
names: aaacertificateservices.comodoca.com/aaacertificateservices.comodoca.com/
www.aaacertificateservices.comodoca.com "
    }
  }
}
```

Log entries for certificate revocation check failures

These entries report outbound traffic that fails the server certificate revocation check during TLS inspection. This log type requires the firewall to be configured with TLS inspection for outbound traffic, and for the TLS inspection to be configured to check the certificate revocation status. For information about configuring certificate revocation checking, see [Using SSL/TLS certificates with TLS inspection configurations in AWS Network Firewall](#) and [Checking certificate revocation status](#).

For revocation checks, the TLS logs report when servers that you're sending traffic to are failing the checks. The logs include the revocation check status, the action taken, source and destination IPs and ports, and the SNI that the revocation check was for. You can use this information to pinpoint the traffic that's experiencing failures and take measures to manage the problem.

In the following example TLS log entry, the revocation check reports that the certificate has been revoked, either by an Online Certificate Status Protocol (OCSP) or a Certificate Revocation Lists (CRL) provider. The certificate is not valid anymore. For this case, the firewall acts according to the configuration that you've specified for these checks. For this example, the action is DROP.

```
{
  "firewall_name": "egress-fw",
  "availability_zone": "us-east-1d",
  "event_timestamp": 1708361189,
```

```
"event": {
  "src_ip": "10.0.2.53",
  "src_port": "55930",
  "revocation_check": {
    "leaf_cert_fpr": "1234567890EXAMPLE0987654321",
    "status": "REVOKED",
    "action": "DROP"
  },
  "dest_ip": "54.92.160.72",
  "dest_port": "443",
  "timestamp": "2024-02-19T16:46:29.441824Z",
  "sni": "revoked-rsa-dv.ssl.com"
}
}
```

In the following example, the revocation check status is UNKNOWN. This can happen for a number of reasons, such as when the check encounters an error retrieving the data from the provider, or the provider not having any record of the certificate. Whatever the reason, the firewall isn't sure whether the certificate is revoked. The firewall again acts according to the configuration that you've specified for these checks.

```
{
  "firewall_name": "egress-fw",
  "availability_zone": "us-east-1d",
  "event_timestamp": 1708361189,
  "event": {
    "src_ip": "10.0.2.53",
    "src_port": "55930",
    "revocation_check": {
      "leaf_cert_fpr": "1234567890EXAMPLE0987654321",
      "status": "UNKNOWN",
      "action": "DROP"
    },
    "dest_ip": "54.92.160.72",
    "dest_port": "443",
    "timestamp": "2024-02-19T16:46:29.441824Z",
    "sni": "revoked-rsa-dv.ssl.com"
  }
}
```

Using SSL/TLS certificates with TLS inspection configurations in AWS Network Firewall

Network Firewall integrates with AWS Certificate Manager (ACM) to make it easy to manage the certificates that you're using to decrypt and re-encrypt your firewall's SSL/TLS traffic. To get started with TLS inspection configurations, you must first import or issue certificates to ACM, and then associate the certificates with your TLS inspection configuration. You can configure certificates for inbound or outbound inspection, or both. To view the maximum number of certificates that you can use, see [AWS Network Firewall quotas](#). This section discusses how to use SSL/TLS certificates with TLS inspection configurations.

General requirements for TLS inspection

The following are general requirements for the certificates that are used for TLS inspection.

Certificate chain order

Imported SSL/TLS certificates require all of the intermediate certificates in the certificate chain that's in the **.pem** file, beginning with the certificate authority (CA) that signed the certificate that you are importing. Typically, you'll find a file on the CA website that lists intermediate and root certificates in the proper chained order. Here's an example:

```
-----BEGIN CERTIFICATE-----  
Intermediate certificate 2  
-----END CERTIFICATE-----  
-----BEGIN CERTIFICATE-----  
Intermediate certificate 1  
-----END CERTIFICATE-----
```

Supported certificates

Network Firewall supports all of the algorithms and key sizes that are supported by AWS Certificate Manager (ACM), and also supports wildcard certificates. However, Network Firewall doesn't support using self-signed certificates for inbound inspection. Using self-signed certificates can result in client-side errors, and using cross-signed certificates can result in asynchronous and synchronous failures. For information about ACM supported algorithms, key sizes, and wildcard certificates, see [ACM certificate characteristics](#) in the *AWS Certificate Manager User Guide*.

If a certificate that's associated with your TLS inspection configuration expires or is deleted, you experience client-side errors in the traffic that Network Firewall processes. Replace certificates that will expire soon to avoid client-side errors.

Server certificates - Inbound SSL/TLS inspection

To configure inbound TLS inspection, you must first issue or import a certificate in AWS Certificate Manager (ACM) for each domain that you want Network Firewall to inspect. After you issue or import the certificates in ACM, you can associate the certificates with your TLS inspection configuration.

For more information about working with certificates in ACM, see [Request a public certificate](#) or [Importing certificates](#) in the *AWS Certificate Manager User Guide*.

The following requirements are specific to the server certificates that are used for inbound inspection.

Certificate issuer

For server certificates for inbound inspection, Network Firewall supports the same certificate authorities (CAs) as Mozilla, except for those that issue cross-signed root certificates. If you import a certificate into ACM, use a certificate issued by a CA on the [Mozilla Included CA Certificate List](#). For more information about getting and installing a certificate, refer to the documentation for your HTTP server software and to the documentation for the CA.

Note

Network Firewall can't validate cross-signed root certificates, such as those issued by Let's Encrypt. Usage of cross-signed certificates can cause asynchronous failures in your firewall.

Deleted or expired certificates

Network Firewall doesn't support the Online Certificate Status Protocol (OCSP) MustStaple TLS extension. Network Firewall also doesn't validate the expiration status of server certificates that are associated with a TLS inspection configuration. Validate the status of your server certificates and use a valid certificate at all times.

Supported certificates

You can either generate or import a server certificate in ACM.

CA certificate - Outbound SSL/TLS inspection

To configure outbound TLS inspection, you must first import a certificate authority (CA) certificate into AWS Certificate Manager (ACM). After you import the CA certificate in ACM, you can associate the CA certificate with your TLS inspection configuration. Network Firewall uses the CA certificate to generate a server certificate, which the service uses to establish trust between the client and the server.

For more information about working with certificates in ACM, see [Importing certificates](#) in the *AWS Certificate Manager User Guide*.

The following requirements are specific to CA certificates that are used for outbound SSL/TLS inspection.

Supported certificates

You can use CA certificates that are imported into ACM for outbound inspection—including private certificates—but you can't generate intermediate CA certificates using ACM. However, we don't support TLS traffic to or from destination servers that use a server certificate signed by a private CA. In other words, we do certificate verification of downstream server certificates to allow TLS traffic where server certificates are signed by public well-known roots contained on the [Mozilla Included CA Certificate List](#). We also don't support certificates issued by AWS Private Certificate Authority.

Revoked certificates

For outbound TLS, you can enable certificate revocation checks, which allows Network Firewall to check revocation status with Online Certificate Status Protocol (OCSP) and Certificate Revocation List (CRL) on behalf of clients. For more information see [Checking certificate revocation status](#).

Checking certificate revocation status

For outbound inspection, Network Firewall can check if the server or intermediate certificate that the server presents in the TLS connection has a revoked or unknown status. When you enable this option, Network Firewall checks the revocation status using the Online Certificate Status Protocol (OCSP) and Certificate Revocation List (CRL) endpoints. These endpoints are specified in the certificate authority (CA) certificate that's downloaded from the server in the SSL/TLS connection. Network Firewall caches the CRL and OCSP responses.

If the certificate has a revoked or unknown status, Network Firewall handles the outbound traffic based on the actions that you set when you configure the revocation status checks.

Actions

When the certificate has a revoked or unknown status, you can choose from the following actions. These actions configure how Network Firewall processes traffic when it determines that the certificate that's presented by the server in the SSL/TLS connection has an unknown or revoked status.

- **Pass** – Allows the connection to continue, and pass subsequent packets to the stateful engine for inspection.
- **Drop** – Network Firewall closes the connection and drops subsequent packets for that connection.
- **Reject** – Network Firewall sends a TCP reject packet back to your client. The service closes the connection and drops subsequent packets for that connection. This option is available only for TCP traffic.

Keep the following considerations in mind when turning on certificate revocation check:

Caching

Network Firewall caches the revocation check status to serve responses faster than the CRL and OCSP servers can. The cache duration depends on the revocation check status, which improves the accuracy of the certificate revocation status checks.

Latency

When you turn on certificate revocation checking for outbound TLS inspection, expect additional latency when you initially connect to a new domain over a firewall instance. This is specific to outbound TLS inspection only. We recommend that you conduct your own testing using your rulesets to ensure that the service meets your performance expectations.

Logging

You can get logs for certificate revocation check failures. To do this, configure TLS logging for your firewall's stateful engine. For additional information, see [Logging for TLS inspection in AWS Network Firewall](#). For information about enabling logging, see [Logging network traffic from AWS Network Firewall](#).

Unsupported features

Revocation status checking doesn't include support for the following:

- Configurations for salted or nonce OCSP responses.
- Network Firewall supports OCSP checks using a different signer key, but doesn't support CRL checks using a different signer key.
- Delegated OCSP checks.
- Delta Certificate Revocation Lists (CRLs).
- File Transfer Protocol (FTP) and Lightweight Directory Access Protocol (LDAP) CRL URLs.
- OCSP and CRL checks requiring referenced parameters from issuer certificates.
- OCSP stapling.

For information about troubleshooting issues related to certificate revocation, see [Troubleshooting TLS inspection in AWS Network Firewall](#).

TLS inspection configuration settings in AWS Network Firewall

When you configure TLS inspection configuration you provide the following settings.

- **Name** – The identifier for the TLS inspection configuration. You assign a unique name to every TLS inspection configuration. You can't change the name of a TLS inspection configuration after you create it.
- **Description** – Optional additional information about the TLS inspection configuration. Fill in information that might help you to remember the purpose of the TLS inspection configuration and how you want to use it. The description is included in TLS inspection configuration lists in the console and the APIs.
- **Associate SSL/TLS certificates** – The certificates to associate with the TLS inspection configuration for inbound and outbound inspection. Network Firewall uses certificates to decrypt and re-encrypt the SSL/TLS traffic that's going to your firewall.
- **Define scope** – Defines the scope of the traffic to decrypt based on source and destination addresses and port ranges in a scope configuration. For each scope configuration that you add, Network Firewall adds a mirrored scope configuration with reverse sources and destinations when it creates the TLS inspection configuration. This allows Network Firewall to decrypt—and subsequently inspect—traffic in both directions, which is required for TLS termination.

- **Customer managed key** (Optional) – Network Firewall encrypts and decrypts the TLS inspection configuration, to protect against unauthorized access. By default, Network Firewall uses AWS owned keys for this. If you want to use your own keys, you can configure customer managed keys from AWS Key Management Service and provide them to Network Firewall. For information about this option, see [Encryption at rest with AWS Key Management Service](#).
- **Certificate revocation status** (Optional) – Network Firewall checks if the certificate that's presented by the server in the TLS session is revoked or has an unknown status. If this is turned on, Network Firewall handles the outbound traffic based on the actions that you configure in the certificate revocation check.
- **Tags** – A tag is an optional label that you assign to an AWS resource. You can use tags to search and filter your resources and to track your AWS costs. For more information about tags, see [Tagging AWS Network Firewall resources](#).

Managing your TLS inspection configuration in Network Firewall

This section describes how to create, update, and delete a TLS inspection configuration in Network Firewall. To turn on TLS inspection for your firewall, create a TLS inspection configuration, add the TLS inspection configuration to a firewall policy, then associate the firewall policy with your firewall.

You can only add a TLS inspection configuration to a new policy, not to an existing policy. However, you can replace an existing TLS inspection configuration with another TLS inspection configuration in a firewall policy. To add a TLS inspection configuration to a firewall policy or update an existing TLS inspection configuration, see [Managing your firewall policy](#).

Note

A TLS inspection configuration is only available for use by the account that you use to create it. It can't be shared across accounts.

Topics

- [Creating a TLS inspection configuration in Network Firewall](#)
- [Updating a TLS inspection configuration in Network Firewall](#)

- [Deleting a TLS inspection configuration in Network Firewall](#)

Creating a TLS inspection configuration in Network Firewall

This procedure explains how to create a TLS inspection configuration using Network Firewall. To follow this procedure, you must have at least one certificate in AWS Certificate Manager (ACM) that's accessible by your AWS account.

To create a TLS inspection configuration using the console

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, under **Network Firewall**, choose **TLS inspection configurations**.
3. Choose **Create TLS inspection configuration**.
4. In the **Associate SSL/TLS certificates** page, configure **Server certificates for inbound SSL/TLS inspection**, **CA certificate for outbound SSL/TLS inspection**, or both.
5. Choose **Next** to go to the TLS inspection configuration's **Describe TLS inspection configuration** page.
6. Enter a **Name** to identify this TLS inspection configuration.

Warning

You can't change the name after you create the TLS inspection configuration.

7. (Optional) Enter a **Description** for the TLS inspection configuration.
8. Choose **Next** to go to the TLS inspection configuration's **Define scope** page.
9. In the **Scope configuration** pane, choose the protocol, source, source port range, destination, and destination port range of the traffic that you want Network Firewall to decrypt. Network Firewall uses the associated certificates to decrypt the SSL/TLS traffic that matches the scope configuration. After Network Firewall decrypts the traffic, the service inspects the traffic according to your firewall policy's stateful rules.

Network Firewall also automatically configures a reverse scope, ensuring that the service inspects the traffic in both directions.

- a. For **Protocol**, choose the protocol to decrypt. Network Firewall currently supports TCP.

- b. For **Source IP**, choose the source IP addresses and ranges to decrypt. You can decrypt by **Custom IP addresses** or by **Any IPv4 address**.
 - c. For **Source port**, choose the source ports and source port ranges to decrypt. You can decrypt by **Custom port ranges** or by **Any port**.
 - d. For **Destination IP**, choose the destination IP addresses and ranges to decrypt. You can decrypt by **Custom IP addresses** or by **Any IPv4 address**.
 - e. For **Destination port**, choose the destination ports and destination port ranges to decrypt. You can decrypt by **Custom port ranges** or by **Any port**.
 - f. Choose **Add scope configuration**. To add more scope configurations, adjust the settings in the **scope configuration** pane, then select **Add scope configuration**.
10. Choose **Next**.
 11. (Optional) On the **Advanced settings** page, under **Customer managed key**, you can change the key that Network Firewall uses to decrypt and encrypt the TLS inspection configuration, to protect against unauthorized access. By default, Network Firewall uses AWS owned keys. If you want to use your own keys, you can configure customer managed keys from the AWS Key Management Service and provide them to Network Firewall. For information about customer managed keys, see [Encryption at rest with AWS Key Management Service](#).
 12. (Optional) In the **Certificate revocation status** section, choose whether Network Firewall should check if the certificate that's presented by the server in the TLS connection has a revoked status. To enable this option, you must first associate a certificate authority (CA) certificate for outbound inspection in the **Associate SSL/TLS certificates** step. You can also configure the actions that Network Firewall takes on outbound traffic if the certificate is revoked or has an unknown status.
 13. Choose **Next**.
 14. (Optional) On the **Add tags** page, enter a key and optional value for any tag that you want to add to this TLS inspection configuration. Tags help you to organize and manage your AWS resources. For more information about tagging your resources, see [Tagging AWS Network Firewall resources](#).
 15. Choose **Next**.
 16. On the **Review and confirm** page, check the TLS inspection configuration settings. If you want to change anything, choose **Edit** for that section. This returns you to the corresponding step in the create TLS inspection configuration wizard. Make your changes, then choose **Next** on each page until you come back to the review and confirm page.
 17. Choose **Create TLS inspection configuration**.

Your new TLS inspection configuration is added to the list in the Network Firewall TLS inspection configurations page.

If you've configured the inspection for certificate revocation checks on outbound traffic, you can log failures for these checks by enabling TLS logging. For information, see [Logging network traffic](#).

To use your TLS inspection configuration in a firewall policy, follow the procedures at [Managing your firewall policy](#).

Updating a TLS inspection configuration in Network Firewall

To change your TLS inspection configuration settings, use the following procedure:

To update a TLS inspection configuration

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, under **Network Firewall**, choose **TLS inspection configurations**.
3. In the **TLS inspection configuration** page, select the name of the TLS inspection configuration that you want to update.
4. On the TLS inspection configuration page, make your changes. You can't update the name of a TLS inspection configuration after creation, but you can change other details. If you want to update the name, you must create a new TLS inspection configuration.
5. Choose **Save** to save your changes.

How Network Firewall propagates your changes

When you make any changes to a firewall, including changes to any of the firewall's components, like rule groups, TLS inspection configurations, and firewall policies, Network Firewall propagates the changes everywhere that the firewall is used. Your changes are applied within seconds, but there might be a brief period of inconsistency when the changes have arrived in some places and not in others. For example, if you modify a rule group so that it drops an additional type of packet, for a firewall that uses the rule group, the new packet type might briefly be dropped by one firewall endpoint while still being allowed by another.

This temporary inconsistency can occur when you first create a firewall and when you make changes to an existing firewall. Generally, any inconsistencies of this type last only a few seconds.

Deleting a TLS inspection configuration in Network Firewall

To delete a TLS inspection configuration, perform the following procedure.

Deleting a TLS inspection configuration

When you delete a TLS inspection configuration, AWS Network Firewall checks to see if it's currently being referenced in a firewall policy. If Network Firewall determines that the TLS inspection configuration is referenced in a policy, the service sends you a warning, and won't delete the TLS inspection configuration. Network Firewall is almost always able to determine whether a resource is being referenced. However, in rare cases, it might not be able to do so. To be sure that the resource that you want to delete isn't in use, check all of your firewall policies before deleting it. TLS inspection configurations referenced in firewall policies can't be deleted.

To delete a TLS inspection configuration

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, under **Network Firewall**, choose **TLS inspection configurations**.
3. In the **TLS inspection configuration** page, select the TLS inspection configuration that you want to delete.
4. Choose **Delete**, and confirm your request.

Your TLS inspection configuration is removed from the list in the **TLS inspection configuration** page.

Sharing AWS Network Firewall firewall policies and rule groups

You can share some Network Firewall resources with other accounts. This permits them to use the protections that you define and maintain to protect their resources too.

The owner of a firewall policy or rule group can share a resource with:

- Specific AWS accounts inside or outside of its organization in AWS Organizations
- An organizational unit inside its organization in AWS Organizations
- Its entire organization in AWS Organizations

Warning

You can't share a firewall policy that's configured to use a [TLS inspection configuration](#).

Contents

- [Prerequisites for sharing firewall policies and rule groups](#)
- [Related services](#)
- [Sharing across Availability Zones](#)
- [Sharing a firewall policy or rule group](#)
- [Unsharing a shared firewall policy or rule group](#)

The owner of a rule group can share a rule group that refers to a [resource group](#), but can't share the resource group itself.

Prerequisites for sharing firewall policies and rule groups

- To share a firewall policy or rule group, you must own it in your AWS account. You cannot share a firewall policy or rule group that has been shared with you.
- To share a firewall policy or rule group with your organization or an organizational unit in AWS Organizations, you must enable sharing with AWS Organizations. For more information, see [Enable Sharing with AWS Organizations](#) in the *AWS RAM User Guide*.

Related services

Firewall policy and rule group sharing integrates with AWS Resource Access Manager (AWS RAM). AWS RAM is a service that enables you to share your AWS resources with any AWS account or through AWS Organizations. With AWS RAM, you share resources that you own by creating a *resource share*. A resource share specifies the resources to share, and the consumers with whom to share them. Consumers can be individual AWS accounts, organizational units, or an entire organization in AWS Organizations.

For more information about AWS RAM, see the [AWS RAM User Guide](#).

Sharing across Availability Zones

To ensure that resources are distributed across the Availability Zones for a Region, we independently map Availability Zones to names for each account. This could lead to Availability Zone naming differences across accounts. For example, the Availability Zone `us-east-1a` for your AWS account might not have the same location as `us-east-1a` for another AWS account.

To identify the location of your firewall policy or rule group relative to your accounts, you must use the *Availability Zone ID* (AZ ID). The AZ ID is a unique and consistent identifier for an Availability Zone across all AWS accounts. For example, `us1-az1` is an AZ ID for the `us-east-1` Region and it is the same location in every AWS account.

To view the AZ IDs for the Availability Zones in your account

1. Open the AWS RAM console at <https://console.aws.amazon.com/ram>.
2. The AZ IDs for the current Region are displayed in the **Your AZ ID** panel on the right-hand side of the screen.

Sharing a firewall policy or rule group

To share a firewall policy or rule group, you must add it to a resource share. A resource share is an AWS RAM resource that lets you share your resources across AWS accounts. A resource share specifies the resources to share, and the consumers with whom they are shared. When you share a firewall policy or rule group using AWS Network Firewall, you add it to an existing resource share. To add the firewall policy or rule group to a new resource share, you must first create the resource share using the [AWS RAM console](#).

If you are part of an organization in AWS Organizations and sharing within your organization is enabled, consumers in your organization are automatically granted access to the shared firewall policies and rule groups. Otherwise, consumers receive an invitation to join the resource share and are granted access to the shared firewall policies and rule groups after accepting the invitation.

You can share a firewall policy or rule group that you own using the AWS RAM console, the AWS Network Firewall API, or the AWS CLI.

To share a firewall policy or rule group that you own using the AWS RAM console

See [Creating a Resource Share](#) in the *AWS RAM User Guide*.

To share a firewall policy or rule group that you own using the AWS CLI

Use the [create-resource-share](#) command.

To share a firewall policy or rule group that you own using the Network Firewall API

Use the `PutResourcePolicy` action. For information about how to use this, see [PutResourcePolicy](#) in the *AWS Network Firewall API Reference*.

Unsharing a shared firewall policy or rule group

To unshare a shared firewall policy or rule group that you own, you must remove it from the resource share. You can do this using the AWS RAM console or the AWS CLI.

To unshare a shared firewall policy or rule group that you own using the AWS RAM console

See [Updating a Resource Share](#) in the *AWS RAM User Guide*.

To unshare a shared firewall policy or rule group that you own using the AWS CLI

Use the [disassociate-resource-share](#) command.

Security in AWS Network Firewall

This section describes AWS Cloud security for Network Firewall. Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The [shared responsibility model](#) describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. The effectiveness of our security is regularly tested and verified by third-party auditors as part of the [AWS compliance programs](#). To learn about the compliance programs that apply to Network Firewall, see [AWS Services in Scope by Compliance Program](#).
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your organization's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Network Firewall. The following topics show you generally how to configure Network Firewall to meet your security and compliance objectives for using an AWS service. You also learn how to use other AWS services that help you to monitor and secure your Network Firewall resources.

Use this general guidance in addition to the guidance for using the AWS Network Firewall service itself. Network Firewall is intended to improve the security of communication into and out of your Amazon Virtual Private Cloud VPCs, and this entire developer guide provides guidance for using Network Firewall.

Topics

- [Data protection in Network Firewall](#)
- [Identity and Access Management for AWS Network Firewall](#)
- [AWS logging and monitoring tools](#)
- [Compliance validation and security best practices for Network Firewall](#)
- [Resilience in Network Firewall](#)
- [Infrastructure security in AWS Network Firewall](#)

- [Access AWS Network Firewall using an interface endpoint](#)

Data protection in Network Firewall

The AWS [shared responsibility model](#) applies to data protection in AWS Network Firewall. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the [Data Privacy FAQ](#). For information about data protection in Europe, see the [AWS Shared Responsibility Model and GDPR](#) blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail. For information about using CloudTrail trails to capture AWS activities, see [Working with CloudTrail trails](#) in the *AWS CloudTrail User Guide*.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see [Federal Information Processing Standard \(FIPS\) 140-3](#).

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with Network Firewall or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Network Firewall entities—such as firewalls, firewall policies, and rule groups—are encrypted at rest, except in certain Regions where encryption is not available, including China (Beijing) and China (Ningxia). Unique encryption keys are used for each Region.

Topics

- [Encryption at rest with AWS Key Management Service](#)

Encryption at rest with AWS Key Management Service

By default, AWS Network Firewall provides encryption for your data at rest using *AWS owned keys* to protect sensitive customer data. Or, you can use symmetric *customer managed keys* that you create, own, and manage to encrypt your data at rest.

- **AWS owned keys** — Network Firewall uses these keys to automatically encrypt personally identifiable data. You can't view, manage, or use AWS owned keys, or audit their use. However, you don't have to take any action or change any programs to protect the keys that encrypt your data. Encryption of data at rest by default helps reduce the operational overhead and complexity involved in protecting sensitive data. At the same time, it enables you to build secure applications that meet strict encryption compliance and regulatory requirements. For more information about using AWS owned keys, see [AWS owned keys](#) in the *AWS Key Management Service Developer Guide*.
- **Customer managed keys** — Network Firewall supports the use of a symmetric customer managed key that you create, own, and manage. You can specify a customer managed key when you create your firewall, firewall policy, and rule group resources. Because you have full control over this type of encryption, you can perform such tasks as:
 - Establishing and maintaining key policies
 - Establishing and maintaining IAM policies and grants
 - Enabling and disabling key policies
 - Rotating key cryptographic material
 - Adding tags
 - Creating key aliases
 - Scheduling keys for deletion

For more information, see [Customer managed keys](#) in the *AWS Key Management Service Developer Guide*.

Note

Network Firewall automatically enables encryption at rest using AWS owned keys to protect personally identifiable data at no charge. Standard AWS KMS charges apply when using customer managed keys. For more information about pricing, see the [AWS Key Management Service pricing](#).

For general information on AWS KMS, see [What is AWS Key Management Service?](#) in the *AWS Key Management Service Developer Guide*.

Using customer managed keys with Network Firewall

The following section provides information about using customer managed keys with Network Firewall.

Topics

- [How AWS Network Firewall uses grants in AWS KMS](#)
- [Creating a customer managed key](#)
- [Specifying a customer managed key for Network Firewall](#)
- [AWS Network Firewall encryption context](#)
- [Monitoring customer managed keys](#)

How AWS Network Firewall uses grants in AWS KMS

Network Firewall requires a [grant](#) to use your customer managed key.

When you create a firewall, firewall policy, or rule group encrypted with a customer managed key, Network Firewall creates a grant on your behalf by sending a [CreateGrant](#) request to AWS KMS. Grants in AWS KMS are used to give Network Firewall access to a KMS key in a customer account.

Network Firewall requires the grant to use your customer managed key for the following internal operations:

- Send [DescribeKey](#) requests to AWS KMS to verify that the symmetric customer managed key ID entered when creating a firewall, firewall policy, or rule group is valid.
- Send [GenerateDataKey](#) requests to AWS KMS to generate data keys encrypted by your customer managed key.
- Send [Decrypt](#) requests to AWS KMS to decrypt the encrypted data keys so that they can be used to encrypt your data.

Important

You can revoke access to the grant or delete the customer managed key at any time. However if you do, the Network Firewall resources that are dependent on the customer managed key won't function. The firewall endpoints encrypted with the customer managed key will drop all packets. To continue to use the resource types that are associated with the customer managed key, you must delete the dependent resources, and then recreate your resources with a key containing adequate permissions.

Creating a customer managed key

You can create a symmetric customer managed key by using the AWS Management Console, or the AWS KMS APIs.

To create a symmetric customer managed key

Follow the steps for [Creating symmetric customer managed key](#) in the *AWS Key Management Service Developer Guide*.

Key policy

Key policies control access to your customer managed key. Every customer managed key must have exactly one key policy, which contains statements that determine who can use the key and how they can use it. When you create your customer managed key, you can specify a key policy. For more information, see [Managing access to customer managed keys](#) in the *AWS Key Management Service Developer Guide*.

To use your customer managed key with your Network Firewall resources, the following API operations must be permitted in the key policy:

- [kms:CreateGrant](#) – Adds a grant to a customer managed key. Grants control access to a specified KMS key, which allows access to [grant operations](#) Network Firewall requires. For more information, see [Using Grants](#) in the *AWS Key Management Service Developer Guide*.

This allows Network Firewall to do the following:

- Call `GenerateDataKey` to generate an encrypted data key and store it, because the data key isn't immediately used to encrypt.
- Call `Decrypt` to use the stored encrypted data key to access encrypted data.
- Set up a retiring principal to allow the service to `RetireGrant`.

The following are policy statement examples you can add for Network Firewall:

```
{
  "Id": "key-consolepolicy-3",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "kms:*",
      "Resource": "*"
    },
    {
      "Sid": "Allow access for Key Administrators",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/Admin"
      },
      "Action": [
        "kms:Create*",
        "kms:Describe*",
        "kms:Enable*",
        "kms:List*",
        "kms:Put*",
        "kms:Update*",
        "kms:Revoke*",
        "kms:Disable*",
```

```

        "kms:Get*",
        "kms:Delete*",
        "kms:TagResource",
        "kms:UntagResource",
        "kms:ScheduleKeyDeletion",
        "kms:CancelKeyDeletion"
    ],
    "Resource": "*"
},
{
    "Sid": "Allow use of the key",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*"
},
{
    "Sid": "Allow attachment of persistent resources",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
    },
    "Action": [
        "kms:CreateGrant",
        "kms:ListGrants",
        "kms:RevokeGrant"
    ],
    "Resource": "*",
    "Condition": {
        "Bool": {
            "kms:GrantIsForAWSResource": "true"
        }
    }
}
]
}

```

For information about [specifying permissions in a policy](#), see the *AWS Key Management Service Developer Guide*. For more information about [troubleshooting key access](#), see the *AWS Key Management Service Developer Guide*.

Specifying a customer managed key for Network Firewall

You can specify a customer managed key for the following resources:

- Firewalls
- Firewall policies
- Rule groups
- TLS inspection configurations

When you create a resource, you can specify the data key by entering an *AWS KMS ID*, which Network Firewall uses to encrypt the identifiable personal data stored by the resource. An AWS KMS ID is a [key identifier](#) for an AWS KMS customer managed key. You can specify a key ID, key ARN, alias name, or alias ARN.

AWS Network Firewall encryption context

An [encryption context](#) is an optional set of key-value pairs that contain additional contextual information about the data. AWS KMS uses the encryption context as additional authenticated data to support authenticated encryption. When you include an encryption context in a request to encrypt data, AWS KMS binds the encryption context to the encrypted data. To decrypt data, you include the same encryption context in the request.

AWS Network Firewall encryption context

Network Firewall uses the same encryption context in all AWS KMS cryptographic operations, where the key is `aws:network-firewall:resource-id` and the value is the resource [Amazon Resource Name](#) (ARN).

Example

```
"encryptionContext": {
  "aws:network-firewall:resource-id": "abcdef-b795-4280-8560-3c2b5e723c41"
}
```


Using encryption context for monitoring

When you use a symmetric customer managed key to encrypt your Network Firewall resources, you can also use the encryption context in audit records and logs to identify how the customer managed key is being used. The encryption context also appears in [logs generated by AWS CloudTrail or Amazon CloudWatch Logs](#).

Using encryption context to control access to your customer managed key

You can use the encryption context in key policies and IAM policies as conditions to control access to your symmetric customer managed key. You can also use encryption context constraints in a grant.

Network Firewall uses an encryption context constraint in grants to control access to the customer managed key in your account or Region. The grant constraint requires that the operations that the grant allows use the specified encryption context.

Example

The following are example key policy statements to grant access to a customer managed key for a specific encryption context. The condition in this policy statement requires that the grants have an encryption context constraint that specifies the encryption context.

```
{
  "KeyId":
    "arn:aws:kms:region:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
  "GrantId": "grant_ID",
  "Name": "11223344-abcd-1111-2222-111222333444",
  "CreationDate": "2022-03-16T14:42:42-04:00",
  "GranteePrincipal": "network-firewall.region.amazonaws.com",
  "RetiringPrincipal": "network-firewall.region.amazonaws.com",
  "IssuingAccount": "111122223333",
  "Operations": [
    "Decrypt",
    "GenerateDataKey",
    "RetireGrant"
  ],
  "Constraints": {
    "EncryptionContextSubset": {
      "aws:network-firewall:resource-id": "11223344-aabb-1122-3344-111222333444"
    }
  }
}
```

Monitoring customer managed keys

When you use an AWS KMS customer managed key with your AWS Network Firewall resources, you can use [AWS CloudTrail](#) or [Amazon CloudWatch Logs](#) to track requests that Network Firewall sends to AWS KMS.

The following example is an AWS CloudTrail event for `CreateGrant` to monitor KMS operations called by Network Firewall to access data encrypted by your customer managed key:

When you use an AWS KMS customer managed key to encrypt your Network Firewall resources, Network Firewall sends a `CreateGrant` request on your behalf to access the KMS key in your AWS account. The grant that Network Firewall creates are specific to the resource associated with the AWS KMS customer managed key. In addition, Network Firewall uses the `RetireGrant` operation to remove a grant when you delete a resource.

The following example event records the `CreateGrant` operation:

CreateGrant

When you use an AWS KMS customer managed key to encrypt your Network Firewall resources, Network Firewall sends a `CreateGrant` request on your behalf to access the customer managed key in your AWS account. The grant that Network Firewall creates are specific to the resource associated with the AWS KMS customer managed key. In addition, Network Firewall uses the `RetireGrant` operation to remove a grant when you delete a resource.

The following example event records the `CreateGrant` operation:

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAIQDTESTANDEXAMPLE:Sampleuser01",
    "arn": "arn:aws:sts::555555555555:assumed-role/Admin/example_Account",
    "accountId": "555555555555",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAIQDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:iam::555555555555:role/Admin",
```

```

        "accountId": "555555555555",
        "userName": "Admin"
    },
    "webIdFederationData": {},
    "attributes": {
        "creationDate": "2022-03-16T18:41:09Z",
        "mfaAuthenticated": "false"
    }
},
"invokedBy": "555555555555"
},
"eventTime": "2022-03-16T18:42:42Z",
"eventSource": "kms.amazonaws.com",
"eventName": "CreateGrant",
"awsRegion": "region",
"sourceIPAddress": "192.0.2.0/24",
"userAgent": "111122223333",
"requestParameters": {
    "keyId":
"arn:aws:kms:region:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
    "retiringPrincipal": "network-firewall.region.amazonaws.com",
    "operations": [
        "Decrypt",
        "GenerateDataKey",
        "RetireGrant"
    ],
    "granteePrincipal": "network-firewall.region.amazonaws.com",
    "constraints": {
        "encryptionContextSubset": {
            "aws:network-firewall:resource-id":
"1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    }
},
"responseElements": {
    "grantId":
"0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE"
},
"requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
"eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
"readOnly": false,
"resources": [
    {
        "accountId": "093688922507",

```

```

        "type": "AWS::KMS::Key",
        "ARN":
"arn:aws:kms:region:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "555555555555",
  "sharedEventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
  "eventCategory": "Management"
}

```

GenerateDataKey

When you enable an AWS KMS customer managed key for your Network Firewall resource, Network Firewall creates a unique table key. It sends a GenerateDataKey request to AWS KMS that specifies the AWS KMS customer managed key for the resource.

The following example event records the GenerateDataKey operation:

```

{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AWSService",
    "invokedBy": "444455556666"
  },
  "eventTime": "2022-03-10T17:16:28Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "GenerateDataKey",
  "awsRegion": "region",
  "sourceIPAddress": "198.51.100.0/24",
  "userAgent": "ExampleDesktop/1.0 (V1; OS)",
  "requestParameters": {
    "keySpec": "AES_256",
    "keyId":
"arn:aws:kms:region:444455556666:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
    "encryptionContext": {
      "aws:network-firewall:resource-id":
"1234abcd-12ab-34cd-56ef-123456SAMPLE",
      "aws:s3:arn": "arn:aws:s3:::service-bucket/stateless-
rulegroup/9876abcd-12ab-23cd-56ef-123456SAMPLE/1/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    }
  }
}

```

```

},
"responseElements": null,
"requestID": "aa000af-00eb-00ce-0e00-ea000gh0fba0SAMPLE",
"eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
"readOnly": true,
"resources": [
  {
    "accountId": "444455556666",
    "type": "AWS::KMS::Key",
    "ARN":
"arn:aws:kms:region:444455556666:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
  }
],
"eventType": "AwsApiCall",
"managementEvent": true,
"recipientAccountId": "444455556666",
"sharedEventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
"eventCategory": "Management"
}

```

Decrypt

When you access an encrypted resource, Network Firewall calls the Decrypt operation to use the stored encrypted data key to access the encrypted data.

The following example event records the Decrypt operation:

```

{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AWSService",
    "invokedBy": "444455556666"
  },
  "eventTime": "2022-03-10T17:16:33Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "Decrypt",
  "awsRegion": "region",
  "sourceIPAddress": "198.51.100.0/24",
  "userAgent": "ExampleDesktop/1.0 (V1; OS)",
  "requestParameters": {
    "encryptionContext": {
      "aws:network-firewall:resource-id":
"1234abcd-12ab-34cd-56ef-123456SAMPLE",

```

```

    "aws:s3:arn": "arn:aws:s3:::service-bucket/stateless-
rulegroup/9876abcd-12ab-23cd-56ef-123456SAMPLE/1/1234abcd-12ab-34cd-56ef-123456SAMPLE"
  },
  "encryptionAlgorithm": "SYMMETRIC_DEFAULT"
},
"responseElements": null,
"requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
"eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
"readOnly": true,
"resources": [
  {
    "accountId": "444455556666",
    "type": "AWS::KMS::Key",
    "ARN":
"arn:aws:kms:region:444455556666:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
  }
],
"eventType": "AwsApiCall",
"managementEvent": true,
"recipientAccountId": "444455556666",
"sharedEventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
"eventCategory": "Management"
}

```

Learn more

The following resources provide more information about data encryption at rest.

- For more information about [AWS Key Management Service basic concepts](#), see the *AWS Key Management Service Developer Guide*.
- For more information about [Security best practices for AWS Key Management Service](#), see the *AWS Key Management Service Developer Guide*.

Identity and Access Management for AWS Network Firewall

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

and *authorized* (have permissions) to use Network Firewall resources. IAM is an AWS service that you can use with no additional charge.

Topics

- [Audience](#)
- [Authenticating with identities](#)
- [Managing access using policies](#)
- [How AWS Network Firewall works with IAM](#)
- [Identity-based policy examples for AWS Network Firewall](#)
- [Resource-based policy examples for AWS Network Firewall](#)
- [AWS managed policies for AWS Network Firewall](#)
- [Troubleshooting AWS Network Firewall identity and access](#)
- [Using service-linked roles for Network Firewall](#)

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in Network Firewall.

Service user – If you use the Network Firewall service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more Network Firewall features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in Network Firewall, see [Troubleshooting AWS Network Firewall identity and access](#).

Service administrator – If you're in charge of Network Firewall resources at your company, you probably have full access to Network Firewall. It's your job to determine which Network Firewall features and resources your service users should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with Network Firewall, see [How AWS Network Firewall works with IAM](#).

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to Network Firewall. To view example Network Firewall identity-based policies that you can use in IAM, see [Identity-based policy examples for AWS Network Firewall](#).

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be *authenticated* (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (IAM Identity Center) users, your company's single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see [How to sign in to your AWS account](#) in the *AWS Sign-In User Guide*.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see [AWS Signature Version 4 for API requests](#) in the *IAM User Guide*.

