|  |  |
| --- | --- |
| Name: | Pratyusha Ampolu |
| Lab User ID: | 23SEK3324\_U09 |
| Date: | 10-01-2024 |
| Application Name: | Vulnerable java application |

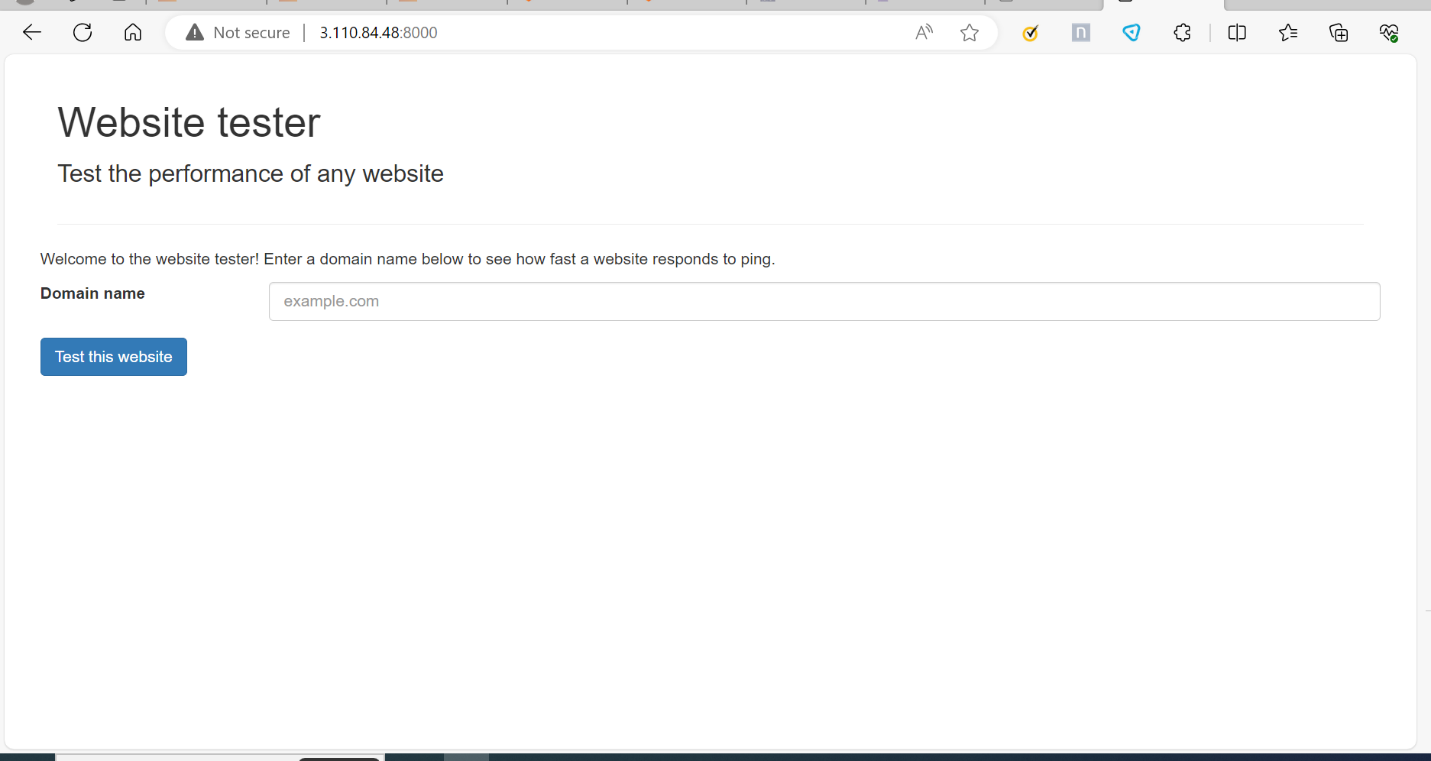
**Follow the below guidelines:**





System Architecture:

(Understand the system and document the physical and logical architecture of the system, use the shapes and icons to capture the system architecture)



My Vm

My docker

Docker container

Browser

IP:8000

Docker image

Define system’s normal behavior:

(Define the steady state of the system is defined, thereby defining some measurable outputs which can indicate the system’s normal behavior)

* The webserver starts listening on defined ports like IP address:8000
* User can access a website hosted on this web server via web browser.
* While opening the page we can see a welcome message and we can find a space to enter the domain name.
* Under that, we can find the button to test the domain name.

