

Cloud security continues to be a critical focus for organizations, since most organizations are moving their infrastructure to the cloud. With the growing importance of secure network architectures in the cloud, AWS Network Firewall (ANFW) offers robust protection to control traffic within an AWS environment. Recently, I completed Labs 1 and 2 of the AWS Hands-on Network Firewall Workshop. This documentation documents my journey through setting up and configuring a protected VPC with public and private workloads, providing insights into AWS Network Firewall's powerful capabilities.

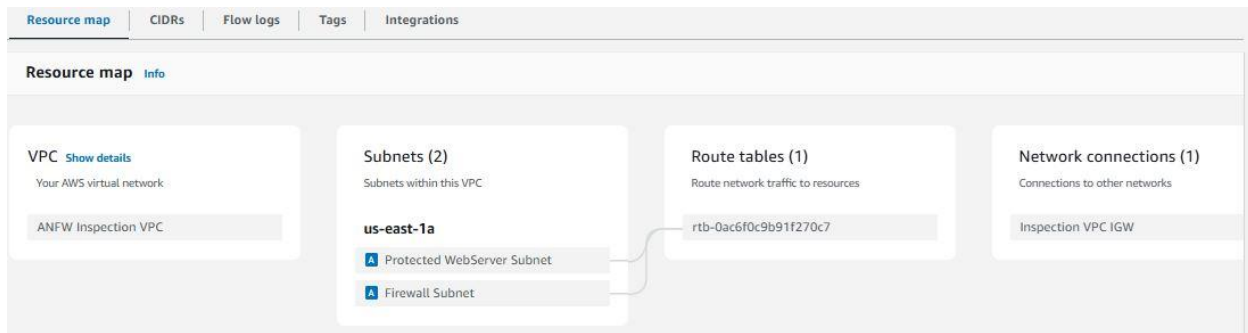
Setting Up the Lab Environment for Lab 1.

I used CloudFormation to create infrastructure needed for this lab. This template created vpc, firewall subnet, protected webserver subnet, internet gateway, webserver and an EC2 instance. The image below shows code sample from template.

```
Parameters:
  AvailabilityZoneSelection:
    Description: Availability Zone
    Type: AWS::EC2::AvailabilityZone::Name
    Default: us-east-1a
  LatestAmiId:
    Description: Latest EC2 AMI from Systems Manager Parameter Store
    Type: 'AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>'
    Default: '/aws/service/ami-amazon-linux-latest/al2023-ami-kernel-default-x86_64'
  InstanceType:
    Description: WebServer EC2 instance type
    Type: String
    Default: t3.micro
Resources:
  # Inspection VPC Deployment for the AWS Network Firewall
  InspectionVPC:
    Type: AWS::EC2::VPC
    Properties:
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
      CidrBlock: 10.1.0.0/16
      Tags:
        - Key: Name
          Value: "ANFW Inspection VPC"
  FirewallSubnet:
    Type: AWS::EC2::Subnet
    Properties:
      AvailabilityZone: !Ref AvailabilityZoneSelection
```

Lab 1: Protected VPC with Public Workload.

The resource map after the the template was loaded is as shown below:



I applied firewall rules to control traffic for the configured vpc. The goal was to set up a basic firewall for a publicly accessible web server, log network activity, and configure initial route tables.

The steps taken were:

1. Creating a firewall.

I created an AWS Network Firewall and attached it to the firewall subnet, as shown below:

The screenshot shows the 'Describe firewall' screen in the AWS Management Console. The left sidebar lists the steps: Step 1 (Describe firewall), Step 2 (Configure VPC and subnets), Step 3 (optional, Configure advanced settings), Step 4 (Associate firewall policy), Step 5 (optional, Add tags), and Step 6 (Review and create). The main content area is titled 'Describe firewall' and includes a description: 'Name and describe your firewall so you can easily identify it and distinguish it from other resources.' The 'Firewall details' section contains the following fields:

- Firewall name:** ANFW-Lab. The name must have 1-128 characters. Valid characters: a-z, A-Z, 0-9 and - (hyphen). The name can't start or end with a hyphen, and it can't contain two consecutive hyphens.
- Description - optional:** Enter firewall description. The description can have 0-256 characters.

At the bottom right, there are 'Cancel' and 'Next' buttons.

VPC > Network Firewall: Firewalls > Create firewall

Step 1
[Describe firewall](#)

Step 2
Configure VPC and subnets

Step 3 - optional
[Configure advanced settings](#)

Step 4
[Associate firewall policy](#)

Step 5 - optional
[Add tags](#)

Step 6
[Review and create](#)

Configure VPC and subnets [Info](#)

The firewall protects the subnets within an Amazon Virtual Private Cloud (VPC) by filtering traffic going between the subnets and locations outside of your VPC. After you create the firewall and its associated firewall policy, configure your VPC to route traffic through the endpoints created by the firewall.

