

Module 2: Security Management in AWS

Demo Document 7

edureka!

edureka!

© Brain4ce Education Solutions Pvt. Ltd.

Title: Configure WAF to Protect Website from Attacks

Use Case

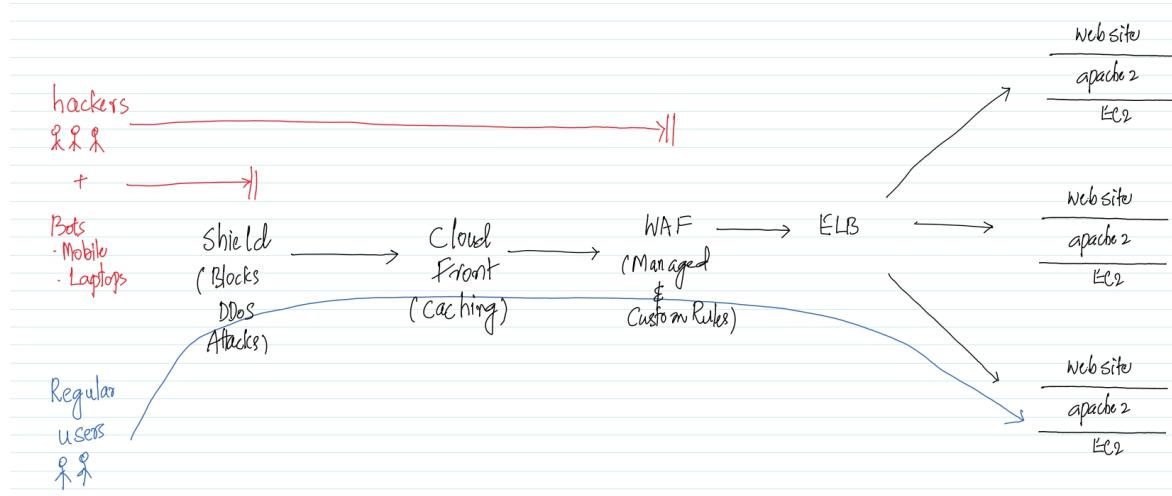
With the proliferation of the internet, many companies are trying to become global. For example, Amazon has its presence in many countries worldwide. But, with expansion comes a greater risk of hacking and data theft. The application stack has become so complex that there is always some sort of vulnerability in the different layers (OS, web, DB, network, storage, balancing, etc.), and the hackers try to exploit them for financial and other benefits. Companies have to be on their toes to keep the hackers at bay while making sure only the legitimate users have access to the website and other applications.

Protecting the publicly exposed services like websites and REST/Web Services based interfaces consumes a lot of effort, time, and money for the organizations. However, AWS provides a few managed services like AWS Shield and AWS WAF (Web Application Firewall) that help in mitigating the security risks to a greater degree. This feature is especially valuable to cash-strapped companies who have a variety of costs to cover.

AWS WAF is a service for protecting the websites and the APIs from some of the common attacks like SQL Injection, XSS (Cross-Site Scripting), SSRF (Server Side Request Forgery) and including those published in OWASP (Open Web Application Security Project). However, it may sometimes call for the need to apply custom rules to meet the application requirements.

AWS Shield provides support against the DDOS (Distributed Denial of Service) attacks. These are the types of attacks when invalid requests are sent in bulk, while the legitimate user requests are throttled for resources. AWS Shield comes in two editions: Standard, which is free of cost, and Advanced, which costs around 3000 USD a month with a commitment of 1 year full-fledged support. By default, Shield Standard is turned on CloudFront and Route53, and there is nothing we have to do except using a CloudFront distribution, which automatically comes with Shield Standard.

Architectural diagram:



In this use case, we will create an EC2 with an SSRF vulnerability application deployed on it. Then, we will try to mitigate the vulnerability using WAF. For this, we need to create an Application Load Balancer and integrate it with WAF.

AWS Services: WAF, ELB, EC2

Solution Steps:

Step 1: Go to the IAM Management Console and create a Role for EC2 with AmazonS3ReadOnlyAccess Policy attached to it. Make sure the name of the role is **Role4EC2-S3RO**.