Hypothesis:

(During an experiment, we need a hypothesis for comparing to a stable control group, and the same applies here too. If there is a reasonable expectation for a particular action according to which we will change the steady state of a system, then the first thing to do is to fix the system so that we accommodate for the action that will potentially have that effect on the system. For eg: "If one of our database servers fails, our service will automatically switch to a backup server, and users will not experience any downtime or data loss.")



Engineers intentionally shut down a database server during non-peak hours to observe how the system reacts. They expect service degradation or failures.

Engineers test the systems capacity by gradually increasing user load until it reaches and surpasses the expected operational limits.

**Known**

.

Through load testing engineers discover that a particular microservice experiences significant latency only when multiple users concurrently access a specific feature.

Engineers randomly throttle network bandwidth or introduce intermittent latency into different parts of the system to simulate unpredictable real world conditions.

**Unknown**

**Unknown**

**Known**

.

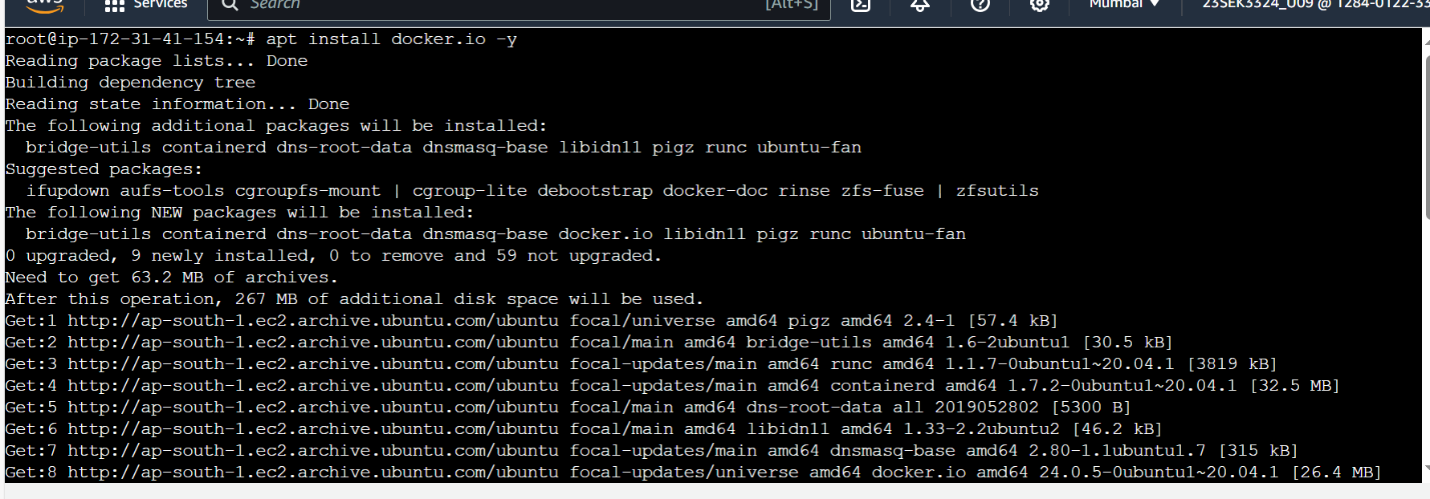
Experiment:

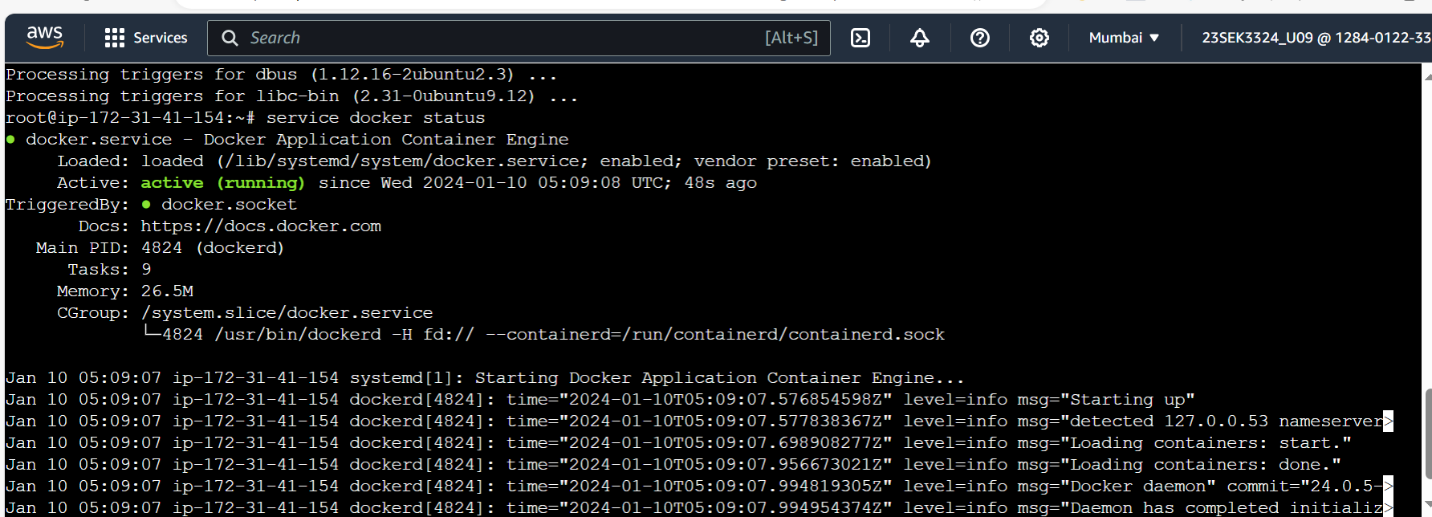
(Document your Preparation, Implementation, Observation and Analysis )

Overview: We are using several tools to perform vulnerability scan.

Methodology:

Step-1: Install docker and make sure that docker is running.





Step-2: Build the image locally or use ghcr.io/datadog/vulnerable-java-application

