ENPM665-MIDTERM

E-Commerce Platform Security Assessment Report

Rakshana Muthukumar [redacted]
Bhavna Kumari [redacted]
Harshit Dharani [redacted]
Semileniola Bernice Salako [redacted]
Priyanka Nateshan Iyer [redacted]

Contents

Executive Summary 4
1.1 Project Objectives 4
1.2 Summary of Findings 4
Security Report 5
2.1 Vulnerability Assessment Report 6
2.1.1 SQL Injection 6
2.1.2 Broken Access Control 8
2.1.3 Reflective Cross Site Scripting (XSS) 9
2.1.4 Sensitive Data Exposure 10
2.1.5 Password Brute Force 13
2.1.6 Verbose Error Exposure 14
2.1.7 Improper Input Validation 16
2.2 Data Security Assessment Report 17
2.2.1 Database Credentials and Storage 17
2.2.2 Encryption at Rest 18
2.2.3 Encryption at Transit 19
2.2.4 No Data Leak Prevention (DLP) 20
2.2.5 Secrets Management 21
2.3 Virtual Machine Vulnerability Report 22
2.3.1 Unrestricted Sudo Access for ec2-user with No Password 22
2.3.2 EBS Volume Not Encrypted 23
2.3.3 EC2 Termination Protection Not Enabled 24
2.3.4 IAM Role Not Attached to EC2 Instance 24
2.3.5 SSH Publicly Accessible + SSH Version Disclosure 26
2.3.6 Usage of Vulnerable Component 27
2.3.7 Weak MariaDB Password + Weak Password Policy 28

2.4 Network Security Assessment Report 29
2.4.1 SSH Open to 0.0.0.0/0 29
2.4.2 Public Subnet and IP 30
2.4.3 Lack of Network Segmentation 32
2.4.4 Lack of DDoS Protection 33
2.4.5 Absence of VPC Flow Logs or Logging 34
2.4.6 Absence of Load Balancer 35
2.4.7 No Intrusion Detection/Prevention System 36
2.5 Disaster Recovery Assessment Report 37
2.5.1 Single Availability Zone 37
2.5.2 No Automated Backup Plans Created 38
2.5.3 No Load Balancers 39
2.5.4 Installation of MariaDB on an EC2 instance instead of AWS RDS . 40
2.5.5 EC2 Instance does not have a Snapshot Policy 41
2.5.6 No Redundancy in the Network 42
2.5.7 No Disaster Recovery Documentation Exists 44
2.6 Conclusion 46

Executive Summary

1.1 Project Objectives

Group 4's primary objective was to assess the security vulnerabilities in the e-commerce application's cloud infrastructure and ensure confidentiality, integrity, and availability of both the application and the associated data.

Group 4 performed the following to accomplish the task:

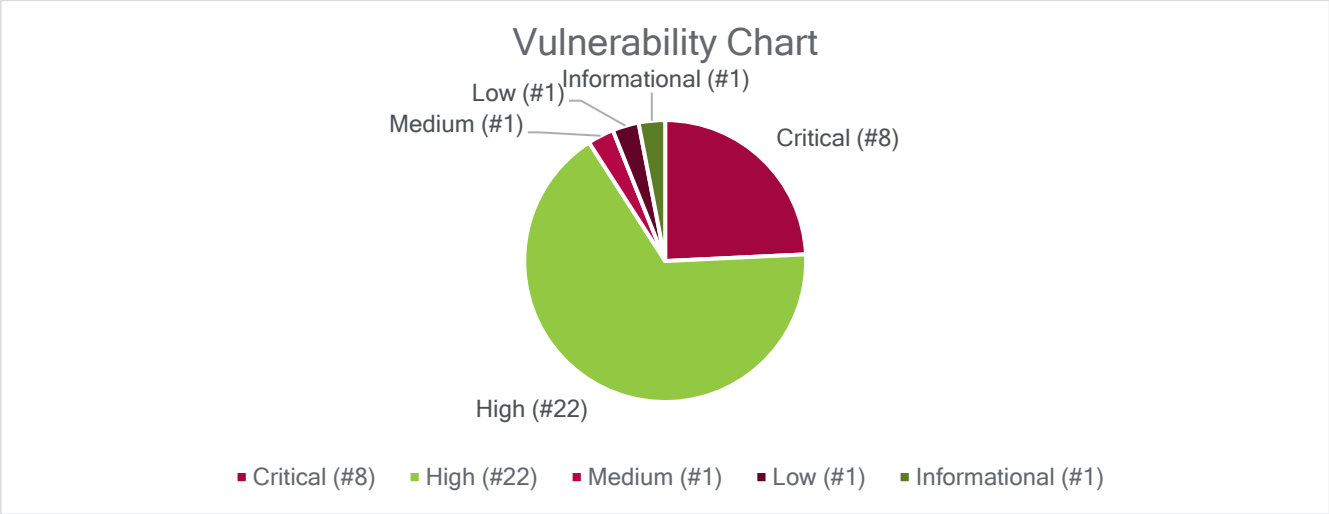
- Identify application and infrastructure-based threats and vulnerabilities
- Identify vulnerabilities, misconfigurations, and gaps in maintaining data security of the platform
- Identify vulnerabilities, misconfigurations, and gaps in the underlying virtual machine
- Identify weaknesses in network segmentation, access controls, and routing.
- Identify risks related to potential data loss and extended downtime.
- Provide recommendations that the e-commerce platform team can implement to mitigate any identified gaps, vulnerabilities, and security issues to meet industry standards.

1.2 Summary of Findings

The identified issues, gaps, and vulnerabilities are given below in the table, along with the risk rating.

#	Vulnerability Name	Severity
1	SQL Injection	Critical
2	Broken Access Control	Critical
3	Database Credentials and Storage	Critical
4	Encryption at Rest	Critical
5	Encryption at Transit	Critical
6	Unrestricted Sudo Access for ec2-user with No Password	Critical
7	EBS Volume Not Encrypted	Critical
8	Single Availability Zone	Critical
9	Reflective Cross Site Scripting (XSS)	High
10	Sensitive Data Exposure	High
11	No Data Leak Prevention (DLP)	High
12	Secrets Management	High
13	EC2 Termination Protection Not Enabled	High
14	IAM Role Not Attached to EC2 Instance	High
15	SSH Publicly Accessible + SSH Version Disclosure	High
16	Usage of Vulnerable Component	High
17	Weak MariaDB Password + Weak Password Policy	High
18	SSH Open to 0.0.0.0/0	High
19	Public Subnet and IP	High
20	Lack of Network Segmentation	High
21	Lack of DDoS Protection	High
22	Absence of VPC Flow Logs or Logging	High
23	Absence of Load Balancer	High

24	No Intrusion Detection/Prevention System	High
25	No Automated Backup Plans Created	High
26	No Load Balancers	High
27	Installation of MariaDB on an EC2 instance instead of AWS RDS	High
28	EC2 Instance does not have a Snapshot Policy	High
29	No Redundancy in the Network	High
30	No Disaster Recovery Documentation Exists	High
31	Password Brute Force	Medium
32	Verbose Error Exposure	Low
32	Improper Input Validation	Informational



Security Report

This report presents a comprehensive security assessment of a vulnerable e-commerce application, known as Juice Shop, hosted on AWS. The application stack, deployed using CloudFormation, includes an EC2 instance running OWASP Juice Shop and MariaDB.

A comprehensive security analysis revealed multiple vulnerabilities and misconfigurations, including publicly exposed ports, hardcoded credentials, a lack of backup and monitoring, and the absence of network segmentation, etc. Each finding is documented with detailed impact, associated risk levels, and actionable recommendations for remediation, aligned with industry-standard security best practices.

2.1 Vulnerability Assessment Report

2.1.1 SQL Injection

Description:

SQL Injection is a security vulnerability that arises when an application uses unsensitized/ untrusted user input directly in a SQL query without adequate validation or escaping.

Observation:

It was observed that the login page in the e-commerce application is vulnerable to SQL Injection. It was possible to inject a true statement using the payload - ' OR TRUE – in the email field to bypass the login mechanism.

Impact:

Using a simple SQLi payload, an attacker may bypass the authentication mechanism of the application and gain unauthorized access to the user's account, leading to a successful account takeover (ATO). Using the administrator account, an attacker may take full control over the application interface and potential access to user data, order history, and internal functionality.

Risk: Critical

Recommendation: It is recommended to

- Use parameterized queries in place of raw SQL queries.
- Implement both client-side and server-side input validation and reject suspicious characters like ', --, or ;.
- Enable detailed authentication logging to detect suspicious login patterns.

Reference:

SQL Injection | OWASP Foundation

Artifacts:

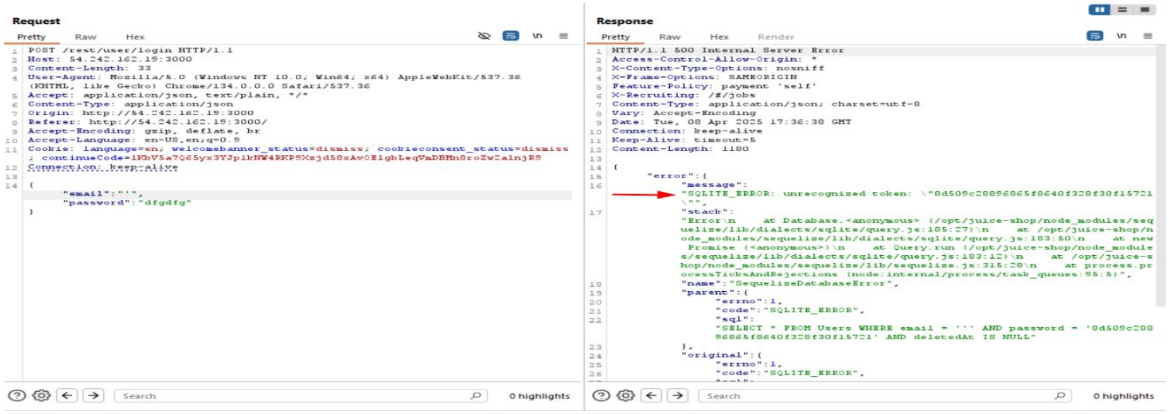


Figure 1: Adding a ' leading to breaking of SQL query resulting in database type exposure

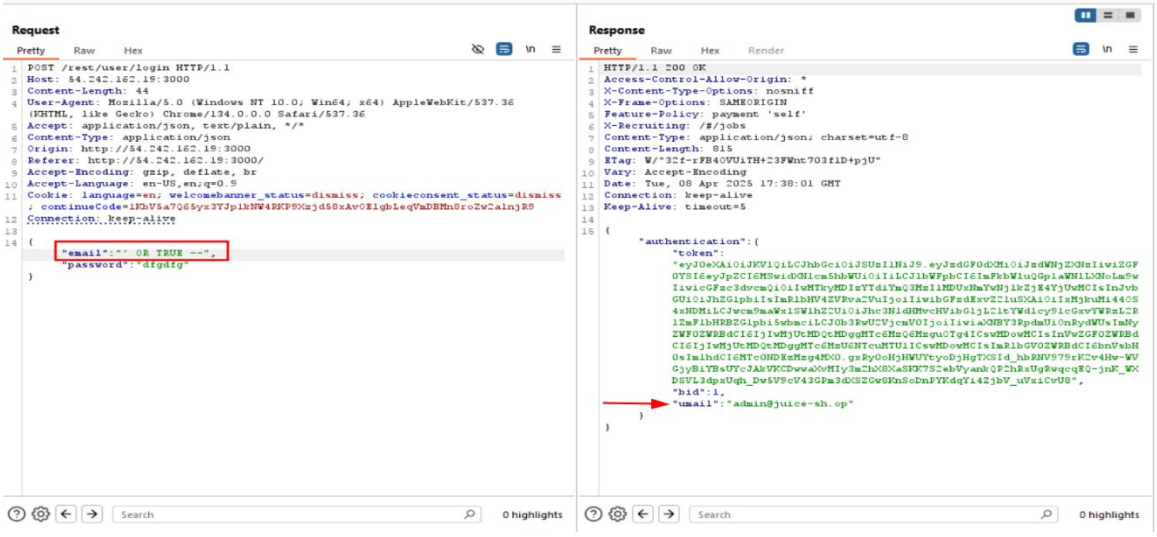


Figure 2: Able to get the administrator email address

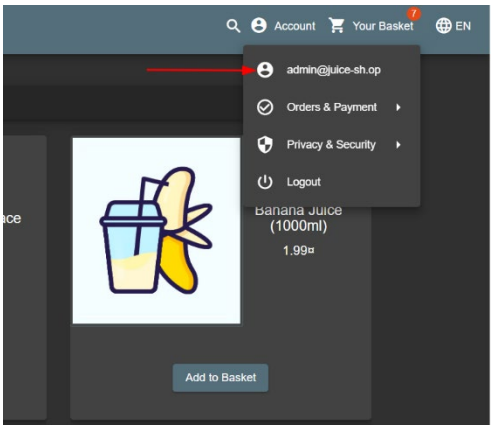


Figure 3: Successfully logged in as administrator

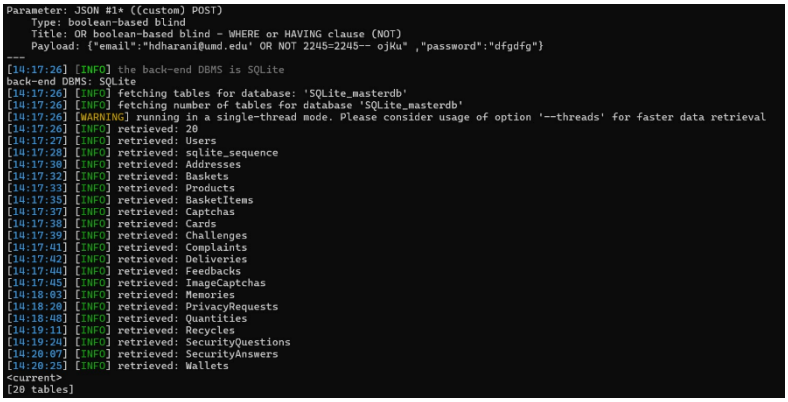


Figure 4: Exploitation leading to tables used in the database

2.1.2 Broken Access Control

Description:

Broken access control occurs when individuals have unauthorized access to data or functions beyond what they have been authorized to do because of improper security checks, which may lead to unauthorized data exposure or privilege escalation.

Observation:

It was observed that viewing the basket using the basket ID has broken access control vulnerabilities, allowing unauthorized viewing of other users' basket content.

Steps to Reproduce:

1. To see users' baskets, capture the requests used to view the basket in Burp suite and send the request to Burp suite's repeater tool.
2. In the repeater tab, the basket ID in the request URL (/rest/basket/<ID>) was changed to multiple arbitrary IDs starting from value 1.
3. This allowed access to other users' baskets when a valid ID was passed.

Impact:

Broken access control can allow attackers to gain unauthorized access to sensitive data, which can lead to data breaches, privilege escalation, and complete system takeover.

Risk: Critical

Recommendation:

It is recommended to implement proper access control checks on both the client side and the server side. User permission should be validated against the user action to ensure proper access control.