The screenshot shows the AWS IAM Roles page. On the left, there's a sidebar with navigation links like Dashboard, Access management, Roles, Policies, etc. The main area shows a 'Summary' for the role 'Role4EC2-S3RO'. Key details include:

- Role ARN: arn:aws:iam::304000509264:role/Role4EC2-S3RO
- Role description: Allows EC2 instances to call AWS services on your behalf.
- Instance Profile ARNs: arn:aws:iam::304000509264:instance-profile/Role4EC2-S3RO
- Path: /
- Creation time: 2020-10-02 10:57 UTC+0530
- Last activity: 2020-10-02 11:09 UTC+0530 (Yesterday)
- Maximum session duration: 1 hour

The 'Permissions' tab is active, showing one policy applied: 'AmazonS3ReadOnlyAccess'. There are tabs for Trust relationships, Tags, Access Advisor, and Revoke sessions. A button to 'Add inline policy' is visible. At the bottom, there are links for Feedback, English (US), and other AWS links.

Step 2: Create an Ubuntu EC2 with the below details and connect to it:

- t2.micro
- Security Group (22/SSH and 80/HTTP) inbound
- The above role attached
- And, Keypair attached for authentication

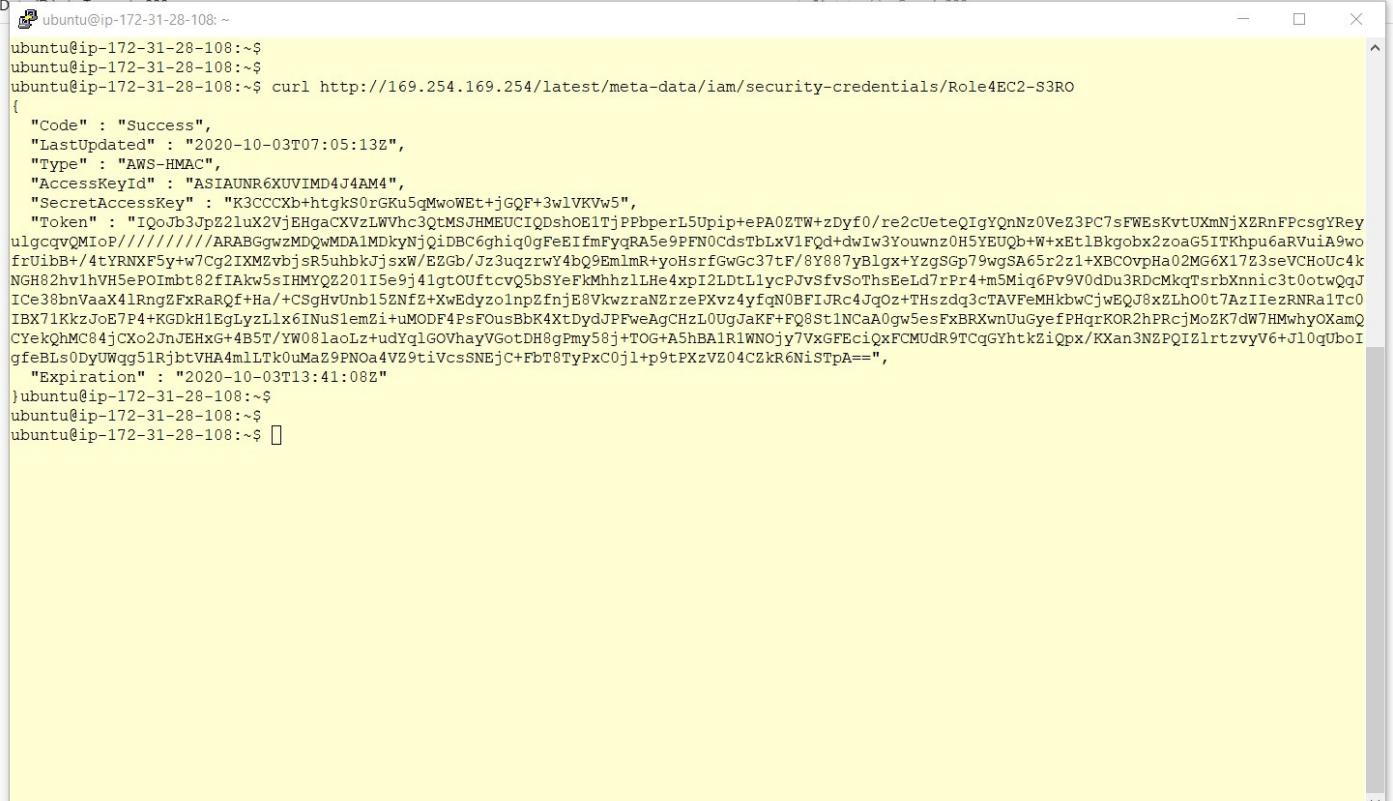
Security Management in AWS

The screenshot shows the AWS EC2 Instances details page for an instance named 'i-0ecfd8a1ba5ad3410'. The instance is running in the 'us-east-1b' availability zone with a Public DNS of 'ec2-34-207-208-218.compute-1.amazonaws.com'. It has an IPv4 address of '34.207.208.218' and an Elastic IP of 'ec2-34-207-208-218.compute-1.amazonaws.com'. The instance type is 't2.micro' and it was launched via a VPC ID 'vpc-3bc6e341'. The platform is 'Ubuntu' and the usage operation is 'RunInstances'. The instance has a key pair named 'my-keypair' and an IAM role named 'Role4EC2-S3RO'. There are no scheduled events or security groups listed.