VPC
For each Availability Zone where you want protection, provide Network Firewall with a public subnet that's dedicated to the firewall endpoint. Only use the firewall subnets that you specify here for the firewall. Don't use them for any other purpose.

VPC
Choose the VPC where you want to create this firewall.
ANFW Inspection VPC

Firewall subnets
Each subnet must have one available IP address. You can't change the subnet's IP address type after creation.

Availability Zone	Subnet	IP address type	
us-east-1a	subnet-0af0368c2...	IPv4	Remove subnet

Add new subnet

Cancel Previous **Next**

Activate

Then I associated an existing firewall policy that I had previously created, to the firewall. All the other setting I left it as default, then I reviewed and created the firewall.

VPC > Network Firewall: Firewalls > Create firewall

Step 1
[Describe firewall](#)

Step 2
[Configure VPC and subnets](#)

Step 3 - optional
[Configure advanced settings](#)

Step 4
Associate firewall policy

Step 5 - optional
[Add tags](#)

Step 6
[Review and create](#)

Associate firewall policy [Info](#)

A firewall policy defines the monitoring and protection behavior for the firewall.

Associated firewall policy
The firewall policy contains a list of rule groups that define how the firewall inspects and manages web traffic. You can configure the associated firewall policy after you create the firewall.

Firewall policy
Either create a new firewall policy, or associate an existing firewall policy.

☐ Create and associate an empty firewall policy

☒ Associate an existing firewall policy

Choose firewall policy
Choose the firewall policy to associate with this firewall.
ANFW-Lab-Policy

Cancel Previous **Next**

2.Creating Route tables.

I created three route tables to direct traffic to and from my NGINX web server through the firewall for inspection. Traffic will flow through the firewall subnet where the firewall policies can be applied before the request reaches the server.

The IGW Ingress route table is used to ensure traffic coming in to the VPC gets routed to the network firewall first and then the firewall can make a determination on what to do with that traffic.

The screenshot shows the 'Create route table' page in the AWS console. The breadcrumb navigation is 'VPC > Route tables > Create route table'. The page title is 'Create route table' with an 'Info' link. A descriptive text states: 'A route table specifies how packets are forwarded between the subnets within your VPC, the internet, and your VPN connection.' The 'Route table settings' section includes a 'Name - optional' field with the value 'IGW-Ingress-Route-Table' and a 'VPC' dropdown menu showing 'vpc-0f9060eae12108e60 (ANFW Inspection VPC)'. The 'Tags' section shows a key-value pair: 'Name' as the key and 'IGW-Ingress-Route-Table' as the value. At the bottom, there are 'Cancel' and 'Create route table' buttons.

The Firewall Route Table. This route table will control flow from the firewall subnet to the rest of the VPC as well as all other traffic (the default route) going to the Internet Gateway (IGW).

The screenshot shows the 'Describe firewall' page in the AWS console. The breadcrumb navigation is 'VPC > Network Firewall: Firewalls > Create firewall'. A sidebar on the left lists steps: 'Step 1 Describe firewall' (active), 'Step 2 Configure VPC and subnets', 'Step 3 - optional Configure advanced settings', 'Step 4 Associate firewall policy', 'Step 5 - optional Add tags', and 'Step 6 Review and create'. The main content area is titled 'Describe firewall' with an 'Info' link. It includes a description: 'Name and describe your firewall so you can easily identify it and distinguish it from other resources.' The 'Firewall details' section has a 'Firewall name' field with the value 'ANFW-Lab' and a 'Description - optional' text area. At the bottom, there are 'Cancel' and 'Next' buttons.

I created the **Protected WebServer Route Table**. This route table directed all traffic from the protected workflow subnet bound for anywhere outside of the VPC to the firewall endpoint so that outbound requests can also be processed by the firewall.

[VPC](#) > [Route tables](#) > Create route table

Create route table [Info](#)

A route table specifies how packets are forwarded between the subnets within your VPC, the internet, and your VPN connection.

Route table settings

Name - *optional*

Create a tag with a key of 'Name' and a value that you specify.

VPC

The VPC to use for this route table.

Tags

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key

Value - *optional*

You can add 49 more tags.

Connect to Ec2 and enable the webserver.

I added a rule in the pre-configured security group to allow http traffic from my IP address.