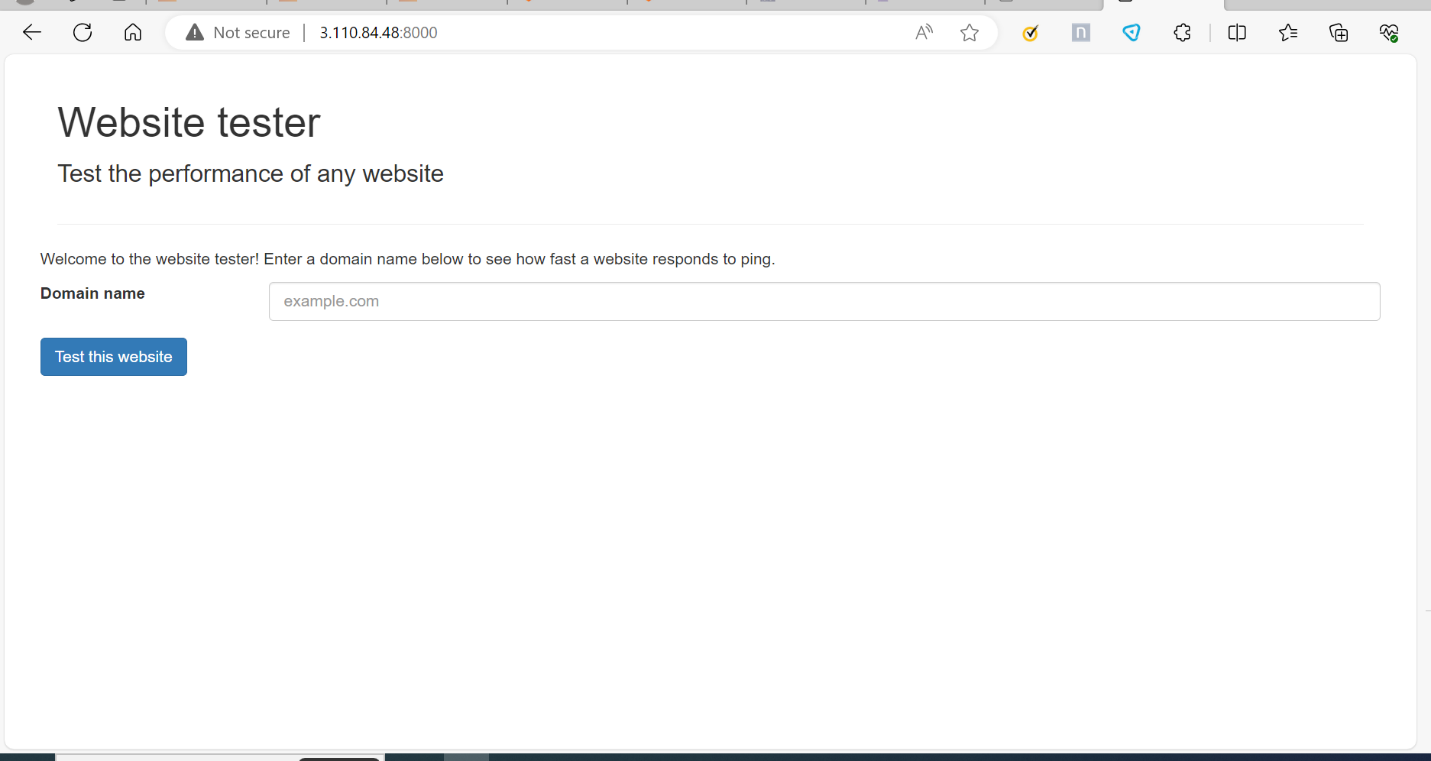
Step-3: Create the container using the image you build before.

The command is to create container is

**docker run --rm -p 8000:8000 ghcr.io/datadog/vulnerable-java-application**

step-4: Live the application using http://<ipaddress>:8000

we get,



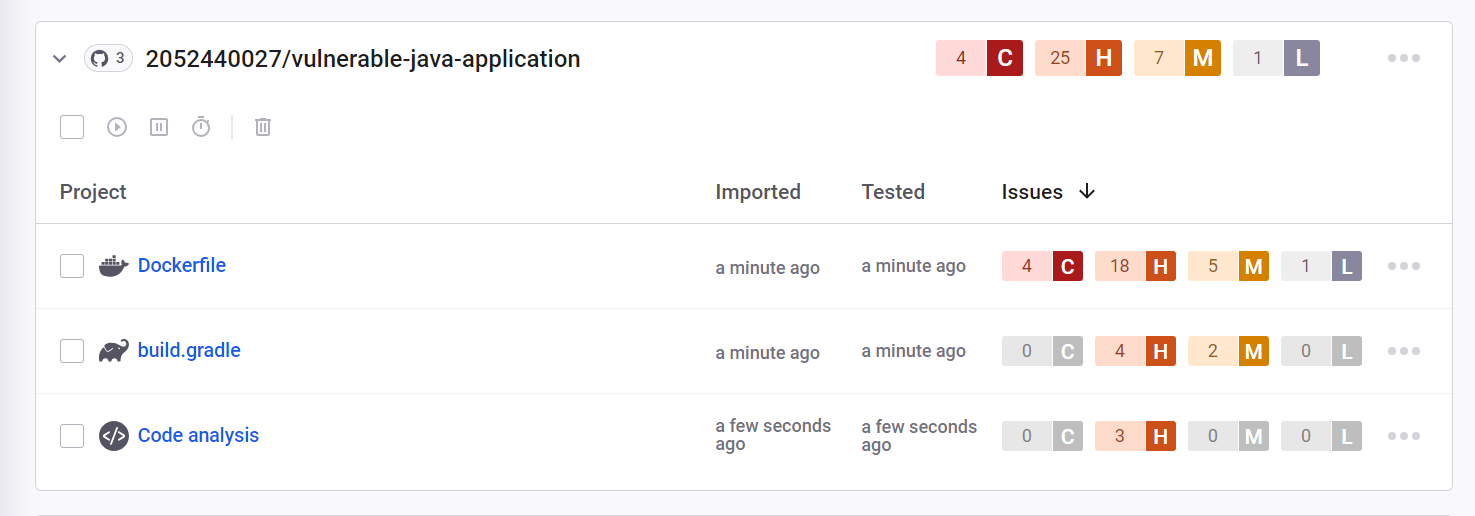
Tools: The following are the tools used in this Project.