Description	Value	Description	Value
Instance ID	i-0ecfd8a1ba5ad3410	Public DNS (IPv4)	ec2-34-207-208-218.compute-1.amazonaws.com
Instance state	running	IPv4 Public IP	34.207.208.218
Instance type	t2.micro	IPv6 IPs	-
Finding	Opt-in to AWS Compute Optimizer for recommendations. Learn more	Elastic IPs	-
Private DNS	ip-172-31-28-108.ec2.internal	Availability zone	us-east-1b
Private IPs	172.31.28.108	Security groups	AllowHTTP, AllowSSH, view inbound rules, view outbound rules
Secondary private IPs	-	Scheduled events	No scheduled events
VPC ID	vpc-3bc6e341 (Default)	AMI ID	ubuntu/images/hvm-ssd/ubuntu-bionic-18.04-amd64-server-20200810 (ami-0bcc094591f354be2)
Platform	Ubuntu	Subnet ID	subnet-67b9612a
Platform details	Linux/UNIX	Network interfaces	eth0
Usage operation	RunInstances	IAM role	Role4EC2-S3RO
Source/dest. check	True	Key pair name	my-keypair
T2/T3 Unlimited	Disabled		

Step 3: Execute the below command on the EC2 and notice the Access Keys are displayed as part of the **AccessKeyId** and **SecretAccessKey** fields. This is displayed because it is part of the EC2 instance metadata and can be retrieved, as mentioned [here](#).

```
curl http://169.254.169.254/latest/meta-data/iam/security-credentials/Role4EC2-S3RO
```



```
ubuntu@ip-172-31-28-108:~$ curl http://169.254.169.254/latest/meta-data/iam/security-credentials/Role4EC2-S3RO
{
    "Code" : "Success",
    "LastUpdated" : "2020-10-03T07:05:13Z",
    "Type" : "AWS-HMAC",
    "AccessKeyId" : "ASIAUNR6XUVIMD4J4AM4",
    "SecretAccessKey" : "K3CCCXb+htgkS0rGKu5qMwoWEt+jGQF+3w1VKVw5",
    "Token" : "IQoJb3Jpz2luX2VjEHgacXVzLWvhc3QtMSJHMEUCiQDshoE1TjPPbperL5Upip+ePA0ZTw+zDyf0/re2cUeteQIgYQnNz0VeZ3PC7sFWEsKvtUXmNjXZRnFPcsgYRey
ulgccgvMIoP//////////ARABGgwzMDQwMDA1MDkyNjQiBc6ghiq0gFeElfmFyqRA5e9FNOCdsTblxV1FQd+dwIw3Youwnz0H5YEUqb+W+xEt1Bkgobx2zoaG51TKhpu6aRVuiA9wo
frUiBt+/4tYRNxF5y+w7Cq2lXM2vbjsR5uhbkJjsxWEZGb/Jz3uqzrwY4b9EmlmR+yoHsrifGwgC37tF/8Y887yb1gx+YzgSGp79wgsA65r2z1+XBCCvpHa02MG6X17Z3seVChouC4k
NGH82hv1hVH5ePoImbt82f1Akw5sIHMYQ2201I5e9j41gtOUftcvq5bSYefKmhzhz1LHe4xpI2LDtLlycPJvSfvSoThsEeLd7rPr4+m5Miq6Pv9V0dDu3RDcMkqTsrbXnnic3t0otwQqJ
ICe38bnVaaX41rnq2FxRaRqf+Ha/+CSghvUnb15ZNf2+XwEdyzolnpZfnjB8VKwzraNZrreFxz4yfgN0BFIJRC4JgOz+THszdq3ctAVFeMhkbcjweQQJ8xZLh00t7AzIEzRNra1fc0
IBX71KzJ0e7P4+KGDKh1EqLyzzlx6INuS1emZi+uMODF4PsFOusBbK4xtDydJFweAgCHzLOUgJaKF+FQ8St1NCaA0gw5esFxBRXnUuGyeFPHqrKOR2hPRcjMoZK7dW7HMwhyoXamQ
CyekQhmC84jCxo2JnJEhxG+4B5T/YW08laolz+udYqlGOvhayVGotDH8gPmy58j+TOG+A5hBA1R1WNOjy7VxGFECiQxFCMUdR9TCqGYhtkZiQpx/KXan3NZPQIZlrtzvyV6+J10qUb0I
gfeBLs0DyUWqg51RjbtVHA4mlLTk0uMa29PNoa4VZ9tiVcssSNEjC+FbT8TyPxC0j1+p9tPXzVZ04CZkR6NiSTpA==",
    "Expiration" : "2020-10-03T13:41:08Z"
}ubuntu@ip-172-31-28-108:~$
```