Regardless of the authentication method that you use, you might be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see [Multi-factor authentication](#) in the *AWS IAM Identity Center User Guide* and [AWS Multi-factor authentication in IAM](#) in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see [Tasks that require root user credentials](#) in the *IAM User Guide*.

Federated identity

As a best practice, require human users, including users that require administrator access, to use federation with an identity provider to access AWS services by using temporary credentials.

A *federated identity* is a user from your enterprise user directory, a web identity provider, the AWS Directory Service, the Identity Center directory, or any user that accesses AWS services by using credentials provided through an identity source. When federated identities access AWS accounts, they assume roles, and the roles provide temporary credentials.

For centralized access management, we recommend that you use AWS IAM Identity Center. You can create users and groups in IAM Identity Center, or you can connect and synchronize to a set of users and groups in your own identity source for use across all your AWS accounts and applications. For information about IAM Identity Center, see [What is IAM Identity Center?](#) in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An [IAM user](#) is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see [Rotate access keys regularly for use cases that require long-term credentials](#) in the *IAM User Guide*.

An [IAM group](#) is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see [Use cases for IAM users](#) in the *IAM User Guide*.

IAM roles

An [IAM role](#) is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. To temporarily assume an IAM role in the AWS Management Console, you can [switch from a user to an IAM role \(console\)](#). You can assume a

role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see [Methods to assume a role](#) in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see [Create a role for a third-party identity provider \(federation\)](#) in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permission sets, see [Permission sets](#) in the *AWS IAM Identity Center User Guide*.
- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see [Cross account resource access in IAM](#) in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
 - **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see [Forward access sessions](#).
- **Service role** – A service role is an [IAM role](#) that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see [Create a role to delegate permissions to an AWS service](#) in the *IAM User Guide*.

- **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see [Use an IAM role to grant permissions to applications running on Amazon EC2 instances](#) in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see [Overview of JSON policies](#) in the *IAM User Guide*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can

perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see [Define custom IAM permissions with customer managed policies](#) in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see [Choose between managed policies and inline policies](#) in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are *IAM role trust policies* and *Amazon S3 bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must [specify a principal](#) in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see [Access control list \(ACL\) overview](#) in the *Amazon Simple Storage Service Developer Guide*.

Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user

or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of an entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see [Permissions boundaries for IAM entities](#) in the *IAM User Guide*.

- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see [Service control policies](#) in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – RCPs are JSON policies that you can use to set the maximum available permissions for resources in your accounts without updating the IAM policies attached to each resource that you own. The RCP limits permissions for resources in member accounts and can impact the effective permissions for identities, including the AWS account root user, regardless of whether they belong to your organization. For more information about Organizations and RCPs, including a list of AWS services that support RCPs, see [Resource control policies \(RCPs\)](#) in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see [Session policies](#) in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see [Policy evaluation logic](#) in the *IAM User Guide*.

How AWS Network Firewall works with IAM

Before you use IAM to manage access to Network Firewall, learn what IAM features are available to use with Network Firewall.

IAM features you can use with AWS Network Firewall

IAM feature	Network Firewall support
Identity-based policies	Yes
Resource-based policies	Yes
Policy actions	Yes
Policy resources	Yes
Policy condition keys (service-specific)	No
ACLs	No
ABAC (tags in policies)	Yes
Temporary credentials	Yes
Forward access sessions (FAS)	Yes
Service roles	No
Service-linked roles	Yes

To get a high-level view of how Network Firewall and other AWS services work with most IAM features, see [AWS services that work with IAM](#) in the *IAM User Guide*.

Identity-based policies for Network Firewall

Supports identity-based policies: Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see [Define custom IAM permissions with customer managed policies](#) in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. You can't specify the principal in an identity-based policy because it applies to the user or role to which it is attached. To learn about all

of the elements that you can use in a JSON policy, see [IAM JSON policy elements reference](#) in the *IAM User Guide*.

Identity-based policy examples for Network Firewall

To view examples of Network Firewall identity-based policies, see [Identity-based policy examples for AWS Network Firewall](#).

Resource-based policies within Network Firewall

Supports resource-based policies: Yes

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are *IAM role trust policies* and *Amazon S3 bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must [specify a principal](#) in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship. When the principal and the resource are in different AWS accounts, an IAM administrator in the trusted account must also grant the principal entity (user or role) permission to access the resource. They grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access to a principal in the same account, no additional identity-based policy is required. For more information, see [Cross account resource access in IAM](#) in the *IAM User Guide*.

The Network Firewall service supports only one type of resource-based policy called a *resource policy*, which is attached to a shared firewall policy or rule group. This policy defines which principals can share firewall policies and rule groups between accounts.

To learn how to attach a resource-based policy to a shared firewall policy or rule group, see [Sharing AWS Network Firewall firewall policies and rule groups](#).

Resource-based policy examples within Network Firewall

To view examples of Network Firewall resource-based policies, see [Resource-based policy examples for AWS Network Firewall](#).

Policy actions for Network Firewall

Supports policy actions: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The Action element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

To see a list of Network Firewall actions, see [Actions defined by AWS Network Firewall](#) in the *Service Authorization Reference*.

Policy actions in Network Firewall use the following prefix before the action:

```
network-firewall
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [  
  "network-firewall:action1",  
  "network-firewall:action2"  
]
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word Describe, include the following action:

```
"Action": "network-firewall:Describe*"
```

To view examples of Network Firewall identity-based policies, see [Identity-based policy examples for AWS Network Firewall](#).

Policy resources for Network Firewall

Supports policy resources: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The Resource JSON policy element specifies the object or objects to which the action applies. Statements must include either a Resource or a NotResource element. As a best practice, specify a resource using its [Amazon Resource Name \(ARN\)](#). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*" 
```

To see a list of Network Firewall resource types and their ARNs, see [Resource types defined by AWS Network Firewall](#) in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see [Actions defined by AWS Network Firewall](#).

Some Network Firewall API actions support multiple resources. For example, EXAMPLE-ACTION accesses EXAMPLE-RESOURCE-1 and EXAMPLE-RESOURCE-2, so a principal must have permissions to access both resources. To specify multiple resources in a single statement, separate the ARNs with commas.

```
"Resource": [
  "EXAMPLE-RESOURCE-1",
  "EXAMPLE-RESOURCE-2" ]
```

To view examples of Network Firewall identity-based policies, see [Identity-based policy examples for AWS Network Firewall](#).

Policy condition keys for Network Firewall

Supports service-specific policy condition keys: No

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element (or `Condition block`) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can create conditional expressions that use [condition operators](#), such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical AND operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical OR operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see [IAM policy elements: variables and tags](#) in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see [AWS global condition context keys](#) in the *IAM User Guide*.

To see a list of Network Firewall condition keys, see [Condition keys for AWS Network Firewall](#) in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see [Actions defined by AWS Network Firewall](#).

To view examples of Network Firewall identity-based policies, see [Identity-based policy examples for AWS Network Firewall](#).

ACLs in Network Firewall

Supports ACLs: No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

ABAC with Network Firewall

Supports ABAC (tags in policies): Yes

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the [condition element](#) of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see [Define permissions with ABAC authorization](#) in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see [Use attribute-based access control \(ABAC\)](#) in the *IAM User Guide*.

Using temporary credentials with Network Firewall

Supports temporary credentials: Yes

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see [AWS services that work with IAM](#) in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see [Switch from a user to an IAM role \(console\)](#) in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see [Temporary security credentials in IAM](#).

Forward access sessions for Network Firewall

Supports forward access sessions (FAS): Yes

When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a

different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see [Forward access sessions](#).

Service roles for Network Firewall

Supports service roles: No

A service role is an [IAM role](#) that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see [Create a role to delegate permissions to an AWS service](#) in the *IAM User Guide*.

Warning

Changing the permissions for a service role might break Network Firewall functionality. Edit service roles only when Network Firewall provides guidance to do so.

Service-linked roles for Network Firewall

Supports service-linked roles: Yes

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing service-linked roles, see [AWS services that work with IAM](#). Find a service in the table that includes a Yes in the **Service-linked role** column. Choose the **Yes** link to view the service-linked role documentation for that service.

Identity-based policy examples for AWS Network Firewall

By default, users and roles don't have permission to create or modify Network Firewall resources. They also can't perform tasks by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS API. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see [Create IAM policies \(console\)](#) in the *IAM User Guide*.

For details about actions and resource types defined by Network Firewall, including the format of the ARNs for each of the resource types, see [Actions, resources, and condition keys for AWS Network Firewall](#) in the *Service Authorization Reference*.

Topics

- [Policy best practices](#)
- [Using the Network Firewall console](#)
- [Allow users to view their own permissions](#)

Policy best practices

Identity-based policies determine whether someone can create, access, or delete Network Firewall resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see [AWS managed policies](#) or [AWS managed policies for job functions](#) in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see [Policies and permissions in IAM](#) in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as AWS CloudFormation. For more information, see [IAM JSON policy elements: Condition](#) in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies

adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see [Validate policies with IAM Access Analyzer](#) in the *IAM User Guide*.

- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see [Secure API access with MFA](#) in the *IAM User Guide*.

For more information about best practices in IAM, see [Security best practices in IAM](#) in the *IAM User Guide*.

Using the Network Firewall console

To access the AWS Network Firewall console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the Network Firewall resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (users or roles) with that policy.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that they're trying to perform.

The AWS Network Firewall console resides inside the Amazon Virtual Private Cloud (Amazon VPC) console, so you must have the permissions documented at [Viewing the Amazon VPC console](#) in the [Amazon Virtual Private Cloud User Guide](#). These permissions allow you to list and view details about the Network Firewall resources in your AWS account.

To ensure that users and roles can still use the Network Firewall console, also attach the Network Firewall *ConsoleAccess* or *ReadOnly* AWS managed policy to the entities. For more information, see [Adding permissions to a user](#) in the *IAM User Guide*.

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsWithUser",
        "iam:ListAttachedUserPolicies",
        "iam:ListUserPolicies",
        "iam:GetUser"
      ],
      "Resource": ["arn:aws:iam::*:user/${aws:username}"]
    },
    {
      "Sid": "NavigateInConsole",
      "Effect": "Allow",
      "Action": [
        "iam:GetGroupPolicy",
        "iam:GetPolicyVersion",
        "iam:GetPolicy",
        "iam:ListAttachedGroupPolicies",
        "iam:ListGroupPolicies",
        "iam:ListPolicyVersions",
        "iam:ListPolicies",
        "iam:ListUsers"
      ],
      "Resource": "*"
    }
  ]
}
```

Resource-based policy examples for AWS Network Firewall

The Network Firewall service supports only one type of resource-based policy called a *resource policy*, which is attached to a shared firewall policy or rule group. This policy defines which principals can share firewall policies and rule groups between accounts.

To learn how to attach a resource policy to a shared rule group or firewall policy, see [Sharing AWS Network Firewall firewall policies and rule groups](#).

Topics

- [Enable sharing of a firewall policy with an account](#)

Enable sharing of a firewall policy with an account

The following example grants permissions to the service principal to create or update a resource policy for a firewall policy that's shared across accounts. In the resource policy, you specify the accounts that you want to share the resource with and the operations that you want the accounts to be able to perform.

For information about sharing resources in Network Firewall, see [Sharing AWS Network Firewall firewall policies and rule groups](#).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "123456789012"
      },
      "Action": ["network-firewall:AssociateFirewallPolicy", "network-
firewall:ListFirewallPolicies"],
      "Resource": "arn:aws:network-firewall:aws-region:123456789012:firewall-
policy/test-action"
    }
  ]
}
```

AWS managed policies for AWS Network Firewall

To add permissions to users, groups, and roles, it is easier to use AWS managed policies than to write policies yourself. It takes time and expertise to [create IAM customer managed policies](#) that provide your team with only the permissions they need. To get started quickly, you can use our

AWS managed policies. These policies cover common use cases and are available in your AWS account. For more information about AWS managed policies, see [AWS managed policies](#) in the *IAM User Guide*.

AWS services maintain and update AWS managed policies. You can't change the permissions in AWS managed policies. Services occasionally add additional permissions to an AWS managed policy to support new features. This type of update affects all identities (users, groups, and roles) where the policy is attached. Services are most likely to update an AWS managed policy when a new feature is launched or when new operations become available. Services do not remove permissions from an AWS managed policy, so policy updates won't break your existing permissions.

Additionally, AWS supports managed policies for job functions that span multiple services. For example, the **ReadOnlyAccess** AWS managed policy provides read-only access to all AWS services and resources. When a service launches a new feature, AWS adds read-only permissions for new operations and resources. For a list and descriptions of job function policies, see [AWS managed policies for job functions](#) in the *IAM User Guide*.

Network Firewall updates to AWS managed policies

View details about updates to AWS managed policies for Network Firewall since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the Network Firewall Document history page.

Change	Description	Date
AWSNetworkFirewallServiceRolePolicy – Update to the existing policy	Updated the AWSNetworkFirewallServiceRolePolicy to support describing ACM certificates for use with TLS inspection configurations. For policy details, see AWSNetworkFirewallServiceRolePolicy .	March 31, 2023

Change	Description	Date
AWSNetworkFirewallServiceRolePolicy – Update to the existing policy	<p>AWS Network Firewall expanded availability of the policy to the AWS GovCloud (US) Regions, AWS GovCloud (US-East) and AWS GovCloud (US-West).</p> <p><code>AWSNetworkFirewallServiceRolePolicy</code> is an access policy that allows Network Firewall to manage Network Firewall related resources on behalf of your AWS account. Network Firewall uses this policy to create, describe, and delete VPC endpoints in support of your firewall management activities.</p> <p>For policy details, see AWSNetworkFirewallServiceRolePolicy.</p> <p>This policy uses the service-linked role <code>AWSServiceRoleForNetworkFirewall</code>. For more information, see Using service-linked roles for Network Firewall.</p>	June 24, 2021
Network Firewall started tracking changes	Network Firewall started tracking changes for its AWS managed policies.	June 24, 2021

Troubleshooting AWS Network Firewall identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with Network Firewall and IAM.

Topics

- [I am not authorized to perform an action in Network Firewall](#)
- [I am not authorized to perform iam:PassRole](#)
- [I want to allow people outside of my AWS account to access my Network Firewall resources](#)

I am not authorized to perform an action in Network Firewall

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional `my-example-widget` resource but doesn't have the fictional `network-firewall:GetWidget` permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
network-firewall:GetWidget on resource: my-example-widget
```

In this case, the policy for the `mateojackson` user must be updated to allow access to the `my-example-widget` resource by using the `network-firewall:GetWidget` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to Network Firewall.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in Network Firewall. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my AWS account to access my Network Firewall resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether Network Firewall supports these features, see [How AWS Network Firewall works with IAM](#).
- To learn how to provide access to your resources across AWS accounts that you own, see [Providing access to an IAM user in another AWS account that you own](#) in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see [Providing access to AWS accounts owned by third parties](#) in the *IAM User Guide*.
- To learn how to provide access through identity federation, see [Providing access to externally authenticated users \(identity federation\)](#) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see [Cross account resource access in IAM](#) in the *IAM User Guide*.

Using service-linked roles for Network Firewall

AWS Network Firewall uses AWS Identity and Access Management (IAM) [service-linked roles](#). A service-linked role is a unique type of IAM role that is linked directly to Network Firewall. Service-linked roles are predefined by Network Firewall and include all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes setting up Network Firewall easier because you don't have to manually add the necessary permissions. Network Firewall defines the permissions of its service-linked roles, and unless defined otherwise, only Network Firewall can assume its roles. The defined permissions include the trust policy and the permissions policy. That permissions policy can't be attached to any other IAM entity.

You can delete a service-linked role only after first deleting its related resources. This protects your Network Firewall resources because you can't inadvertently remove permission to access the resources.

For information about other services that support service-linked roles, see [AWS services that work with IAM](#) and look for the services that have **Yes** in the **Service-Linked Role** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

Service-linked role permissions for Network Firewall

Network Firewall uses the service-linked role named **AWSServiceRoleForNetworkFirewall** – An access policy that allows AWS Network Firewall to manage Network Firewall related resources on behalf of your AWS account. Network Firewall uses its service-linked-role to create, describe, and delete VPC endpoints in support of your firewall management activities. Network Firewall is the only service that uses this service-linked role, and Network Firewall doesn't use any other service's service-linked role. This service-linked role is used in the Network Firewall managed policy `AWSNetworkFirewallServiceRolePolicy`. For more information, see [AWS managed policies for AWS Network Firewall](#).

The `AWSServiceRoleForNetworkFirewall` service-linked role trusts the `network-firewall.amazonaws.com` service principal to assume the role. The following is the JSON trust policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "network-firewall.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

The role permissions policy allows Network Firewall to perform the following actions:

- Describe and create Amazon EC2 VPC resources for firewall management.
- Describe ACM certificates for use with TLS inspection configurations.
- Create and manage resource groups.
- Periodically check the VPC CIDR blocks and management of firewall endpoints in the VPC.

- Describe Amazon EC2 instances and Amazon EC2 network interfaces for use in resource groups.

The following is the JSON role permissions policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcs",
        "ec2:CreateVpcEndpoint",
        "ec2:DescribeVpcEndpoints",
        "ec2:DescribeInstances",
        "ec2:DescribeNetworkInterfaces"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "acm:DescribeCertificate",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "resource-groups:ListGroupResources",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "tag:GetResources",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:CalledViaLast": "resource-groups.amazonaws.com"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateTags"
      ]
    }
  ]
}
```

```

    ],
    "Resource": "arn:aws:ec2:*:*:vpc-endpoint/*",
    "Condition": {
        "StringEquals": {
            "ec2:CreateAction": "CreateVpcEndpoint",
            "aws:RequestTag/AWSNetworkFirewallManaged": "true"
        }
    }
},
{
    "Effect": "Allow",
    "Action": [
        "ec2:DeleteVpcEndpoints"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "aws:ResourceTag/AWSNetworkFirewallManaged": "true"
        }
    }
}
]
}

```

When you enable logging for a firewall, Network Firewall uses a log delivery service, which might create a service-linked role in your account named `AWSServiceRoleForLogDelivery` to deliver logs.

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. For more information, see [Service-Linked Role Permissions](#) in the *IAM User Guide*.

Creating a service-linked role for Network Firewall

You don't need to manually create a service-linked role for AWS Network Firewall. When you create a AWS Network Firewall firewall in the AWS Management Console, the AWS CLI, or the AWS API, if your account doesn't have the Network Firewall service-linked role yet, Network Firewall creates the role for you. If you manage your firewall resources through AWS Firewall Manager, Firewall Manager creates the service-linked role for accounts that are within scope of the Firewall Manager policy. If you want to, you can create the role through the IAM console. If you delete the service-linked role, the next time you create an Network Firewall resource, Network Firewall creates one for you again.

Editing a service-linked role for Network Firewall

Network Firewall doesn't allow you to edit the `AWSServiceRoleForNetworkFirewall` service-linked role. After you create a service-linked role, you can't change the name of the role because various entities might reference it. However, you can edit the description of the role using IAM. For more information, see [Editing a Service-Linked Role](#) in the *IAM User Guide*.

Deleting a service-linked role for Network Firewall

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored or maintained. You must clean up the resources that require your service-linked role before you can manually delete it. You can delete the service-linked role used by Network Firewall if you no longer want to use the service. To delete the role, you must delete all firewalls, firewall policies, stateful rule groups, and stateless rule groups, in all Regions where you have them defined.

Note

If the Network Firewall service is using the role when you try to delete the resources, then the deletion might fail. If that happens, wait for a few minutes and try the operation again.

To delete all Network Firewall resources

1. On the Amazon VPC console, update all route tables that send traffic through your firewall endpoints, to remove the endpoints from the traffic flow. For information about managing route tables for your VPC, see [Route tables](#) in the *Amazon Virtual Private Cloud User Guide*.
2. On the Network Firewall console, remove your firewalls, firewall policies, stateful rules groups, and stateless rule groups. For information, see [Deleting a firewall in AWS Network Firewall](#), [Deleting a firewall policy in AWS Network Firewall](#), and [Deleting a stateless rule group](#).

