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

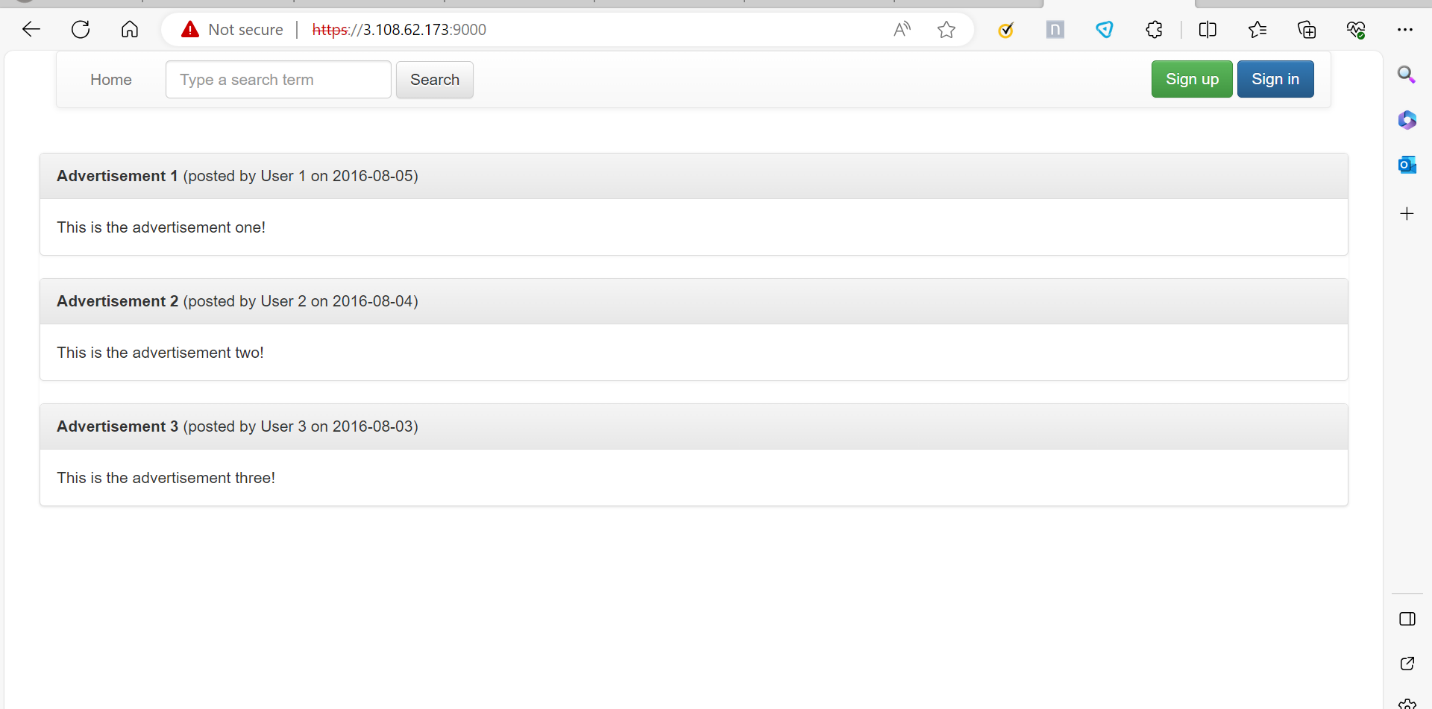
**Follow the below guidelines:**





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:9000

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)

* This is a simple and self-contained java web application with security flaws.
* The application uses Spring Boot and an embedded H2 database that resets every time it starts.
* If you break it just restart and everything will be reset.
* The application will run on HTTPS port 9000.

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.")



Even when a server is suddenly stopped

the application is accessible.

The systems resource limits, especially in terms of CPU, memory and storage are well understood under normal conditions.

**Known**

.

.