Step 4: Execute the below commands in the EC2 to install Ruby and Sinatra, a DSL (Domain Specific Language) to create web applications in Ruby. Installation of Sinatra will take a few minutes to complete.

```
sudo apt-get update  
sudo apt-get install ruby -y  
sudo gem install sinatra
```

Step 5: On the EC2, create a file **server.rb** with the below content. This is what the program does

- Creates an HTTP endpoint (webserver) on port 80
- Takes a URL as an input (query parameter) via HTTP
- Invokes the same URL via the open function
- Responds back to the response from the URL

```
require 'sinatra'  
require 'open-uri'  
  
get '/' do  
  format 'RESPONSE: %s', open(params[:url]).read  
end
```

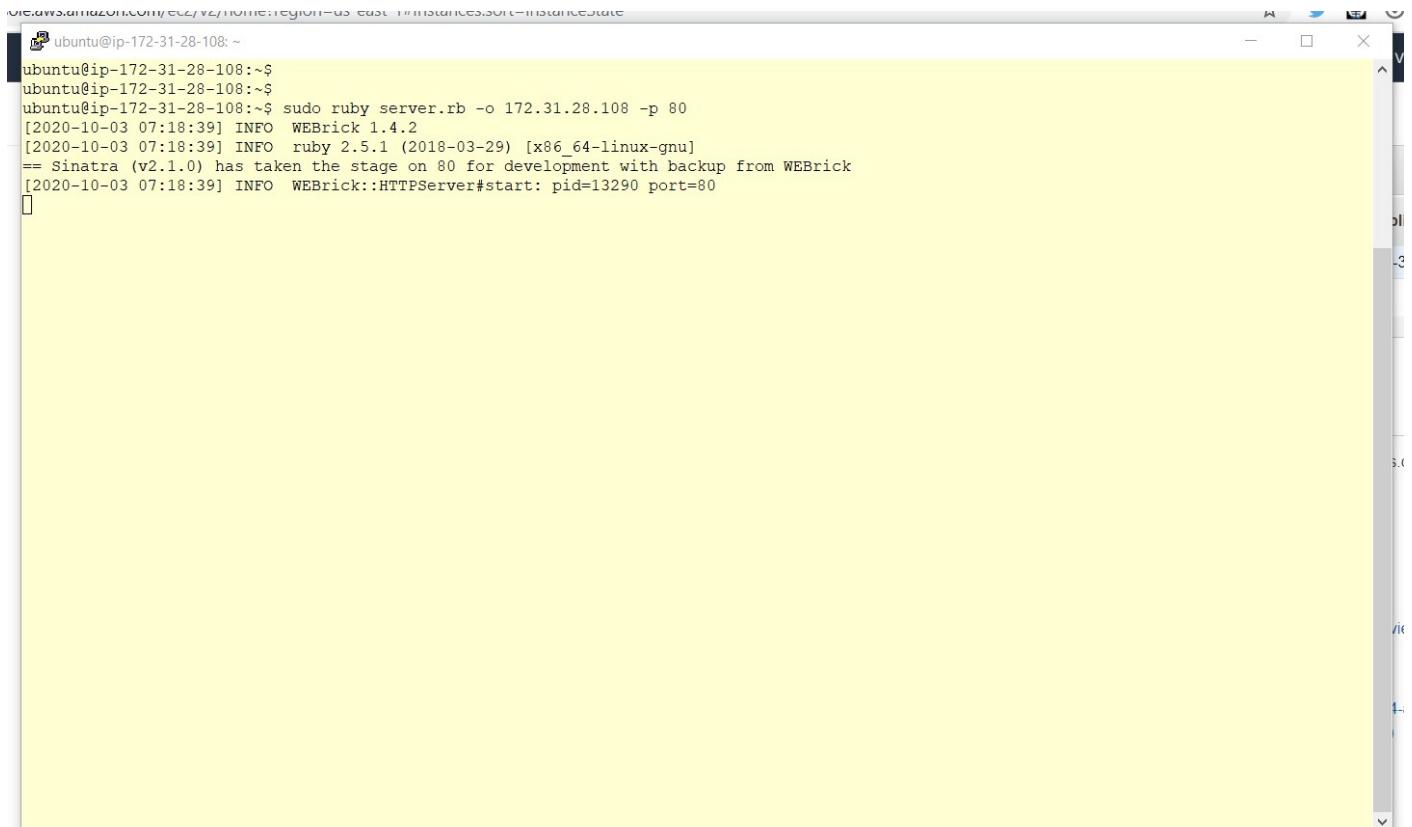


The screenshot shows a terminal window titled 'sole.aws.amazon.com/ec2/v2/home?region=us-east-1#instances;sort=instancestate'. The terminal session is as follows:

```
ubuntu@ip-172-31-28-108: ~  
ubuntu@ip-172-31-28-108:~$  
ubuntu@ip-172-31-28-108:~$ cat server.rb  
require 'sinatra'  
require 'open-uri'  
  
get '/' do  
  format 'RESPONSE: %s', open(params[:url]).read  
end  
ubuntu@ip-172-31-28-108:~$  
ubuntu@ip-172-31-28-108:~$  
ubuntu@ip-172-31-28-108:~$
```

Step 6: Run the program using the below command. Make sure to replace the IP address with the private IP address of the EC2. The program will go in an infinite loop, and if any other command has to be run on the EC2, then a new session has to be opened.

`sudo ruby server.rb -o 172.31.28.108 -p 80`



The screenshot shows a terminal window titled "Ubuntu@ip-172-31-28-108:~". The user runs the command `sudo ruby server.rb -o 172.31.28.108 -p 80`. The output shows the server starting up with WEBrick 1.4.2, Sinatra v2.1.0, and a message indicating it has taken the stage on port 80. The terminal window has a yellow background and a dark border.

```
ubuntu@ip-172-31-28-108:~$ sudo ruby server.rb -o 172.31.28.108 -p 80
[2020-10-03 07:18:39] INFO  WEBrick 1.4.2