1.SNYK

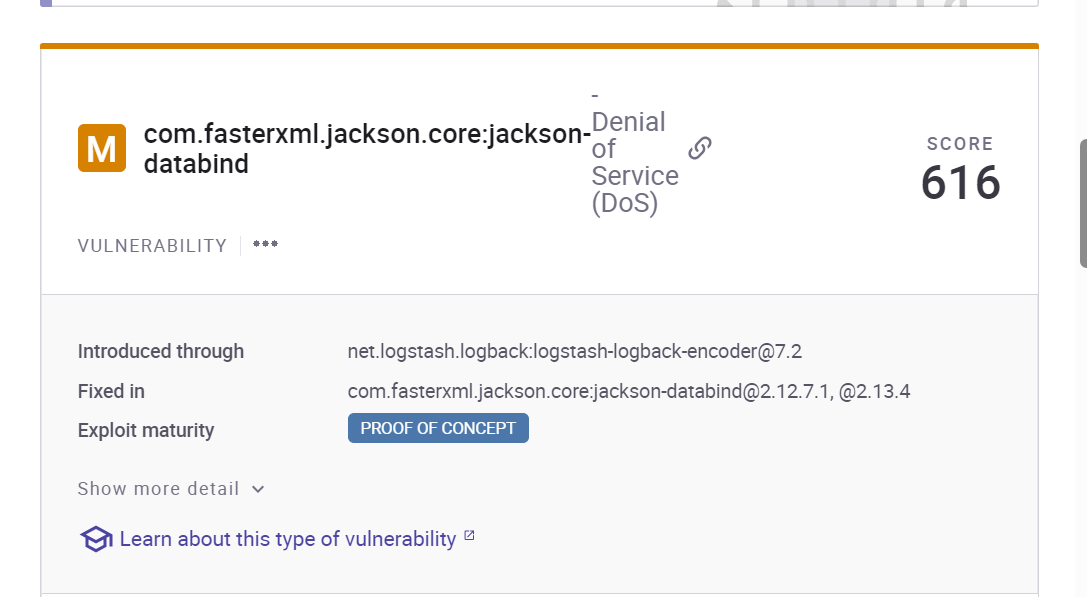
2.OWASP ZAP

Observation:

SNYK:



We got, 4 critical, 25 High, 7 medium and 1 low.