This removes all resources that Network Firewall used the service-linked role for.

To manually delete the service-linked role using IAM

Use the IAM console, the IAM CLI, or the IAM API to delete the `AWSServiceRoleForNetworkFirewall` service-linked role. For more information, see [Deleting a Service-Linked Role](#) in the *IAM User Guide*.

Supported Regions for Network Firewall service-linked roles

Network Firewall supports using service-linked roles in all of the Regions where the service is available. For a Region list, see [AWS Regions and Endpoints](#).

AWS logging and monitoring tools

This section provides an overview of the tools available for logging and monitoring in AWS Network Firewall for standard AWS security purposes. For more information about logging and monitoring in Network Firewall see [Logging and monitoring in AWS Network Firewall](#).

Monitoring is an important part of maintaining the reliability, availability, and performance of Network Firewall and your AWS solutions. You should collect monitoring data from all parts of your AWS solution so that you can more easily debug a multi-point failure if one occurs. AWS provides several tools for monitoring your Network Firewall resources and responding to potential incidents:

Amazon CloudWatch Alarms

Using CloudWatch alarms, you watch a single metric over a time period that you specify. If the metric exceeds a given threshold, CloudWatch sends a notification to an Amazon SNS topic or AWS Auto Scaling policy. For more information, see [AWS Network Firewall metrics in Amazon CloudWatch](#).

AWS CloudTrail Logs

CloudTrail provides a record of actions taken by a user, role, or an AWS service in Network Firewall. Using the information collected by CloudTrail, you can determine the request that was made to Network Firewall, the IP address from which the request was made, who made the request, when it was made, and additional details. For more information, see [Logging calls to the AWS Network Firewall API with AWS CloudTrail](#).

AWS Trusted Advisor

Trusted Advisor draws upon best practices learned from serving hundreds of thousands of AWS customers. Trusted Advisor inspects your AWS environment and then makes recommendations when opportunities exist to save money, improve system availability and performance, or help close security gaps. All AWS customers have access to five Trusted Advisor checks. Customers with a Business or Enterprise support plan can view all Trusted Advisor checks. For more information, see [AWS Trusted Advisor](#).

Compliance validation and security best practices for Network Firewall

Third-party auditors assess the security and compliance of Network Firewall, as part of a number of AWS compliance programs. For general information about compliance programs, see [AWS Compliance Programs](#). You can download third-party audit reports using AWS Artifact. For more information, see [Downloading Reports in AWS Artifact](#).

You can see the current compliance status of Network Firewall and other AWS services at [AWS Services in Scope by Compliance Program](#).

When you use Network Firewall, your compliance responsibility is determined by the sensitivity of your data, your organization's compliance objectives, and applicable laws and regulations. If your use of Network Firewall is subject to compliance with standards like HIPAA, PCI, or FedRAMP, AWS provides the following resources to help you:

- [Security and Compliance Quick Start Guides](#) – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- [Architecting for HIPAA Security and Compliance Whitepaper](#) – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- [AWS Compliance Resources](#) – This collection of workbooks and guides might apply to your industry and location.
- [AWS Config](#) – This AWS service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- [AWS Security Hub](#) – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.
- Monitor your usage of Network Firewall as it relates to security best practices by using [AWS Security Hub](#). Security Hub uses security controls to evaluate resource configurations and security standards to help you comply with various compliance frameworks. For more information about using Security Hub to evaluate Network Firewall resources, see [AWS Network Firewall controls](#) in the *AWS Security Hub User Guide*.

Resilience in Network Firewall

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between Availability Zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see [AWS global infrastructure](#).

Infrastructure security in AWS Network Firewall

As a managed service, AWS Network Firewall is protected by AWS global network security. For information about AWS security services and how AWS protects infrastructure, see [AWS Cloud Security](#). To design your AWS environment using the best practices for infrastructure security, see [Infrastructure Protection](#) in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access Network Firewall through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the [AWS Security Token Service](#) (AWS STS) to generate temporary security credentials to sign requests.

Access AWS Network Firewall using an interface endpoint

You can create a private connection between your VPC and AWS Network Firewall. You can access AWS Network Firewall as if it were in your VPC, without the use of an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC don't need public IP addresses to access AWS Network Firewall.

For more information, see [Access an AWS service using an interface VPC endpoint](#) in the *AWS PrivateLink Guide*.

Considerations for AWS Network Firewall

Before you set up an interface endpoint for AWS Network Firewall, review [Interface endpoint properties and limitations](#) in the *AWS PrivateLink Guide*.

AWS Network Firewall supports making calls to all of its API actions through the interface endpoint.

Before you set up interface VPC endpoints for Network Firewall, be aware of the following considerations:

- VPC endpoints currently don't support cross-Region requests. Ensure that you create your endpoint in the same Region where you plan to issue your API calls to Network Firewall.
- VPC endpoints only support Amazon-provided DNS through Amazon Route 53. If you want to use your own DNS, you can use conditional DNS forwarding. For more information, see [DHCP Options Sets](#) in the *Amazon VPC User Guide*.
- The security group attached to the VPC endpoint must allow incoming connections on port 443 from the private subnet of the VPC.
- VPC interface endpoints are supported in all AWS Regions supported by Network Firewall.

Create an interface VPC endpoint for AWS Network Firewall

You can create an interface VPC endpoint using the Amazon VPC Console. For more information, see [Create an interface endpoint](#) in the *AWS PrivateLink Guide*.

When you create an interface VPC endpoint, use the following service name:

```
com.amazonaws.region.network-firewall
```

For example:

```
com.amazonaws.us-west-2.network-firewall
```

Create a VPC endpoint policy for AWS Network Firewall

An endpoint policy is an IAM resource that you can attach to an interface endpoint. The default endpoint policy allows full access to AWS Network Firewall through the interface endpoint. To control the access allowed to AWS Network Firewall from your VPC, attach a custom endpoint policy to the interface endpoint.

An endpoint policy specifies the following information:

- The principals that can perform actions (AWS accounts, users, and IAM roles).
- The actions that can be performed.
- The resources on which the actions can be performed.

Example: VPC endpoint policy for AWS Network Firewall

The following is an example of a custom endpoint policy. When you attach this policy to your interface VPC endpoint, it grants access to the AWS Network Firewall actions for all principals on all resources.

```
{
  "Statement": [
    {
      "Principal": "*",
      "Effect": "Allow",
      "Action": [
        "network-firewall:ListFirewalls",
        "network-firewall:DescribeFirewall"
      ],
      "Resource": "*"
    }
  ]
}
```

Logging and monitoring in AWS Network Firewall

Logging and monitoring help you to maintain the reliability, availability, and performance of AWS Network Firewall. You can monitor how the service is being used and you can monitor network traffic and traffic filtering done by the stateful rule groups in your Network Firewall firewalls.

AWS provides a number of tools that you can use to monitor Network Firewall. You can configure some of these tools to do the monitoring for you, while other tools require manual intervention. We recommend that you automate monitoring tasks as much as possible.

You can use the following automated monitoring tools with Network Firewall:

- *Amazon CloudWatch* provides metrics for the AWS resources and the applications that you run on AWS. Monitoring and alarms are real time. You can collect and track metrics, create customized dashboards, and set alarms that notify you or take actions when a specified metric reaches a threshold that you specify. For example, you can have CloudWatch track CPU usage or other metrics of your Amazon EC2 instances and automatically launch new instances when needed. For more information, see the [Amazon CloudWatch User Guide](#).
- *Amazon CloudWatch Logs* provides logging for sources such as Amazon EC2 instances and CloudTrail. CloudWatch Logs can monitor information in the log files and notify you when certain thresholds are met. You can also archive your log data in highly durable storage. For more information, see the [Amazon CloudWatch Logs User Guide](#).
- *AWS CloudTrail* captures API calls and related events made by or on behalf of your AWS account and delivers the log files to an Amazon S3 bucket that you specify. You can identify which users and accounts called AWS, the source IP address from which the calls were made, and when the calls occurred. For more information, see the [AWS CloudTrail User Guide](#).
- *AWS Config* lets you view the configuration of your AWS resources in your AWS account. The available information includes how the resources are related to one another and how they were configured in the past, so that you can see how the configurations and relationships change over time. For more information, see the [AWS Config Developer Guide](#).

Topics

- [Logging network traffic from AWS Network Firewall](#)
- [Logging calls to the AWS Network Firewall API with AWS CloudTrail](#)
- [AWS Network Firewall metrics in Amazon CloudWatch](#)

Logging network traffic from AWS Network Firewall

You can configure AWS Network Firewall logging for your firewall's stateful engine. Logging gives you detailed information about network traffic, including the time that the stateful engine received a packet, detailed information about the packet, and any stateful rule action taken against the packet. The logs are published to the log destination that you've configured, where you can retrieve and view them.

Note

Firewall logging is only available for traffic that you forward to the stateful rules engine. You forward traffic to the stateful engine through stateless rule actions and stateless default actions in the firewall policy. For information about these actions settings, see [Firewall policy settings in AWS Network Firewall](#) and [Defining rule actions in AWS Network Firewall](#).

Metrics provide some higher-level information for both stateless and stateful engine types. For more information, see [AWS Network Firewall metrics in Amazon CloudWatch](#).

You can record the following types of logs from your Network Firewall stateful engine.

- Flow logs are standard network traffic flow logs. Each flow log record captures the network flow for a specific standard stateless rule group.
- Alert logs report traffic that matches your stateful rules that have an action that sends an alert. A stateful rule sends alerts for the rule actions DROP, ALERT, and REJECT. For more information, see [Stateful actions](#).
- TLS logs report events that are related to TLS inspection. These logs require the firewall to be configured for TLS inspection. For information, see [Inspecting SSL/TLS traffic with TLS inspection configurations in AWS Network Firewall](#).

You can use the same or different logging destination for each log type. You enable logging for a firewall after you create it. For information about how to do this, see [Updating a AWS Network Firewall logging configuration](#).

Topics

- [Contents of a AWS Network Firewall log](#)
- [Timing of AWS Network Firewall log delivery](#)

- [Permissions to configure AWS Network Firewall logging](#)
- [Pricing for AWS Network Firewall logging](#)
- [AWS Network Firewall logging destinations](#)
- [Logging in AWS Network Firewall with server-side encryption and customer-provided keys](#)
- [Updating a AWS Network Firewall logging configuration](#)

Contents of a AWS Network Firewall log

The Network Firewall logs contain the following information:

- **firewall_name** – The name of the firewall that's associated with the log entry.
- **availability_zone** – The Availability Zone of the firewall endpoint that generated the log entry.
- **event_timestamp** – The time that the log was created, written in epoch seconds at Coordinated Universal Time (UTC).
- **event** – Detailed information about the event. This information includes the event timestamp converted to human readable format, event type, network packet details, and, if applicable, details about the stateful rule that the packet matched against.
 - **Alert and flow events** – Alert and flow events are produced by Suricata, the open source threat detection engine that the stateful rules engine runs on. Suricata writes the event information in the Suricata EVE JSON output format, with the exception of the AWS managed `tls_inspected` attribute.
 - Flow log events use the EVE output type `netflow`. The log type `netflow` logs uni-directional flows, so each event represents traffic going in a single direction.
 - Alert log events using the EVE output type `alert`.
 - If the firewall that's associated with the log uses TLS inspection and the firewall's traffic uses SSL/TLS, Network Firewall adds the custom field `"tls_inspected": true` to the log. If your firewall doesn't use TLS inspection, Network Firewall omits this field.

For detailed information about these Suricata events, see [EVE JSON Output](#) in the [Suricata User Guide](#).

- **TLS events** – TLS events are produced by a dedicated stateful TLS engine, which is separate from Suricata. TLS events have the output type `tls`. The logs have a JSON structure that's similar to the Suricata EVE output.

These events require the firewall to be configured for TLS inspection. For information, see [Inspecting SSL/TLS traffic with TLS inspection configurations in AWS Network Firewall](#).

TLS logs report the following types of errors:

- TLS errors, with the custom field "tls_error": containing the error details. Currently, this category includes Server Name Indication (SNI) mismatches and SNI naming errors. Typically these errors are caused by problems with customer traffic or with the customer's client or server. For example, errors caused when the client hello SNI is NULL or doesn't match the subject name in the server certificate.
- Revocation check errors, with the custom field "revocation_check": containing the check failure details. These report outbound traffic that fails the server certificate revocation check during TLS inspection. This requires the firewall to be configured with TLS inspection for outbound traffic, and for the TLS inspection to be configured to check the certificate revocation status. The logs include the revocation check status, the action taken, and the SNI that the revocation check was for. For information about configuring certificate revocation checking, see [Using SSL/TLS certificates with TLS inspection configurations in AWS Network Firewall](#).

For detailed information about these Suricata events, see [EVE JSON Output](#) in the [Suricata User Guide](#).

Example alert log entry

The following listing shows an example alert log entry for Network Firewall.

```
{
  "firewall_name": "test-firewall",
  "availability_zone": "us-east-1b",
  "event_timestamp": "1602627001",
  "event": {
    "timestamp": "2020-10-13T22:10:01.006481+0000",
    "flow_id": 1582438383425873,
    "event_type": "alert",
    "src_ip": "203.0.113.4",
    "src_port": 55555,
    "dest_ip": "192.0.2.16",
    "dest_port": 111,
    "proto": "TCP",
    "alert": {
```

```
        "action": "allowed",
        "signature_id": 5,
        "rev": 0,
        "signature": "test_tcp",
        "category": "",
        "severity": 1
    }
}
```

Example TLS log entry

The following listing shows an example TLS log entry for a failed certificate revocation check.

```
{
  "firewall_name": "egress-fw",
  "availability_zone": "us-east-1d",
  "event_timestamp": 1708361189,
  "event": {
    "src_ip": "10.0.2.53",
    "src_port": "55930",
    "revocation_check": {
      "leaf_cert_fpr": "1234567890EXAMPLE0987654321",
      "status": "REVOKED",
      "action": "DROP"
    },
    "dest_ip": "54.92.160.72",
    "dest_port": "443",
    "timestamp": "2024-02-19T16:46:29.441824Z",
    "sni": "revoked-rsa-dv.ssl.com"
  }
}
```

Timing of AWS Network Firewall log delivery

A log file or log stream generally contains information about the requests that your firewall received during a given time period. The timing of Network Firewall log delivery varies by location type, averaging 3-6 minutes for Amazon CloudWatch Logs and Amazon Data Firehose and 8-12 minutes for Amazon Simple Storage Service buckets. In some cases, logs may take longer than these averages. When log entries are delayed, Network Firewall saves them and then logs them according to the date and time of the period in which the requests occurred, not the date and time when the logs are delivered.

Note

If your firewall doesn't filter traffic for a period of time, you don't receive logs for that period.

When creating a log file or stream, Network Firewall consolidates information for your firewall from all the endpoints that received traffic during the time period that the log covers.

Permissions to configure AWS Network Firewall logging

You must have the following permissions to make any changes to your firewall logging configuration. These settings are included in the permissions requirements for each logging configuration type, under [AWS Network Firewall logging destinations](#).

```
{
  "Action": [
    "logs:CreateLogDelivery",
    "logs:GetLogDelivery",
    "logs:UpdateLogDelivery",
    "logs>DeleteLogDelivery",
    "logs:ListLogDeliveries"
  ],
  "Resource": [
    "*"
  ],
  "Effect": "Allow",
  "Sid": "FirewallLogging"
}
```

The permissions required for logging configuration are in addition to the standard permissions required to use the Network Firewall API. For information about the standard permissions that are required to use Network Firewall, see [Managing access using policies](#).

Pricing for AWS Network Firewall logging

You are charged for Amazon CloudWatch *vended logs*, on top of the basic charges for using Network Firewall. Vended logs are specific AWS service logs published by AWS on your behalf at volume discount pricing. Your logging costs can vary depending on factors such as the destination

type that you choose and the amount of data that you log. For example, flow logging sends logs for all of the network traffic that reaches your firewall's stateful rules, but alert logging sends logs only for network traffic that your stateful rules drop or explicitly alert on. For information about CloudWatch vended log pricing, see [Logs](#) on the *Amazon CloudWatch pricing* page. For information about Network Firewall pricing, see [Network Firewall pricing](#).

AWS Network Firewall logging destinations

This section describes the logging destinations that you can choose from for your Network Firewall logs. Each section provides guidance for configuring logging for the destination type and information about any behavior that's specific to the destination type. After you've configured your logging destination, you can provide its specifications to the firewall logging configuration to start logging to it.

For information about how to update the logging destination for an existing logging configuration, see [Updating a firewall's logging configuration](#).

Topics

- [Sending AWS Network Firewall logs to Amazon Simple Storage Service](#)
- [Sending AWS Network Firewall logs to Amazon CloudWatch Logs](#)
- [Sending AWS Network Firewall logs to Amazon Data Firehose](#)

Sending AWS Network Firewall logs to Amazon Simple Storage Service

To send your firewall logs to Amazon S3, you need to set up an Amazon S3 bucket as the destination for the logs. In your bucket configuration for the firewall, you can optionally include a prefix, to immediately follow the bucket name. When you enable logging to Amazon S3 in Network Firewall, you provide the bucket name and, if you are using one, the prefix. For information about creating your logging bucket, see [Create a Bucket](#) in the *Amazon Simple Storage Service User Guide*.

Note

Network Firewall supports encryption with Amazon S3 buckets for key type Amazon S3 key (SSE-S3) and for AWS Key Management Service (SSE-KMS) AWS KMS keys. Network Firewall doesn't support encryption for AWS Key Management Service keys that are managed by AWS.

Network Firewall collects log records, consolidates them into log files, and then publishes the log files to the Amazon S3 bucket at 5-minute intervals. Each log file contains log records for the network traffic recorded in the previous five minutes.

The maximum file size for a log file is 75 MB. If the log file reaches the file size limit within the 5-minute period, the log stops adding records to it, publishes it to the Amazon S3 bucket, and then creates a new log file.

A single log file contains interleaved entries with multiple connection identifier (source IP address, source port, destination IP address, destination port, and protocol) records. To see all the log files for your firewall, look for entries aggregated by the firewall name and your account ID.