[2020-10-03 07:18:39] INFO  ruby 2.5.1 (2018-03-29) [x86_64-linux-gnu]
== Sinatra (v2.1.0) has taken the stage on 80 for development with backup from WEBrick
[2020-10-03 07:18:39] INFO  WEBrick::HTTPServer#start: pid=13290 port=80
```

Step 7: Open the below URL in a browser and note that the AccessKeyId and the SecretAccessKey are displayed, as shown below. This happens because the program executed above will take a URL, invoke it, and send it back. There is no check on what the URL is and the internal network can be probed. Make sure to replace 34.207.208.218 with the Public IP of the EC2 instance.

<http://34.207.208.218?url=http://169.254.169.254/latest/meta-data/iam/security-credentials/Role4EC2-S3RO>

```
RESPONSE: { "Code": "Success", "LastUpdated": "2020-10-03T07:05:13Z", "Type": "AWS-HMAC", "AccessKeyId": "ASIAUNR6XUVIMD4J4AM4", "SecretAccessKey": "K3CCCXb+htgkS0rGKu5qMwoWEt+jGQF+3wIVKw5", "Token": "IQoJb3jPZ2huX2VjEHgaCXVzLWVhc3QtMSJHMEUCIQDshOEIJPPbperLSUpip+ePA0ZTW+zDyf0/re2cUeteQIgYQnNz0VeZ3PC7sFWEsKvtUXmNjXZRnFPcsgYReylgcqvQMIIoP//////////ARABGgwzMDQwMDA1MDkyNjQiDIExpiration": "2020-10-03T13:41:08Z" }
```

Step 8: Go to the EC2 Management Console and create a Target Group called MyTG and attach the above EC2.

The screenshot shows the AWS EC2 Target Groups console. On the left sidebar, under 'Load Balancing', 'Target Groups' is selected. In the main area, a new target group named 'MyTG' is being created. The 'Basic configuration' section includes fields for 'Target type' (set to 'instance'), 'Protocol' (set to 'HTTP : 80'), 'VPC' (set to 'vpc-3bc6e341'), and 'Load balancer' (set to '-'). The 'Targets' tab is active, showing a single registered target: an Amazon Linux instance with ID 'i-0ecfd8a1ba5ad3410', port '80', and status 'unused'. A note below the table states 'Target group is not configured to receive traffic from the load balancer...'. The bottom of the screen shows standard AWS navigation links for Feedback, English (US), Privacy Policy, and Terms of Use.