Reference:

A01 Broken Access Control - OWASP Top 10:2021

Artifacts:

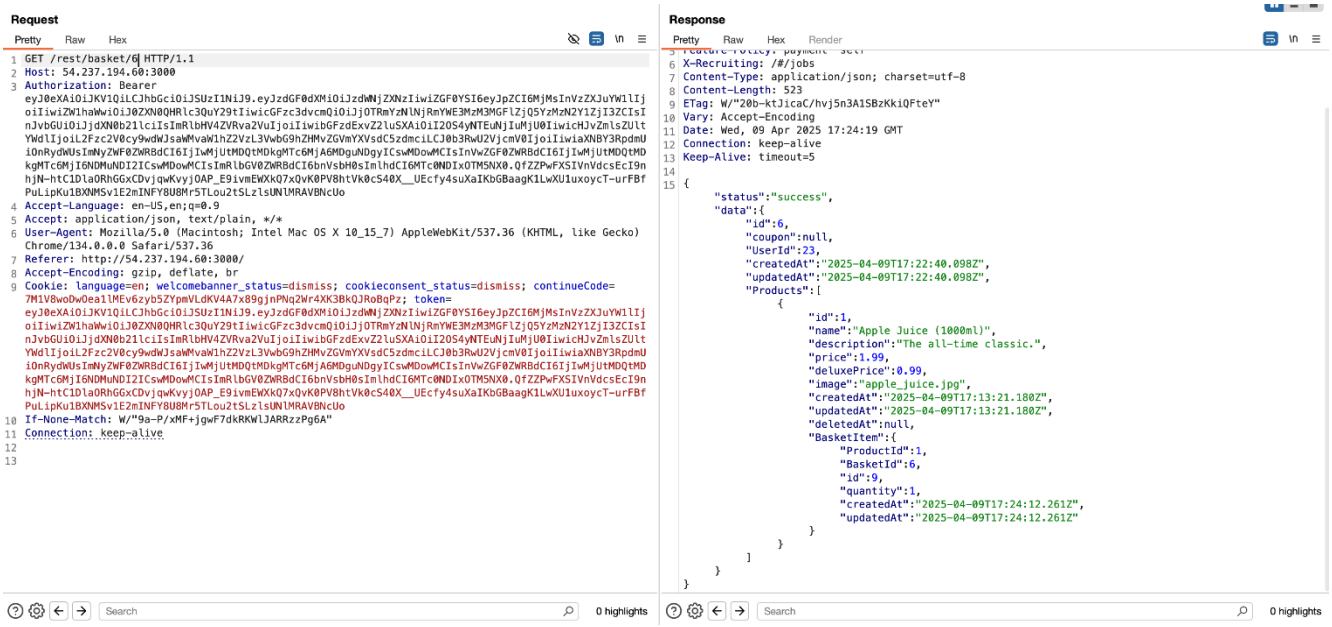


Figure 5: Request and response to retrieve current user's basket

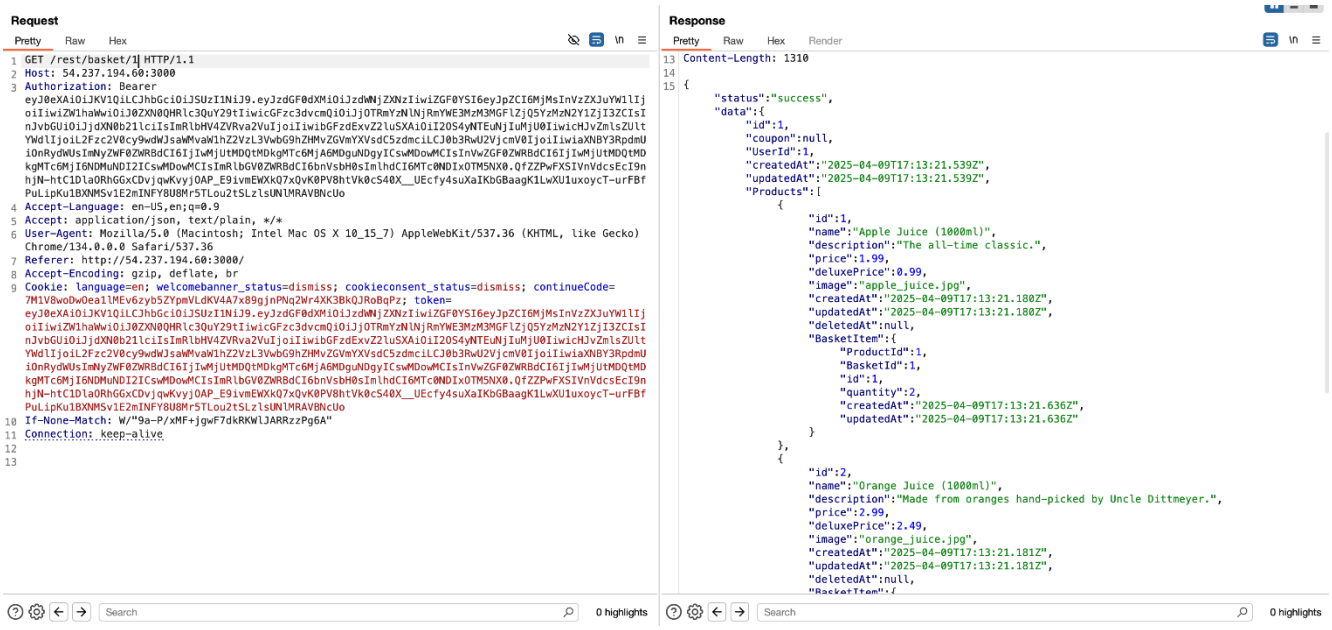


Figure 6: Accessed other user's cart by modifying the basket ID

2.1.3 Reflective Cross Site Scripting (XSS)

Description:

Reflective DOM-based XSS is a vulnerability that causes the payload to be executed as part of the Document Object Model (DOM) on the client side, without server-side interaction.

Observation:

It was observed that the product search functionality in the e-commerce application is vulnerable to XSS. It was possible to trigger the vulnerability using the payload *<img src=x onerror=alert('XSS')>*.

Impact:

XSS enables attackers to inject malicious scripts into web pages accessed by users, which may lead to credentials leakage, session hijacking, and data breaches.

Risk: High

Recommendation:

It is recommended to perform

- **Input Validation and Sanitization:** Implement strict input validation to ensure data matches the expected format.
- **Output Encoding:** Always encode data before sending the response to the client-side.
- **Content Security Policy (CSP):** Use HttpOnly and Secure flags on cookies to protect against client-side vulnerabilities.

Reference:

Cross Site Scripting (XSS) | OWASP Foundation

Artifacts:

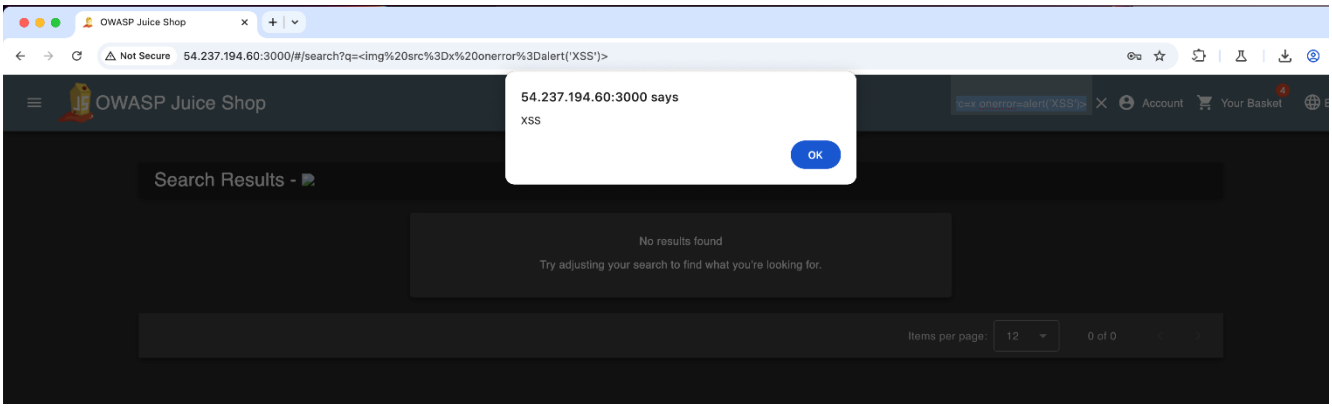


Figure 7: Successful XSS in search

2.1.4 Sensitive Data Exposure

Description:

Sensitive data exposure occurs when sensitive or confidential information is exposed to unauthorized parties. This can happen through vulnerabilities in applications, insecure data storage, poor encryption practices, or improper handling of data in transit or at rest.

Observation:

It was observed that the hyperlink to the legal document under the About Us section was pointing to http://18.205.188.221:3000/ftp/legal.md. When accessing just the /ftp/ folder, it was possible to list all the files and folders under the ftp directory. It was noted that when trying to access the files, the application was restricted to downloading/ viewing only files with the extension .pdf and .md file formats. It was possible to bypass the file extension check with a

null byte (%00) injection. To access the coupons_2013.md.bak file, URL encoded null byte (%2500) was injected with a .pdf extension. The final URL to access the file was http://18.205.188.221:3000/ftp/coupons_2013.md.bak%2500.pdf

Impact:

Directory listing vulnerability may significantly increase the attack surface of the e-commerce application by exposing internal files and folder structure to the public internet. This can lead to sensitive data exposure. Additionally, the lack of input sanitization leads to the injection of a null byte (%00) and bypasses the file type check.

Risk: High

Recommendation:

It is recommended to modify the web server configuration and restrict access to sensitive directories by using file permissions and web server rules. Implement proper input sanitization to validate and remove unexpected characters using modern frameworks.

Reference:

CWE - CWE-548: Exposure of Information Through Directory Listing (4.17)

Artifacts:

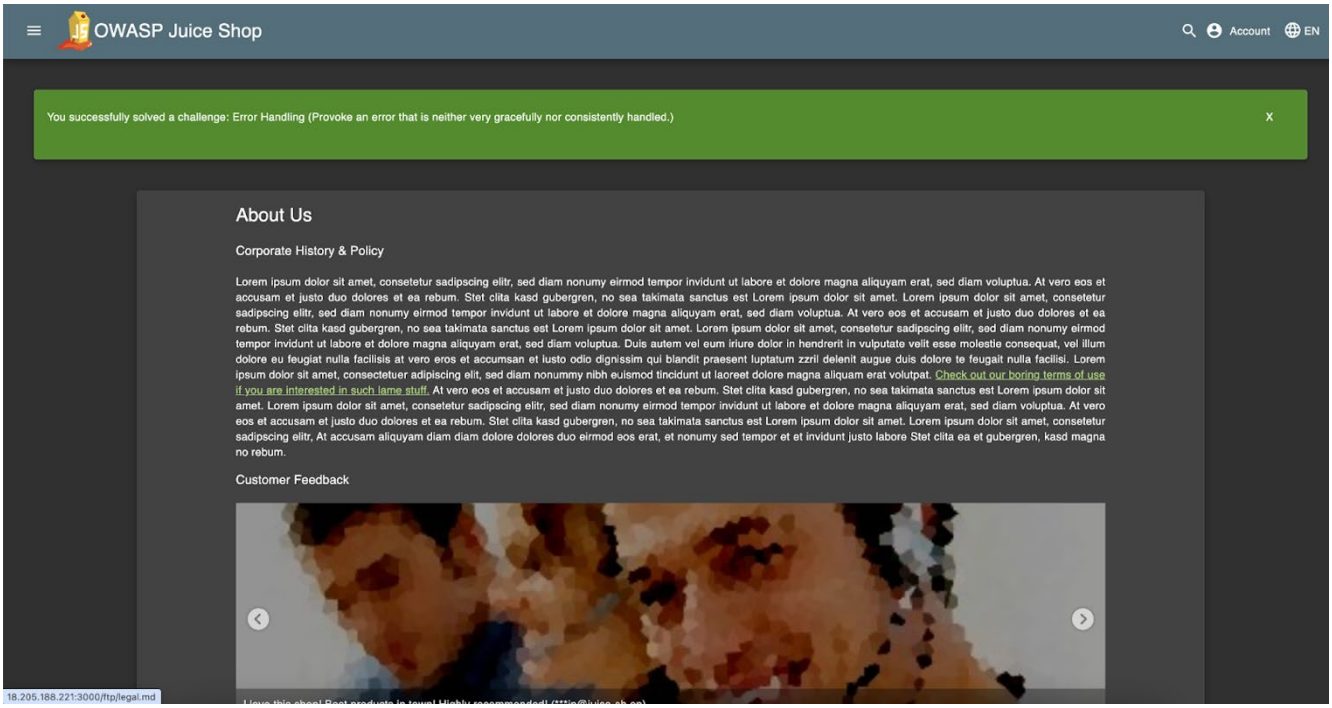


Figure 8: Hyperlink showing the presence of /ftp directory

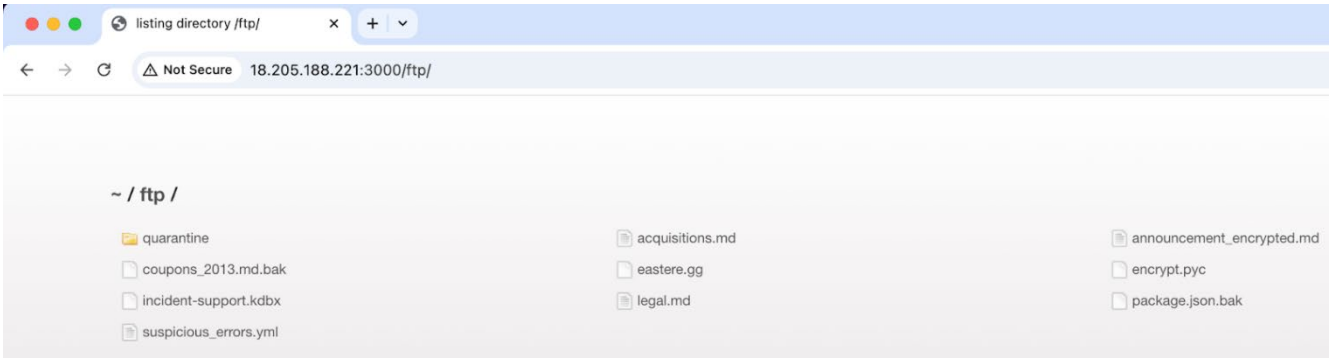


Figure 9: Directory listing in /ftp

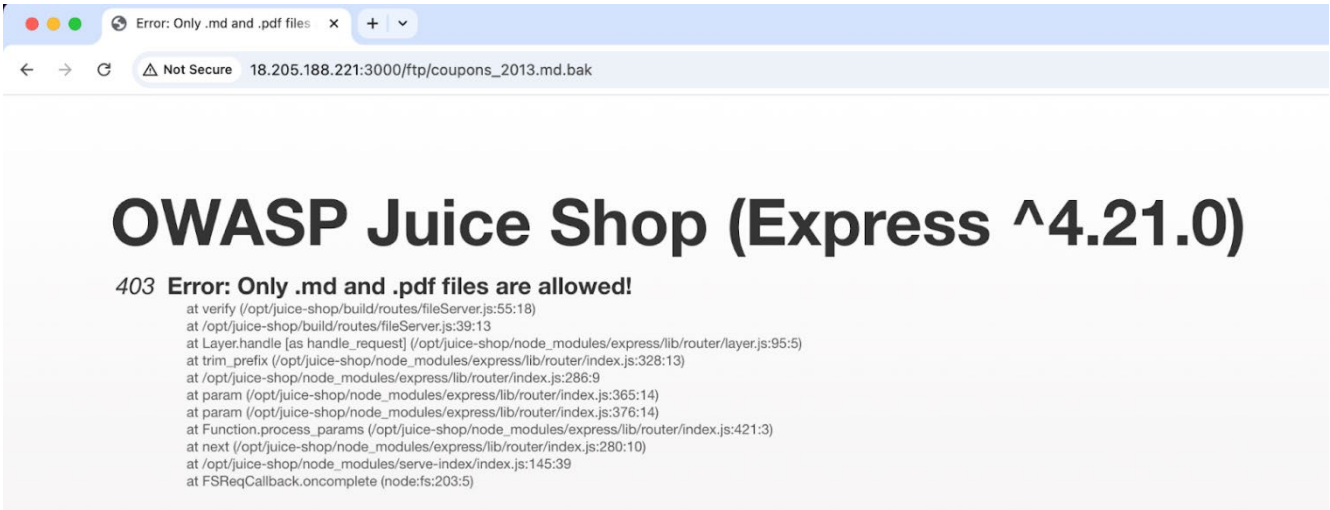


Figure 10: Error message showing that only .md and .pdf files are allowed

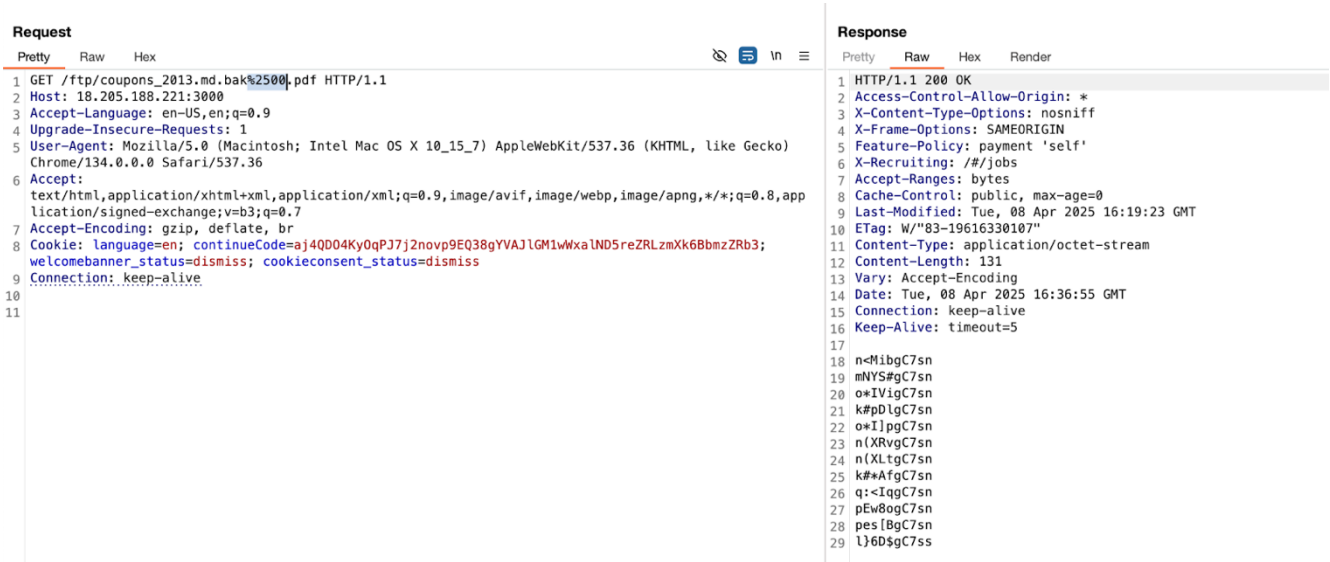


Figure 11: Validation check on the file extension was bypassed by using null byte

2.1.5 Password Brute Force

Description:

Password brute force attacks systematically attempt multiple passwords or passphrases to gain unauthorized access to a system. This is often done using automated tools that try thousands or millions of combinations rapidly.