Scaling mechanisms are assumed to handle increased load, but there may be specific scaling challenges or limitations in certain scenarios that have not been identified.

We don’t know what will happen if we shut down the docker.

**Unknown**

**Unknown**

**Known**

.

**Experiment**:

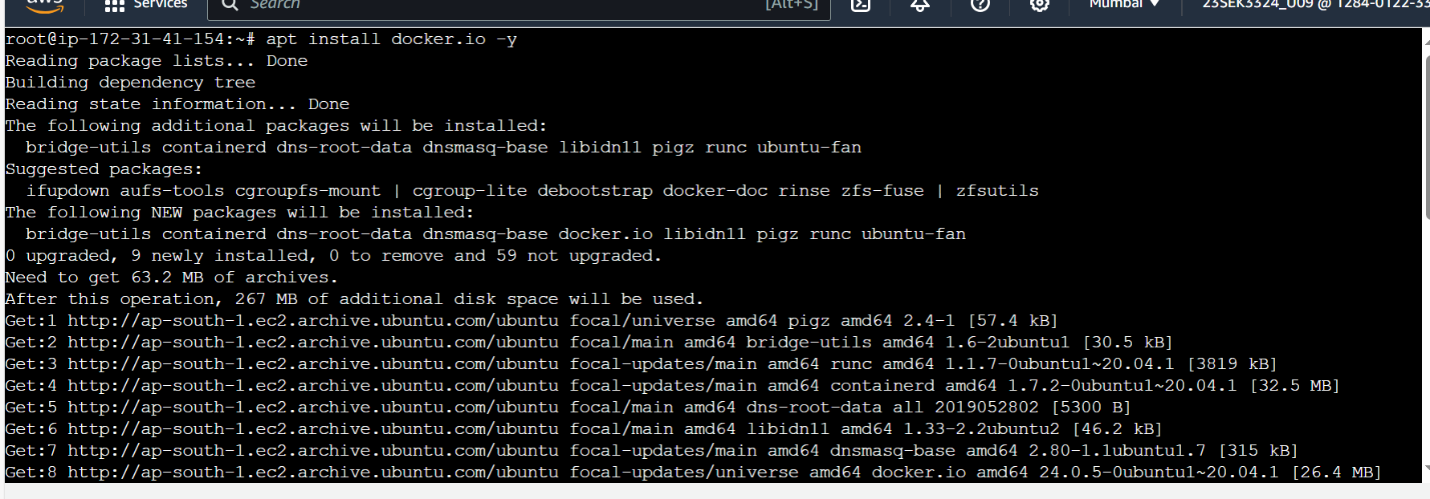
(Document your Preparation, Implementation, Observation and Analysis )

Overview: Vulnerable java web application is java based vulnerable application. We are performing several tools against this application to know the behavior.

Methodology:

Step-1: Create an EC2 instance and install the docker in that.