Step 9: In the same EC2 Management Console, create an Application Load Balancer and specify the Target Group as the default forwarding destination, as shown below. Under the listeners Tab, it should appear, as shown below.

The screenshot shows the AWS EC2 Application Load Balancers console. On the left sidebar, under 'Instances', 'Launch Templates' is selected. In the main area, a new load balancer named 'MyALB' is being created. The 'Listeners' tab is active, showing a single listener for 'HTTP : 80' with a security policy of 'N/A' and an SSL certificate of 'N/A'. The 'Default: forwarding to MyTG' rule is listed. The bottom of the screen shows standard AWS navigation links for Feedback, English (US), Privacy Policy, and Terms of Use.

Step 10: Note down the DNS name of the Load Balancer.

Name	DNS name	State	VPC ID	Availability Zones	Type	Created At
MyALB	MyALB-641995498.us-east-1.elb.amazonaws.com	active	vpc-3bc6e341	us-east-1c, us-east-1f, ...	application	October 3

Step 11: Open the below URL in a browser. The Access Keys are displayed in the browser as there is still a vulnerability in the application. These Access Keys can be used to read the data from S3 as we have given the Read-Only permissions to the EC2 via the IAM Role. Make sure to replace elb.com with the DNS name of the Load Balancer.

<http://elb.com?url=http://169.254.169.254/latest/meta-data/iam/security-credentials/Role4EC2-S3RO>

```
RESPONSE: { "Code" : "Success", "LastUpdated" : "2020-10-03T07:25:48Z", "Type" : "AWS-HMAC", "AccessKeyId" : "ASIAUNR6XUVIOFACRCM5", "SecretAccessKey" : "liNorVtq2eJ4JdaqhjYGM5A7cQJxRV/VgHgH18kS", "Token" : "IQoJb3JpZ2luX2VjEHgaCXVzLWVhc3QtMSJHMEUCIQCgwk7axrkA7fl7msB1wr9kar2Ki95bIdchYz462qiMwwIgbfxdvbm+/MuAzIrjZTBxHfW/pm7MgiXtPpZidL+IrJ0qvQMiof//////////ARABGgwzMDQwMDA1MDkyNjQiDBPzL" "Expiration" : "2020-10-03T13:45:48Z" }
```

Step 12: Now, we will try to mitigate the vulnerability in the application using a WAF. Go to the WAF Management Console and click on **Create web ACL**.

The screenshot shows the AWS WAF Management Console. On the left, there's a sidebar with 'AWS WAF' selected. The main content area has a dark header 'AWS WAF' and a sub-header 'Protect your web applications from common web exploits'. Below this is a paragraph about AWS WAF's purpose. To the right, there's a 'Get started with AWS WAF' section with a 'Create web ACL' button. A modal window titled 'Pricing (US)' is open, listing prices: '\$5 per web ACL per month (prorated hourly)', '\$1 per rule per month (prorated hourly)', and '\$0.6 per million request processed'. At the bottom, there are links for 'Feedback', 'English (US) ▾', '© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.', 'Privacy Policy', and 'Terms of Use'.

Step 13: Give a Name and provide some description, as shown below.

The screenshot shows the 'Describe web ACL and associate it to AWS resources' step in the WAF console. On the left, a sidebar lists steps: Step 1 (selected), Step 2, Step 3, Step 4, and Step 5. The main form is titled 'Web ACL details' and contains fields for 'Name' (MyCRM-ACL), 'Description - optional' (ACL for the CRM Application), 'CloudWatch metric name' (MyCRM-ACL), 'Resource type' (set to 'Regional resources'), and 'Region' (set to 'US East (N. Virginia)'). At the bottom, there are links for 'Feedback', 'English (US) ▾', '© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.', 'Privacy Policy', and 'Terms of Use'.

Step 14: In the same screen, click on **Add AWS resources**.

The screenshot shows the AWS CloudFront ACL configuration page. The 'Associated AWS resources - optional' section is visible, containing a search bar and a table with columns for Name, Resource type, and Region. The table displays the message 'No results'.

Name	Resource type	Region
No results There are no results to display		

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

Step 15: Make sure to select **Application Load Balancer** and **MyALB** from the previously created Load Balancer from the list. Click on **Add**.

The screenshot shows the 'Add AWS resources' modal. Under 'Resource type', 'Application Load Balancer' is selected. In the 'Select the resources you want to associate with the web ACL' section, 'Name' and 'MyALB' are checked. The 'Add' button is highlighted at the bottom right.

Name
MyALB

At the bottom right of the modal, there are 'Cancel' and 'Add' buttons.