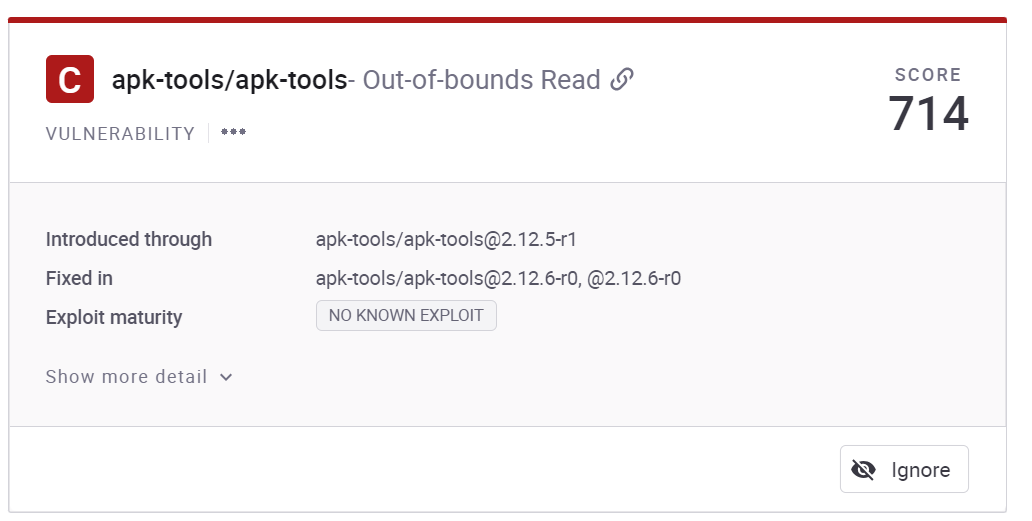
Issue: Denial of service

Solution:

Monitor network and system logs for unusual traffic patterns.

If possible, work with your internet service provider(ISP) to increase network bandwidth to absorb the attack traffic.

Keep all systems including operating systems and software up to date with the latest security patches.

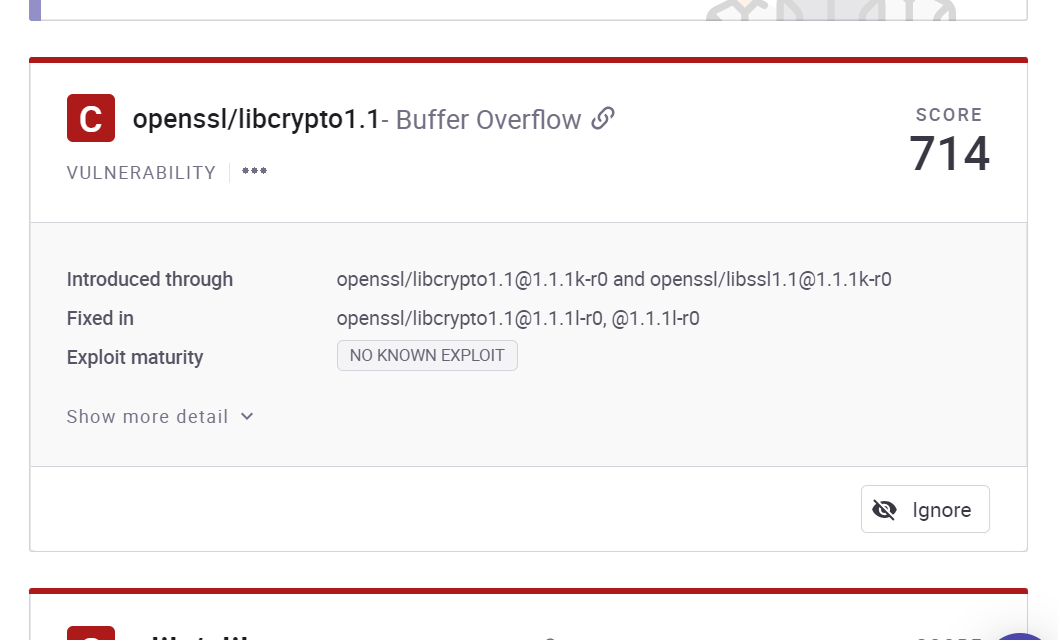


Issue: Out-of-bounds Read

Solution:

Resolving this issue involves carefully examining the code, identifying the specific location of the issue and implementing corrective measures to prevent unintended memory access.

Regular code reviews, testing, and static analysis tools should be integrated into the development process to catch such vulnerabilities early and avoid potential security threats.