Observation:

It was observed that the web application was not locking out accounts on multiple incorrect password attempts; it was possible to identify the admin account password by brute forcing the account with a list of well-known passwords.

Impact:

Successful brute force attacks can lead to unauthorized access to user or administrator accounts, resulting in potential data breaches, identity theft, leakage of sensitive information, or privilege escalation.

Risk: Medium

Recommendation:

It is recommended to

- Implement an account lockout or temporary suspension after a defined number of failed login attempts
- Use CAPTCHA
- Enforce strong password policies (minimum length, complexity)
- Enable multi-factor authentication (MFA)
- Monitor and alert on suspicious login activity (e.g., multiple failures from the same IP)
- Use rate limiting and progressive delays between login attempts

Reference:

Brute Force Attack | OWASP Foundation

Authentication - OWASP Cheat Sheet Series

Artifacts:

Sniper attack | Start attack

Target http://54.237.194.60:3000 | Update Host header to match target

Positions | Add § | Clear § | Auto §

```
1  POST /rest/user/login HTTP/1.1
2  Host: 54.237.194.60:3000
3  Content-Length: 52
4  Accept-Language: en-US,en;q=0.9
5  Accept: application/json, text/plain, */*
6  Content-Type: application/json
7  User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36
8  Origin: http://54.237.194.60:3000
9  Referer: http://54.237.194.60:3000/
10 Accept-Encoding: gzip, deflate, br
11 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=
   V8eRDNrW29v7qjXJMan16xd8Xf3euQWhLaFyEdgbY3EpBz4lkKZQP5mwyOoL
12 Connection: keep-alive
13
14 {"email":"admin@juice-sh.op","password":"$hello1234$"}
```

Payloads

Payload position: All payload positions

Payload type: Simple list

Payload count: 478

Request count: 478

Payload configuration

This payload type lets you configure a simple list of strings that are used as payloads.

Paste | Load... | Remove | Clear | Deduplicate | Add | Add from list...

aaa, abc123, acc, access, adfexc, adm, admin, admin123

Enter a new item

Payload processing

You can define rules to perform various processing tasks on each payload before it is used.

Add | Edit | Remove | Up | Down

Enabled | Rule

Payload encoding

This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.

Figure 12: Login request

Results | Positions

Capture filter: Capturing all items | Apply capture filter

View filter: Showing all items

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
0		401	31			413	
1	aaa	401	36			413	
2	abc123	401	37			413	
3	acc	401	208			413	
4	access	401	161			413	
5	adfexc	401	140			413	
6	adm	401	213			413	
7	admin	401	267			413	
8	admin123	200	259			1202	
9	admin2	401	273			413	

Request | Response

Pretty | Raw | Hex | Render

```
1  HTTP/1.1 200 OK
2  Access-Control-Allow-Origin: *
3  X-Content-Type-Options: nosniff
4  X-Frame-Options: SAMEORIGIN
5  Feature-Policy: payment 'self'
6  X-Recruiting: /#/jobs
7  Content-Type: application/json; charset=utf-8
8  Content-Length: 816
9  ETag: W/"330-GXSAUX8OJLzqjvvnAcTk1w2YKsc"
10 Vary: Accept-Encoding
11 Date: Wed, 09 Apr 2025 17:40:00 GMT
12 Connection: keep-alive
13 Keep-Alive: timeout=5
14
15 {
     "authentication":{
       "token":
       "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIs
       InJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiI2OS4yNTEuNjIuMjU0IiwicHJvZmlsZUltYWdlIjoiYXNzZXRzL3B1YmxpYy9pbWFnZXMvdXBsb2Fkcy9kZWZhdWx0QWRtaW4ucG5nIiwidG90cFNlY3JldCI6IiIsImlzQWN0aXZlIj
       p0cnVlLCJjcmVhdGVkQXQiOiIyMDI1LTA0LTA5IDE3OjEzOjE5LjkzMSArMDA6MDAiLCJ1cGRhdGVkQXQiOiIyMDI1LTA0LTA5IDE3OjM0OjQxLjk1NSArMDA6MDAiLCJkZWxldGVkQXQiOm51bGx9LCJpYXQiOjE3NDQyMjA0MDB9.kyvs4mlhpKCaYB0GiBsYFVKWHAe
       MG46qKHtozRwAfTrAzlM1e4OCqYBvmuuCIz1mWR8zSB6Ot9E21cn4AglEboAYa4sDazrSvg7mnqTGVa2RZD9amhM-KN8SI5x7VbsvozDiLec1XQkToUc15wAkR2ux1yLup7dD5XMvOLlgGfE",
       "bid":1,
       "umail":"admin@juice-sh.op"
     }
   }
```

Figure 13: Multiple password attempts for admin user and successful authentication using brute force method

2.1.6 Verbose Error Exposure

Description:

When an application displays overly detailed error information in the front end of the application, it may include stack traces, server paths, database details, framework information, or internal code snippets. Exposing sensitive details in production can provide attackers with valuable information about the system, leading to more targeted attacks.

Observation:

It was observed that when an error occurs by tampering with the parameters used in the web application, it throws a detailed error message in the front end, revealing sensitive information about the technology stack being used by the application.

Impact:

A Verbose error message can reveal underlying technologies (e.g., language, frameworks, server type) and expose internal file structures or database schemas, which can be used by the attacker to perform more targeted attacks.

Risk: Low

Recommendation:

It is recommended to use a generic error message for any errors that may occur in the application.

Reference:

Improper Error Handling | OWASP Foundation

Error Handling - OWASP Cheat Sheet Series

Artifacts:

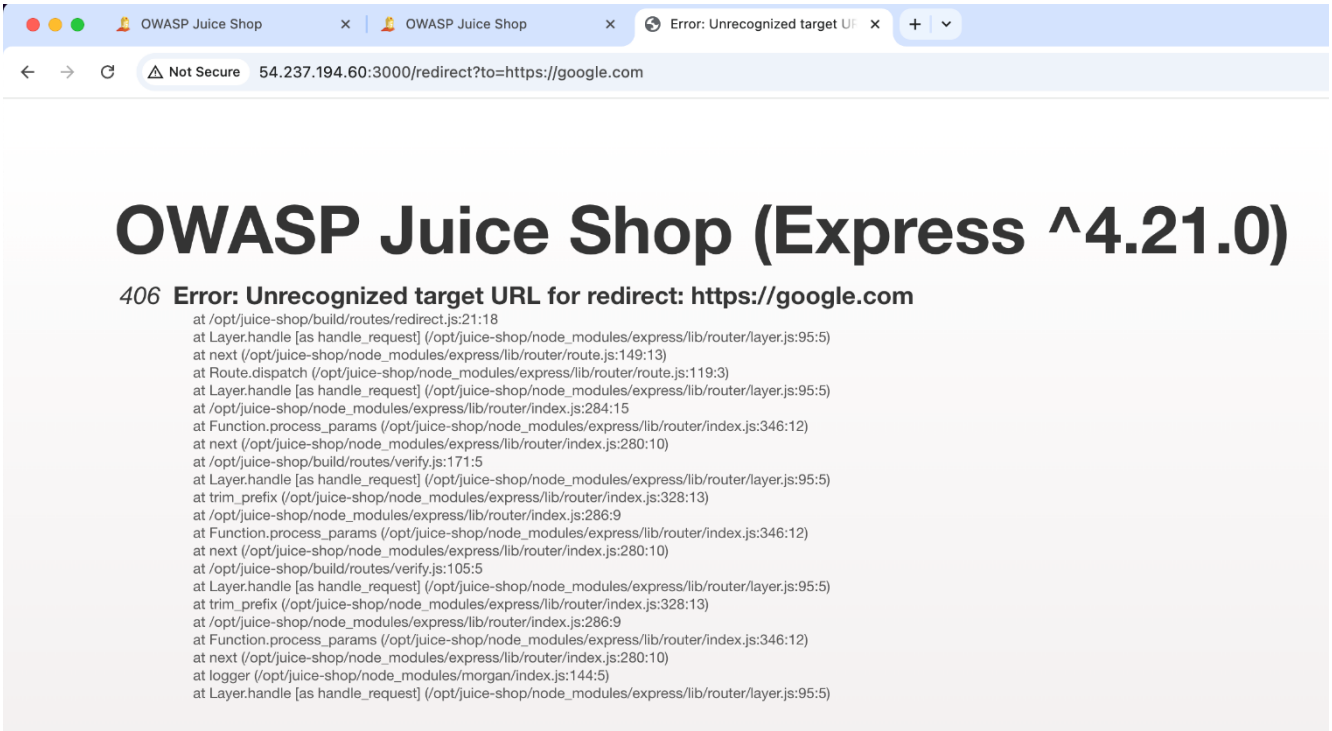


Figure 14: Verbose error message

2.1.7 Improper Input Validation

Description:

Inadequate input validation occurs when an application fails to adequately check or sanitize user inputs.

Observation:

It was observed that the application was not properly validating the inputs in the account registration functionality. It was possible to bypass the same password validation and register the account with a password different from the password in the confirm password field.

Steps to Reproduce:

1. Enter the password in the password field
2. Repeat the password in the confirm password field
3. Go back and update the password in the password field to a different value
4. Register the account, and it will be processed successfully

Impact: Informational

Allowing users to register when the **"password"** and **"confirm password"** fields don't match leads to a bad UX and increased Account Lockouts as there will be user confusion.

Risk: Informational

Reference:

OWASP Secure Coding Practices - Quick Reference Guide | Secure Coding Practices | OWASP Foundation

Input Validation - OWASP Cheat Sheet Series

Recommendation:

It is recommended to implement proper input validation checks across the application to prevent any user input-based security vulnerabilities.

Artifacts:

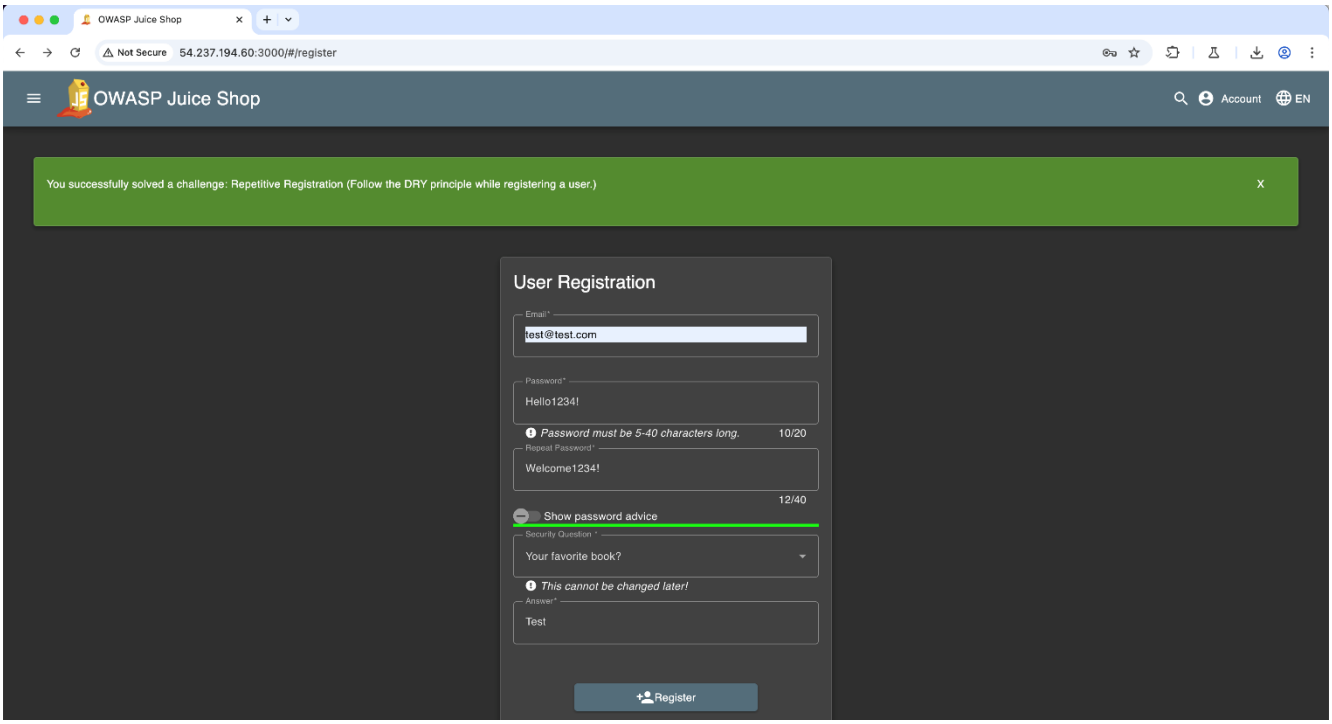


Figure 15: Password and Reset having different values

2.2 Data Security Assessment Report

2.2.1 Database Credentials and Storage

Impact:

An attacker can read the UserData script, which contains the plaintext root password to the database, if they manage to get access to the EC2 instance. The password is easily guessable or brute-forced even without EC2 access because it is both weak and hardcoded. The attacker now has complete control of the database, allowing them to read/edit private application data (orders, logs, and user information), add harmful content, and change how the program behaves.

Risk: Critical

Recommendation:

To manage secrets, use SSM Parameter Store or AWS Secret Manager. Use environment variables or secret references in place of hardcoded passwords. Use secure passwords that are generated at random.

Reference:

AWS Secrets Manager best practices

Artifacts:

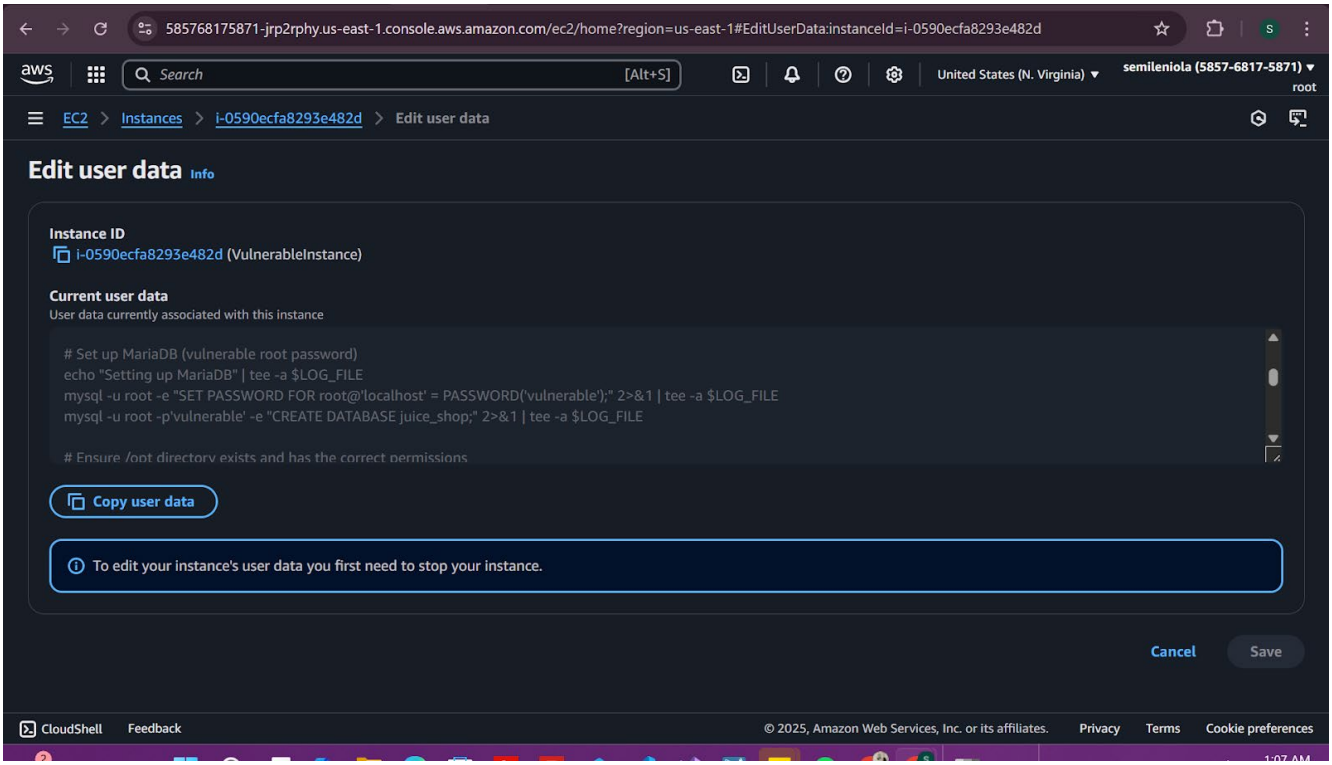


Figure 16: Plain text password in configuration

2.2.2 Encryption at Rest

Impact:

The EBS volumes that are connected to the EC2 instances contain the database and application data. When encryption isn't enabled in the EBS volume, it leaves the application data unprotected as the data will be stored in plain text. The unencrypted data may be readily readable if a snapshot is taken and viewed by an attacker.