Log files are saved in the specified Amazon S3 bucket using a folder structure that's determined by the log's ID, Region, Network Firewall log type, and the date. The bucket folder structure uses the following format:

```
s3-bucket-name/optional-s3-bucket-prefix/AWSLogs/aws-account-id/network-firewall/log-type/Region/firewall-name/timestamp/
```

Similarly, the log file name is determined by the flow log's ID, Region, and the date and time it was created. File names use the following format:

```
aws-account-id_network-firewall_log-type_Region_firewall-name_timestamp_hash.log.gz
```

In the specification of the folder and file name, the following apply:

- The log type is either alert, flow, or tls.
- The timestamp uses the YYYYMMDDTHHmmZ format.
- If you don't provide a specification for the S3 bucket prefix, the log file bucket folder structure will be similar to the following:

```
s3-bucket-name/AWSLogs/aws-account-id
```

- If you specify slash (/) for the S3 bucket prefix, or provide a prefix that begins with a slash, the log file bucket folder structure will contain a double slash (//), like the following for a prefix set to a single slash:

```
s3-bucket-name//AWSLogs/aws-account-id
```

The following shows an example flow log file in Amazon S3 for AWS account 11111111111, firewall name test-firewall, bucket name s3://amzn-s3-demo-bucket, and bucket prefix flow-logs.

```
s3://amzn-s3-demo-bucket/flow-logs/AWSLogs/11111111111/network-firewall/flow/us-east-1/test-firewall/2020/10/01/19/11111111111_network-firewall_flow_us-east-1_test-firewall_202010011920_44442222.log.gz
```

Permissions to publish logs to Amazon S3

You must have the following permissions settings to configure your firewall to send logs to Amazon S3.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:CreateLogDelivery",
        "logs:GetLogDelivery",
        "logs:UpdateLogDelivery",
        "logs>DeleteLogDelivery",
        "logs:ListLogDeliveries"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow",
      "Sid": "FirewallLogging"
    },
    {
      "Sid": "FirewallLoggingS3",
      "Action": [
        "s3:PutBucketPolicy",
        "s3:GetBucketPolicy"
      ],
      "Resource": [
        Amazon S3 bucket ARN
      ],
      "Effect": "Allow"
    }
  ]
}
```

```
}

```

By default, Amazon S3 buckets and the objects that they contain are private. Only the bucket owner can access the bucket and the objects stored in it. The bucket owner, however, can grant access to other resources and users by writing an access policy.

If the user creating the log owns the bucket, the service automatically attaches the following policy to the bucket to give the log permission to publish logs to it:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AWSLogDeliveryWrite",
      "Effect": "Allow",
      "Principal": {"Service": "delivery.logs.amazonaws.com"},
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3::bucket-name/optional-folder/AWSLogs/account-id/
*
",
      "Condition": {"StringEquals": {"s3:x-amz-acl": "bucket-owner-full-
control"}}
    },
    {
      "Sid": "AWSLogDeliveryAclCheck",
      "Effect": "Allow",
      "Principal": {"Service": "delivery.logs.amazonaws.com"},
      "Action": "s3:GetBucketAcl",
      "Resource": "arn:aws:s3:::bucket-name"
    }
  ]
}
```

If the user creating the log doesn't own the bucket, or doesn't have the `GetBucketPolicy` and `PutBucketPolicy` permissions for the bucket, the log creation fails. In this case, the bucket owner must manually add the preceding policy to the bucket and specify the log creator's AWS account ID. For more information, see [How Do I Add an S3 Bucket Policy?](#) in the *Amazon Simple Storage Service User Guide*. If the bucket receives logs from multiple accounts, add a `Resource` element entry to the `AWSLogDeliveryWrite` policy statement for each account.

For example, the following bucket policy allows AWS accounts 111122223333 and 444455556666 to publish logs to a folder named `flow-logs` in a bucket named `amzn-s3-demo-bucket`:


```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AWSLogDeliveryWrite",
      "Effect": "Allow",
      "Principal": {"Service": "delivery.logs.amazonaws.com"},
      "Action": "s3:PutObject",
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket:/flow-logs/AWSLogs/111122223333/",
        "arn:aws:s3:::amzn-s3-demo-bucket:/flow-logs/AWSLogs/444455556666/"
      ],
      "Condition": {"StringEquals": {"s3:x-amz-acl": "bucket-owner-full-
control"}}
    },
    {
      "Sid": "AWSLogDeliveryAclCheck",
      "Effect": "Allow",
      "Principal": {"Service": "delivery.logs.amazonaws.com"},
      "Action": "s3:GetBucketAcl",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
    }
  ]
}
```

Amazon S3 log file access

In addition to the required bucket policies, Amazon S3 uses access control lists (ACLs) to manage access to the log files created by a Network Firewall log. By default, the bucket owner has `FULL_CONTROL` permissions on each log file. The log delivery owner, if different from the bucket owner, has no permissions. The log delivery account has `READ` and `WRITE` permissions. For more information, see [Access Control List \(ACL\) Overview](#) in the *Amazon Simple Storage Service User Guide*.

The log files are compressed. If you open the files using the Amazon S3 console, Amazon S3 decompresses the log records and displays them. If you download the log files, you must decompress them to view the records.

Sending AWS Network Firewall logs to Amazon CloudWatch Logs

To send logs to Amazon CloudWatch Logs, you create a CloudWatch Logs log group. When you enable logging in Network Firewall, you provide the log group name. After you enable logging for

your firewall, AWS Network Firewall delivers logs to the CloudWatch Logs log group in log streams. Each log stream contains an hour of log records.

You can use any name for your CloudWatch Logs log group. Configure the log group in the same Region as the firewall and using the same account as you use to manage the firewall.

For information about configuring a CloudWatch Logs log group, see [Working with Log Groups and Log Streams](#).

Names of alert and flow logs

When you configure your Network Firewall firewall to send alert and flow logs to the log group, the resulting log streams have the following naming format:

```
/aws/network-firewall/log-type/firewall-name_YYYY-MM-DD-HH
```

In the specification, the log type is either alert or flow.

The following shows an example log stream created on October 1, 2020, at 5 pm for alert logging for firewall test-firewall.

```
/aws/network-firewall/alert/test-firewall_2020-10-01-17
```

Names of TLS logs

When you configure your Network Firewall firewall to send TLS logs to the log group, the resulting log streams have the following naming format:

```
/aws/network-firewall/tls/firewall-name
```

The following shows the log stream for TLS logging for the example firewall test-firewall.

```
/aws/network-firewall/tls/test-firewall
```

Permissions to publish logs to CloudWatch Logs

You must have the following permissions settings to configure your firewall to send logs to a CloudWatch Logs log group.

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

```
"Statement": [
  {
    "Action": [
      "logs:CreateLogDelivery",
      "logs:GetLogDelivery",
      "logs:UpdateLogDelivery",
      "logs>DeleteLogDelivery",
      "logs:ListLogDeliveries"
    ],
    "Resource": [
      "*"
    ],
    "Effect": "Allow",
    "Sid": "FirewallLogging"
  },
  {
    "Sid": "FirewallLoggingCWL",
    "Action": [
      "logs:PutResourcePolicy",
      "logs:DescribeResourcePolicies",
      "logs:DescribeLogGroups"
    ],
    "Resource": [
      "CloudWatch Logs log group ARN"
    ],
    "Effect": "Allow"
  }
]
```

Sending AWS Network Firewall logs to Amazon Data Firehose

To send logs to Amazon Data Firehose, you first need to set up a Firehose delivery stream. As part of that process, you choose a destination for storing your logs. After you enable logging for your firewall, AWS Network Firewall delivers logs to the destination through the HTTPS endpoint of Amazon Data Firehose. One AWS Network Firewall log corresponds to one Amazon Data Firehose record.

Configure an Amazon Data Firehose delivery stream for your firewall as follows.

- Create it using the same account as you use to manage the firewall.
- Create it in the same Region as the firewall.

- Configure it for direct put, which allows applications to access the delivery stream directly. In the Amazon Data Firehose console, for the delivery stream **Source** setting, choose **Direct PUT or other sources**. Through the API, set the delivery stream property `DeliveryStreamType` to `DirectPut`.

For information about how to create an Amazon Data Firehose delivery stream and review the stored logs, see [Creating an Amazon Data Firehose delivery stream](#) and [What is Amazon Data Firehose?](#)

When you successfully enable logging to an Amazon Data Firehose data stream, Network Firewall creates a service linked role with the necessary permissions to write logs to it. For more information, see [Using service-linked roles](#).

Permissions to publish logs to Amazon Data Firehose

You must have the following permissions to configure your firewall to send logs to an Amazon Data Firehose delivery stream.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:CreateLogDelivery",
        "logs:GetLogDelivery",
        "logs:UpdateLogDelivery",
        "logs>DeleteLogDelivery",
        "logs:ListLogDeliveries"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow",
      "Sid": "FirewallLogging"
    },
    {
      "Sid": "FirewallLoggingFH",
      "Action": [
        "iam:CreateServiceLinkedRole"
      ],
      "Resource": "*",
    }
  ]
}
```

```
        "Effect": "Allow"
    },
    {
        "Sid": "FirewallLoggingFH",
        "Action": [
            "firehose:TagDeliveryStream"
        ],
        "Resource": "Amazon Data Firehose delivery stream ARN",
        "Effect": "Allow"
    }
]
}
```

Logging in AWS Network Firewall with server-side encryption and customer-provided keys

If your logging destination uses server-side encryption with keys that are stored in AWS Key Management Service (SSE-KMS) and you use a customer managed key (KMS key), you must give Network Firewall permission to use your KMS key. To do this, you add a key policy to the KMS key for your chosen destination to permit Network Firewall logging to write your log files to the destination.

Policy for an Amazon S3 bucket

Add the following key policy to your KMS key to allow Network Firewall to log to your Amazon S3 bucket.

```
{
  "Sid": "Allow Network Firewall to use the key",
  "Effect": "Allow",
  "Principal": {
    "Service": [
      "delivery.logs.amazonaws.com"
    ]
  },
  "Action": "kms:GenerateDataKey*",
  "Resource": "*"
}
```

Note

Network Firewall supports encryption with Amazon S3 buckets for key type Amazon S3 key (SSE-S3) and for AWS Key Management Service (SSE-KMS) AWS KMS keys. Network Firewall doesn't support encryption for AWS Key Management Service keys that are managed by AWS.

Policy for a CloudWatch Logs log group

For a CloudWatch Logs log group, the service principal requires access to the logs for the Region. This is the same as for all encrypted CloudWatch Logs log streams. For more information about log data encryption in CloudWatch Logs, see [Encrypt Log Data in CloudWatch Logs Using AWS KMS](#).

Add the following key policy to your KMS key to allow Network Firewall to log to your CloudWatch Logs log group.

```
{
  "Effect": "Allow",
  "Principal": {
    "Service": "logs.{region}.amazonaws.com"
  },
  "Action": [
    "kms:Encrypt*",
    "kms:Decrypt*",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:Describe*"
  ],
  "Resource": "*"
}
```

Policy for a Firehose delivery stream

For Firehose delivery streams, you allow the service principal to generate keys so that it can put the logging records.

Add the following key policy to your KMS key to allow Network Firewall to log to your Firehose delivery stream.

```
{
```

```
"Sid": "Allow Network Firewall logs to use the key",
"Effect": "Allow",
"Principal": {
  "Service": [
    "delivery.logs.amazonaws.com"
  ]
},
"Action": "kms:GenerateDataKey*",
"Resource": "*"
}
```

Updating a AWS Network Firewall logging configuration

To update your firewall's logging configuration through the Network Firewall AWS Management Console, use the procedure in this section. For the API, see the Network Firewall API action, `UpdateLoggingConfiguration`.

Note

Firewall logging is only available for traffic that you forward to the stateful rules engine. You forward traffic to the stateful engine through stateless rule actions and stateless default actions in the firewall policy. For information about these actions settings, see [Firewall policy settings in AWS Network Firewall](#) and [Defining rule actions in AWS Network Firewall](#).

To update a firewall's logging configuration through the console

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, under **Network Firewall**, choose **Firewalls**.
3. In the **Firewalls** page, choose the name of the firewall that you want to edit. This takes you to the firewall's details page.
4. Choose the tab **Firewall details**, then in the **Logging** section, choose **Edit**.
5. Adjust the **Log type** selections as needed. To disable logging for a firewall, deselect all options.
 - **Flow** – Sends logs for all network traffic that the stateless engine forwards to the stateful rules engine.

- **Alert** – Sends logs for traffic that matches any stateful rule whose action is set to `Alert`, `Drop`, or `Reject`. For more information about stateful rules and rule groups, see [Rule groups in AWS Network Firewall](#).
- **TLS** – Sends logs for events related to TLS inspection. Network Firewall currently logs failures in certificate revocation checks for outbound traffic and TLS errors.

These logs require the firewall to be configured for TLS inspection. For more information, see [Inspecting SSL/TLS traffic with TLS inspection configurations in AWS Network Firewall](#).

6. For each selected log type, choose the destination type, then provide the information for the logging destination that you prepared following the guidance in [Firewall logging destinations](#).

In order to change the destination for an existing **Log type**, you must first disable logging for the policy. Then, edit the policy and specify the new destination(s) for the **Log type**.

7. Choose **Save** to save your changes and return to the firewall's detail page.

Logging calls to the AWS Network Firewall API with AWS CloudTrail

AWS Network Firewall is integrated with AWS CloudTrail, a service that provides a record of API calls to Network Firewall by a user, role, or an AWS service. CloudTrail captures all API calls for Network Firewall as events. The calls captured include calls from the Network Firewall console and code calls to the Network Firewall API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Network Firewall. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine information including the request that was made to Network Firewall, the IP address from which the request was made, who made the request, and when the request was made.

To learn more about CloudTrail, see the [AWS CloudTrail User Guide](#).

AWS Network Firewall information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in Network Firewall, it's recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see [Viewing events with CloudTrail event history](#).

For an ongoing record of events in your AWS account, including events for Network Firewall, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- [Overview for creating a trail](#)
- [CloudTrail supported services and integrations](#)
- [Configuring Amazon SNS notifications for CloudTrail](#)
- [Receiving CloudTrail log files from multiple Regions](#) and [Receiving CloudTrail log files from multiple accounts](#)

All Network Firewall actions are logged by CloudTrail. These actions are documented in the [Actions](#) section of the [AWS Network Firewall API Reference](#). For example, calls to the actions `CreateFirewall`, `ListFirewalls`, and `DeleteFirewall` generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the [CloudTrail userIdentity element](#).

CloudTrail log file examples

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following are examples of CloudTrail log entries for Network Firewall operations.

Example: CloudTrail log entry for CreateFirewall

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLEPrincipalId",
    "arn": "arn:aws:sts::444455556666:assumed-role/Admin/EXAMPLE",
    "accountId": "444455556666",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "EXAMPLEPrincipalId",
        "arn": "arn:aws:iam::444455556666:role/Admin",
        "accountId": "444455556666",
        "userName": "Admin"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-08-13T03:07:52Z"
      }
    }
  },
  "eventTime": "2020-08-13T03:07:53Z",
  "eventSource": "network-firewall.amazonaws.com",
  "eventName": "CreateFirewall",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.4",
  "userAgent": "aws-cli/1.18.117 Python/3.6.10
Linux/4.9.217-0.1.ac.205.84.332.metal1.x86_64 boto3/1.17.40",
  "requestParameters": {
    "firewallName": "firewall01",
    "firewallPolicyArn": "arn:aws:network-firewall:us-west-2:444455556666:firewall-
policy/policy01",
    "vpcId": "vpc-11112222",
    "subnetMappings": [
      {
        "subnetId": "subnet-44443333",
        "requestedCapacity": "10"
      }
    ]
  },
  "deleteProtection": false
}
```

```

    },
    "responseElements": {
      "firewall": {
        "firewallName": "firewall01",
        "firewallArn": "arn:aws:network-firewall:us-west-2:444455556666:firewall/
firewall01",
        "firewallPolicyArn": "arn:aws:network-firewall:us-west-2:444455556666:firewall-
policy/policy01",
        "vpcId": "vpc-11112222",
        "subnetMappings": [
          {
            "subnetId": "subnet-44443333",
            "requestedCapacity": "10"
          }
        ],
        "deleteProtection": false
      },
      "firewallStatus": {
        "status": "PROVISIONING",
        "configurationSyncStateSummary": "PENDING"
      }
    },
    "requestID": "43a8cad0-68b6-45d2-b6f3-28cf0e195d47",
    "eventID": "7d575a14-ec3f-43c8-8735-eaadd21fd9d1",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "recipientAccountId": "444455556666"
  }
}

```

Example: CloudTrail log entry for ListFirewalls

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLEPrincipalId",
    "arn": "arn:aws:sts::444455556666:assumed-role/Admin/EXAMPLE",
    "accountId": "444455556666",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",

```

```

    "principalId": "EXAMPLEPrincipalId",
    "arn": "arn:aws:iam::444455556666:role/Admin",
    "accountId": "444455556666",
    "userName": "Admin"
  },
  "webIdFederationData": {},
  "attributes": {
    "mfaAuthenticated": "false",
    "creationDate": "2020-08-13T03:07:55Z"
  }
}
},
"eventTime": "2020-08-13T03:07:55Z",
"eventSource": "network-firewall.amazonaws.com",
"eventName": "ListFirewalls",
"awsRegion": "us-west-2",
"sourceIPAddress": "203.0.113.4",
"userAgent": "aws-cli/1.18.117 Python/3.6.10
Linux/4.9.217-0.1.ac.205.84.332.metal1.x86_64 botocore/1.17.40",
"requestParameters": {
  "maxResults": 10
},
"responseElements": null,
"requestID": "1ac1567a-fa84-49ac-b5aa-6016052ad646",
"eventID": "79b95fd6-a288-49b1-a907-b61ed99b94c0",
"readOnly": true,
"eventType": "AwsApiCall",
"recipientAccountId": "444455556666"
}

```

Example: CloudTrail log entry for DeleteFirewall

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLEPrincipalId",
    "arn": "arn:aws:sts::444455556666:assumed-role/Admin/EXAMPLE",
    "accountId": "444455556666",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",

```

```

        "principalId": "EXAMPLEPrincipalId",
        "arn": "arn:aws:iam::444455556666:role/Admin",
        "accountId": "444455556666",
        "userName": "Admin"
    },
    "webIdFederationData": {},
    "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-08-19T16:09:29Z"
    }
}
},
"eventTime": "2020-08-19T16:18:43Z",
"eventSource": "network-firewall.amazonaws.com",
"eventName": "DeleteFirewall",
"awsRegion": "us-west-2",
"sourceIPAddress": "198.51.100.190",
"userAgent": "Apache-HttpClient/UNAVAILABLE (Java/1.8.0_232)",
"requestParameters": {
    "firewallArn": "arn:aws:network-firewall:us-west-2:444455556666:firewall/
DeleteMeFast"
},
"responseElements": {
    "firewall": {
        "firewallName": "DeleteMeFast",
        "firewallArn": "arn:aws:network-firewall:us-west-2:444455556666:firewall/
DeleteMeFast",
        "firewallPolicyArn": "arn:aws:network-firewall:us-
west-2:444455556666:firewall-policy/123",
        "vpcId": "vpc-11112222",
        "subnetMappings": [
            {
                "subnetId": "subnet-99990000",
                "requestedCapacity": "14"
            },
            {
                "subnetId": "subnet-77776666",
                "requestedCapacity": "12"
            }
        ],
        "deleteProtection": true,
        "description": "HIDDEN_DUE_TO_SECURITY_REASONS"
    },
    "firewallStatus": {

```

```
    "status": "DELETING",
    "configurationSyncStateSummary": "PENDING",
    "syncStates": {
      "us-west-2c": {
        "attachment": {
          "subnetId": "subnet-99990000",
          "networkInterfaceId": "eni-01e59ab6f6064c453",
          "status": "SCALING"
        },
        "config": {}
      },
      "us-west-2d": {
        "attachment": {
          "subnetId": "subnet-77776666",
          "networkInterfaceId": "eni-04c3ac8c04076ed36",
          "status": "SCALING"
        },
        "config": {}
      }
    }
  },
  "requestID": "299b886e-23da-4c77-8beb-0853a0a08bcf",
  "eventID": "142b089a-8aca-4183-8326-5ff32a38876e",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "recipientAccountId": "444455556666"
}
```

AWS Network Firewall metrics in Amazon CloudWatch

You can monitor AWS Network Firewall using CloudWatch, which collects raw data and processes it into readable, near real-time metrics. CloudWatch stores your metrics for 15 months, so that you can access historical information for added perspective on how your web application or service is performing. You can also set alarms that watch for certain thresholds, and send notifications or take actions when those thresholds are met. For more information, see the [Amazon CloudWatch User Guide](#).

Use the following procedures to view the metrics for Network Firewall.

To view metrics using the CloudWatch console

Metrics are grouped first by the service namespace, and then by the various dimension combinations within each namespace. The CloudWatch namespace for Network Firewall is `AWS/NetworkFirewall`.

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Metrics**.
3. On the **All metrics** tab, choose the Region and then choose `AWS/NetworkFirewall`.

To view metrics using the AWS CLI

- For Network Firewall, at a command prompt use the following command:

```
aws cloudwatch list-metrics --namespace "AWS/NetworkFirewall"
```

AWS Network Firewall metrics

The `AWS/NetworkFirewall` namespace includes the following metrics.