[EC2](#) > [Security Groups](#) > [sg-0288833c98ba3ea5f - Instance Security Group](#) > Edit inbound rules

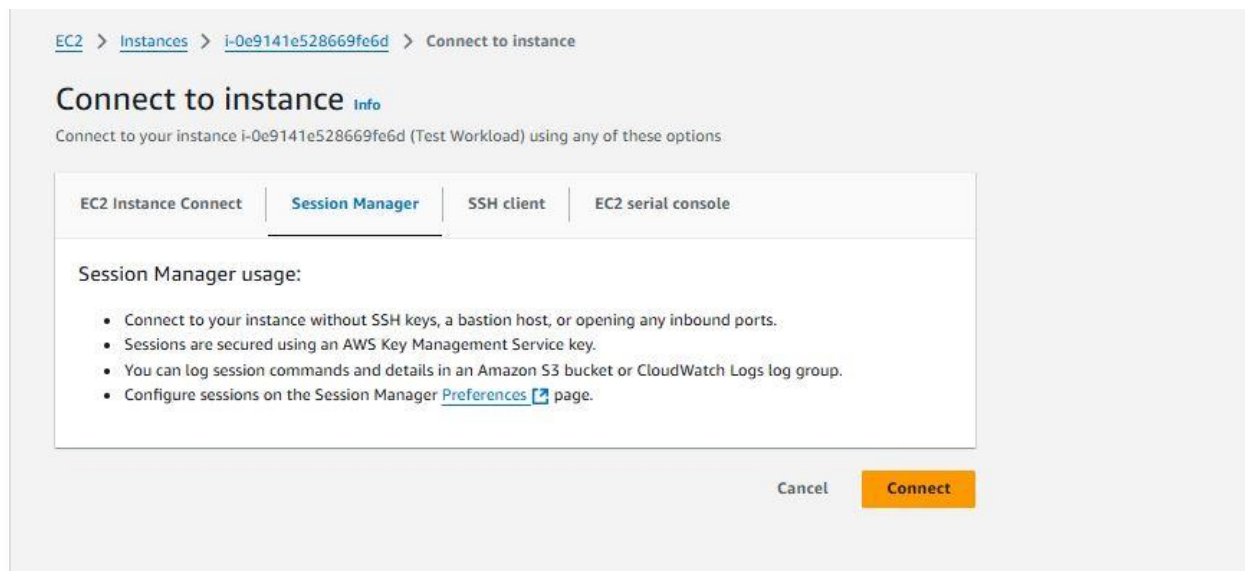
Edit inbound rules [Info](#)

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules [Info](#)

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info	
sgr-0620bd55c10215a87	All ICMP - IPv4	ICMP	All	Custom	<input type="text" value="10.0.0.0/8"/>	<input type="button" value="Delete"/>
-	HTTP	TCP	80	My IP	<input type="text" value="41.90.4.89/32"/>	<input type="button" value="Delete"/>

I then connected to my EC2 instance via the session manager.



Typed the following code after it opened, to install NGINX to accept HTTP requests and test ingress web server traffic to my Test Workload through the ANFW:

```
sudo dnf install nginx -y
```

```
sudo systemctl enable nginx
```

```
sudo systemctl start nginx
```

```
(4/7): gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64.rpm          3.2 MB/s | 308 kB  00:00
(5/7): nginx-filessystem-1.26.2-1.amzn2023.0.1.noarch.rpm      479 kB/s | 9.9 kB  00:00
(6/7): nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch.rpm       983 kB/s | 21 kB  00:00
(7/7): nginx-core-1.26.2-1.amzn2023.0.1.x86_64.rpm           6.1 MB/s | 670 kB  00:00
-----
Total                                                         4.6 MB/s | 1.1 MB  00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing                :                               1/1
  Running scriptlet: nginx-filessystem-1:1.26.2-1.amzn2023.0.1.noarch 1/7
  Installing           : nginx-filessystem-1:1.26.2-1.amzn2023.0.1.noarch 1/7
  Installing           : nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch 2/7
  Installing           : libunwind-1.4.0-5.amzn2023.0.2.x86_64 3/7
  Installing           : gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64 4/7
  Installing           : nginx-core-1:1.26.2-1.amzn2023.0.1.x86_64 5/7
  Installing           : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch 6/7
  Installing           : nginx-1:1.26.2-1.amzn2023.0.1.x86_64 7/7
  Running scriptlet: nginx-1:1.26.2-1.amzn2023.0.1.x86_64 7/7
  Verifying           : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch 1/7
  Verifying           : gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64 2/7
  Verifying           : libunwind-1.4.0-5.amzn2023.0.2.x86_64 3/7
  Verifying           : nginx-1:1.26.2-1.amzn2023.0.1.x86_64 4/7
  Verifying           : nginx-core-1:1.26.2-1.amzn2023.0.1.x86_64 5/7
  Verifying           : nginx-filessystem-1:1.26.2-1.amzn2023.0.1.noarch 6/7
  Verifying           : nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch 7/7

Installed:
  generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch  gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64  libunwind-1.4.0-5.amzn2023.0.2.x86_64  nginx-1:1.26.2-1.amzn2023.0.1.x86_64
  nginx-core-1:1.26.2-1.amzn2023.0.1.x86_64        nginx-filessystem-1:1.26.2-1.amzn2023.0.1.noarch  nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch

Complete!
sh-5.2$ sudo systemctl enable nginx
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr/lib/systemd/system/nginx.service.
sh-5.2$ sudo systemctl start nginx
sh-5.2$
```