Risk: Critical

Recommendation:

It is recommended to enable EBS encryption and incorporate encryption into the CloudFormation template.

Reference:

Amazon EBS encryption

Artifacts:

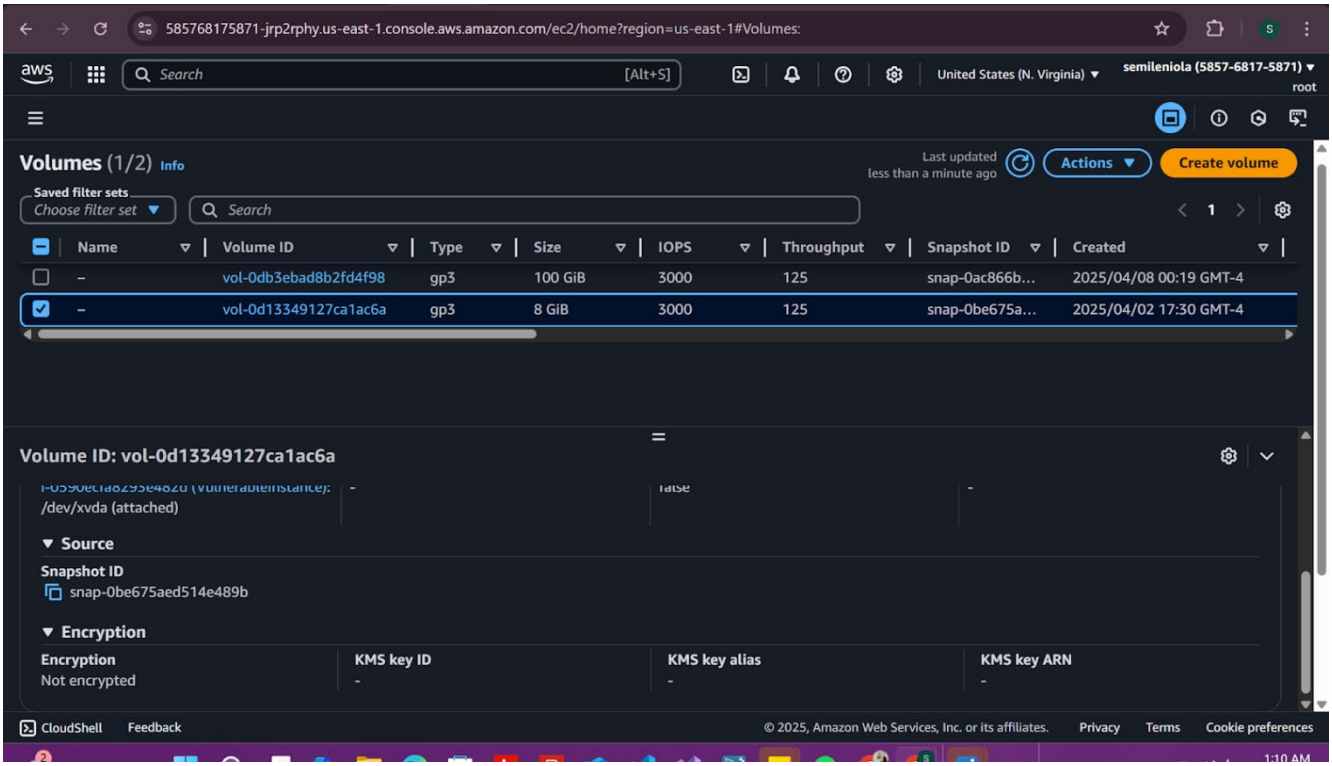


Figure 17: EBS Volume not encrypted

2.2.3 Encryption at Transit

Impact:

Although the database is currently only accessible locally (on the same EC2), this configuration is still vulnerable. If no encryption in transit is enabled and the program is then scaled out to more instances or containers, login credentials, session tokens and other data are transmitted across the network in plaintext from the application to the database, Sensitive information can be accessed or altered by any attacker who has access to the internal network or who uses Man-in-the-Middle(MITM) tactics to intercept traffic, The service is also in violation of secure design principles such as OWASP, CIS, or Zero Trust models due to its lack of SSL/TLS support.

Risk: Critical

Recommendation:

It is recommended to enable SST/TLS for MariaDB to create an encrypted connection.

Reference:

Connecting to your MariaDB DB instance on Amazon RDS with SSL/TLS from the MySQL command-line client (encrypted)

Artifacts:

```
[ec2-user@ip-10-0-1-132 ~]$ sudo cat /etc/my.cnf.d/mariadb-server.cnf
#
# These groups are read by MariaDB server.
# Use it for options that only the server (but not clients) should see
#
# See the examples of server my.cnf files in /usr/share/mysql/
#

# this is read by the standalone daemon and embedded servers
[server]

# this is only for the mysqld standalone daemon
# Settings user and group are ignored when systemd is used.
# If you need to run mysqld under a different user or group,
# customize your systemd unit file for mysqld/mariadb according to the
# instructions in http://fedoraproject.org/wiki/Systemd
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
log-error=/var/log/mariadb/mariadb.log
pid-file=/run/mariadb/mariadb.pid


#
# * Galera-related settings
#
[galera]
# Mandatory settings
#wsrep_on=ON
#wsrep_provider=
#wsrep_cluster_address=
#binlog_format=row
#default_storage_engine=InnoDB
#innodb_autoinc_lock_mode=2
#
# Allow server to accept connections on all interfaces.
#
```

Figure 18: No SSL/ TLS configuration

2.2.4 No Data Leak Prevention (DLP)

Impact:

Lack of Data Loss Prevention (DLP) protection in the infrastructure increases the risk of sensitive data being exposed or improperly accessed. Without DLP mechanism, the application may leak sensitive information, like personally identifiable information (PII).

Risk: High

Recommendation:

It is recommended to enable Amazon Macie, which automatically discovers, classifies, and protects sensitive information.

Reference:

Sensitive Data Discovery and Protection - Amazon Macie - AWS

Artifact:

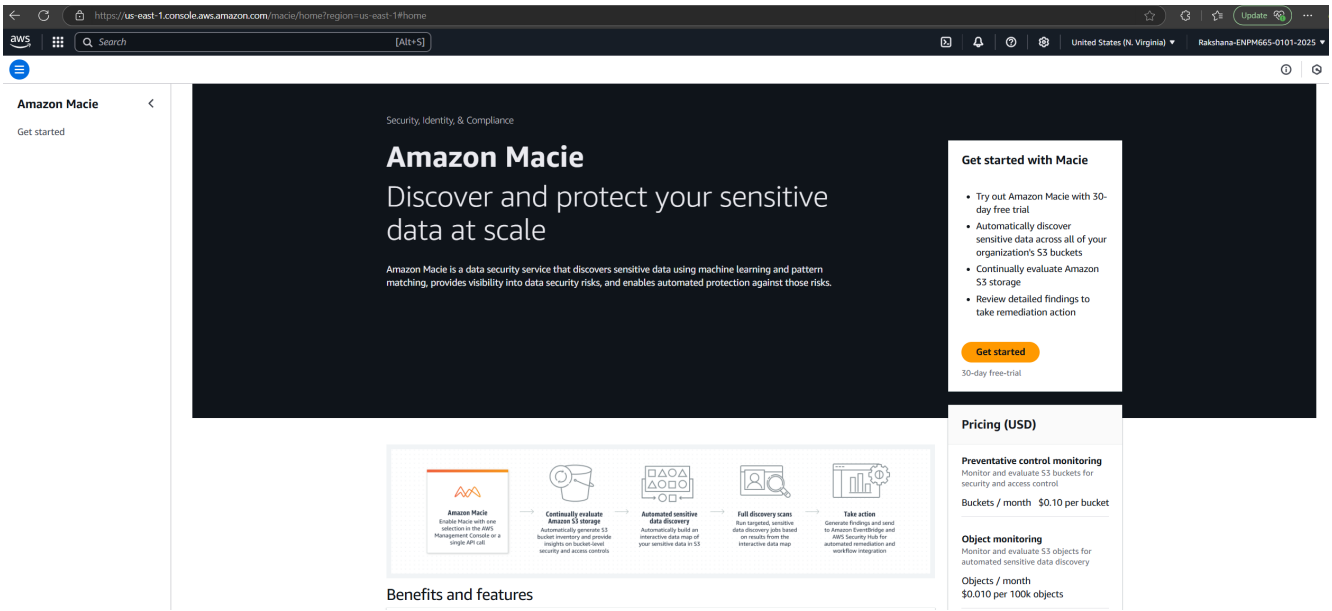


Figure 19: Amazon Macie not enabled

2.2.5 Secrets Management

Impact:

If the instance is compromised, the database is at significant risk when credentials are stored in plain text within EC2 UserData or scripts. It is simple to recover the hardcoded password and obtain complete access to the database by anyone with the access to the EC2, whether via SSH, configuration errors, or internal abuse. In addition to exposing private information, this also goes against compliance guidelines and security best practices, leaving the system open to insider attacks and data breaches.

Risk: High

Recommendation:

Instead of hardcoding login credentials into scripts, save them safely using AWS Secrets Manager or Parameter save

Reference:

AWS Systems Manager Parameter Store

2.3 Virtual Machine Vulnerability Report

2.3.1 Unrestricted Sudo Access for ec2-user with No Password

Description:

This issue occurs when the default ec2-user account is granted unrestricted sudo access without requiring a password. This configuration allows the user to execute any command as root without authentication, effectively giving full administrative control over the system.

Observation:

It was observed that the ec2-user account has passwordless sudo access configured via the /etc/sudoers file.

Impact:

Granting unrestricted sudo privileges without requiring a password reduces accountability and introduces a high risk of privilege escalation or abuse if the user account is compromised. An attacker gaining access to the ec2-user can immediately escalate privileges to root without authentication, potentially compromising the entire system. This weak configuration can result in full system takeover, making it trivial for an attacker to disable security controls, alter system configurations, and install persistent malware.

Risk: Critical

Recommendation:

It is recommended to

- Restrict sudo access to only required commands and configure sudo to require a password.
- Use individual accounts for administrative actions.

Reference:

Best practices for Amazon EC2 - Amazon Elastic Compute Cloud

Artifacts:

Figure 20: sudo access to all without password requirement

2.3.2 EBS Volume Not Encrypted

Description:

Unencrypted EBS volumes store data in plaintext on the underlying storage infrastructure, meaning data at rest is not protected.

Observation:

It was observed that the EBS volume attached to the EC2 instance is not encrypted.

Impact:

Unencrypted EBS volumes risk data exposure in case of snapshot leaks or improper deprovisioning. Sensitive data stored on the disk can be accessed or leaked if the volume or snapshot is exposed. Unencrypted data volumes can be cloned or restored elsewhere, leading to unauthorized access to sensitive files or configuration data that supports further attacks on system integrity.

Risk: High

Recommendation:

It is recommended to enable encryption on all EBS volumes using AWS-managed or customer-managed KMS keys. Ensure encryption is enforced via service control policies or launch templates.

Reference:

Amazon EBS encryption - Amazon EBS

Artifacts:

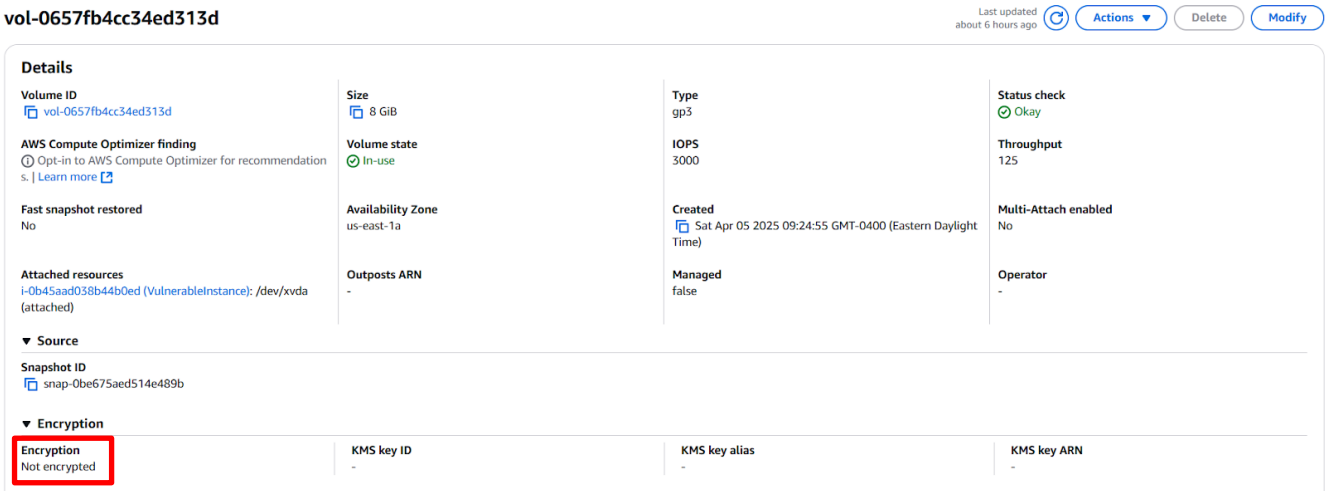


Figure 21: EBS volume not encrypted

2.3.3 EC2 Termination Protection Not Enabled

Description:

EC2 Termination Protection is a feature in AWS that prevents an instance from being accidentally or unintentionally terminated.

Observation:

It was observed that the EC2 instance does not have termination protection enabled.

Impact:

Termination protection prevents accidental or unauthorized termination of EC2 instances. Critical instances may be unintentionally deleted, leading to loss of availability and potential data corruption if systems are terminated without proper shutdown or backup. It could also interrupt dependent services.

Risk: High

Recommendation:

It is recommended to enable termination protection on critical EC2 instances via the AWS Console or CLI.

Reference:

Terminate Amazon EC2 instances - Amazon Elastic Compute Cloud

Artifacts:

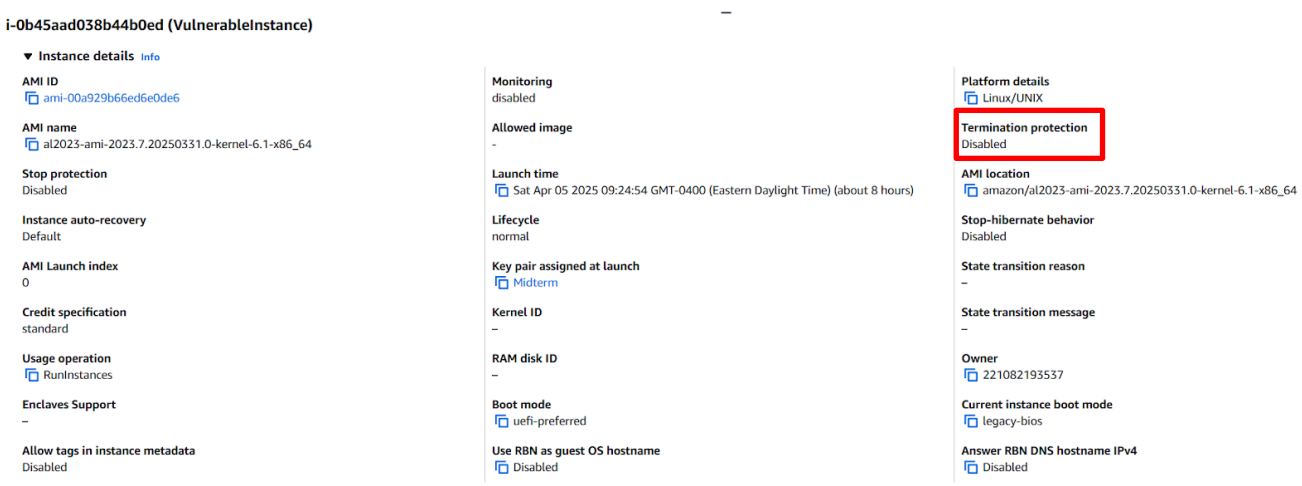


Figure 22: EC2 termination not enabled

2.3.4 IAM Role Not Attached to EC2 Instance

Description:

This issue occurs when an EC2 instance is launched without an IAM (Identity and Access Management) role attached to it.