Metric	Description
<code>DroppedPackets</code>	Number of packets dropped due to rule actions. Reporting criteria: There is a nonzero value. Valid statistics: Sum
<code>InvalidDroppedPackets</code>	Number of packets dropped for failing packet validation due to issues with the packet. Reporting criteria: There is a nonzero value. Valid statistics: Sum
<code>OtherDroppedPackets</code>	Number of packets dropped due to reasons other than those described by <code>InvalidDroppedPack</code>

Metric	Description
	<p>ets or DroppedPackets , including packets that are throttled due to bursting traffic.</p> <p>Reporting criteria: There is a nonzero value.</p> <p>Valid statistics: Sum</p>
Packets	<p>Number of packets inspected for a firewall policy or stateless rulegroup for which a custom action is defined. This metric is only used for the dimension CustomAction .</p> <p>Reporting criteria: There is a nonzero value.</p> <p>Valid statistics: Sum</p>
PassedPackets	<p>Number of packets that the Network Firewall firewall allowed through to their destinations.</p> <p>Reporting criteria: There is a nonzero value.</p> <p>Valid statistics: Sum</p>
ReceivedPackets	<p>Number of packets received by the Network Firewall firewall.</p> <p>Reporting criteria: There is a nonzero value.</p> <p>Valid statistics: Sum</p>
RejectedPackets	<p>The number of packets rejected due to Reject stateful rule actions. For information about stateful actions, see Stateful actions.</p> <p>Reporting criteria: There is a nonzero value.</p> <p>Valid statistics: Sum</p>

Metric	Description
StreamExceptionPolicyPackets	<p>The number of packets matching the firewall policy's stream exception policy. You can configure stream exception policy settings while creating a firewall policy in the console, or by the StatefulEngineOptions structure when using the API. For more information about stream exception policy settings, see the Stream exception policy option in the Creating a firewall policy in AWS Network Firewall procedure.</p> <p>Reporting criteria: There is a nonzero value.</p> <p>Valid statistics: Sum</p>
TLSDroppedPackets	<p>Number of packets dropped by Network Firewall while inspecting SSL/TLS packets. The value of this metric might differ between stateless and stateful rule processing due to the TCP and TLS connection termination that occurs prior to stateful packet inspection.</p> <p>Reporting criteria: There is a nonzero value.</p> <p>Valid statistics: Sum</p>
TLSErrors	<p>Number of errors observed by Network Firewall while inspecting SSL/TLS packets.</p> <p>Reporting criteria: There is a nonzero value.</p> <p>Valid statistics: Sum</p>

Metric	Description
TLSPassedPackets	<p>Number of packets passed by Network Firewall while inspecting SSL/TLS packets. The value of this metric might differ between stateless and stateful rule processing due to the TCP and TLS connection termination that occurs prior to stateful packet inspection.</p> <p>Reporting criteria: There is a nonzero value.</p> <p>Valid statistics: Sum</p>
TLSReceivedPackets	<p>Number of SSL/TLS packets received by the Network Firewall firewall. The value of this metric might differ between stateless and stateful rule processing due to the TCP and TLS connection termination that occurs prior to stateful packet inspection.</p> <p>Reporting criteria: There is a nonzero value.</p> <p>Valid statistics: Sum</p>
TLSRejectedPackets	<p>Number of packets rejected by Network Firewall while inspecting SSL/TLS packets. The value of this metric might differ between stateless and stateful rule processing due to the TCP and TLS connection termination that occurs prior to stateful packet inspection.</p> <p>Reporting criteria: There is a nonzero value.</p> <p>Valid statistics: Sum</p>

Metric	Description
<p>TLSRevocationStatusOKConnections</p>	<p>The number of SSL/TLS connections to TLS servers whose certificates have been confirmed as not revoked.</p> <p>Reporting criteria: There is a nonzero value.</p> <p>Valid statistics: Sum</p>
<p>TLSRevocationStatusRevokedConnections</p>	<p>The number of SSL/TLS connections to TLS servers whose certificates have been confirmed as revoked.</p> <p>Reporting criteria: There is a nonzero value.</p> <p>Valid statistics: Sum</p>
<p>TLSRevocationStatusUnknownConnections</p>	<p>The number of SSL/TLS connections to TLS servers whose certificates revocation status is unknown or could not be determined by the firewall. This can occur when the OCSP responder for a server certificate returns an unknown status, or when the firewall is unable to connect to the CRL or OCSP endpoints provided in the certificate.</p> <p>Reporting criteria: There is a nonzero value.</p> <p>Valid statistics: Sum</p>
<p>TLSTimedOutConnections</p>	<p>Number of SSL/TLS connections that timed out during SSL/TLS inspection by Network Firewall.</p> <p>Reporting criteria: There is a nonzero value.</p> <p>Valid statistics: Sum</p>

AWS Network Firewall dimensions

Network Firewall can use the following dimension combinations to categorize your metrics:

Dimension	Description
AvailabilityZone	Availability Zone in the Region where the Network Firewall firewall is active.
CustomAction	Dimension for a publish metrics custom action that you defined. You can define this for a rule action in a stateless rule group or for a stateless default action in a firewall policy.
Engine	Rules engine that processed the packet. The value for this is either <code>Stateful</code> or <code>Stateless</code> .
FirewallName	Name that you specified for the Network Firewall firewall.

Troubleshooting AWS Network Firewall

The topics in this chapter can help you troubleshoot problems with configuring and using AWS Network Firewall.

Contents

- [Troubleshooting general issues in AWS Network Firewall](#)
 - [Firewall endpoint creation or deletion fails](#)
 - [Availability Zone is unsupported](#)
 - [How do I check if I have asymmetric routing?](#)
 - [I'm using Network Firewall with AWS Transit Gateway and Network Firewall is dropping traffic](#)
 - [High latency and intermittent packet drops when traffic passes through Network Firewall](#)
- [Troubleshooting logging in AWS Network Firewall](#)
 - [My firewall isn't logging all traffic that matches pass action rules](#)
 - [I don't see Alert logs that match drop action rules in my rule group](#)
 - [I don't see any TLS logs even though I have TLS logging enabled](#)
- [Troubleshooting rules in AWS Network Firewall](#)
 - [Rules with the HOME_NET variable are not working as expected with managed rule groups](#)
 - [I created a rule to allow only outbound traffic from HOME_NET to EXTERNAL_NET, but EXTERNAL_NET was also able to initiate a connection back to HOME_NET. How do I prevent this from happening?](#)
 - [I'm using strict ordering, but stateful rules near the bottom of my ruleset appear to be handling traffic before rules near the top of my ruleset](#)
 - [I've configured a drop action rule but traffic still goes through the firewall](#)
- [Troubleshooting TLS inspection in AWS Network Firewall](#)
 - [Outbound TLS - Blocked connections to servers with revoked certificates](#)
 - [Outbound TLS - Passing traffic for specific target server with revoked certificates by adjusting scope](#)
 - [Troubleshooting connection issues with AWS service endpoints \(including the AWS Systems Manager agent\)](#)
 - [Troubleshooting TLS - Connections dropping or resetting](#)

Troubleshooting general issues in AWS Network Firewall

Use the information here to help you diagnose and fix common issues when you work with Network Firewall.

Firewall endpoint creation or deletion fails

If Network Firewall can't create or delete a firewall endpoint in the subnet because of an error, the service displays a status message describing how to resolve the issues. You can view the status message in the console or API. For more information about troubleshooting firewall creation and deletion issues, see [Troubleshooting firewall endpoint failures](#).

Availability Zone is unsupported

When you try to create a firewall, you get the following error “Availability zone subnet(s) cannot be updated or added because this Availability Zone is not supported by Network Firewall”.

You might be trying to create the Network Firewall in a constrained Availability Zone—a zone in which our ability to expand is constrained. We cannot support Network Firewall in these Availability Zones. You can move your resources to an unconstrained Availability Zone so that your resources and your Network Firewall are in the same zone.

How do I check if I have asymmetric routing?

AWS Network Firewall doesn't support asymmetric routing. Request and response traffic must be routed to the same firewall endpoint. The recommended best practice is to route traffic to the firewall endpoint closest to the client in both directions. Failure to route traffic symmetrically to the same firewall endpoint prevents stateful features from working correctly, such as application layer inspection. You can use the Amazon VPC [Reachability Analyzer](#), or in Network Firewall, you can use either the [stateless rule group analyzer](#) or flow and alert logging as outlined in the following section.

To test whether routing is symmetric using Network Firewall flow and alert logs:

1. Create a firewall using the [Creating a firewall](#) procedure, and associate an empty **Strict order** rule evaluation order policy to it. The stateless default action should be **Forward to stateful rule groups** for both full and fragmented packets. For this test, you don't need to enable any stateful default actions.

- From the console, select the firewall you created in the previous step. On the **Firewall details** tab, navigate to the **Logging** tab and select **Edit**. Choose **Alert** and **Flow**, and select a separate Amazon CloudWatch log group for each log type. For example, choose *my-firewall-flow-logs* and *my-firewall-alert-logs*. If the log groups don't exist, choose **Create log group** to be redirected to the CloudWatch log groups console. Once created, you can refresh the log group list in the firewall logging console. After you select the log groups, select **Save**.
- Create a Suricata-compatible rule group using the [Creating a stateful rule group](#), and include the following Suricata rule:

```
alert tcp any any -> any any (msg:"Routing is symmetric. You can safely remove this test rule."; flow:established; sid:123456;)
```

- From an instance behind the firewall, attempt to connect to `https://www.amazon.com` using cURL or a web browser. The test should succeed based on the firewall policy and rule configuration you created in the previous steps. If the request and response traffic routes symmetrically through the same firewall endpoint, the alert rule should match the established TCP flow and an alert log event should be published to the CloudWatch log group. Be aware that there might be a delay to publish the alert log event to the log destination.
- From the CloudWatch log groups console, search the firewall alert log group for `www.amazon.com`.
- An alert log event similar to the following means the stateful engine saw the TCP 3-way handshake bidirectionally (`flow:established`) and was able to reassemble the flow as application layer protocol TLS. This is a positive indication that routing is configured symmetrically.

```
{
  "firewall_name": "protected-vpc-fw",
  "availability_zone": "us-east-1a",
  "event_timestamp": "1122334455",
  "event": {
    "app_proto": "tls",
    "src_ip": "18.154.234.51",
    "src_port": 443,
    "event_type": "alert",
    "alert": {
      "severity": 3,
      "signature_id": 123456,
      "rev": 0,

```

```

        "signature": "Routing is symmetric. You can safely remove this test
rule.",
        "action": "allowed",
        "category": ""
    },
    "flow_id": 49218262306386,
    "dest_ip": "10.170.19.217",
    "proto": "TCP",
    "tls": {
        "subject": "CN=www.amazon.com",
        "issuerdn": "C=US, O=DigiCert Inc, CN=DigiCert Global CA G2",
        "serial": "05:21:67:4F:03:57:5F:5B:A5:BD:6B:2B:CC:A0:EB:4B",
        "fingerprint":
"91:78:b8:6a:9b:40:ad:af:ad:6b:25:ad:a3:d2:39:e5:39:af:86:9a",
        "sni": "www.amazon.com",
        "version": "TLS 1.2",
        "notbefore": "2023-01-17T00:00:00",
        "notafter": "2024-01-16T23:59:59",
        "ja3": {},
        "ja3s": {}
    },
    "dest_port": 60460,
    "timestamp": "2023-07-27T14:56:49.250792+0000"
}
}

```

7. (Optional) To further confirm that routing is configured symmetrically, you can copy the `flow_id` value from the alert log event (for example, `77665544332211`). From the CloudWatch log groups console, search the firewall flow log group for the `flow_id` value from the alert log event. For flow `77665544332211` in this example, separate firewall flow log events were published for the request and response traffic. This is a positive indication that routing is configured symmetrically. If you only see a flow log event for the request side of the conversation, the routing in the response path may be directing traffic around the firewall endpoint. It could also be an indication that routing is configured symmetrically but a Network ACL or security group are blocking the inbound traffic on the server.

Request flow (client > server):

```

{
    "firewall_name": "protected-vpc-fw",
    "availability_zone": "us-east-1a",
    "event_timestamp": "3344556677",

```



```
"event": {
  "tcp": {
    "tcp_flags": "1b",
    "syn": true,
    "fin": true,
    "psh": true,
    "ack": true
  },
  "app_proto": "tls",
  "src_ip": "10.170.19.217",
  "src_port": 60460,
  "netflow": {
    "pkts": 13,
    "bytes": 1527,
    "start": "2023-07-27T14:54:02.888402+0000",
    "end": "2023-07-27T14:54:02.923400+0000",
    "age": 0,
    "min_ttl": 254,
    "max_ttl": 254
  },
  "event_type": "netflow",
  "flow_id": 77665544332211,
  "dest_ip": "18.154.234.51",
  "proto": "TCP",
  "dest_port": 443,
  "timestamp": "2023-07-27T14:56:49.250816+0000"
}
}
```

Response flow (server > client):

```
{
  "firewall_name": "protected-vpc-fw",
  "availability_zone": "us-east-1a",
  "event_timestamp": "2233445566",
  "event": {
    "tcp": {
      "tcp_flags": "1b",
      "syn": true,
      "fin": true,
      "psh": true,
      "ack": true
    },
    "app_proto": "tls",
```

```
"src_ip": "18.154.234.51",
"src_port": 443,
"netflow": {
  "pkts": 16,
  "bytes": 9125,
  "start": "2023-07-27T14:54:02.888402+0000",
  "end": "2023-07-27T14:54:02.923400+0000",
  "age": 0,
  "min_ttl": 246,
  "max_ttl": 246
},
"event_type": "netflow",
"flow_id": 49218262306386,
"dest_ip": "10.170.19.217",
"proto": "TCP",
"dest_port": 60460,
"timestamp": "2023-07-27T14:56:49.250823+0000"
}
```

For more information about asymmetric routing, see [Avoiding asymmetric routing with AWS Network Firewall](#).

I'm using Network Firewall with AWS Transit Gateway and Network Firewall is dropping traffic

Network Firewall is a managed service that uses a Gateway Load Balancer to distribute traffic flows across backend firewall appliances for high availability and scale. Network Firewall requires symmetric routing, and needs to see the forward and return traffic flow to track flow state and apply stateful rules. Many customers use a centralized architecture to inspect VPC-to-VPC or VPC-to-on-premise traffic. In a centralized architecture, Network Firewall is deployed in an inspection VPC and customers use a AWS Transit Gateway transit gateway to route the traffic through inspection VPC.

If you have a centralized architecture, make sure that appliance mode is enabled on the transit gateway. When you enable the appliance mode, a transit gateway selects a single network interface in the appliance VPC, using a flow hash algorithm, to send traffic to same backend firewall appliance for the life of the flow. The transit gateway uses the same network interface for the return traffic. This ensures that bi-directional traffic is routed symmetrically; in other words, it's routed through the same Availability Zone in the VPC attachment for the life of the flow. For

information about appliance mode in Transit Gateway, see [Example: Appliance in a shared services VPC](#) in the *Amazon VPC Transit Gateways*.

High latency and intermittent packet drops when traffic passes through Network Firewall

Network latency and packet drops can occur for multiple reasons such as a sudden burst in traffic, asymmetric routing, applications with longer TCP timeouts, or issues occurring outside of the Network Firewall service. Try these steps to identify the root cause:

1. **Monitor CloudWatch metrics** – You can monitor [CloudWatch metrics for AWS PrivateLink](#) such as `ActiveConnections`, `BytesProcessed`, and `PacketsDropped` for the Gateway Load Balancer endpoint interfaces in your VPC. Create a ticket with AWS Support if the `PacketsDropped` metric is increasing. To understand what traffic is traversing a Gateway Load Balancer endpoint, you can enable [custom format](#) Amazon VPC flow logs for each interface. For traffic routed to Network Firewall, you can review stateless and stateful [Network Firewall metrics](#) such as `ReceivedPackets` to understand the volume of packets when high latency is observed.
2. **Asymmetrically-forward connections** – Check if you have asymmetrically-forward connections on your network. For more information about how to check asymmetric routing, see [How do I check if I have asymmetric routing?](#). When forwarding traffic from the stateless engine to stateful, you must forward the request and response traffic bi-directionally. Be aware that unidirectional pass rules can create asymmetric forwarding when the policy's stateless default action is **Forward to stateful rule groups**.

Make sure that your stateless rules forward traffic symmetrically to the stateful engine using the **forward to stateful rule groups** action. Often this means writing pairs of rules to match both forward and return direction traffic. Asymmetrically forwarded packets are subject to the [stream exception policy](#) action, which by default is configured to **drop**. You can check the `StreamExceptionPolicyPackets` metric in CloudWatch, which counts the number of times that stream exception policy is invoked. If you have asymmetrically-forwarded connections, this metric count will be high during the time you noticed high latency or packet drops. You can change the stream exception policy configuration in the firewall policy to determine how Network Firewall handles asymmetrically-forwarded connections. The following stream exception policy actions correlate to the specified CloudWatch metrics:

- **DROP** – Matching packets are dropped and counted in both the `StreamExceptionPolicyPackets` metric as well as the `DroppedPackets` metric.
- **REJECT** – Matching packets receive a TCP reset response and are counted in both the `StreamExceptionPolicyPackets` metric as well as the `RejectedPackets` metric. Any further packets from these rejected connections are dropped and reflected in the `DroppedPackets` metric.
- **CONTINUE** Matching packets are counted in the `StreamExceptionPolicyPackets` metric but continue to be processed by rules. Therefore if you use the `CONTINUE` value in the stream exception policy, elevated values of this metric aren't necessarily a cause for concern. Please note that Network Firewall will continue to inspect these connections using your firewall rules and a connection can be dropped if it matches with a drop action rule.

For information about using CloudWatch metrics for Network Firewall, see [Metrics in CloudWatch](#).

3. **Long TCP idle timeouts** – Check if you have applications with long TCP timeouts. Network Firewall uses a Gateway Load Balancer (GWLb) endpoint to distribute flows to backend firewall instances. By default, Gateway Load Balancer has a [fixed idle timeout](#) of 350 seconds for TCP flows. Once the idle timeout is reached or a TCP connection is closed for a flow, it is removed from the firewall's connection state table. This can result in the flow timing out on the client side. Subsequent non-SYN TCP packets for a removed flow may be dropped by Gateway Load Balancer. New TCP connection requests using the same 5-tuple (source/destination IP, source/destination port and protocol) might be routed to a different backend firewall instance than before. In this case, since the Gateway Load Balancer timeout is lower than the timeout value on your application, Gateway Load Balancer removes the flow without the application being aware it was dropped.

To prevent this from happening, we recommend either configuring the **TCP keep-alive** setting on your client and server's application or operating system or configuring the idle timeout period in the Network Firewall console. If you choose to update the TCP keep-alive setting, set the value to a longer period than the default 350 seconds. This ensures that the client and server keep the flow alive if there's inactivity, or the flow is removed from Network Firewall. If you choose to update the idle timeout value in the Network Firewall console, set the value to a longer period than the default 350 seconds. This ensures that your firewall can go without detecting traffic for longer than 350 seconds without dropping traffic. For more information about Gateway Load Balancer, see [Gateway Load Balancers](#) in the *Elastic Load Balancing User*

Guide. For more information about the idle timeout setting in Network Firewall, see [Firewall policy settings](#).

4. **Amazon Virtual Private Cloud NAT gateway port allocation errors** – Sometimes you'll see increased latency or packet drops on your traffic if your Amazon VPC NAT gateway is running out of ports to establish new connections which can result in packet drops and high latency. You can look at the NAT gateway `ErrorPortAllocation` errors to ensure NAT gateway is able to support new connections. If you notice NAT gateway `ErrorPortAllocation` errors, create a support case and the AWS Support team can help you resolve it. To troubleshoot your NAT gateway, you can look at `IdleTimeoutCount`, `PacketsDropCount` and other metrics as described in the [Monitor NAT gateways with CloudWatch](#) topic in the *Amazon VPC Developer Guide*.