I then accessed my webserver in a new tab using it's public IP address and this was the output:

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

3. Set up firewall logging.

Now that all routes are configured and traffic is being inspected by the ANFW I set up logging to monitor ingress and egress traffic.

First navigate to the VPC Dashboard and select Firewalls in the left pane under Network Firewalls. Select the **Firewall Details** tab and scroll down to the **Logging** section, then click **Edit** as shown below:

Note: Set the flow log and alert log destination to cloudwatch logs.

Edit firewall logging configuration Info

Configure the log type and the log destination. Logs are generated only for stateful rule groups.

Logging configuration

Log type
You can choose to emit alert logs, flow logs, or both.

☒ Alert
☒ Flow
☐ TLS

Alert log destination

Log destination
You can send each log type to a S3 bucket, a CloudWatch log group, or a Kinesis Data Firehose delivery stream.

☐ S3

CloudWatch log group

Use: "anfw_lab"

/aws/route53/AnfwDemo-dnsfw-queries-loggroup

anfw_lab

Flow log destination

On the Log groups page click on the **Create log group** button in the upper right hand corner from the tab shown on the diagram above:

CloudWatch X

CloudWatch > Log groups > Create log group

Create log group

Log group details info

CloudWatch Logs offers two log classes: Standard and Infrequent Access. [Learn more about the features offered by each log class.](#)

Log group name
anfw_lab

Retention setting
1 week (7 days)

Log class info
Standard

KMS key ARN - optional

Tags
A tag is a label that you assign to an Amazon Web Services resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your Amazon Web Services costs.

First, generate some flow logs to the Test Workload Web Server by navigating to it though a browser. Then check the log stream from cloudwatch.

```
2024-11-14T12:11:11.000Z {"firewall_name":"ANFW-Lab","availability_zone":"us-east-1a","event_timestamp":"1731586271","event":{"tcp":{"tcp_flags":"02","syn":true},"app_proto":...

{
  "firewall_name": "ANFW-Lab",
  "availability_zone": "us-east-1a",
  "event_timestamp": "1731586271",
  "event": {
    "tcp": {
      "tcp_flags": "02",
      "syn": true
    },
    "app_proto": "unknown",
    "src_ip": "147.185.133.99",
    "src_port": 53688,
    "netflow": {
      "pkts": 1,
      "bytes": 44,
      "start": "2024-11-14T12:05:18.152428+0000",
      "end": "2024-11-14T12:05:18.152428+0000",
      "age": 0,
      "min_ttl": 57,
      "max_ttl": 57
    }
  }
}
```

Activate Windows
Go to Settings to activate Windows.

4. Configure firewall policy

I created and test two types of stateful rule categories: standard and domain-list.

After creating the standard stateful rule, this is how it looked like:

Rules (1)

Move up

Move down

Delete

Find rules

<

1

>

⚙

	Protocol	Source	Destination	Source port	Destination port	Direction	Action
<input type="radio"/>	TCP	ANY	ANY	ANY	8080	Forward	Drop

Cancel

Previous

Next

I then generated some traffic to my web server on port 8080 and checked my **Alert** logs stream in the associated CloudWatch log group.

```
2024-11-14T12:39:51.000Z {"firewall_name":"ANFW-Lab","availability_zone":"us-east-1a","event_timestamp":"1731587991","event":{"src_ip":"207.98.244.3","src_port":29011,"event_type":"alert","alert":{"severity":3,"signature_id":2,"rev":0,"signature":"","action":"blocked","category":""}},{"flow_id":2199880719188764,"dest_ip":"10.1.3.4","proto":"TCP","verdict":{"action":"drop"},"dest_port":8080,"pkt_src":"geneve encapsulation","timestamp":"2024-11-14T12:39:51.708889+0000","direction":"to_server"}}
```

For stateful domain list, I configured the following:

Step 1

Choose rule group type

Step 2

Describe rule group

Step 3

Configure rules

Step 4 - optional

Configure advanced settings

Step 5 - optional

Add tags

Step 6

Review and create

Configure rules

An AWS Network Firewall rule group is a reusable set of criteria for inspecting and handling network traffic.