To install the docker, we can use the command

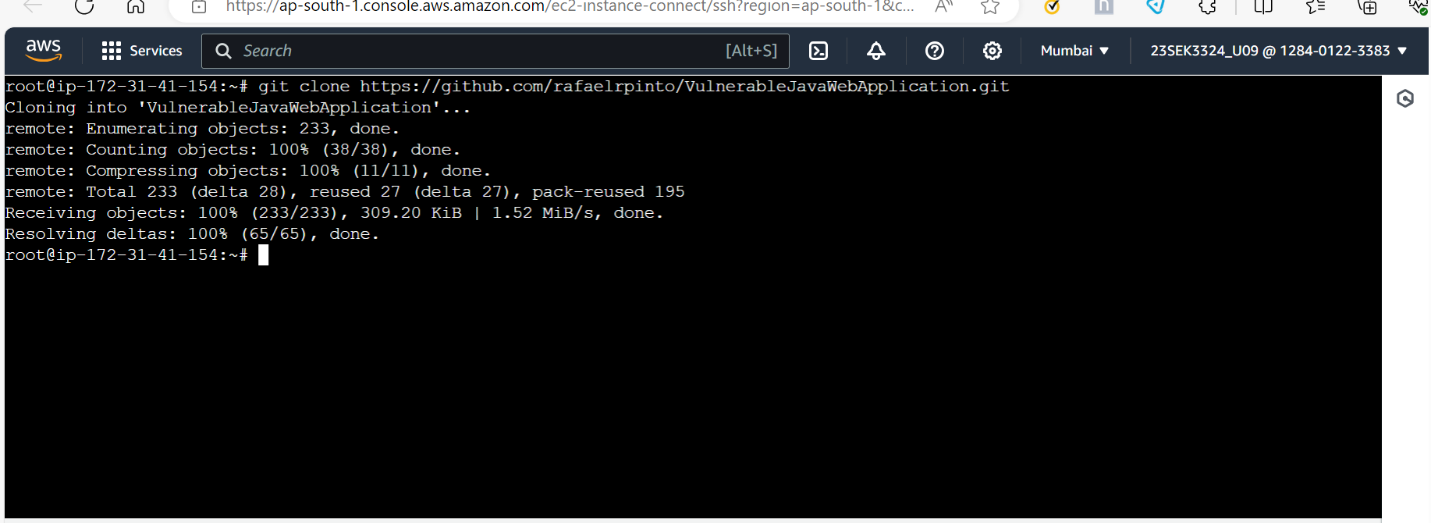


Make sure that docker is running

A screenshot of a computer

Description automatically generated

Step-2: clone the git repo into the VM.



Step-3: Build a docker image using the the docker file in the repo.