Troubleshooting logging in AWS Network Firewall

Use the information here to help you diagnose common issues that you might encounter when working with logging in Network Firewall.

My firewall isn't logging all traffic that matches pass action rules

Network Firewall only logs traffic that match drop, reject, and alert action rules. Rules with a pass action silently allow the traffic in a way that any proceeding rules will not alert on that traffic. If you want to log all traffic for troubleshooting purposes, use the [strict evaluation order](#), and add an alert action rule before the pass action rule. For example, to log allowed traffic to `https://checkip.amazonaws.com`, add an alert action rule before the pass action rule:

```
alert tls $HOME_NET any -> $EXTERNAL_NET any (ssl_state:client_hello; tls.sni;
content:"checkip.amazonaws.com"; startswith; nocase; endswith; msg:"alert on TLS
allowlisted FQDNs"; flow:to_server; sid:111111; rev:1;)
```

```
pass tls $HOME_NET any -> $EXTERNAL_NET any (ssl_state:client_hello; tls.sni;
content:"checkip.amazonaws.com"; startswith; nocase; endswith; msg:"matching TLS
allowlisted FQDNs"; flow:to_server; sid:222222; rev:1;)
```

For more information about rule actions, see [Defining rule actions](#).

I don't see Alert logs that match drop action rules in my rule group

A stateful rule sends alerts for the rule actions drop, alert, and reject. Alert logs are generated for the traffic that match your stateful rules and have an action that sends an alert. Network Firewall can send alert logs to Amazon S3, CloudWatch, and Firehose. Make sure you have enabled logging by [updating the firewall's logging configuration](#). If you are using [strict evaluation order](#) with a default action set to **Drop established**, make sure you have also enabled an alert action so that the firewall will generate alert messages for this traffic. You can select **Alert all** to log an alert message on all packets, or select **Alert established** to log an alert message only on packets that are in established connections. Flows dropped by the **Drop established** or **Drop all** stateful default actions won't generate alert logs if you don't configure a stateful default alert action.

I don't see any TLS logs even though I have TLS logging enabled

TLS logging reports only TLS errors and outbound traffic that fails certificate revocation checks.

Double check your firewall policy configuration. Your firewall policy must be configured for [TLS inspection](#). Additionally, for revocation check logging, your firewall must be configured with [outbound SSL/TLS inspection](#) enabled and [certificate revocation status checking](#) enabled.

If your firewall policy is correctly configured, then it's possible that your traffic isn't generating anything for the logs.

- For TLS errors, currently the logs report SNI mismatches and naming errors. If your traffic doesn't have these errors, the logs might not report anything in this category.
- For revocation checks:
 - TLS inspection performs checks only on outbound traffic, and only for servers that you don't own or control.
 - TLS inspection only creates a log record when the check returns a status of REVOKED or UNKNOWN.

If you don't have traffic of this type, or if the certificates for this type of traffic are all passing the revocation checks, Network Firewall won't log anything into your TLS logs for revocation checks.

Troubleshooting rules in AWS Network Firewall

Use the information here to help you diagnose common issues that you might encounter when working with rules in Network Firewall.

Rules with the HOME_NET variable are not working as expected with managed rule groups

If your firewall is deployed in a centralized inspection VPC the HOME_NET variable uses the Classless Inter-Domain Routing (CIDR) ranges of VPC where the firewall is deployed. You can override the HOME_NET in your firewall policy to add the CIDR ranges of your application (spoke) VPCs that you want to filter. Once you define the HOME_NET variable, Network Firewall maintains that the value of EXTERNAL_NET in the rule group is the negation of the HOME_NET value that you define at the firewall policy level. Policy variables affect all rule groups that don't already define HOME_NET. For more information, see [Firewall policy settings in AWS Network Firewall](#).

I created a rule to allow only outbound traffic from HOME_NET to EXTERNAL_NET, but EXTERNAL_NET was also able to initiate a connection back to HOME_NET. How do I prevent this from happening?

You have to ensure that your firewall rules are not wide open to allow unintended traffic. The following example ensures that only trusted traffic is allowed to pass through the firewall in Network Firewall.

Example 1 - Do not use

In this example, Network Firewall allows TCP traffic from HOME_NET to EXTERNAL_NET. However, this rule also allows inbound traffic from EXTERNAL_NET to HOME_NET because <> means "any direction."

```
pass tcp $HOME_NET any <> $EXTERNAL_NET any. (msg:"sid1"; sid:1;)
```

Example 2 - Do not use

In this example traffic from HOME_NET is allowed to initiate TCP connections to EXTERNAL_NET, but since the rule is missing the flow:to_server keyword, there are scenarios where this rule will also allow EXTERNAL_NET to initiate connections to HOME_NET.

```
pass tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"sid1"; sid:1;)
```

You should write rules such that only trusted traffic is allowed through the firewall, and only in the direction you intend.

Example 3

If combined with the Network Firewall stateful default action **Drop established**, this rule ensures that traffic initiated by clients in HOME_NET to servers in EXTERNAL_NET is allowed, but clients in EXTERNAL_NET cannot initiate a connection to servers in HOME_NET.

```
pass tcp $HOME_NET any -> $EXTERNAL_NET 22 (msg:"sid1"; flow:to_server; sid:1;)
```

I'm using strict ordering, but stateful rules near the bottom of my ruleset appear to be handling traffic before rules near the top of my ruleset

When network connections traverse Network Firewall's stateful rule engine, it will see the lower level protocol being used (for example TCP) before it sees the application layer protocol (for example HTTP). This means that it's possible to have a blanket TCP (layer 4) rule that comes later in the ruleset take action on traffic that an earlier HTTP (layer 7) rule in the ruleset was intended to handle. Example 1 below may accidentally allow unwanted traffic because rule 2 (TCP) will take action on the port 80 traffic before rule 1 (HTTP), since the TCP part of the connection will be seen by the firewall first. Example 2 uses the `flow:to_server;` keyword to ensure that only the intended traffic is allowed and "baddomain.com (<http://baddomain.com/>)" continues to be blocked.

Example 1 - Do not use

```
# Rule 1 is intended to block http traffic to baddomain.com
reject http $HOME_NET any -> $EXTERNAL_NET 80 (http.host; content:"baddomain.com";
startswith; endswith; sid:1;)
# Rule 2 accidentally allows the TCP port 80 traffic even before any application protocol
is inspected
pass tcp $HOME_NET any -> $EXTERNAL_NET 80 (sid:2;)
```

Example 2

```
# Rule 1 blocks http traffic to baddomain.com
reject http $HOME_NET any -> $EXTERNAL_NET 80 (http.host; content:"baddomain.com";
startswith; endswith; sid:1;)
```



```
# Rule 2 waits until later in the connection (when the application protocol can also be
  detected) before allowing it
pass tcp $HOME_NET any -> $EXTERNAL_NET 80 (flow:to_server; sid:2;)
```

I've configured a drop action rule but traffic still goes through the firewall

The AWS Network Firewall stateful rule engine supports action order, previous known as default order, and strict rule evaluation order. Action rule order dictates that Network Firewall evaluates firewall rules based on a rule action. Network Firewall evaluates rules with a pass first, followed by drop, reject, and alert action rules. The stateful default action for a policy configured to use action order is pass. It's likely that the traffic matched one of the pass rules in your rule group and therefore it was not dropped by Network Firewall. We recommend that you use strict order so that Network Firewall will evaluate the rules in the order in which they are defined. When configuring a policy that uses strict order, configure the stateful default actions as **Drop established** and **Alert established**. For more information about evaluation order, see [Managing evaluation order for Suricata compatible rules in AWS Network Firewall](#).

Troubleshooting TLS inspection in AWS Network Firewall

Use the information here to help you diagnose common issues that you might encounter when working with TLS inspection in Network Firewall.

Outbound TLS - Blocked connections to servers with revoked certificates

If you've enabled the certificate revocation status check for your outbound TLS workflow, and your client starts getting blocked connections to a specific target server, use one of the following procedures to determine whether the firewall has blocked the connection based on the certificate revocation check action. For more information about actions, see [Checking certificate revocation status](#).

The check differs depending on whether you have TLS logging enabled.

Troubleshooting with TLS logging enabled

When you have TLS logging enabled, you can read the TLS logs to determine which servers are experiencing failed revocation checks.

1. Look in your TLS logs for records reporting the action, the certificate revocation check status, and the Server Name Indication (SNI) of the server whose certificate failed the revocation check.
2. Use the SNI from the logs to determine which target servers are experiencing the failed revocation checks.

Example output for a certificate that's been revoked by the server:

```
{
  "firewall_name": "egress-fw",
  "availability_zone": "us-east-1d",
  "event_timestamp": 1708361189,
  "event": {
    "src_ip": "10.0.2.53",
    "src_port": "55930",
    "revocation_check": {
      "leaf_cert_fpr": "1234567890EXAMPLE0987654321",
      "status": "REVOKED",
      "action": "DROP"
    },
    "dest_ip": "54.92.160.72",
    "dest_port": "443",
    "timestamp": "2024-02-19T16:46:29.441824Z",
    "sni": "revoked-rsa-dv.ssl.com"
  }
}
```

Troubleshooting without TLS logging

If you don't have TLS logs to refer to, follow these steps to determine which servers are experiencing failed revocation checks.

1. Run a `curl` command from the client to the target server that has the blocked connection. Note the output of the commands.

Example output for **Drop** action:

```
[root@ip-10-0-2-230 ~]# curl -ikv https://revoked-rsa-dv.ssl.com
* Trying 54.92.160.72:443...
* Connected to revoked-rsa-dv.ssl.com (54.92.160.72) port 443
* ALPN: curl offers h2,http/1.1
```

```
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
* TLSv1.3 (IN), TLS handshake, Server hello (2):
...
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* old SSL session ID is stale, removing
* TLSv1.3 (IN), TLS alert, close notify (256):
* Empty reply from server
* Closing connection
* TLSv1.3 (OUT), TLS alert, close notify (256):
curl: (52) Empty reply from server
```

Example output for **Reject** action:

```
[root@ip-10-0-2-230 ~]# curl -ikv https://revoked-rsa-dv.ssl.com
* Trying 54.92.160.72:443...
* Connected to revoked-rsa-dv.ssl.com (54.92.160.72) port 443
* ALPN: curl offers h2,http/1.1
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
* TLSv1.3 (IN), TLS handshake, Server hello (2):
...
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* old SSL session ID is stale, removing
* Recv failure: Connection reset by peer
* OpenSSL SSL_read: Connection reset by peer, errno 104
* Closing connection
* Send failure: Broken pipe
curl: (56) Recv failure: Connection reset by peer
```

2. Switch the action from **Drop** to **Reject**, or vice versa depending on the action you originally configured.
3. Run the `curl` command again on the client, and note the output.
4. Compare the outputs before and after you changed the configured action. If the `curl` output with a **Reject** action shows `connection reset by server` and the output with **Drop** action doesn't, it implies that the firewall has determined the certificate for that TLS server has a revoked status, and has blocked the connection based on the configured action.

Outbound TLS - Passing traffic for specific target server with revoked certificates by adjusting scope

You can pass traffic to hosts with revoked certificate by updating the scope configuration in your TLS inspection configuration. Because a service endpoint hostname might resolve and be served using more than one IP address, it's important to ensure that the list of destinations contain all of the CIDRs corresponding to the service being investigated. In the following example, the server name with the revoked certificate is `revoked-rsa-dv.ssl.com` with IP address `54.92.160.72/32`, and is associated with the scope configuration on a TLS configuration as follows:

```
{
  ...
  "Scopes": [{
    "Sources": [{"AddressDefinition": "0.0.0.0/0"}],
    "Destinations": [{"AddressDefinition": "54.92.160.72/32"}],
    "SourcePorts": [{"FromPort": 0, "ToPort": 65535}],
    "DestinationPorts": [{"FromPort": 443, "ToPort": 443}],
    "Protocols": [6]
  }], {
    "Sources": [{"AddressDefinition": "0.0.0.0/0"}],
    "Destinations": [{"AddressDefinition": "55.0.0.1/24"}],
    "SourcePorts": [{"FromPort": 0, "ToPort": 65535}],
    "DestinationPorts": [{"FromPort": 443, "ToPort": 443}],
    "Protocols": [6]
  }]
  ...
}
```

When you update the scope configuration to not send traffic for that target through TLS inspection, you can exclusively pass that traffic for the revoked target through the stateful engine without changing to a global **Pass** action for all revocation check failures. This keeps the traffic in the remaining scopes, flowing through the TLS inspection, without the existing **Drop** or **Reject** actions.

```
{
  ...
  "Scopes": [{
    "Sources": [{"AddressDefinition": "0.0.0.0/0"}],
    "Destinations": [{"AddressDefinition": "55.0.0.1/24"}],
```

```

    "SourcePorts": [{"FromPort": 0, "ToPort": 65535}],
    "DestinationPorts": [{"FromPort": 443, "ToPort": 443}],
    "Protocols": [6]
  }]
  ...
}

```

For information about scope configurations, see [TLS inspection configuration settings in AWS Network Firewall](#).

You can verify your scope configuration changes using the alert or TLS logs. For information about logging, see [Logging and monitoring in AWS Network Firewall](#).

Verify with TLS logs

Review the TLS logs to verify that the servers you've removed from the scope of TLS inspection are no longer showing up. TLS logging reports revocation check failures for outbound traffic and SNI mismatches and naming errors.

Verify with alert logs

Add an alert rule to match the SNI of the revoked certificate, like in the following example.

Example rule:

```

alert tls any any -> 54.92.160.72 443 (msg:"revoked cert rule match"; tls.sni;
content:"revoked-rsa-dv.ssl.com"; ssl_state:client_hello; sid:1;)

```

Example alert log:

```

{
  "firewall_name": "test-network-firewall",
  ...
  "event": {
    "timestamp": "2023-10-25T15:14:11.010637+0000",
    "app_proto": "tls",
    "proto": "TCP",
    "src_ip": "10.0.2.230",
    "src_port": 55784,
    "dest_ip": "54.92.160.72",
    "dest_port": 443,
    "event_type": "alert",
    "alert": {

```

```
    "severity": 1,
    "signature_id": 1,
    "signature": "revoked cert rule match",
    ...
  },
  "tls": {
    "subject": "CN=revoked-rsa-dv.ssl.com",
    "issuerdn": "C = US, ST = Texas, L = Houston, O = SSL Corporation, CN = SSL.com
Root Certification Authority RSA",
    ...
  },
  ...
}
}
```

Troubleshooting connection issues with AWS service endpoints (including the AWS Systems Manager agent)

When TLS outbound inspection is enabled, depending on the firewall's VPC architecture, instances behind a firewall might be unable to connect to some public AWS service endpoints (including Systems Manager). This can occur for the following common reasons.

- Some AWS service clients (and some other applications) can use an internal set of trusted root certificates that is different from the host operating system default trusted root certificates. This can cause certificate verification errors on the client for the certificate provided by the firewall in the TLS connection. To fix this, set the `AWS_CA_BUNDLE` environment variable on the client device to the directory that has the root certificate for the CA certificate that has been configured on the firewall for outbound inspection. For more information about this environment variable, see [AWS CLI supported environment variables](#) in the *AWS CLI Version 2 User Guide*. Alternately, add the `--ca-bundle` parameter to your CLI calls to avoid the issue.
- Additionally, because the firewall decrypts TLS traffic before stateful inspection, be sure that your firewall has [appropriate stateful rules](#) to allow decrypted HTTP (not TLS or HTTPS) traffic to reach the endpoints to which clients connect.

Troubleshooting TLS - Connections dropping or resetting

When TLS inspection is enabled, if you experience connection drops, it's possible that the cause is TLS errors. You can use the TLS logs to determine if this is the case and to troubleshoot the problems that are logged. To do this, enable TLS logs and then check the logs for TLS error entries.

In the following example TLS log entry, the error is in the client hello.

```
{
  "firewall_name": "firewall-tls-test",
  "availability_zone": "us-east-1d",
  "event_timestamp": 1719943304,
  "event": {
    "timestamp": "2024-07-02T18:01:44.412778Z",
    "src_ip": "10.0.2.53",
    "src_port": "59844",
    "dest_ip": "10.0.1.27",
    "dest_port": "443",
    "sni": "-",
    "tls_error": {
      "error_message": "Server name is not found in client hello."
    }
  }
}
```

For information about enabling TLS logging, see [Logging network traffic from AWS Network Firewall](#).

For more information about the logs for this type of error, see [Logging for TLS inspection in AWS Network Firewall](#).

Tagging AWS Network Firewall resources

A *tag* is a custom attribute label that you assign or that AWS assigns to an AWS resource. Each tag has two parts:

- A *tag key*, for example `CostCenter`, `Environment`, or `Project`. Tag keys are case sensitive.
- An optional *tag value*, for example, `111122223333` or `Production`. Omitting the tag value is the same as using an empty string. Tag values are case sensitive.

You can use tags to do the following:

- Identify and organize your AWS resources. Many AWS services support tagging, so you can assign the same tag to resources from different services to indicate that the resources are related. For example, you could assign the same tag to an Amazon Virtual Private Cloud VPC that you assign to an firewall and firewall policy in AWS Network Firewall.
- Track your AWS costs. To do this, you activate tags on the AWS Billing and Cost Management dashboard. AWS uses the tags to categorize your costs and deliver a monthly cost allocation report to you. For more information, see [Use cost allocation tags](#) in the *AWS Billing User Guide*.

The following sections provide more information about tags for AWS Network Firewall.

Supported resources in Network Firewall

The following resources in Network Firewall support tagging:

- Firewalls
- Firewall policies
- Rule groups

For information about adding and managing tags, see [Managing tags](#).

Tag naming and usage conventions

The following basic naming and usage conventions apply to using tags with Network Firewall resources:

- Each resource can have a maximum of 50 tags.
- For each resource, each tag key must be unique, and each tag key can have only one value.
- The maximum tag key length is 128 Unicode characters in UTF-8.
- The maximum tag value length is 256 Unicode characters in UTF-8.
- Allowed characters are letters, numbers, spaces representable in UTF-8, and the following characters: . : + = @ _ / - (hyphen). Amazon EC2 resources allow any characters.
- Tag keys and values are case sensitive. As a best practice, decide on a strategy for capitalizing tags, and consistently implement that strategy across all resource types. For example, decide whether to use `Costcenter`, `costcenter`, or `CostCenter`, and use the same convention for all tags. Avoid using similar tags with inconsistent case treatment.
- The `aws :` prefix is prohibited for tags; it's reserved for AWS use. You can't edit or delete tag keys or values with this prefix. Tags with this prefix do not count against your tags per resource quota.

Managing tags

For ease of use and best results, use the Tag Editor in the AWS Resource Groups console. It provides a central, unified way to create and manage your tags. For more information, see [Working with Tag Editor](#).

You can also use AWS Network Firewall to apply tags while you are creating and managing your Network Firewall firewalls, firewall policies, and rule groups.

Using the AWS Network Firewall REST API

AWS Network Firewall is a stateful, managed, network firewall and intrusion detection and prevention service for your virtual private cloud (VPC) that you create in Amazon Virtual Private Cloud (Amazon VPC). With Network Firewall, you can filter traffic at the perimeter of your VPC. This includes filtering traffic going to and coming from an internet gateway, NAT gateway, or over VPN or AWS Direct Connect. Network Firewall uses the open source intrusion prevention system (IPS), Suricata, for stateful inspection. Network Firewall supports Suricata compatible rules.

This section describes how to make requests to the Network Firewall API for creating and managing firewalls in Network Firewall. This section covers the components of requests, the content of responses, and how to authenticate requests.