Observation:

It was observed that the EC2 instance is not associated with any IAM role.

Impact:

IAM roles provide instances with secure access to AWS services without hardcoding credentials. Scripts or services running on the instance may use hardcoded keys or lack access, increasing the risk of key leaks or privilege issues. If credentials are leaked, attackers can gain unauthorized access to AWS APIs, modify resources, or exfiltrate data, undermining trust and control over the environment.

Risk: High

Recommendation:

Attach an IAM role with the least privilege permissions to EC2 instances. Use IAM policies tailored to required AWS resource access.

Reference:

Use an IAM role to grant permissions to applications running on Amazon EC2 instances - AWS Identity and Access Management

Security best practices in IAM - AWS Identity and Access Management

IAM roles for Amazon EC2 - Amazon Elastic Compute Cloud

Artifacts:

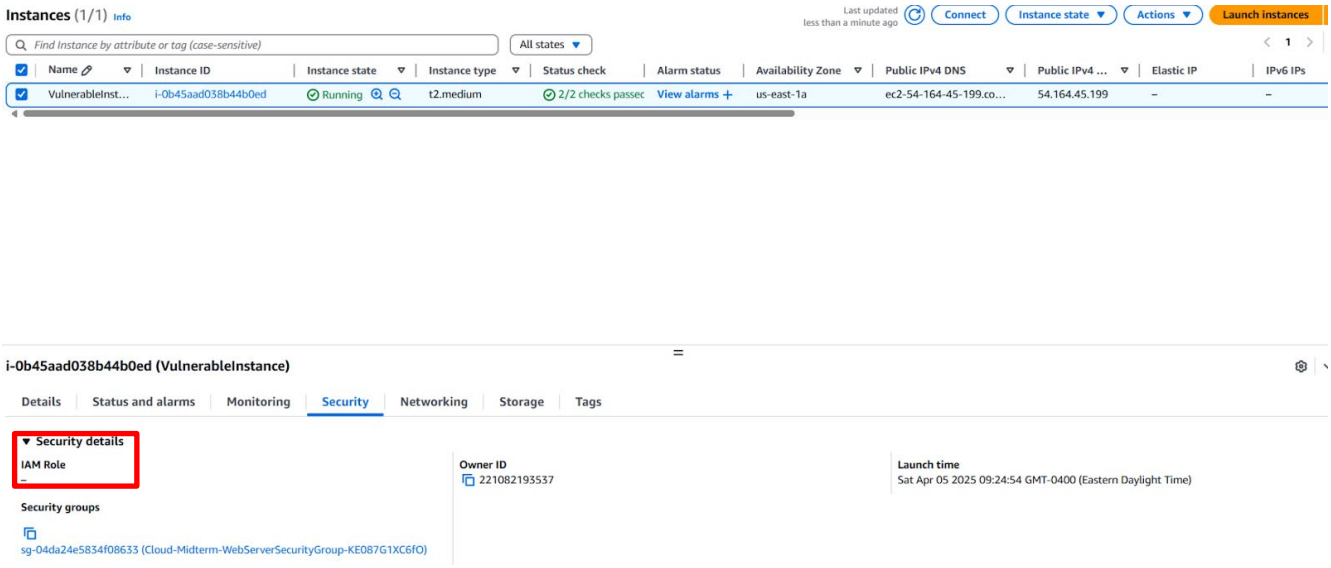


Figure 23: No IAM role attached

2.3.5 SSH Publicly Accessible + SSH Version Disclosure

Description:

The instance allows SSH (port 22) access from any IP address (0.0.0.0/0), making it exposed to the internet and potentially vulnerable to brute force and automated scanning attacks. The SSH service responds with version information during connection attempts, which may help attackers identify known vulnerabilities

Observation:

It was observed that the EC2 instance allows SSH from any IP address (0.0.0.0/0), and the SSH banner discloses version information.

Impact:

Disclosing the SSH version can help attackers craft specific exploits. Attackers can target known vulnerabilities in the disclosed version or launch brute-force attacks on public-facing SSH. Attackers may exploit known vulnerabilities in the disclosed version, leading to unauthorized access to the system.

Risk: High

Recommendation:

It is recommended to restrict SSH access to specific IPs (e.g., corporate VPN). Disable SSH version banners and use key-based authentication only.

Reference:

Top 10 security best practices for securing backups in AWS - AWS Prescriptive Guidance

sshd_config - How to Configure the OpenSSH Server?

OWASP Top Ten 2017 | A6:2017-Security Misconfiguration | OWASP Foundation

Artifacts:

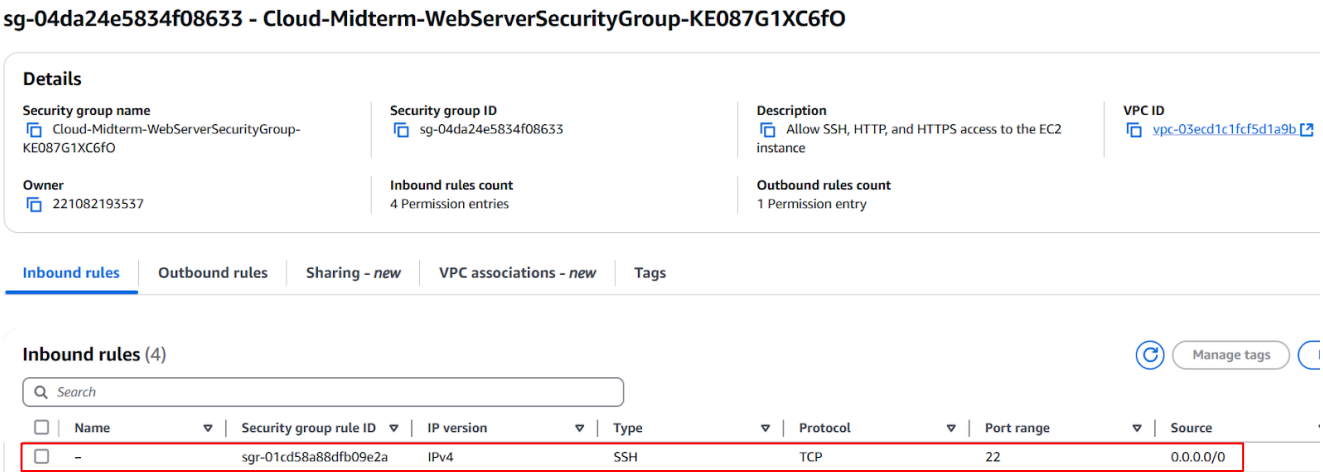


Figure 24: SSH accessible over public internet (0.0.0.0/0)

```
┌──(kali㉿kali)-[~]
└─$ nmap -sC -sV -Pn 54.164.45.199
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-05 14:47 EDT
Stats: 0:00:21 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 58.90% done; ETC: 14:48 (0:00:15 remaining)
Stats: 0:01:40 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 67.03% done; ETC: 14:50 (0:00:49 remaining)
Nmap scan report for ec2-54-164-45-199.compute-1.amazonaws.com (54.164.45.199)
Host is up (0.016s latency).
Not shown: 995 filtered tcp ports (no-response), 3 filtered tcp ports (host-unreach)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.7 (protocol 2.0)
| ssh-hostkey:
|   256 f5:08:a9:98:6f:1c:b2:7d:88:a5:bb:85:70:27:d7:52 (ECDSA)
|_  256 dd:0b:7e:ef:41:d6:07:14:61:b9:eb:26:da:d2:41:e6 (ED25519)
3000/tcp open  ppp?
```

Figure 25: SSH version disclosure

2.3.6 Usage of Vulnerable Component

Description:

Utilizing vulnerable components in an application or system, such as outdated libraries, frameworks, or plug-ins, can expose the application and its underlying infrastructure to known vulnerabilities. The attackers may exploit such components to gain unauthorized access, run arbitrary code, or compromise system integrity.

Observation:

It was observed that the vulnerable version of cross-spawn was being used in the instance that could be exploited by attackers to create a regular expression based denial of service attack.

Impact:

Attackers may exploit known vulnerabilities in outdated software or components to access sensitive data, launch denial-of-service attacks, or execute remote code. Most of the vulnerabilities in third-party components are often well-documented publicly, exploitation can be low-effort and high-impact.

Risk: High

Recommendation:

It is recommended to keep the components up-to-date and use software composition analysis (SCA) tools like OWASP Dependency-Check as part of the CI/CD pipeline to detect any vulnerable libraries/ components.

Reference:

A06 Vulnerable and Outdated Components - OWASP Top 10:2021

OWASP Dependency-Check | OWASP Foundation

Artifacts:

Components affected by critical level vulnerabilities

1. CVE-2024-21538 - cross-spawn (Denial of Service Vulnerability, affecting the availability of the instance)

```
[ec2-user@ip-10-0-1-9 ~]$ cat /usr/lib/node_modules/npm/node_modules/cross-spawn/package.json
{
  "name": "cross-spawn",
  "version": "7.0.3",
  "description": "Cross platform child_process#spawn and child_process#spawnSync",
  "keywords": [
    "spawn",
    "spawnSync",
    "windows",
    "cross-platform",
    "path-ext",
    "shebang",
    "cmd",
    "execute"
  ],
```

Figure 26: Vulnerable version of cross-spawn used

2.3.7 Weak MariaDB Password + Weak Password Policy

Description:

This problem is related to the use of weak and easily guessable passwords for MariaDB root account alongside an inadequately enforced password policy.

Observation:

It was observed that MariaDB root user account password hash can be extracted from the database table, and it was possible to crack the hash. It was also observed that there is no minimum password length requirement for accounts that are created in the instance, and the password expiration is set to 99,999 days.

Impact:

Weak database credentials and lack of password complexity enforcement increase the risk of brute-force or credential stuffing attacks. Unauthorized database access may lead to sensitive data exposure. Compromised database credentials may allow an attacker to modify or remove data, escalate privileges, and compromise database integrity.

Risk: High

Recommendation:

It is recommended to enforce strong password policies (minimum length, complexity). Rotate weak credentials and store secrets in AWS Secrets Manager.

Reference:

Securing MariaDB - MariaDB Knowledge Base

Artifacts:

```
MariaDB [(none)]> SELECT User, Host, authentication_string FROM mysql.user;
+-------------+-----------+-------------------------------------------+
| User        | Host      | authentication_string                     |
+-------------+-----------+-------------------------------------------+
| mariadb.sys | localhost |                                           |
| root        | localhost | *86C7B5631B394A08C8DA239E586E7292D29BF7FA |
| mysql       | localhost | invalid                                   |
+-------------+-----------+-------------------------------------------+
3 rows in set (0.016 sec)
```

Figure 27: Password hash for root MariaDB user

```
┌──(kali㉿kali)-[~]
└─$ echo "root:*86C7B5631B394A08C8DA239E586E7292D29BF7FA" > mysql_hash.txt

┌──(kali㉿kali)-[~]
└─$ john --format=mysql-sha1 mysql_hash.txt --wordlist=/usr/share/wordlists/metasploit/password.lst
Using default input encoding: UTF-8
Loaded 1 password hash (mysql-sha1, MySQL 4.1+ [SHA1 128/128 AVX 4x])
Warning: no OpenMP support for this hash type, consider --fork=4
Press 'q' or Ctrl-C to abort, almost any other key for status
vulnerable       (root)
1g 0:00:00:00 DONE (2025-04-05 12:04) 50.00g/s 4256Kp/s 4256Kc/s 4256KC/s vulgate..vulnerable
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Figure 28: Hash cracked with John the Ripper tool

```
# Password aging controls:
#
#       PASS_MAX_DAYS   Maximum number of days a password may be used.
#       PASS_MIN_DAYS   Minimum number of days allowed between password changes.
#       PASS_MIN_LEN    Minimum acceptable password length.
#       PASS_WARN_AGE   Number of days warning given before a password expires.
#
PASS_MAX_DAYS   99999
PASS_MIN_DAYS   0
PASS_WARN_AGE   7