A screen shot of a computer

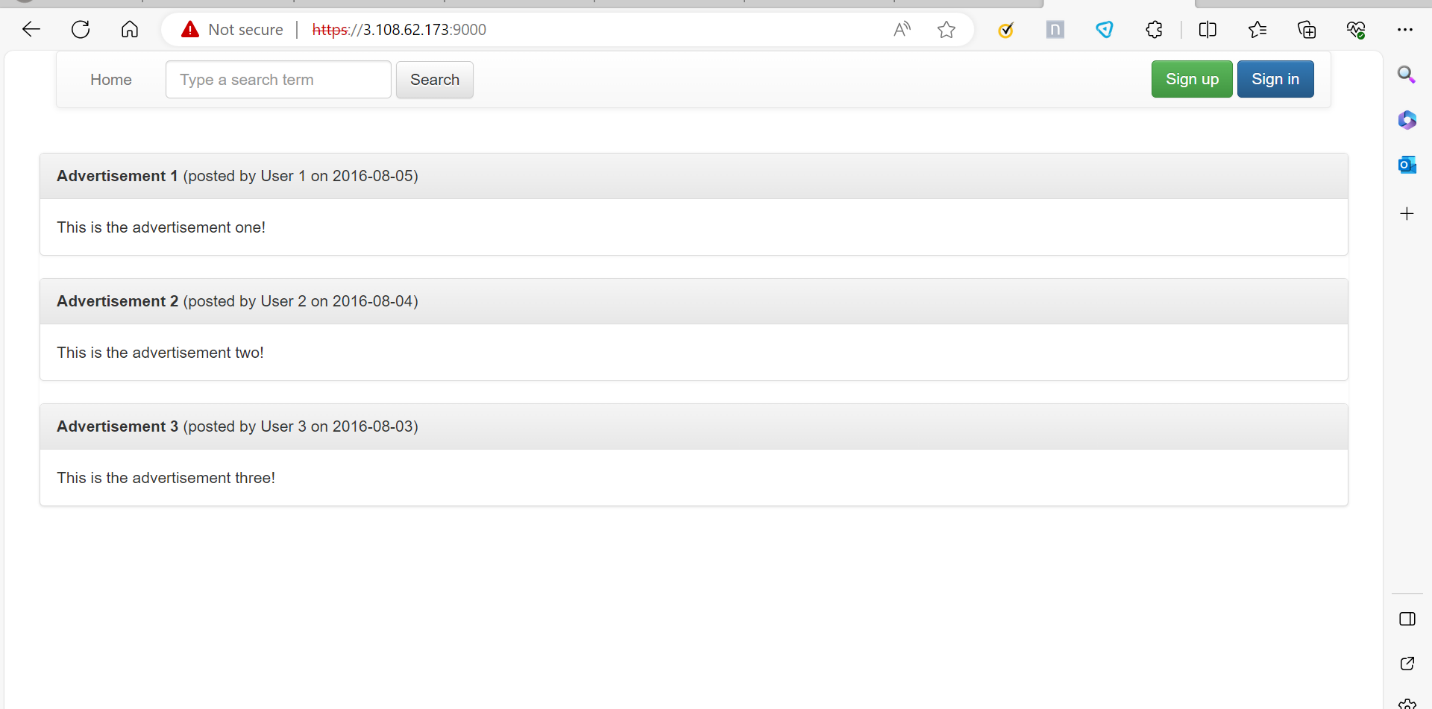
Description automatically generated

Step-4: Create a container for the image we build.

A screenshot of a computer

Description automatically generated

Step-5: we can live the application using https://<ipaddress>:9000/

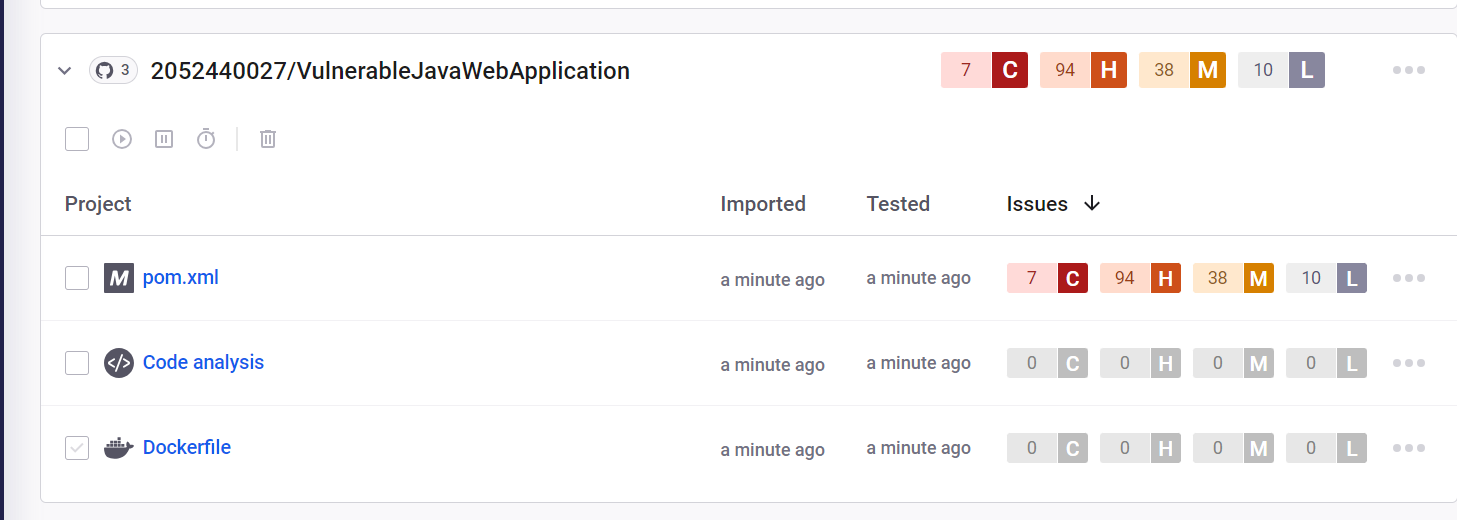


Tools: The following tools are used in this project.