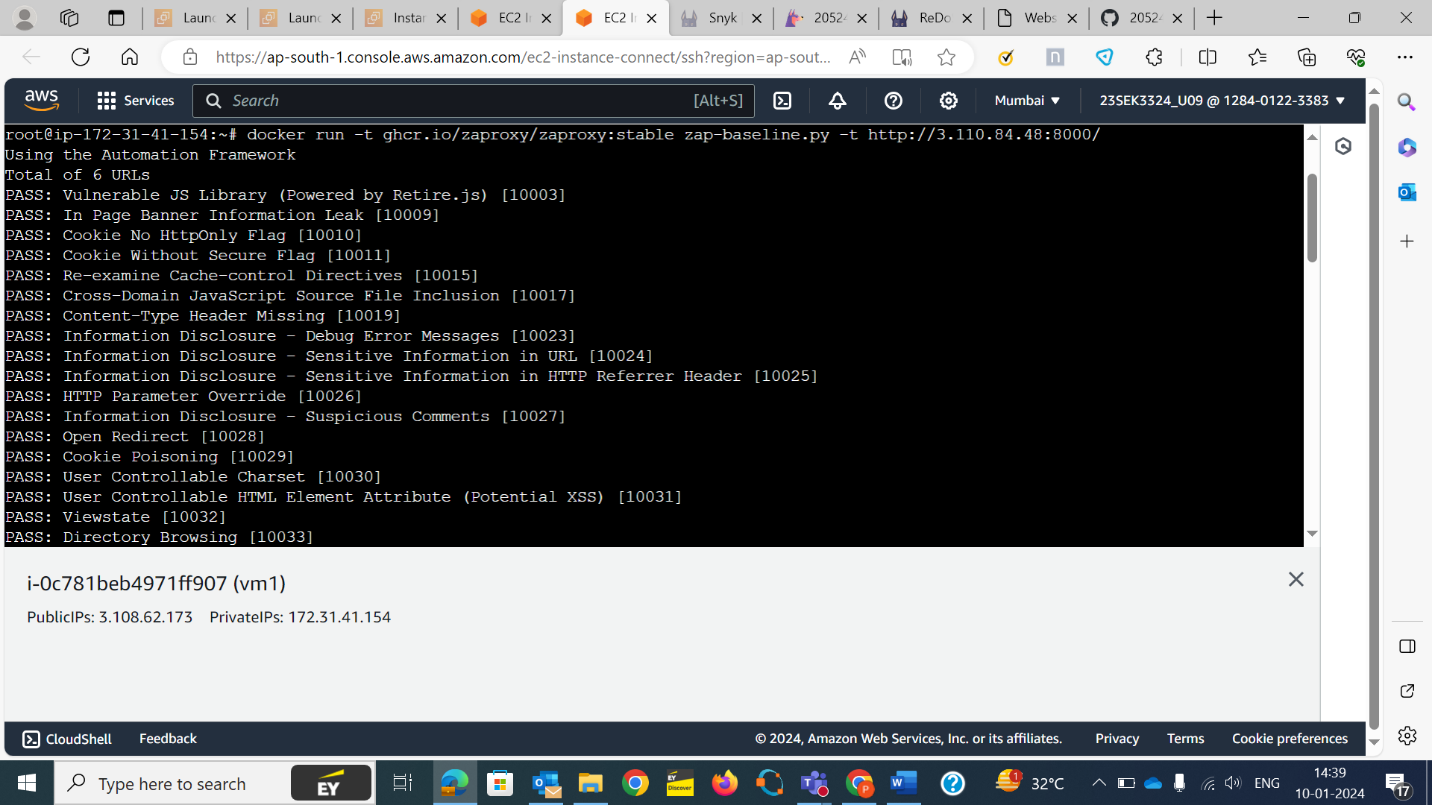


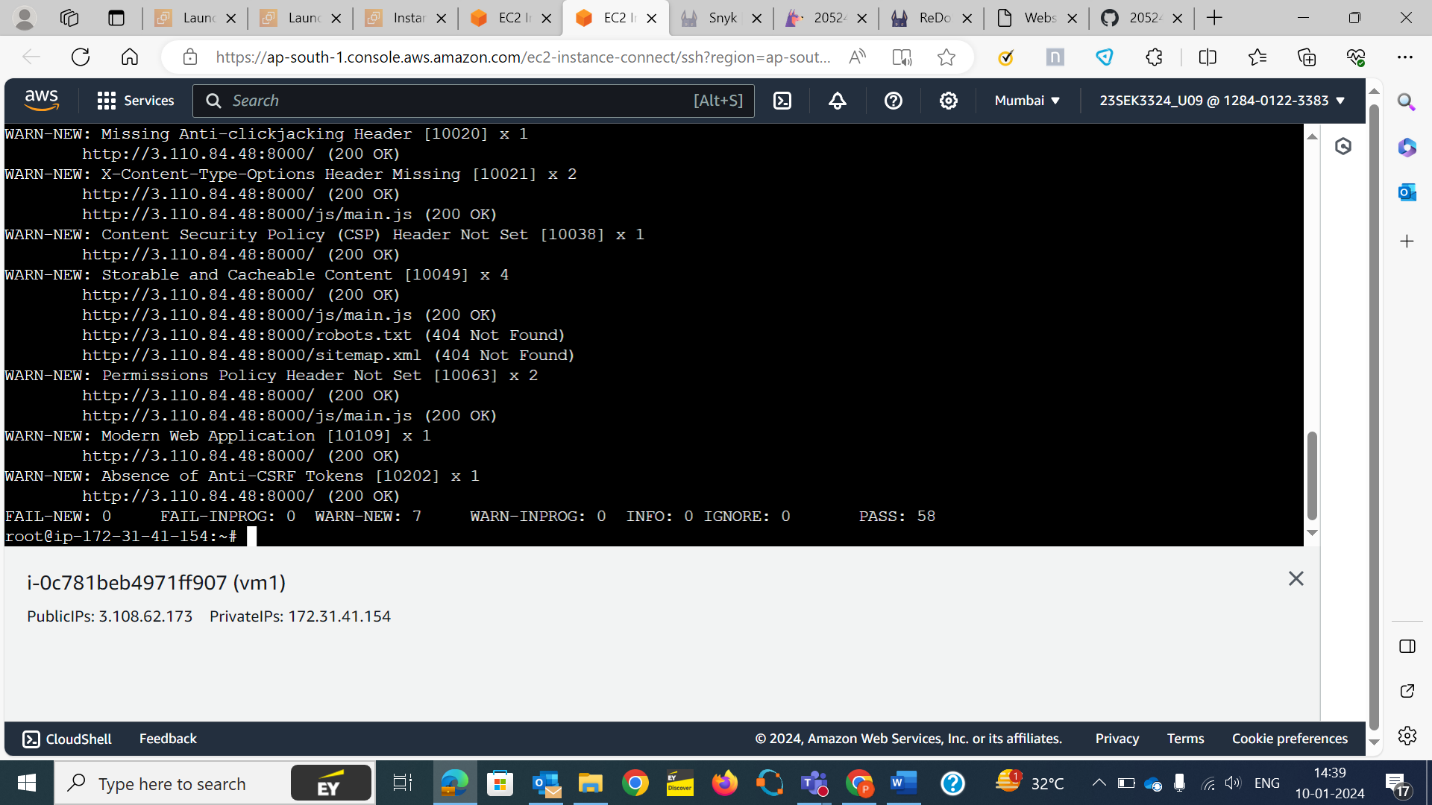
Issue: Buffer overflow

Solution:

This issue can be resolved by adopting secure coding practices such as bound checking and using memory safe programming languages like Rust, Ada etc can reduce the risk of buffer overflows.

Using OWASP ZAP:





Issue: Missing Anti-Clickjacking header

Solution: To address the missing anti-clickjacking header issue, you can add the X-Frame options header to your web servers responses.

This header helps prevent clickjacking attacks by controlling whether your website can be embedded in an frame.

Issue: Permissions policy header not set

Solution:

This issue can be resolved by including permissions-policy header in your servers responses. This header allows you to control which browser features your site can access.

Adjust the permissions based on your needs and make sure to restart or reload your web server for the changes to take effect.