# Currently PASS_MIN_LEN is not supported
```

Figure 29: EC2 instance password policy

2.4 Network Security Assessment Report

2.4.1 SSH Open to 0.0.0.0/0

Observation:

It was observed that the EC2 security group allows SSH access (port 22) from any IP address globally (0.0.0.0/0), which exposes the server to the public internet.

Impact:

Any attacker can attempt to connect via SSH, including using brute-force, password-guessing, or exploiting known vulnerabilities in the SSH daemon. This also violates the principle of least privilege.

Risk: High

Recommendation:

It is recommended to

- Restrict SSH access to specific trusted IPs (e.g., corporate network).
- Replace SSH access with AWS Systems Manager (SSM) Session Manager, which offers auditable, keyless access without needing to expose port 22.
- Rotate SSH key pairs regularly and enforce strong authentication policies.

Reference:

Amazon EC2 key pairs and Amazon EC2 instances - Amazon Elastic Compute Cloud

Artifacts:

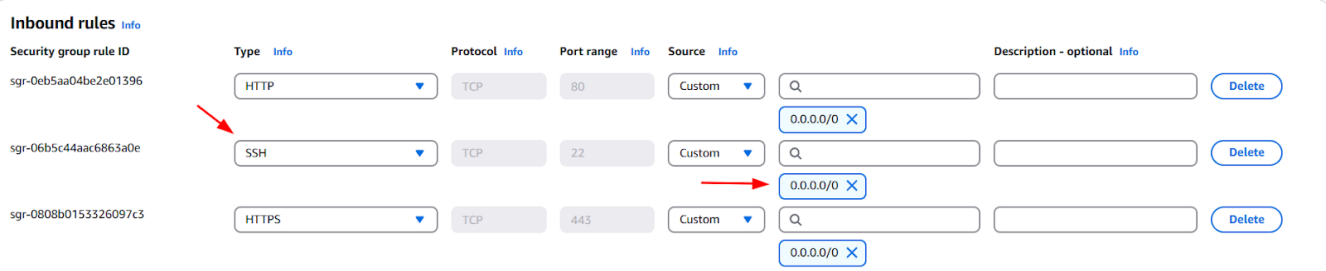


Figure 30: SSH Port open globally

2.4.2 Public Subnet and IP

Observation:

It was observed that the EC2 instance hosting the Juice Shop application is deployed into a public subnet with a public IPv4 address automatically assigned. This means the server is directly exposed to the internet without any intermediary (e.g., load balancer, API gateway, or proxy layer) to filter or inspect incoming traffic.

Impact:

The entire server, including web app, database, and OS-level services, is exposed to external traffic. This invites scanning, exploitation, and targeted attacks (e.g., HTTP fuzzing, known CVEs). Since both the app and DB are co-located, this configuration also exposes the backend layer without restrictions.

Risk: High

Recommendation:

It is recommended to

- Move application servers to a private subnet.
- Use a load balancer in a public subnet to route traffic to the backend.
- Apply security groups to control what traffic reaches backend resources.
- Limit public IP usage to critical edge services only.

Reference:

Example: VPC with servers in private subnets and NAT - Amazon Virtual Private Cloud

Artifacts:

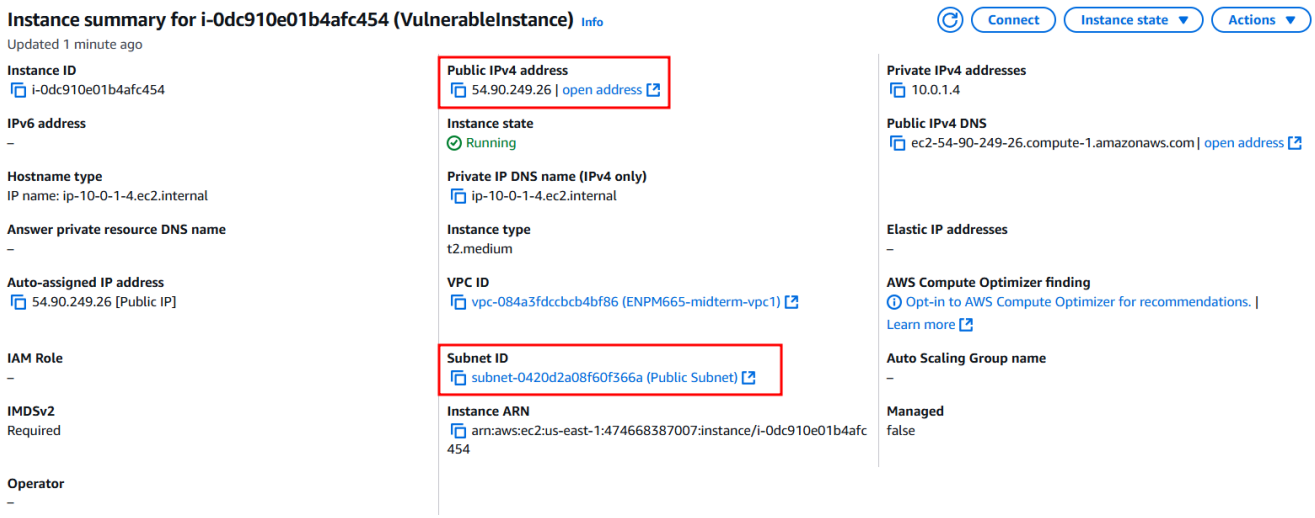


Figure 31: EC2 instance having public subnet and public IP

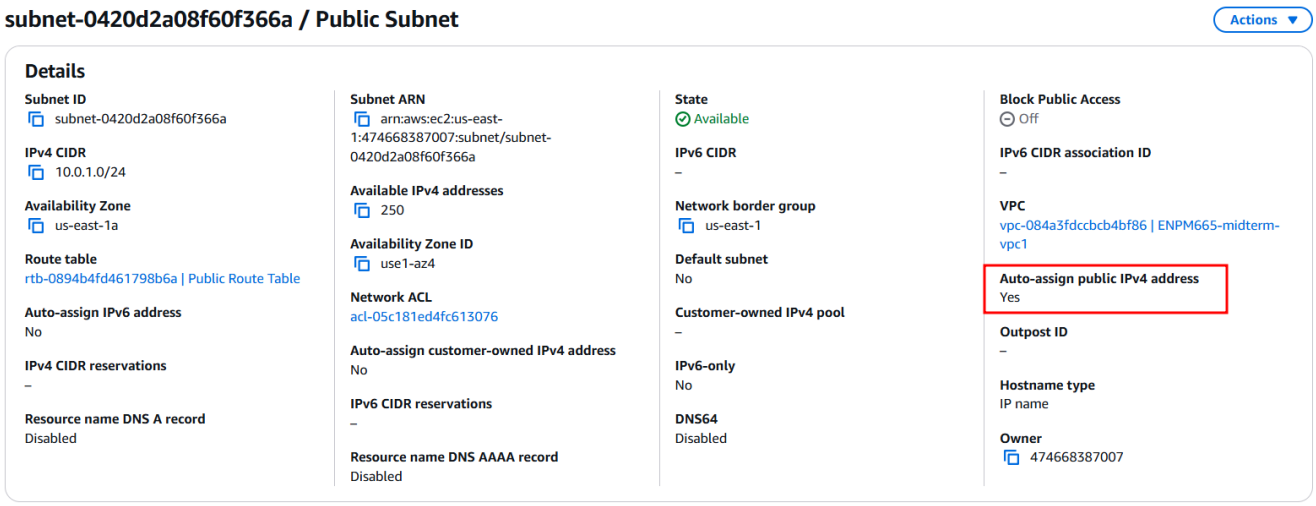


Figure 32: Auto assigning of public IP address enabled

2.4.3 Lack of Network Segmentation

Observation:

It was observed that the deployed infrastructure uses a flat network topology with all application components – the Juice Shop web server and its MariaDB database – running on a single EC2 instance within a public subnet, under one security group. There are no private subnets, separate tiers, or traffic isolation mechanisms.

Impact:

If an attacker compromises the web server (e.g., via SQLi, RCE), they immediately gain access to the database and potentially the underlying OS. A flat topology means a single successful exploit could compromise the entire environment. Exposing internal services (e.g., DB) in the same security context as public-facing services (e.g., web server) violates cloud architecture best practices.

Risk: High

Recommendation:

It is recommended to redesign the network architecture using a multi-tier VPC layout with proper segmentation as follows

- **Public Subnet:** For Load Balancer only (accepts internet traffic)
- **Private Subnet (App Tier):** For EC2 instances running the Juice Shop web application
- **Private/Isolated Subnet (DB Tier):** For hosting the MariaDB database; no direct internet access
- Use security groups to allow only specific port access (e.g., web tier → DB port 3306)

Reference:

Overview of Amazon Web Services

Artifacts:

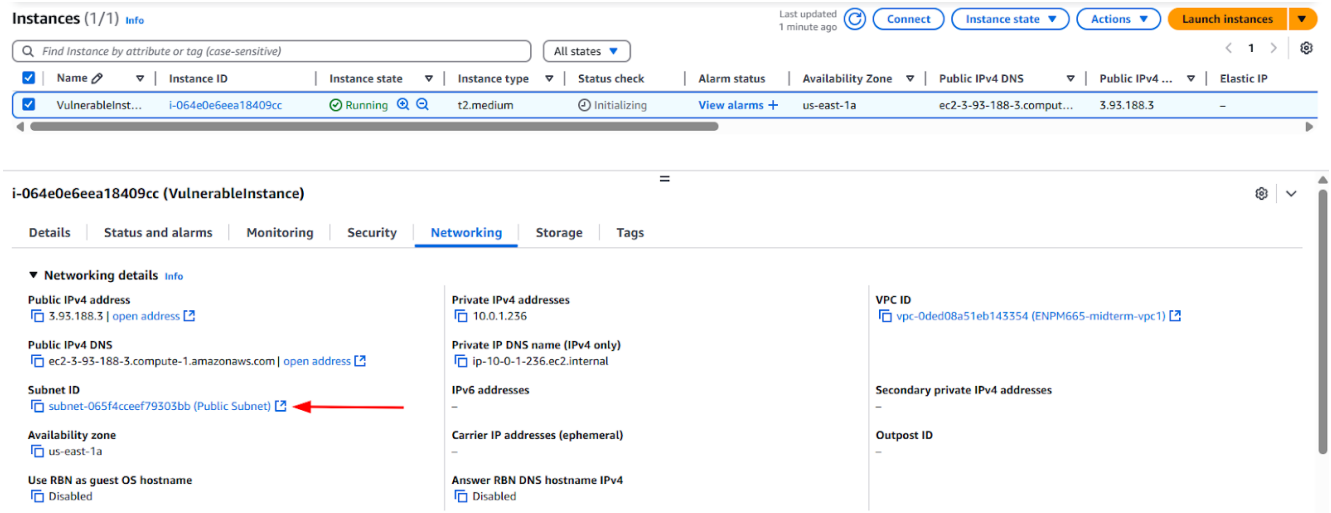


Figure 33: A blank "Protected Resources" page shows the absence of AWS Shield or WAF.

Figure 34: Public subnet has a direct route to the internet

2.4.4 Lack of DDoS Protection

Observation:

It was observed that no AWS Shield, AWS WAF, CloudFront, or rate-limiting measures are implemented to prevent or mitigate DDoS attacks.

Impact:

A malicious actor can easily overwhelm the application or its EC2 instance with repeated requests, potentially taking it offline. Since the EC2 instance is directly exposed, there are no buffers or autoscaling to handle load surges.

Risk: High

Recommendation:

It is recommended to

- Enable AWS Shield Standard (free) or Advanced (paid).
- Use AWS WAF with rules for rate limiting, IP reputation lists, and common exploits (SQLi, XSS).
- Deploy CloudFront as a CDN in front of the app to absorb and cache legitimate traffic.

Reference:

How AWS Shield and Shield Advanced work

Artifacts:

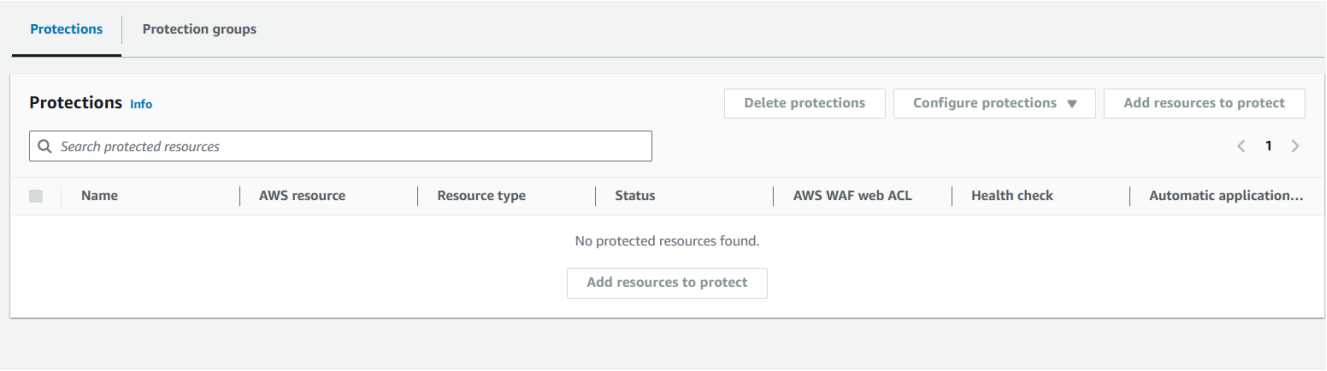


Figure 35: A blank "Protected Resources" page shows the absence of AWS Shield or WAF.

2.4.5 Absence of VPC Flow Logs or Logging

Observation:

It was observed that there are no VPC Flow Logs, CloudWatch Logs, or centralized logging services in place to track network traffic or instance behavior.

Impact:

Security teams will have zero visibility into traffic patterns, making it impossible to detect unusual outbound connections, port scans, data exfiltration, internal lateral movement etc.

Risk: High

Recommendation:

It is recommended to

- Enable VPC Flow Logs to capture accepted/rejected traffic.
- Stream logs to CloudWatch or S3 for storage and analysis.
- Integrate AWS GuardDuty to detect threats based on logs.

Reference:

Logging IP traffic using VPC Flow Logs - Amazon Virtual Private Cloud

Artifacts:

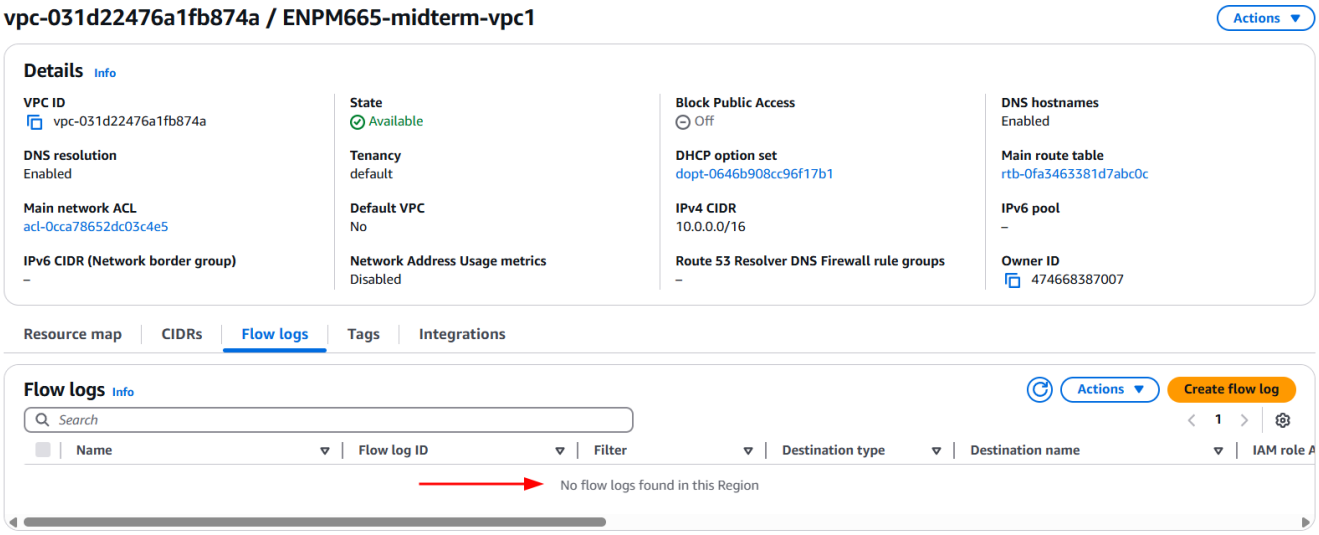


Figure 36: Flow logs absent

2.4.6 Absence of Load Balancer

Observation:

It was observed that the current deployment routes traffic directly from the internet to a public EC2 instance hosting the Juice Shop application. There is no Web Application Firewall (WAF) in place to filter or inspect HTTP/S traffic, and there is no Load Balancer (ALB or NLB) to provide centralized traffic routing, SSL termination, or redundancy.

Impact:

The absence of load balance can lead to

- No protection against OWASP Top 10 attacks, including SQL injection, cross-site scripting (XSS), and request smuggling.
- Gap in implementing IP reputation filtering, geo-blocking, or rate limiting.
- No TLS offloading, which means HTTPS traffic adds CPU load to the EC2 instance.