Domain list rule [Info](#)

Allow or deny traffic based on the domain name list.

Domain names
List the domain names you want to inspect and either allow or deny.

www.google.com
search.google.com

Enter one domain name per line.

CIDR ranges
The source traffic CIDR ranges to inspect.

☒ **Default**
Use the CIDR range of the VPC where Network Firewall is deployed.

☐ **Custom**
Set your own list of CIDR ranges.

Protocols
The protocols to inspect.

☒ HTTP

☒ HTTPS

Action
Action to take when a request matches the domain names in this group.

☒ **Allow**

☐ **Deny**

Cancel

Previous

Next

This **Domain list** allows the domains added, and effectively denies all others except those in other rule groups associated with this policy.

Test the domain using the following command:

```
curl https://www.google.com --max-time 5
```

output:

```
sh-5.2$ curl http://www.google.com --max-time 5
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="en">
```

The output is successful since the domain name is specified in the approved list. One which is not approved is blocked as shown below:

```
sh-5.2$ curl http://google.com --max-time 5
curl: (28) Operation timed out after 5001 milliseconds with 0 bytes received
```

Lab 2: Protected VPC with Private Workload

This lab focused on enhancing the security of the environment by transitioning the public workload into a protected private subnet and strengthening firewall policies to control access further.

Steps Taken:

1.Setting Up the environment for lab 2.

Similar to Lab 1, the CloudFormation template set up the necessary infrastructure and resources, with a focus on updating the VPC configuration to support private access.

- Navigate to the CloudFormation console by following [this link](#). and select the **NetworkFirewallLab** stack radio button. Once the stack is selected click the **Update** button to update the stack.

Update stack

Prerequisite - Prepare template

Prepare template
Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.

☐ Use existing template
Proceed with the template you are already using for this stack.

☒ Replace existing template
Replace your existing template with a new template.

☐ Edit in Application Composer
Edit your template in a visual builder.

Specify template

A template is a JSON or YAML file that describes your stack's resources and properties.

Template source
Selecting a template generates an Amazon S3 URL where it will be stored.

☐ Amazon S3 URL

☒ Upload a template file

Upload a template file

ANFW_Lab2.yml

JSON or YAML formatted file

S3 URL: https://s3.us-east-1.amazonaws.com/cf-templates-1d7fhql18ja9-us-east-1/2024-05-06T023334.155Z6a9-ANFW_Lab2.yml

After it is updated, this is how the vpc resource map will look like:



2. Update route tables.

I updated the IGW-Ingress-Route-Table as it now needs to point to the new Protected Subnet instead of the Protected Workload Subnet. This is so incoming traffic to the Network Load Balancer will be routed to the firewall.

Select routes to modify it.

The screenshot shows the AWS Route Tables console. The 'Route tables (1/5)' list includes the 'IGW-Ingress-Route-Table' (rtb-0ac6f0c9b91f270c7) which is associated with the 'Inspection VPC IGW'. The 'Routes' tab for this route table is selected, showing a single route with a destination of '0.0.0.0/0' and a target of 'Inspection VPC IGW'.

Name	Route table ID	Explicit subnet associati...	Edge associations	Main	VPC	Own...
Protected-WebServer-Route-Table	rtb-0ac6f0c9b91f270c7	subnet-0ac6f0c9b91f270c7	-	No	vpc-0ac6f0c9b91f270c7	ANFW Inspectio...
-	rtb-0ac6f0c9b91f270c7	-	-	Yes	vpc-0ac6f0c9b91f270c7	ANFW Inspectio...
Firewall-Route-Table	rtb-0ac6f0c9b91f270c7	subnet-0ac6f0c9b91f270c7	-	No	vpc-0ac6f0c9b91f270c7	ANFW Inspectio...
-	rtb-0ac6f0c9b91f270c7	-	-	Yes	vpc-0ac6f0c9b91f270c7	ANFW Inspectio...
IGW-Ingress-Route-Table	rtb-0ac6f0c9b91f270c7	-	igw-0ac6f0c9b91f270c7	No	vpc-0ac6f0c9b91f270c7	ANFW Inspectio...

rtb-0ac6f0c9b91f270c7 / IGW-Ingress-Route-Table

Details

Route table ID	Main	Explicit subnet associations	Edge associations
rtb-0ac6f0c9b91f270c7	No	-	igw-0ac6f0c9b91f270c7 / Inspection VPC IGW

VPC
vpc-0ac6f0c9b91f270c7 | ANFW Inspection VPC

Change the destination as shown below:

VPC > Route tables > rtb-04fcd4a33f8b8b292 > Edit routes

Edit routes

Destination	Target	Status	Propagated
10.1.0.0/16	local	Active	No
10.1.2.0/28	Gateway Load Balancer Endpoint	Active	No

Use: "10.1.2.0/28"

0.0.0.0/0

0.0.0.0/8

0.0.0.0/16

0.0.0.0/24

0.0.0.0/32

072db5054424601cb

Cancel Preview Save changes

I update the **Protected-WebServer-Route-Table** to send all its Internet-bound traffic to the NAT Gateway in the protected subnet.