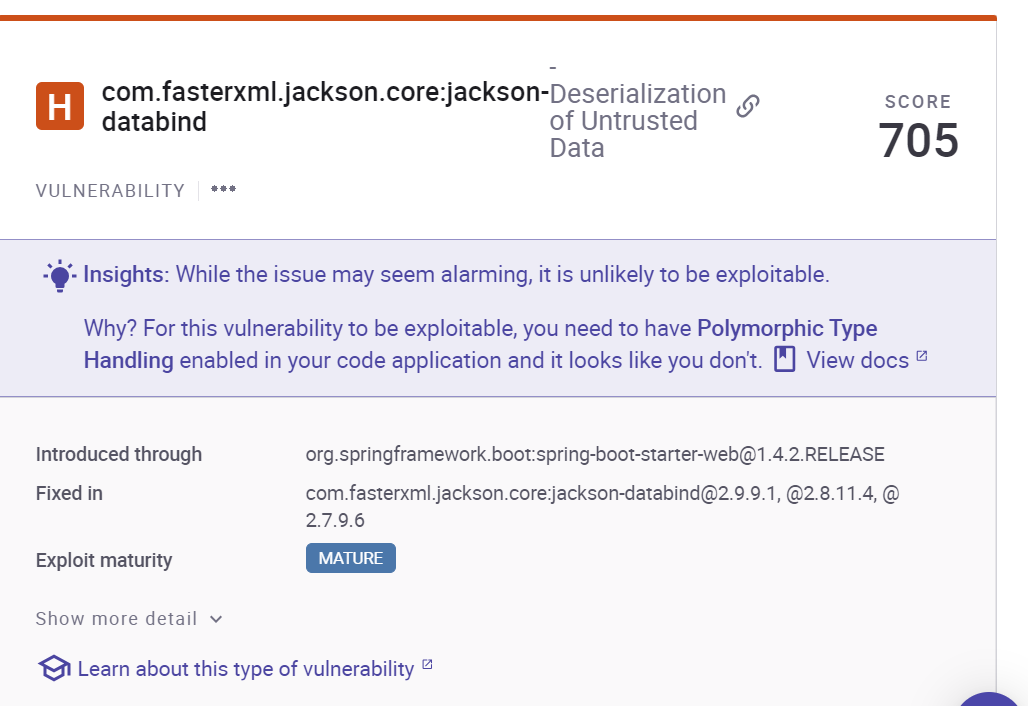
1.SNYK:

2.OWASP ZAP

By using the snyk tool we can found vulnerabilities of our repository.



Issue:



Solution: Avoid having unnecessary types or gadgets available that can be leveraged for malicious ends. This limits the potential for the unintended or unauthorized types and gadgets to be leveraged by the attacker. Add only acceptable diseases to an allowlist.

Issue: Improper input validation

Solution: Directly convert your input type into the expected data type, such as using a conversion function that translates a string into a number.

After converting to a expected data type, ensure that the inputs values fall within the expected range of allowable values and that multi field consistencies are maintained.

Issue: code injection occurs when malicious code is inserted through user input, posing a threat to system security.