Risk: High

Recommendation:

It is recommended to

- Deploy an Application Load Balancer (ALB) in the public subnet.
- Attach an AWS WAF WebACL to the ALB.
- Use Managed WAF Rules from AWS Marketplace for additional protections (e.g., Bot Control, Account Takeover Prevention).

Reference:

AWS WAF

Associating or disassociating a web ACL with an AWS resource - AWS WAF, AWS Firewall Manager, and AWS Shield Advanced

Artifacts:

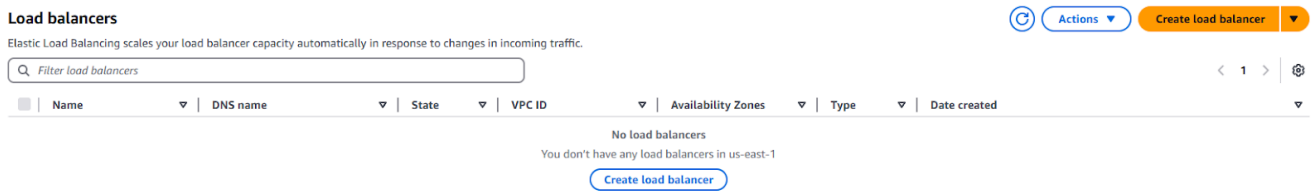


Figure 37: Absence of Load Balancer

2.4.7 No Intrusion Detection/Prevention System

Observation:

It was observed that the current cloud deployment does not include any Intrusion Detection System (IDS) or Intrusion Prevention System (IPS) at either the network or host level. There is no monitoring in place to detect suspicious activity, unauthorized access attempts, traffic anomalies, or signs of compromise within the VPC or EC2 instance.

Impact:

Malicious behavior can go undetected, including port scanning, brute-force login attempts, lateral movement, and data exfiltration. The infrastructure is blind to early-stage attack indicators, such as C2 beaconing, reconnaissance activity, or insider threats. Non-compliance with security frameworks like PCI-DSS, HIPAA, and CIS that recommend or require intrusion detection.

Risk: High

Recommendation:

It is recommended to

- Enable Amazon GuardDuty. GuardDuty provides detection for port scans, malware C2 activity, credential compromise, etc.
- Install lightweight host-based intrusion detection agents such as Wazuh
- Use AWS Network Firewall or third-party IPS to inspect and control traffic between subnets.
- Integrate with AWS Security Hub to centralize findings and automate response (e.g., Lambda to isolate instances)

Reference:

What is Amazon GuardDuty? - Amazon GuardDuty

AWS Security Hub

AWS Security Best Practices

Artifacts:

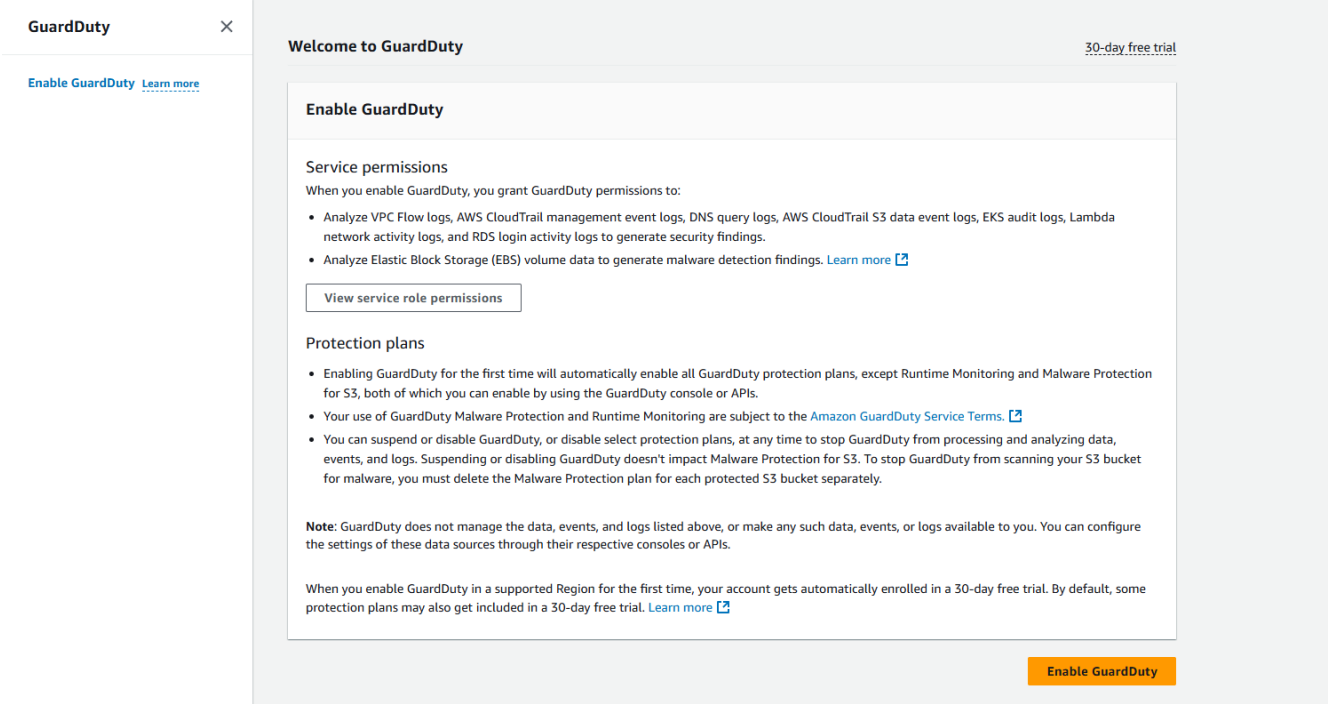


Figure 38: Amazon GuardDuty not enabled

2.5 Disaster Recovery Assessment Report

2.5.1 Single Availability Zone

Observation:

It was observed that the entire AWS Setup has been created within a single availability zone - us-east-1 a.

Impact:

As the application is available in a single zone - us-east-1a, if this availability zone goes down, the application will not be accessible by its users, resulting in downtime.

Risk: Critical

Recommendation:

The application should be deployed in a multi-availability zone or cross-region zone setup, with AWS features like Elastic Load Balancers (ELB), Auto Scaling, etc., enabled. One of the availability zones/cross-region zones should be dedicatedly configured as a disaster recovery zone, containing replicas of VPCs, subnets, instances, etc., present in the other zones.

Reference:

Use multiple Availability Zones - Real-Time Communication on AWS

Artifacts:

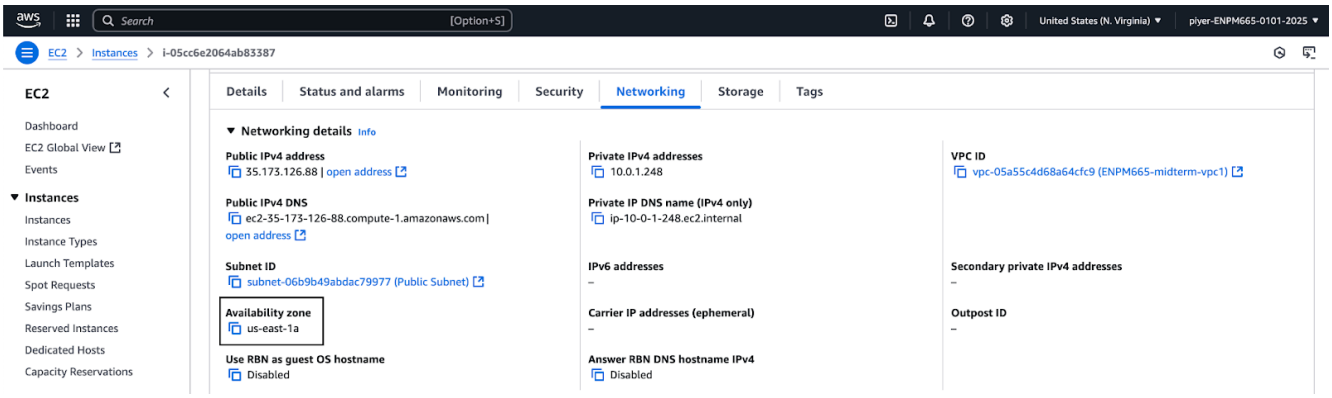


Figure 39: Only 1 Availability Zone Present in the AWS Environment

2.5.2 No Automated Backup Plans Created

Observation:

It was observed that AWS Backup does not have automated Backup Plans defined.

Impact:

In case of accidental deletion/corruption of data due to human error, disasters, etc., there is no way of recovering it unless manual backups are taken, as there are no plans configured to automatically take backups of the EC2 instance. This could lead to the loss of critical business data.

Risk: High

Recommendation:

Creation of backup plans that automate taking backups of instances periodically and store them in Amazon S3 buckets or EBS snapshots. They should have a retention period set up (35 days/1 year/5 years/7 years), depending on the business requirement. Since the website (OWASP Juice Shop) is a live website, there should be daily full backup + incremental backups, and the retention period should be between 30-90 days to avoid the loss of critical data. Following this approach will ensure that even in cases of accidental deletion, not all data will be lost.

Reference:

Backup As A Service - AWS Backup - AWS

Artifacts:

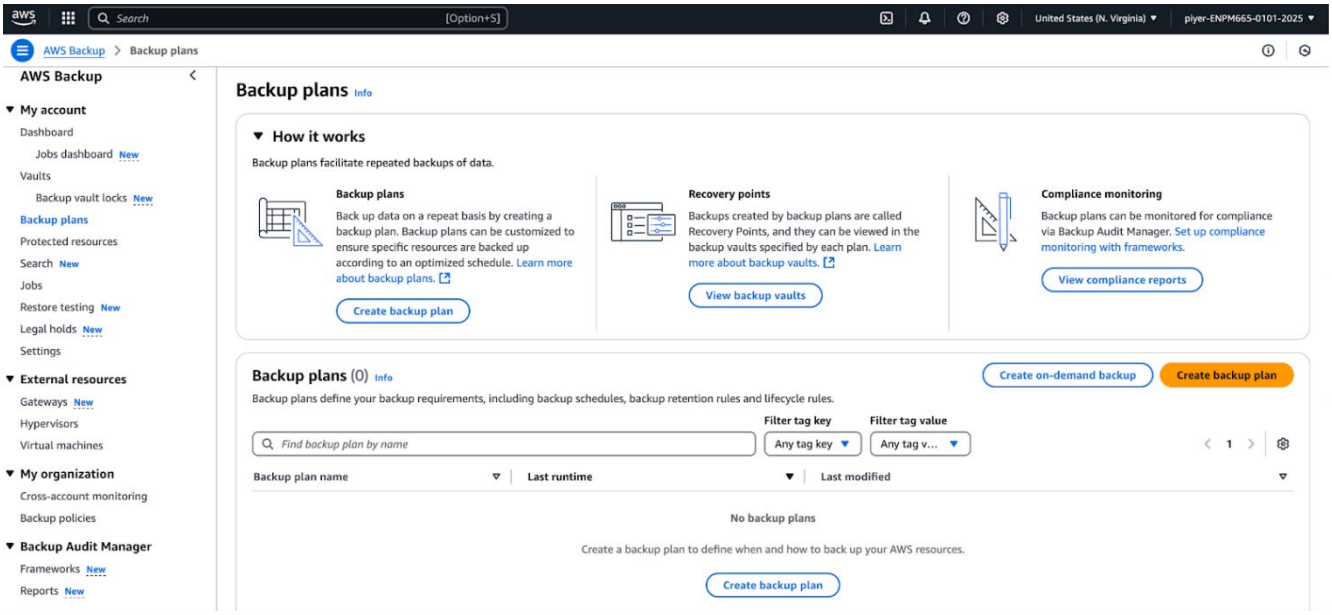


Figure 40: Absence of automated Backup Plans in AWS Backup

2.5.3 No Load Balancers

Observation:

It was observed that there are no Load Balancers in the AWS Environment to help distribute traffic.

Impact:

Due to the absence of Load Balancers, failure in the primary server will result in the downtime of the entire application, affecting the availability of resources to users and potentially leading to loss of revenue for the organization.

Risk: High

Recommendation:

There should be Application Load Balancers or Network Load Balancers that would help distribute traffic, eliminate the risk of a single point of failure, and increase redundancy.

Reference:

What is Elastic Load Balancing? - Elastic Load Balancing

Artifacts:

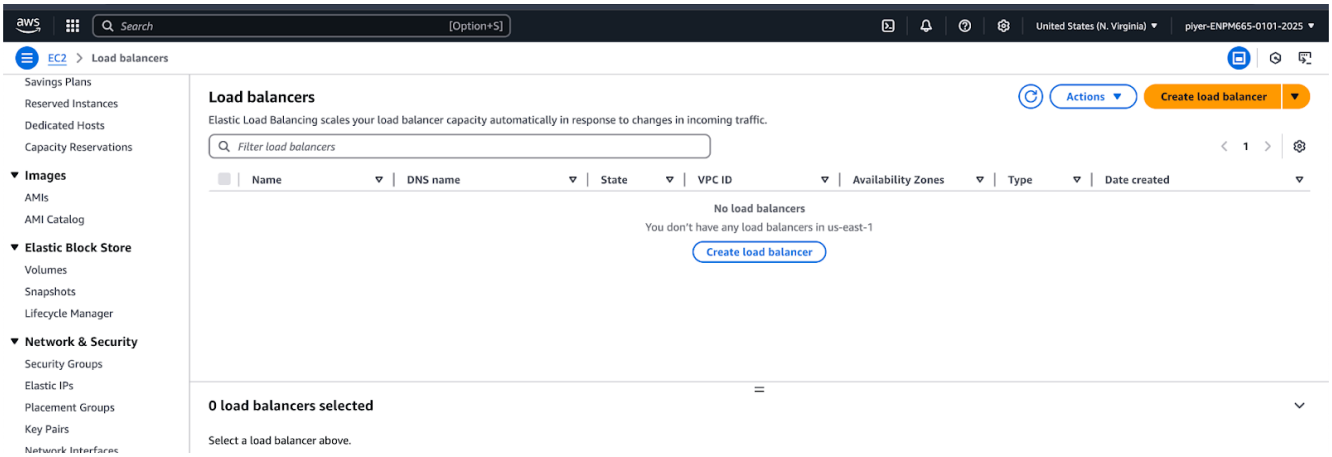


Figure 41: No Load Balancers created to distribute traffic

2.5.4 Installation of MariaDB on an EC2 instance instead of AWS RDS

Observation:

It was observed that the database used for the application instance - MariaDB - has been installed on an EC2 system and not on the dedicated AWS RDS platform.

Impact:

The relational database used by the application, MariaDB, is directly running on an EC2 instance instead of AWS RDS. If the instance becomes unavailable or crashes, the database will no longer be accessible, and all its data will be lost. Running it on an instance also means that there are no automated backups, failovers, or high availability.

Since the database and the application both run on the same instance, that will make it very easy for the entire application to be compromised if the instance is compromised.

Risk: High

Recommendation:

It is recommended to

- The database should be migrated to AWS RDS. This would provide additional security and features like automated backups, monitoring, and multi availability zone failovers.
- If it is required to be on an EC2 instance, there should at least be a dedicated instance running just the database. The database should not be running on any instance with other application services on it.

Reference:

Managed SQL Database - Amazon Relational Database Service (RDS) - AWS

Artifacts:

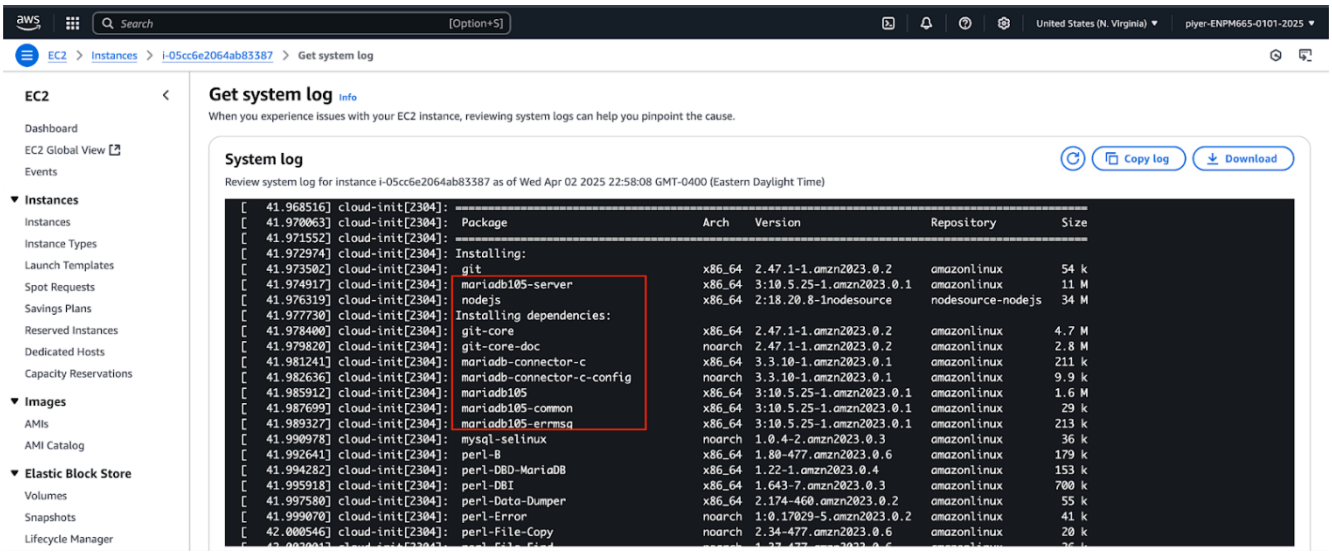


Figure 42: System Logs showcasing installation of MariaDB on the EC2 Instance instead of AWS RDS