VPC > Route tables > rtb-0250ad70162a56d09 > Edit routes

Edit routes

Destination	Target	Status	Propagated
10.1.0.0/16	local	Active	No
0.0.0.0/0	NAT Gateway	Active	No

Use: "nat-0b43b08416780e673"

nat-0b43b08416780e673 (egress-natgw-NetworkFirewallLab)

Add route

Cancel Preview Save changes

3. Create protected subnet routes.

From the Route tables screen, click on the Create route table button on the top left of the screen. On the Create route table form set the Name to be Protected-Route-Table. Select ANFW Inspection VPC from the Select a VPC dropdown. Then, click on the Create route table button, edit the routes and add a subnet association as shown below:

This is to allow inbound connectivity to the protected web server in the protected subnet through the Network Load Balancer. Outbound connectivity should be established through the NAT Gateway.

Create route table Info

A route table specifies how packets are forwarded between the subnets within your VPC, the internet, and your VPN connection.

Route table settings

Name - *optional*

Create a tag with a key of 'Name' and a value that you specify.

Protected-Route-Table

VPC

The VPC to use for this route table.

vpc- (ANFW Inspection VPC) ▼

Tags

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key

Q Name X

Value - *optional*

Q Protected-Route-Table X

Remove

Add new tag

You can add 49 more tags.

Cancel

Create route table

VPC > Route tables > rtb- > Edit routes

Edit routes

Destination	Target	Status	Propagated
10.1.0.0/16	<div>local ▼</div> <div>Q local X</div>	Active	No
<div>Q 0.0.0.0/0 X</div>	<div>Gateway Load Balancer Endpoint ▼</div> <div>Q vpc- X</div>	-	No

Add route

Remove

Cancel

Preview

Save changes

VPC > Route tables > rtb- > Edit subnet associations

Edit subnet associations

Change which subnets are associated with this route table.

Available subnets (1/3)

Q Filter subnet associations

	Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID
<input type="checkbox"/>	Firewall Subnet	subnet-	10.1.1.0/28	-	rtb- / Firewall-Route-T...
<input type="checkbox"/>	Protected WebServer Subnet	subnet-	10.1.3.0/28	-	rtb- / Protected-WebSe...
<input checked="" type="checkbox"/>	Protected Subnet	subnet-	10.1.2.0/28	-	Main (rtb-)

Selected subnets

subnet- / Protected Subnet X

Cancel

Save associations

Testing the setup.

I tested the new setup with the private workload and confirm network connectivity via the Network Load Balancer (NLB).

I navigated to to the EC2 Dashboard. Then in the Resources window, as seen below, I select Load balancers.

Resources

EC2 Global view

You are using the following Amazon EC2 resources in the US West (Oregon) Region:

Instances (running)	1	Auto Scaling Groups	0	Dedicated Hosts	0
Elastic IPs	1	Instances	2	Key pairs	0
Load balancers	1	Placement groups	0	Security groups	3
Snapshots	0	Volumes	1		

I selected the NLB for this lab and copied the **DNS name** from the **Description** tab in the configuration details as seen below.

EC2 > Load balancers

Load balancers (1/1)

Actions

Create load balancer

Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Filter load balancers

< 1 >

<input checked="" type="checkbox"/>	Name	DNS name	State	VPC ID	Availability Zones	Type
<input checked="" type="checkbox"/>	NetworkFirewallLab-NLB	NetworkFirewallLab-NLB-8...	Active	vpc-	us-east-1a (use1-az2)	network

Load balancer: NetworkFirewallLab-NLB

Details

Listeners

Network mapping

Resource map - new

Security

Monitoring

Integrations

Attributes

Tags

Details

Load balancer type

Network

Status

Active

VPC

vpc-

IP address type

IPv4

Scheme

Internet-facing

Hosted zone

Availability Zones

subnet-east-1a (use1-az2)us-

Date created

May 5, 2024, 21:37 (UTC-05:00)

Load balancer ARN

arn:aws:elasticloadbalancing:us-east-1:

DNS name Info

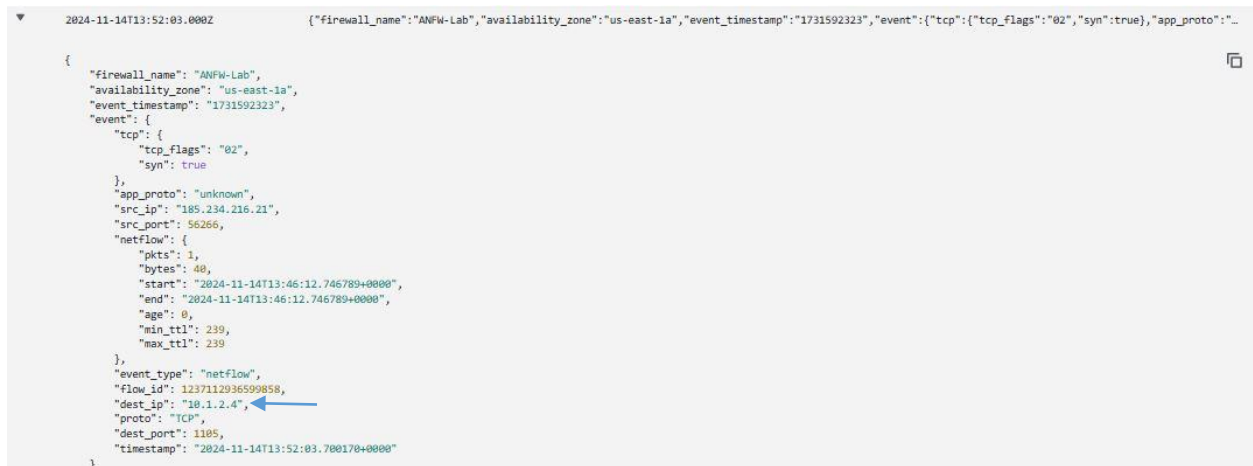
NetworkFirewallLab-NLB-Record.elb.us-east-1.amazonaws.com (A)

I copied it on my web browser and got the output as shown below:



After that I checked the logs generated from my activity on cloudwatch.

I can validate the correct logs are delivered as the the destination IP address changed from 10.1.3.4
To that of my NLB, 10.1.2.4.



4.Strengthen Firewall Policy.

I configured both Stateless rules and Stateful IPS rules to strengthen the configured firewall policy in **Lab 1** which consisted of Stateful 5-tuple rules and Stateful Domain list rules only.

Stateless rule groups evaluate packets in isolation, while stateful rule groups evaluate them in the context of their traffic workflow.

- **Stateless** – Defines standard, 5-tuple criteria for examining a packet on its own, with no additional context.
- **Stateful Suricata-compatible IPS rules** – Defines intrusion prevention system (IPS) rules in the rule group in an open source compatible format (e.g. Suricata, Snort).

Benefit of combining rules: Stateless rules are evaluated first, and can provide a coarse layer of security protecting you from accidentally making Stateful rules too permissive. Stateful IPS network traffic inspection enables identification of malicious activity from protocol anomalies to malware and botnets using signature-based detection. AWS Network Firewall IPS rules use an open source Suricata syntax with thousands of existing community maintained rulesets available.

Configure stateless rules.

First navigate to VPC → Firewalls → *Your Lab Firewall*. This will bring you to the Firewall Overview page. Select the Firewall policy settings tab, then select Create stateless rule group in the Actions tab on the top right of the Stateless rule groups section.

The screenshot shows the AWS Network Firewall console interface. At the top, there are three tabs: 'Firewall details', 'Firewall policy settings' (which is selected), and 'Monitoring'. Below the tabs, the 'Stateless default actions' section is visible, with an 'Edit' button. It shows two action settings: 'Actions for full packets' and 'Actions for fragmented packets', both set to 'Forward to stateful rule groups'. Below this is the 'Stateless rule groups (0)' section. It has a table with columns for 'Priority', 'Name', and 'Capacity'. The table is currently empty. To the right of the table, there is an 'Actions' dropdown menu with options: 'Create stateless rule group' (highlighted), 'Add unmanaged stateless rule groups', and 'Disassociate from policy'. Below the table, a message states 'No stateless rule groups' and 'Choose Add rule groups to add stateless rule groups to the policy.'

Add three rules as shown below:

Priority	Protocol	Source	Destination	Source port range	Destination port range	Action	Custom action	Masks	Flags
1	TCP	0.0.0.0/0	0.0.0.0/0	0:65535	80	Pass	-	-	-
2	TCP	0.0.0.0/0	0.0.0.0/0	80	0:65535	Pass	-	-	-
3	ICMP	0.0.0.0/0	0.0.0.0/0	-	-	Forward	-	-	-

Navigate to the **Stateless default actions** section and click **Edit**. Then save it after editing.