Step 16: Click on Next.

Step 3
Set rule priority

Step 4
Configure metrics

Step 5
Review and create web ACL

Description - optional
ACL for the CRM Application

CloudWatch metric name
MyCRM-ACL

Resource type
 Regional resources (Application Load Balancer, API Gateway, AWS AppSync)

Region
US East (N. Virginia)

Associated AWS resources - optional

<input type="checkbox"/>	Name	Resource type	Region
<input type="checkbox"/>	MyALB	Application Load Balancer	US East (N. Virginia)

Cancel **Next**

Step 17: Click on Add rules → Add my own rules and rule groups.

Step 1
Describe web ACL and associate it to AWS resources

Step 2
Add rules and rule groups

Step 3
Set rule priority

Step 4
Configure metrics

Step 5
Review and create web ACL

Add rules and rule groups

A rule defines attack patterns to look for in web requests and the action to take when a request matches the patterns. Rule groups are reusable collections of rules. You can use managed rule groups offered by AWS and AWS Marketplace sellers. You can also write your own rules and use your own rule groups.

Rules

If a request matches a rule, take the corresponding action. The rules are prioritized in order they appear.

<input type="button"/> Edit	<input type="button"/> Delete	Add rules ▲
<input type="checkbox"/>	Name	Action
<input type="checkbox"/>	Add managed rule groups	
<input type="checkbox"/>	Add my own rules and rule groups	

No rules.
You don't have any rules added.

Web ACL rule capacity units used
The total capacity units used by the web ACL can't exceed 1500.
0/1500 WCLUs

Default web ACL action for requests that don't match any rules

Default action

Step 18: Make sure to select the **Rule builder** and specify the rule name as **RuleForSSRF** and type of rule as **Regular rule**.

The screenshot shows the AWS WAF interface for creating a web ACL. On the left, there's a sidebar with steps: Step 1 (Describe web ACL and associate it to AWS resources), Step 2 (Add rules and rule groups: Add my own), Step 3 (Set rule priority), Step 4 (Configure metrics), and Step 5 (Review and create web ACL). The main area is titled "Add my own rules and rule groups". Under "Rule type", the "Rule builder" option is selected. In the "Rule" section, the "Name" field contains "RuleForSSRF" and the "Type" dropdown is set to "Regular rule". There are tabs for "Rule visual editor" and "Rule JSON editor". A note at the bottom says: "You can use the JSON editor for complex statement nesting, for example to nest two OR statements inside an AND statement. The visual editor handles one level of nesting. For web ACLs and rule groups with complex nesting, the visual editor is disabled." A "Validate" button is also present.

Step 19: In the **If a request**, select **matches the statement** and then select

Query string for Inspect
Contains string for Match type
169.254.169.254 for String to match

The screenshot shows the "If a request" configuration screen. It has a dropdown menu showing "matches the statement". Below it, the "Statement" section is expanded. It includes fields for "Inspect" (set to "Query string"), "Match type" (set to "Contains string"), and "String to match" (set to "169.254.169.254"). There's also a "Text transformation" section with a note about AWS WAF applying transformations before evaluating the request, and a dropdown set to "None". A "Add text transformation" button is available. At the bottom, there's a "Then" section which is currently collapsed.

Step 20: Under the **Action**, select **Block**. Click on **Add rule**.

The screenshot shows the AWS WAF Rule creation interface. In the 'Match type' section, 'Contains string' is selected with the value '169.254.169.254'. In the 'Action' section, 'Block' is selected. At the bottom right, there are 'Cancel' and 'Add rule' buttons.

Step 21: The rule will be created, as shown below. Click on **Next**.

The screenshot shows the 'Add rules and rule groups' step in the AWS WAF wizard. It displays a table of rules with one entry: 'RuleForSSRF' with capacity 10 and action Block. Below the table, it shows 'Web ACL rule capacity units used' at 10/1500 WCUs. At the bottom, it shows the 'Default web ACL action for requests that don't match any rules' set to Block. Navigation buttons 'Cancel', 'Previous', and 'Next' are at the bottom right.

Step 22: There is only one rule that needs to be set on priority. Click on **Next**.

Step 1
Describe web ACL and associate it to AWS resources

Step 2
Add rules and rule groups

**Step 3
Set rule priority**

Step 4
Configure metrics

Step 5
Review and create web ACL

Rules
If a request matches a rule, take the corresponding action. The rules are prioritized in order they appear.

Name	Capacity	Action
RuleForSSRF	10	Block

Cancel Previous **Next**

Step 23: Click on **Next**.