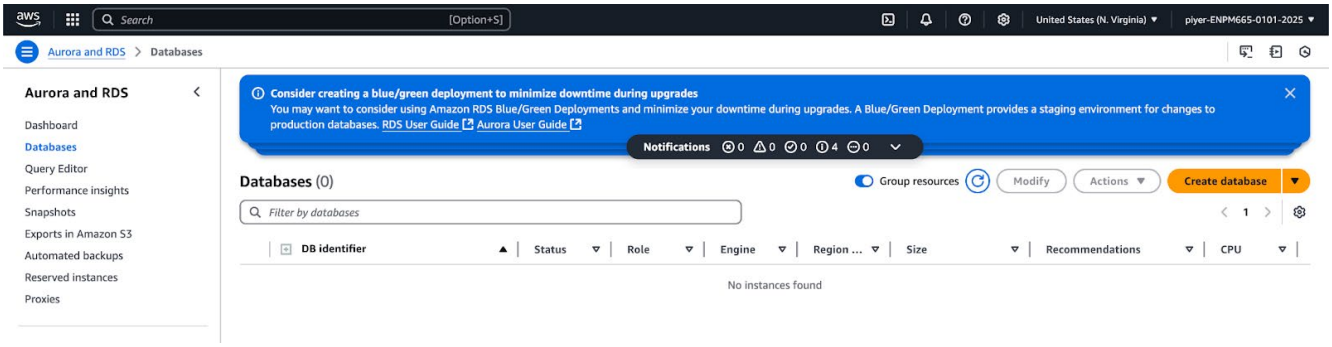


Figure 43: No Databases present in AWS Aurora and RDS

2.5.5 EC2 Instance does not have a Snapshot Policy

Observation:

It was observed that there was no Snapshot Policy established/ defined in the Lifecycle Manager to automate the creation, retention, or deletion of snapshots in the environment.

Impact:

Since there is no Snapshot Policy in place, there is no mechanism for rolling back the instance to a previous secure state. In the event of an instance crash, the whole application instance would need to be manually rebuilt and reinstalled, which leads to downtime and inaccessibility of the application.

Risk: High

Recommendation:

It is recommended to create snapshots of the EC2 instance periodically using Amazon Image (AMI) snapshots so that in case of failure, the system can be restored and returned quickly to its original state, reducing downtime.

Reference:

Create Amazon EBS snapshots - Amazon EBS

Artifacts:

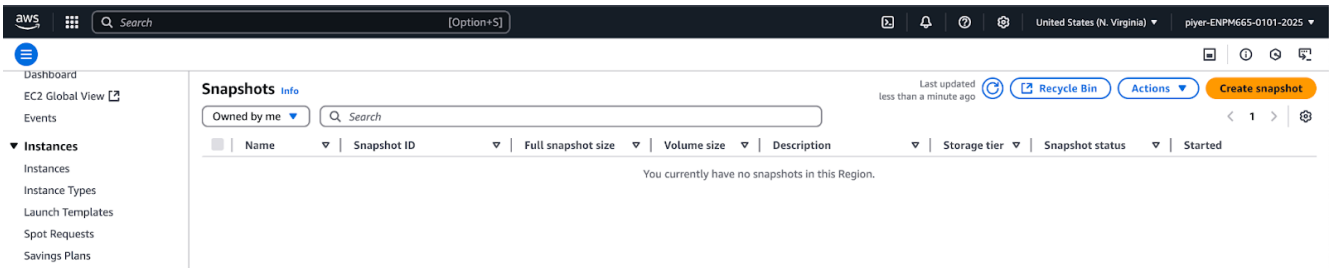


Figure 44: No Snapshots taken for the EC2 Instance

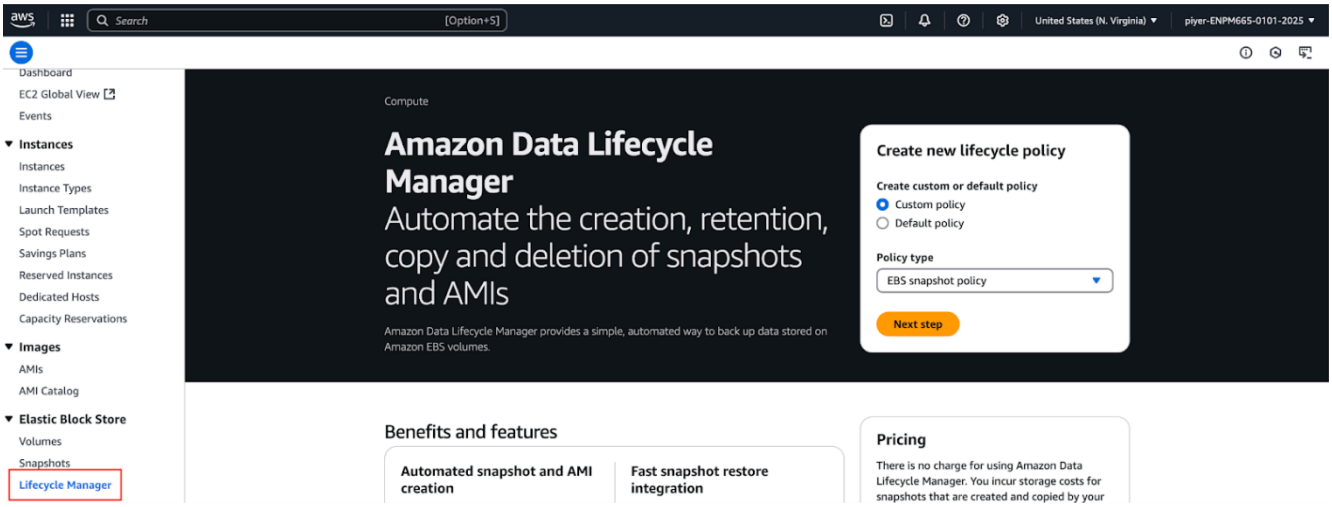


Figure 45: No Policy in the Lifecycle Manager for the automatic creation/deletion/retention of snapshots

2.5.6 No Redundancy in the Network

Observation:

It was observed that there is only a single instance of every networking component in the environment - VPC, Subnet, Route Table, Internet Gateway.

Impact:

Since there is only one VPC in the network, if the primary VPC fails, there is no disaster recovery site or failover VPC that the application can run on. If there is an outage in the only subnet in the network, the application instance running in that subnet will also stop functioning. Failure in the internet gateway/route table will shut off any network access.

Risk: High

Recommendation:

It is recommended that

- There should be more than one VPC, and VPC Peering should be configured between them for easy accessibility and availability of a secondary site.

- Multiple subnets across multiple VPCs and availability zones should be deployed.
- Multiple route tables should be configured so that there is no dependency on a single route.
- There should be more than one internet gateway with failover routes enabled.
- Having a replica of the network or having redundant networking components will improve business continuity and maintain application availability.

Reference:

Solutions on AWS for Networking

Artifacts:

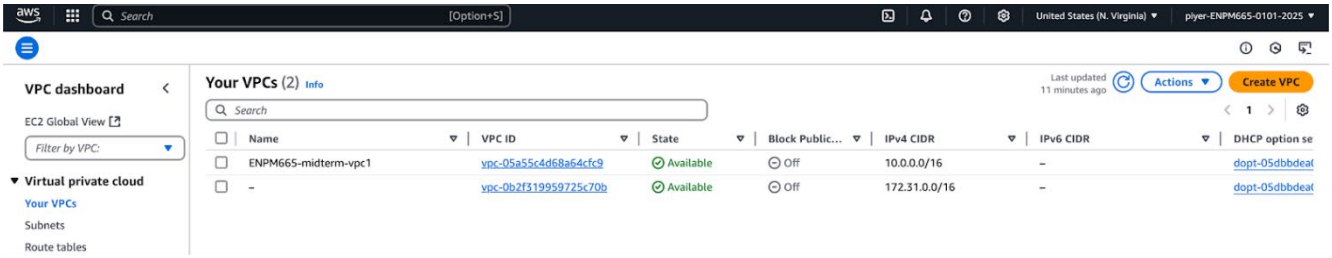


Figure 46: Only 1 VPC Present in the AWS Environment

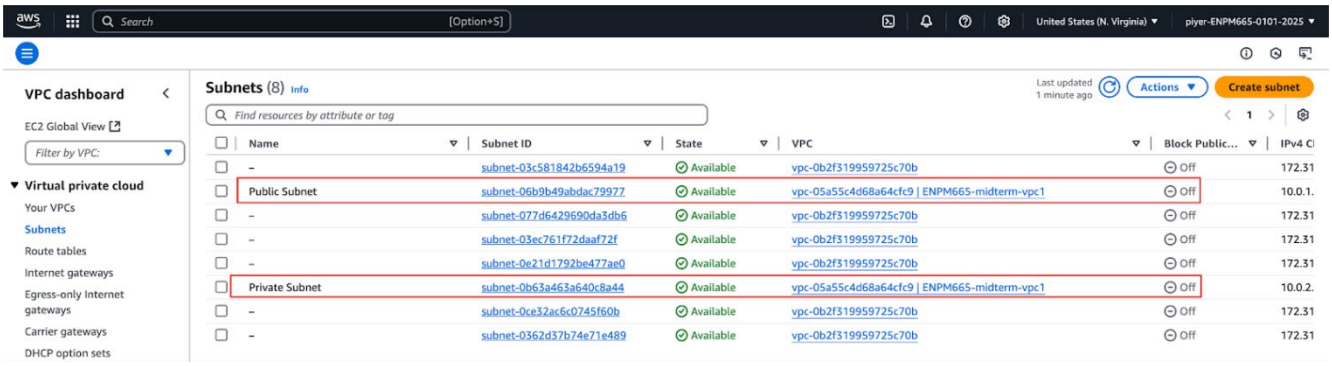


Figure 47: Only 1 Public and Private Subnet Present in the AWS Environment

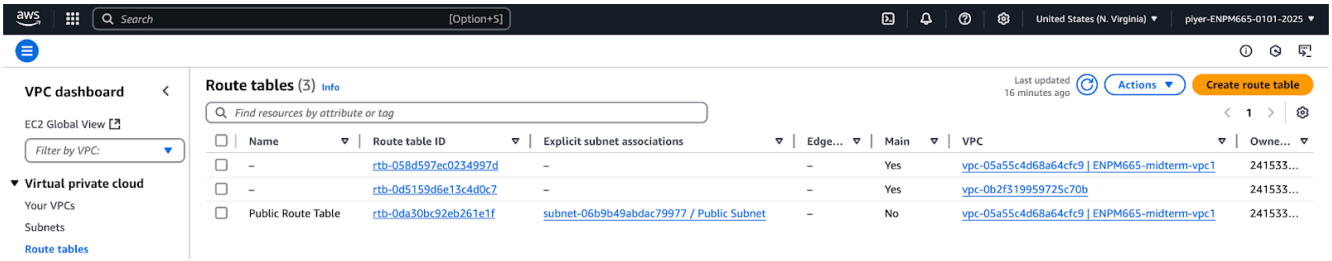


Figure 48: Only 1 Route Table Present in the AWS Environment

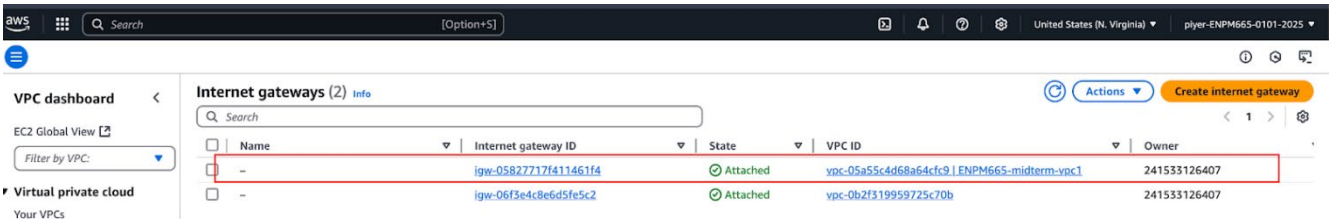


Figure 49: Only 1 Internet Gateway Present in the AWS Environment

2.5.7 No Disaster Recovery Documentation Exists

Observation:

It was observed that Critical Disaster Recovery Documents such as Business Continuity Plans (BCPs), Incident Response Plans, etc., have not been created.

Impact:

There are no concrete disaster recovery plans made for the application infrastructure. Having no Business Continuity Plans (BCPs), Incident Response Plans, RTO (Recovery Time Objective) & RPO (Recovery Point Objective) defined, no operational runbooks/playbooks indicate that in the event of a disaster, as there is no formal process in place, it would result in extended downtime and impact business continuity.

Risk: High

Recommendation:

It is recommended that

- A formalized incident response plan should be created so that the e-commerce application can be brought back to normal functionality with as little disruption as possible.
- Having runbooks/playbooks enables quicker recovery as response and recovery steps like how to restore from a backup, how to bring the network connectivity back up, etc. would be formally documented with detailed steps.
- RTO helps understand how much downtime our systems can afford to have without causing business harm, and RPO defines how much data loss can be handled by the business. Defining these beforehand helps in setting expectations for disaster recovery.

Reference:

Disaster Recovery Service - AWS Elastic Disaster Recovery - AWS

Artifacts:

Resources (10)

Search resources

Logical ID	Physical ID	Type	Status	Module
AttachGateway	IGW|vpc-05a55c4d68a64cfc9	AWS::EC2::VPCGatewayAttachment	CREATE_COMPLETE	-
InternetGateway	igw-05827717f411461f4	AWS::EC2::InternetGateway	CREATE_COMPLETE	-
MidtermVPC	vpc-05a55c4d68a64cfc9	AWS::EC2::VPC	CREATE_COMPLETE	-
PrivateSubnet	subnet-0b63a463a640c8a44	AWS::EC2::Subnet	CREATE_COMPLETE	-
PublicRoute	rtb-0da30bc92eb261e1f|0.0.0.0/0	AWS::EC2::Route	CREATE_COMPLETE	-
PublicRouteTable	rtb-0da30bc92eb261e1f	AWS::EC2::RouteTable	CREATE_COMPLETE	-
PublicSubnet	subnet-06b9b49abdac79977	AWS::EC2::Subnet	CREATE_COMPLETE	-
PublicSubnetRouteTableAssociation	rtbassoc-05bfb63c54d58cfae	AWS::EC2::SubnetRouteTableAssociation	CREATE_COMPLETE	-
VulnerableEC2Instance	i-05cc6e2064ab83387	AWS::EC2::Instance	CREATE_COMPLETE	-
WebServerSecurityGroup	sg-03b62de5d58eae561	AWS::EC2::SecurityGroup	CREATE_COMPLETE	-

Figure 50: No template/resources/parameters for DR infrastructure exists (eg. AWS::SSM::Document) in the CloudFormation Stack.

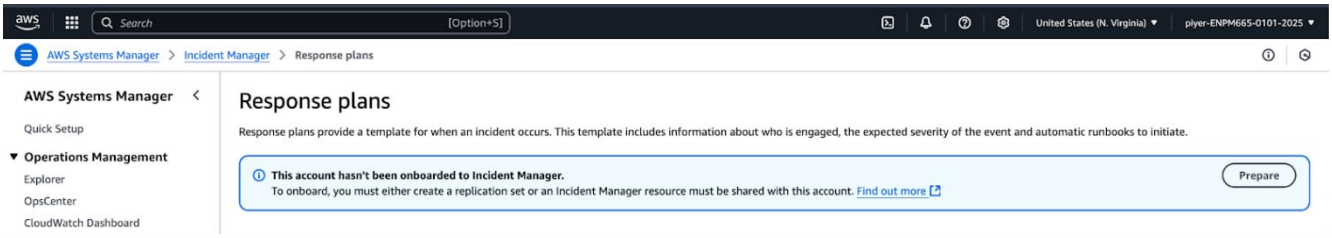


Figure 51: No Incident Response Plan exists in the Incident Response Manager.

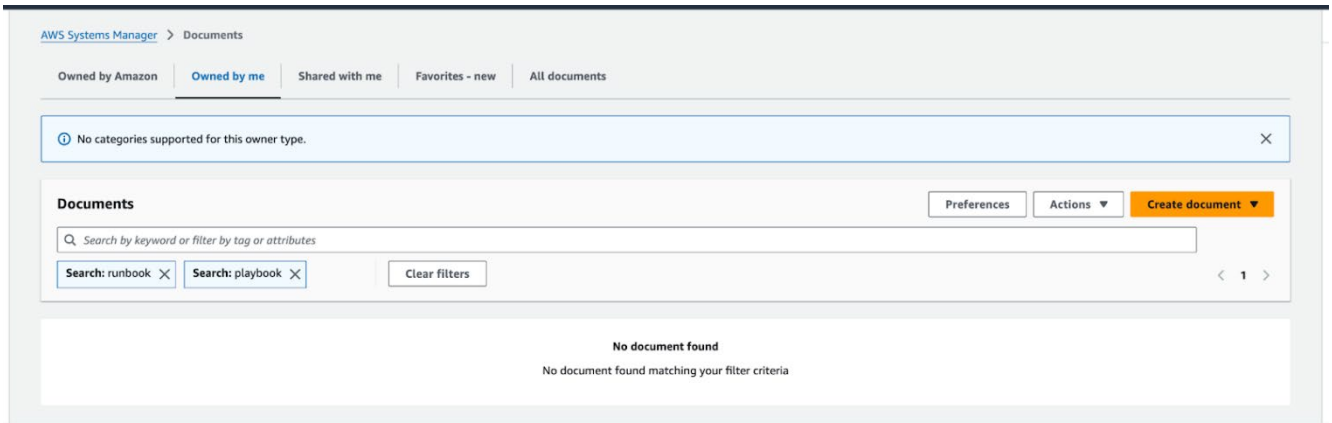


Figure 52: No runbooks/playbooks exist in the AWS System Manager.

2.6 Conclusion

Group 4's comprehensive assessment of the e-commerce application and cloud infrastructure has successfully identified several critical security vulnerabilities across the five pillars of AWS security. Through comprehensive security analysis, several issues and vulnerabilities were uncovered related to misconfigurations, weaknesses in data security, network segmentation, data loss, and extended downtime.

The detailed recommendations provide actionable steps to address these vulnerabilities, strengthen access controls, enhance encryption, and improve overall infrastructure security. By implementing these measures, the e-commerce platform can ensure confidentiality, integrity, and availability of both the application and its data while aligning with industry's best practices.