Stateless default actions

Fragmented packets

☒ Use the same actions for all packets

☐ Use different actions for full packets and fragmented packets

Rule action

☐ Pass

☒ Drop

☐ Forward to stateful rule groups

Publish metrics - *optional*

Publish a custom Amazon CloudWatch metric to monitor the usage of your stateless rule groups.

☐ Enable

Cancel

Save

Configure stateful IPS Rules as shown below:

i A firewall policy can be associated with multiple firewalls. Modifying a firewall policy affects all firewalls that reference it. To use rule groups that are managed for you, see [AWS Partner Network \(APN\) integrations](#). [↗](#)

Step 1

[Choose rule group type](#)

Step 2

Describe rule group

Step 3

[Configure rules](#)

Step 4 - optional

[Configure advanced settings](#)

Step 5 - optional

[Add tags](#)

Step 6

[Review and create](#)

Describe rule group

Name and describe your rule group so you can easily identify it and distinguish it from other resources.

Rule group details

Name

Enter a name for the rule group that's unique within your stateful rule groups.

The name must have 1-128 characters. Valid characters: a-z, A-Z, 0-9 and - (hyphen). The name can't start or end with a hyphen, and it can't contain two consecutive hyphens.

Description - optional

This description appears when you view this rule group's details. It can help you quickly identify what your rule group is used for.

The description can have 0-256 characters.

Capacity

The number of rules you expect to have in this rule group during its lifetime. You can't change capacity after rule group creation, so leave room to grow.

The capacity must be greater than or equal to 1 and less than 30,000.

[Cancel](#)[Previous](#)[Next](#)

For step 3, scroll down to the **Suricata compatible rule string** section and copy and paste the following Suricata IPS rule into the text box:

```
alert icmp any any -> any any (msg:"ICMP traffic detected"; flow:to_server; sid: 889;)
```

Suricata compatible rule string [Info](#)

Suricata is an open source network IPS that includes a standard rule-based language for traffic inspection.

Suricata compatible rule string

[Copy rules](#)[Cancel](#)[Previous](#)[Next](#)

The Suricata IPS rule will enable the firewall to log an alert in response to any ICMP traffic with the following message: "ICMP traffic detected".

To test, connect to EC2 Instance (Test Workload) via Session Manager as previously done in the lab. Then copy and paste the following ping command into your secure browser-based shell window.

```
ping -c 5 www.amazon.com
```

```
sh-5.2$ ping -c 5 www.amazon.com
PING d3ag4hukkh62yn.cloudfront.net (23.62.25.178) 56(84) bytes of data.

--- d3ag4hukkh62yn.cloudfront.net ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4123ms

sh-5.2$
```

The pings timed out. This is because a stateless **alert** rule will generate an alert and then pass the traffic to the stateful rules engine. Since the stateful rules engine has no rules defined for ICMP traffic, it will drop the packet in accordance with its **Drop established** default action.

Below is the log activity from cloud watch on the traffic generated above.

```
2024-11-14T15:07:28.000Z {"firewall_name": "ANFW-Lab", "availability_zone": "us-east-1a", "event_timestamp": "1731596848", "event": {"icmp_type": 8, "src_ip": "47.254.245.116", "src_port": 0, "event_type": "alert", "alert": {"severity": 3, "signature_id": 889, "rev": 0, "signature": "ICMP traffic detected", "action": "allowed", "category": ""}}, {"flow_id": 259270992178239, "dest_ip": "10.1.2.4", "proto": "ICMP", "verdict": {"action": "drop"}}, {"icmp_code": 0, "dest_port": 0, "pkt_src": "geneve encapsulation", "timestamp": "2024-11-14T15:07:28.191438+0000", "direction": "to_server"}]
```

Key Take away in lab 1.

Through this lab, I learned the basics of AWS Network Firewall and how to enforce access control for public-facing resources. The integration with CloudWatch provided insights into network traffic and potential security events.

Key take away in lab 2.

Lab 2 reinforced the importance of a layered security approach by isolating workloads in private subnets and tightening firewall rules. It highlighted the flexibility of AWS Network Firewall policies to adapt as security needs evolve.

Conclusion.

Completing Labs 1 and 2 of the AWS Hands-on Network Firewall Workshop gave me hands-on experience in setting up and securing AWS environments with a network firewall. Through these labs, I gained practical skills in configuring and logging firewall traffic, managing route tables, and applying security best practices to both public and private workloads. Additionally, I learned how to use a **CloudFormation template** to automate the creation of essential AWS resources, streamlining the setup process and ensuring consistency across environments. I look forward to exploring more advanced topics in AWS Network Firewall to deepen my knowledge and further secure cloud-based applications.