Step 1
Describe web ACL and associate it to AWS resources

Step 2
Add rules and rule groups

Step 3
Set rule priority

**Step 4
Configure metrics**

Step 5
Review and create web ACL

Amazon CloudWatch metrics
CloudWatch metrics allow you to monitor web requests, web ACLs, and rules.

Rules CloudWatch metric name

RuleForSSRF

Request sampling options
If you disable request sampling, you can't view requests that match your web ACL rules.

Options

- Enable sampled requests
- Disable sampled requests
- Enable sampled requests with exclusions

Cancel Previous **Next**

Step 24: Review the web ACL and click on **Create web ACL**.

Step 1
Describe web ACL and associate it to AWS resources

Step 2
Add rules and rule groups

Step 3
Set rule priority

Step 4
Configure metrics

Step 5
Review and create web ACL

Review and create web ACL

Step 1: Describe web ACL and associated AWS resources

Web ACL details

Name	Scope
MyCRM-ACL	REGIONAL
Description	Region
ACL for the CRM Application	us-east-1
CloudWatch metric name	
MyCRM-ACL	

Steps 2 and 3: Add rules and set rule priority

Rules

If a request matches a rule, take the corresponding action. The rules are prioritized in order they appear.

Name	Capacity	Action
RuleForSSRF	10	Block

Feedback English (US) ▾ © 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Web ACL rule capacity units used
The total capacity units used by the web ACL can't exceed 1500.
10/1500 WCU's

Default web ACL action for requests that don't match any rules

Default action
 Allow
 Block

Step 4: Configure metrics

Amazon CloudWatch metrics

Rules	CloudWatch metric name
RuleForSSRF	RuleForSSRF

Sampled requests

Sampled requests
Enabled

Sampled requests for web ACL default actions
Enabled

Cancel Previous **Create web ACL**

Feedback English (US) ▾ © 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Step 25: The Web ACL would be created, as shown below.

The screenshot shows the AWS WAF & Shield console. On the left, there's a sidebar with 'WAF & Shield' selected. Under 'AWS WAF', 'Web ACLs' is also selected. The main area shows a table titled 'Web ACLs' with one entry: 'MyCRM-ACL'. The table has columns for 'Name', 'Description', and 'ID'. The 'Name' column shows 'MyCRM-ACL', 'Description' shows 'ACL for the CRM Application', and 'ID' shows 'd0440334-627b-450c-b72c-f432fb997538'. At the top right of the table, there are buttons for 'Copy ARN', 'Delete', and 'Create web ACL'. The status bar at the bottom indicates 'Feedback English (US) © 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. Privacy Policy Terms of Use'.

Step 26: Open the below URL in the browser, and **403 Forbidden** is displayed in the browser. This happens because 169.254.169.254 is there the URL and this is blocked by the WAF. Make sure to replace elb.com with the DNS name of the Load Balancer.

`http://elb.com?url=http://169.254.169.254/latest/meta-data/iam/security-credentials/Role4EC2-S3RO`

403 Forbidden

Step 27: FYI, before deleting the ACL, we need to make sure that the **Associated AWS resources** are deleted, or else AWS won't allow the ACL to be deleted.

The screenshot shows the AWS WAF & Shield console. On the left, there's a sidebar with options like 'Getting started', 'Web ACLs' (which is selected and highlighted in orange), 'IP sets', 'Regex pattern sets', 'Rule groups', 'AWS Marketplace', 'Switch to AWS WAF Classic', 'AWS Shield', and 'AWS Firewall Manager'. The main area shows a breadcrumb path: AWS WAF > Web ACLs > MyCRM-ACL. Below the path, the title 'MyCRM-ACL' is displayed. There are four tabs: 'Overview' (disabled), 'Rules' (disabled), 'Associated AWS resources' (selected and highlighted in orange), and 'Logging and metrics'. To the right of the tabs is a button 'Download web ACL as JSON'. Under the 'Associated AWS resources' section, there's a search bar 'Find associated AWS resources' and a table with one row. The table has columns for 'Name', 'Resource type', and 'Region'. The single entry is 'MyALB' (Application Load Balancer) in the US East (N. Virginia) region. There are buttons 'Disassociate' and 'Add AWS resources' at the top of the table, along with navigation arrows and a refresh icon. At the bottom of the page, there are links for 'Feedback', 'English (US)', '© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.', 'Privacy Policy', and 'Terms of Use'.

Conclusion:

We had a vulnerability in the application, and it was displaying the Access Keys in the browser, which can be exploited to get the data from S3. Although there is still a vulnerability in the application, we mitigated it using a WAF custom rule.