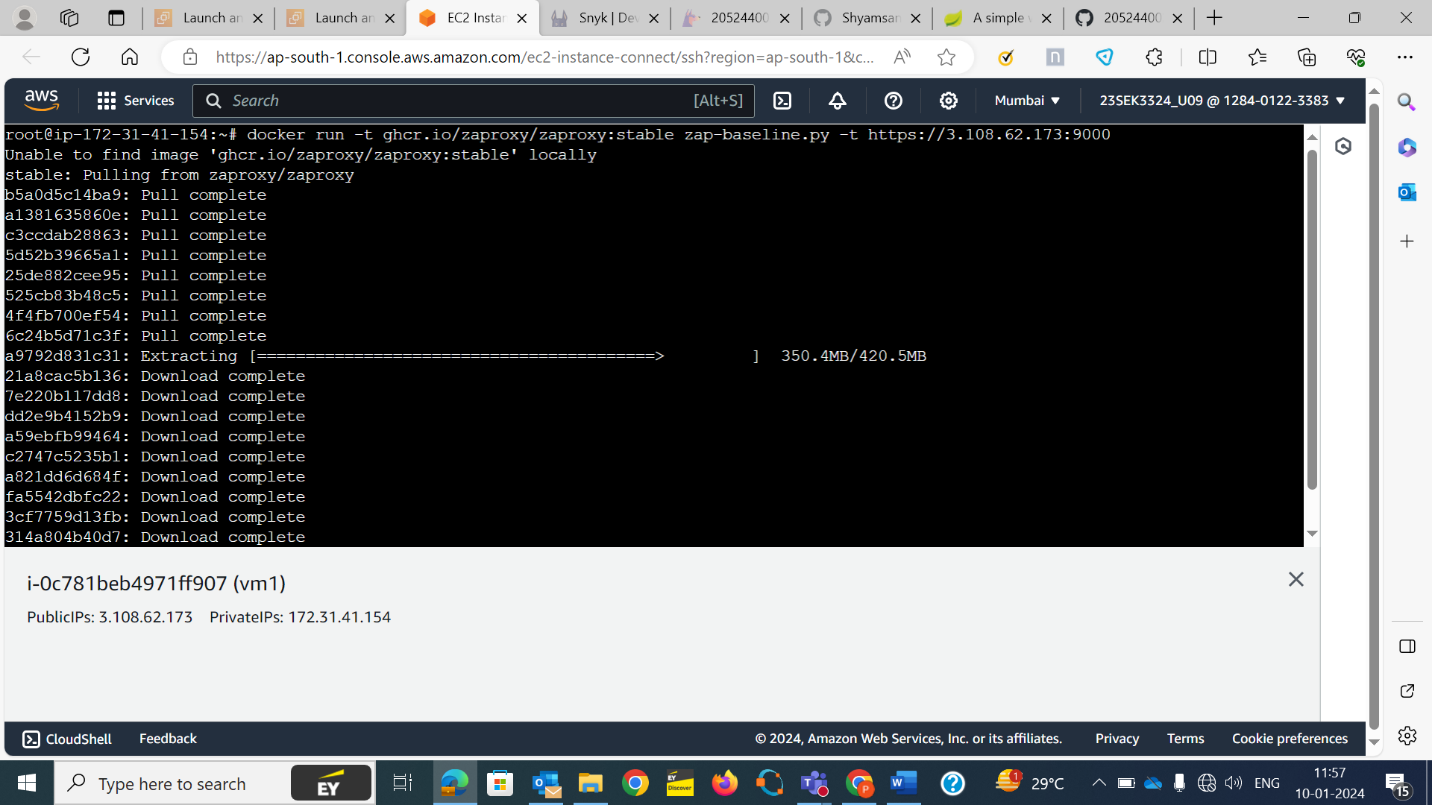
Prevention involves re-evaluating dynamic code execution, particularly in scenarios involving user input, and applying secure code practices. Regularly updating the library is essential for mitigation.

Solution: To mitigate the arbitrary code execution vulnerability , promptly update to the latest security version. Additionally, adopt secure coding practices, validate user inputs and limit dynamic code execution.

By using zap we can scan vulnerabilities.

Step-1: Setup owasp zap in the VM.

Cmd: docker run -t ghcr.io/zaproxy/zaproxy:stable zap-baseline.py -t https://3.108.62.173:9000



Issue: Session management response identified

The given response has been identified as containing a session management token. If the request in the context which has a session management method set to Auto-Detect , then this rule will change the session management to use the tokens identified.

Solution: This is an informational alert rather than a vulnerability and so there nothing to fix.

Issue:

Depecated feature policy header set: The header has now been renamed to permissions-policy.

Solution: Ensure that your web server, application server, load balancer etc is configured to set the permissions-policy header instead of the feature-policy header.