For general guidance on accessing the AWS APIs, see the [AWS APIs](#) in the *AWS General Reference*.

Note

If you use a programming language that has an AWS SDK, use the SDK rather than trying to work your way through the APIs. The SDKs make authentication simpler, integrate more easily with your development environment, and provide easy access to Network Firewall commands. For more information about the AWS SDKs, see [Setting up tool access](#) in the topic [Setting up](#).

Topics

- [Making HTTPS requests to AWS Network Firewall](#)
- [HTTP responses](#)
- [Authenticating requests](#)

Making HTTPS requests to AWS Network Firewall

Network Firewall requests are HTTPS requests, as defined by [RFC 2616](#). Like any HTTP request, a request to Network Firewall contains a request method, a URI, request headers, and a request body. The response contains an HTTP status code, response headers, and sometimes a response body.

Request URI

The request URI is always a single forward slash, /.

HTTP headers

Network Firewall requires the following information in the header of an HTTP request.

Host (Required)

The endpoint that specifies where your resources are created. You can find the various endpoints in [AWS service endpoints](#). For example, the value of the Host header for Network Firewall for a CloudFront distribution is `network-firewall.amazonaws.com:443`.

x-amz-date or Date (Required)

The date used to create the signature that is contained in the Authorization header. Specify the date in ISO 8601 standard format, in UTC time, as shown in the following example:

```
x-amz-date: 20151007T174952Z
```

You must include either `x-amz-date` or `Date`. (Some HTTP client libraries don't let you set the `Date` header). When an `x-amz-date` header is present, Network Firewall ignores any `Date` header when authenticating the request.

The timestamp must be within 15 minutes of the AWS system time when the request is received. If it isn't, the request fails with the `RequestExpired` error code to prevent someone else from replaying your requests.

Authorization (Required)

The information required for request authentication. For more information about constructing this header, see [Authenticating requests](#).

X-Amz-Target (Required)

The operation, provided as a concatenation of the following values:

- `NetworkFirewall_`
- The API version without punctuation
- A period (.)
- The name of the operation

Example:

```
NetworkFirewall_20201112.CreateFirewall
```

Content-Type (Conditional)

The type and version of the content. Specify the version of JSON, as shown in the following example:

```
Content-Type: application/x-amz-json-1.0
```

Condition: Required for POST requests.

Content-Length (Conditional)

The length of the message, without the headers, according to RFC 2616.

Condition: Required if the request body itself contains information. Most toolkits add this header automatically.

The following is an example header for an HTTP request to create a firewall in Network Firewall:

```
POST / HTTP/1.1
Host: network-firewall.amazonaws.com:443
X-Amz-Date: 20151007T174952Z
Authorization: AWS4-HMAC-SHA256
                Credential=AKIAIOSFODNN7EXAMPLE/20151007/us-east-2/network-firewall/
aws4_request,
                SignedHeaders=host;x-amz-date;x-amz-target,

                Signature=145b1567ab3c50d929412f28f52c45dbf1e63ec5c66023d232a539a4afd11fd9
X-Amz-Target: NetworkFirewall_20201112.CreateFirewall
Accept: */*
Content-Type: application/x-amz-json-1.0; charset=UTF-8
Content-Length: 231
Connection: Keep-Alive
```

HTTP request body

Many Network Firewall API actions require you to include JSON-formatted data in the body of the request.

HTTP responses

All Network Firewall API actions include JSON-formatted data in the response.

Here are some important headers in the HTTP response and how you should handle them in your application, if applicable.

HTTP/1.1

This header is followed by a status code. Status code 200 indicates a successful operation.

Type: String

x-amzn-RequestId

A value created by Network Firewall that uniquely identifies your request, for example, K2QH8DN0U907N97FNA2GDLL80BVV4KQNS05AEMVJF66Q9ASUAAJG. If you have a problem with Network Firewall, AWS can use this value to troubleshoot the problem.

Type: String

Content-Length

The length of the response body in bytes.

Type: String

Date

The date and time that Network Firewall responded, for example, Wed, 07 Oct 2019 12:00:00 GMT.

Type: String

Error responses

If a request results in an error, the HTTP response contains the following values:

- A JSON error document as the response body
- A Content-Type header
- The applicable 3xx, 4xx, or 5xx HTTP status code

The following is an example of a JSON error document:

```
HTTP/1.1 400 Bad Request
x-amzn-RequestId: b0e91dc8-3807-11e2-83c6-5912bf8ad066
x-amzn-ErrorType: ValidationException
Content-Type: application/json
Content-Length: 125
Date: Mon, 26 Nov 2012 20:27:25 GMT

{"message": "1 validation error detected: Value null at 'TargetString' failed to satisfy constraint: Member must not be null"}
```

Authenticating requests

If you use a language that AWS provides an SDK for, we recommend that you use the SDK. All the AWS SDKs greatly simplify the process of signing requests and save you a significant amount of time when compared with using the Network Firewall API. In addition, the SDKs integrate easily with your development environment and provide easy access to related commands.

Network Firewall requires that you authenticate every request that you send by signing the request. To sign a request, you calculate a digital signature using a cryptographic hash function, which returns a hash value based on the input. The input includes the text of your request and your secret access key. The hash function returns a hash value that you include in the request as your signature. The signature is part of the Authorization header of your request.

Network Firewall supports authentication using [AWS Signature Version 4](#). Follow the process for signing your request at see the [Signing AWS requests with Signature Version 4](#) in the *AWS General Reference*.

After receiving your request, Network Firewall recalculates the signature using the same hash function and input that you used to sign the request. If the resulting signature matches the signature in the request, Network Firewall processes the request. If not, Network Firewall rejects the request.

AWS Network Firewall quotas

AWS Network Firewall is subject to the following quotas (formerly referred to as limits). These quotas are the same for all AWS Regions in which Network Firewall is available. Each Region is subject to these quotas individually. The quotas are not cumulative across Regions.

Network Firewall has the following default quotas on the maximum number of entities you can have per account per Region. You can request an increase in these adjustable quotas through the [Service Quotas console](#).

Resource	Default quota
Maximum number of firewalls per account per Region.	5
Maximum number of firewall policies per account per Region.	20
Maximum number of stateful rule groups per account per Region.	50
You can't use all of these rule groups in a single firewall policy. See the immutable maximum number of stateful rule groups per firewall policy in the table that follows this one.	
Maximum number of stateless rule groups per account per Region.	50
You can't use all of these rule groups in a single firewall policy. See the immutable maximum number of stateless rule groups per firewall policy in the table that follows this one.	
Maximum number of stateful rules per firewall policy per account per Region. This is the total across all rule groups that are referenced by the policy.	30,000
Maximum number of TLS inspection configurations per account per Region.	20

Network Firewall has the following quotas that can't be changed.

Resource	Quota
Maximum character length of a Suricata rule. Each variable value in the rule counts towards this limit.	8,192
Maximum size of a Suricata-compatible rules string for a rule group, in bytes.	2,000,000
Maximum stateful rule group capacity. For more information, see Setting rule group capacity in AWS Network Firewall .	30,000
Maximum number of IP set references per Suricata compatible stateful rule group. For information about IP set references, see IP set references in Suricata compatible AWS Network Firewall rule groups .	5
Maximum number of stateful rule groups per firewall policy.	20
Maximum number of stateless rule groups per firewall policy.	20
Maximum stateless rule group capacity. For more information, see Setting rule group capacity in AWS Network Firewall .	30,000
Maximum number of custom actions per stateless rule group.	10
Maximum number of stateless rules per firewall policy. This is the total across all rule groups that are referenced by the policy.	30,000
Maximum network traffic bandwidth per firewall endpoint. If you require more traffic bandwidth, you can split your resources into subnets and create a firewall in each subnet.	100 Gbps
Required number of firewall policies per firewall.	1
Maximum number of firewalls that can use the same firewall policy.	1,000
Maximum number of firewall policies that can use the same rule group.	1,000
Maximum number of TLS inspection configurations per firewall policy.	1

Resource	Quota
Maximum number of firewall policies that can use the same TLS inspection configuration.	1,000
Maximum number of certificate authority (CA) certificates per TLS inspection configuration. CA certificates are used for outbound SSL/TLS inspection.	1
Maximum number of server certificates per TLS inspection configuration. Server certificates are used for inbound SSL/TLS inspection.	10

Resources

The following related resources can help you as you work with this service.

AWS resources

Amazon Web Services provides the following resources specific to AWS Network Firewall.

- [Network Firewall discussion forum](#) – A community-based forum for developers to discuss technical questions related to Network Firewall.
- [Network Firewall product information](#) – The primary web page for information about Network Firewall, including features and pricing.
- [Amazon Virtual Private Cloud product information](#) – The primary web page for information about Amazon VPC, including features and pricing.

Amazon Web Services provides the following general guides, forums, and other resources.

- [Classes & Workshops](#) – Links to role-based and specialty courses, in addition to self-paced labs to help sharpen your AWS skills and gain practical experience.
- [AWS Developer Center](#) – Explore tutorials, download tools, and learn about AWS developer events.
- [AWS Developer Tools](#) – Links to developer tools, SDKs, IDE toolkits, and command line tools for developing and managing AWS applications.
- [Getting Started Resource Center](#) – Learn how to set up your AWS account, join the AWS community, and launch your first application.
- [Hands-On Tutorials](#) – Follow step-by-step tutorials to launch your first application on AWS.
- [AWS Whitepapers](#) – Links to a comprehensive list of technical AWS whitepapers, covering topics such as architecture, security, and economics and authored by AWS Solutions Architects or other technical experts.
- [AWS Support Center](#) – The hub for creating and managing your AWS Support cases. Also includes links to other helpful resources, such as forums, technical FAQs, service health status, and AWS Trusted Advisor.
- [AWS Support](#) – The primary webpage for information about AWS Support, a one-on-one, fast-response support channel to help you build and run applications in the cloud.

- [Contact Us](#) – A central contact point for inquiries concerning AWS billing, account, events, abuse, and other issues.
- [AWS Site Terms](#) – Detailed information about our copyright and trademark; your account, license, and site access; and other topics.

Document history for AWS Network Firewall

The following table describes important changes to this documentation.

Change	Description	Date
Added support for IPv6 service endpoints	AWS Network Firewall now supports dual-stack endpoints.	December 20, 2024
Updated supported Suricata version to 7.0.	AWS Network Firewall supports Suricata version 7.0. This might require changes in your use of Network Firewall. For information about the update to this version, see Upgrading 6.0 to 7.0 and the Suricata User Guide .	November 24, 2024
New idle timeout configuration	You can now configure the TCP idle timeout in your firewall policy settings.	October 30, 2024
Interface endpoints through AWS PrivateLink	You can use AWS PrivateLink to create a private connection between your VPC and AWS Network Firewall, without requiring access through an internet connection.	September 12, 2024
Country code filtering in stateful rules	You can now use the Suricata <code>geoip</code> keyword in stateful rules, to filter for the country codes associated with IP addresses.	August 28, 2024

Removed caveat regarding QUIC protocol detection	Network Firewall now supports QUIC protocol detection.	August 16, 2024
TLS logging	You can now use the TLS log type to log TLS errors and outbound traffic that fails a TLS inspection server certificate revocation check. This is a new log type, in addition to the existing alert and flow log types.	July 25, 2024
Stateful rules match on TLS.SNI for decrypted traffic	With TLS inspection, Network Firewall now matches on the TLS.SNI keyword in stateful rules, even when it decrypts traffic.	June 25, 2024
Quota on stateful rules per policy is adjustable	The Network Firewall service quota for stateful rules per firewall policy is now adjustable.	May 22, 2024
Removed Regional availability constraint for outbound SSL/TLS inspection	Network Firewall now supports inspection of outbound SSL/TLS traffic in all Regions that Network Firewall is available in. For information about available Regions, see AWS Network Firewall endpoints and quotas in the <i>Amazon Web Services General Reference</i> .	December 19, 2023

Added caveat regarding IP-only rule syntax	Unless you include ! with your destination IP, Suricata treats the rule as an IP-only rule.	November 17, 2023
New stateless rule group analyzer	Network Firewall now has a stateless rule group analyzer that identifies stateless rules that have asymmetric routing.	November 2, 2023
Outbound SSL/TLS inspection is available in Israel (Tel Aviv) and Europe (Ireland)	Network Firewall now supports inspection of outbound SSL/TLS traffic in the Israel (Tel Aviv) Region and the Europe (Ireland) Region.	October 26, 2023
New troubleshooting chapter	Added a chapter on troubleshooting problems with configuring and using Network Firewall.	October 20, 2023
New tls_inspected flag	Network Firewall now adds a tls_inspected field to firewall logs to indicate when there's TLS traffic flowing across a firewall that's enabled with TLS inspection.	October 12, 2023
New stream exception policy topic	Added information about a firewall policy's stream exception policy.	October 12, 2023
New Suricata rule examples	Added examples of Suricata rules that can be used with Network Firewall.	October 6, 2023

New CloudWatch metrics	New metrics for tracking TLS packet count: <code>TLSDroppedPackets</code> , <code>TLSPassedPackets</code> , and <code>TLSRejectedPackets</code> .	October 2, 2023
Added unsupported certificate type	Network Firewall doesn't support cross-signed root certificates in TLS inspection configurations.	September 25, 2023
Updated console procedures for creating rule groups	Updated the console procedures to reflect the new console user experience.	August 31, 2023
Updated console procedures for creating a firewall policy	Updated the console procedure to reflect the new console user experience.	August 31, 2023
Updated console procedures for creating a firewall	Updated the console procedure to reflect the new console user experience.	August 31, 2023
Added two new error states	Added two error states regarding invalid certificates in TLS inspection configurations.	August 24, 2023
New CloudWatch metrics	<code>TLSTimedOutConnections</code> is the number of SSL/TLS connections that timed out during SSL/TLS inspection by Network Firewall. <code>TLSErrors</code> is the number of errors observed by Network Firewall while inspecting SSL/TLS packets.	June 26, 2023

Adding note about pass behavior	If a packet within a flow matches a rule containing pass action, then Suricata doesn't scan the other packets in that flow and passes the unscanned packets.	June 9, 2023
Added caveat regarding QUIC protocol detection	Network Firewall doesn't currently support QUIC protocol detection.	May 25, 2023
TLS inspection configurations now available in all Regions	TLS inspection configurations are now available in all Regions that AWS Network Firewall is available in. For more information, see What's New with AWS .	May 9, 2023
New stream exception REJECT option	You can now choose to reject traffic in your midstream exception configurations.	May 4, 2023
New firewall policy option	You can now override the Suricata HOME_NET variable with your own CIDRs. This is helpful when working using a centralized deployment model.	May 3, 2023
TLS inspection configurations now available in additional Regions	TLS inspection configurations are now available in additional Regions. For more information, see What's New with AWS .	April 27, 2023

New chapter on TLS inspection configurations	Network Firewall now supports TLS inspection configurations. Use TLS inspection configurations with your firewall policy to enable decryption and re-encryption of the SSL/TLS traffic going through your firewall.	March 30, 2023
New CloudWatch TLSReceivedPackets metric	TLSReceivedPackets is the number of TLS packets received by the Network Firewall firewall.	March 30, 2023
AWS managed policy updates - Update to an existing policy	Updated AWSNetworkFirewallServiceRolePolicy to support describing ACM certificates for use with TLS inspection configurations.	March 30, 2023
New topic on asymmetric routing	Provides information about how to prevent asymmetric routing issues within your firewall.	March 28, 2023
Updated the IAM guidance for AWS Network Firewall	Updated guide to align with the IAM best practices . For more information, see Security best practices in IAM .	February 15, 2023
New resource type for IP set references	You can now include resource groups in your IP set references.	February 14, 2023

New Network Firewall resource groups top-level resource	Network Firewall now supports referencing resource groups in stateful rule groups. Resource groups ensure that your rules stay in sync as your AWS resources change.	February 14, 2023
Added note regarding 5-tuple traffic direction keyword	When you create a 5-tuple rule from the console, the rule doesn't automatically add the direction keyword <code>to_server</code> .	February 2, 2023
Added caveat regarding EXTERNAL_NET	If customers override <code>HOME_NET</code> , they must also override <code>EXTERNAL_NET</code> to equal the negation of <code>HOME_NET</code> .	February 2, 2023
New subnet IP address type	You can now configure your subnets to use IPv4, IPv6, or dualstack IP addresses.	January 17, 2023
New stateful rule action	Network Firewall now supports the stateful rule action <code>reject</code> , in addition to the actions <code>pass</code> , <code>drop</code> , and <code>alert</code> .	January 9, 2023
New CloudWatch RejectedPackets metric	<code>RejectedPackets</code> tracks the number of packets rejected due to <code>Reject</code> stateful rule actions.	January 9, 2023
New status message field	Use a firewall's status message to troubleshoot why an endpoint is failing.	December 28, 2022

Added evaluation order for stateful domain list rule groups	You can now configure evaluation order for your own stateful domain list rule groups.	December 21, 2022
New stream exception configuration for firewall policies	You can now select how Network Firewall handles traffic when there's a midstream break in network traffic.	October 5, 2022
Added maximum number of IP set references	You can use as many as five IP set references per Suricata compatible stateful rule group.	October 5, 2022
Added maximum network traffic bandwidth per firewall endpoint	The maximum network traffic bandwidth per firewall endpoint is 100 Gbps.	September 19, 2022
Added two new threat signature categories	Added support for Malware Coin Mining and Phishing.	July 29, 2022
New topic on using IP set references	IP set references enable you to reference an IP set resource, such as an Amazon VPC prefix list, in your Suricata compatible stateful rules.	July 21, 2022
Updated endpoint capacity	Network Firewall now supports as much as 100 Gbps of network traffic per firewall endpoint.	June 17, 2022

Added caveat regarding inner packet inspection for tunneling protocols	The Network Firewall stateful rule engine supports inner packet inspection for tunneling protocols. To block the tunnelled traffic, you can write rules against the tunnel layer or against the inner packet.	June 14, 2022
Added warning regarding AWS KMS customer managed keys	If you revoke access to the grant or delete the customer managed keys, endpoints encrypted using the customer managed keys will drop all packets.	June 2, 2022
Updated AWS managed rule groups for Network Firewall	Added documentation for each rule in the AWS managed rule groups for Network Firewall.	April 28, 2022
Added support for threat signature managed rule groups	AWS Network Firewall now supports threat signature AWS Managed Rule Groups.	April 28, 2022
New topic on encryption using AWS KMS customer managed keys	Network Firewall now supports the use of customer managed keys to encrypt data at rest.	April 26, 2022
Added maximum character length for Suricata rules	The maximum character length of a Suricata rule is 8,192.	March 22, 2022
Added support for managed rule groups	AWS Network Firewall now supports AWS Managed Rule Groups.	December 9, 2021

Optional strict evaluation order for Suricata compatible stateful rule groups	This release adds support for strict ordering for stateful rule groups. Using strict ordering, stateful rule groups are evaluated in the exact order in which you provide them in the firewall policy.	October 1, 2021
Expanded availability of AWS managed policy	Network Firewall expanded the availability of the managed policy <code>AWSNetworkFirewallServiceRolePolicy</code> to AWS GovCloud (US) Regions.	June 24, 2021
Increased stateless rule group capacity	The capacity for stateless rule groups is increased from 10,000 to 30,000.	June 10, 2021
Reorganized stateful rule groups sections and expanded examples	Domain list rule groups and the standard stateless rule groups provide easy entry forms for Suricata compatible rule strings, and the documentation didn't indicate this. Reorganized stateful rule group sections, clarified the information, and added examples showing the correlation between the easy entry forms and the resulting Suricata compatible rule strings.	April 28, 2021
JA3 keywords support	JA3 keywords are now supported by Network Firewall.	April 28, 2021

[First release of AWS Network Firewall](#)

Network Firewall is now available to provide firewall protection for your Amazon Virtual Private Cloud VPCs.

November 16, 2020