|  |  |
| --- | --- |
| Name: | Arpan Nyati |
| Lab User ID: | 23SEK3324\_U04 |
| Date: | 10-01-24 |
| 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)

Web Server

User Browser

Ping Service

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 Vulnerable Java Application is a website ping testing application. Under normal operation conditions, it takes a domain name as input and return the ping statistics. But the code is buggy and there are server side request vulnerability and command injection vulnerability.

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.")



**Known**

System is prone to expose metadata of the server on malicious code injection

When the applications is running in AWS, Azure or GCP, it can often be exploited to retrieve instance metadata credentials.

942236

**Unknown**

Command execution in container with malicious input

The validation is buggy, we can start the input with domain name and execute in the container.

**Unknown**

**Known**

Experiment:

(Document your Preparation, Implementation, Observation and Analysis )

**Preparation**

1. Created an AWS t2.medium instance with Ubuntu OS

2. Connected using SSH through command line

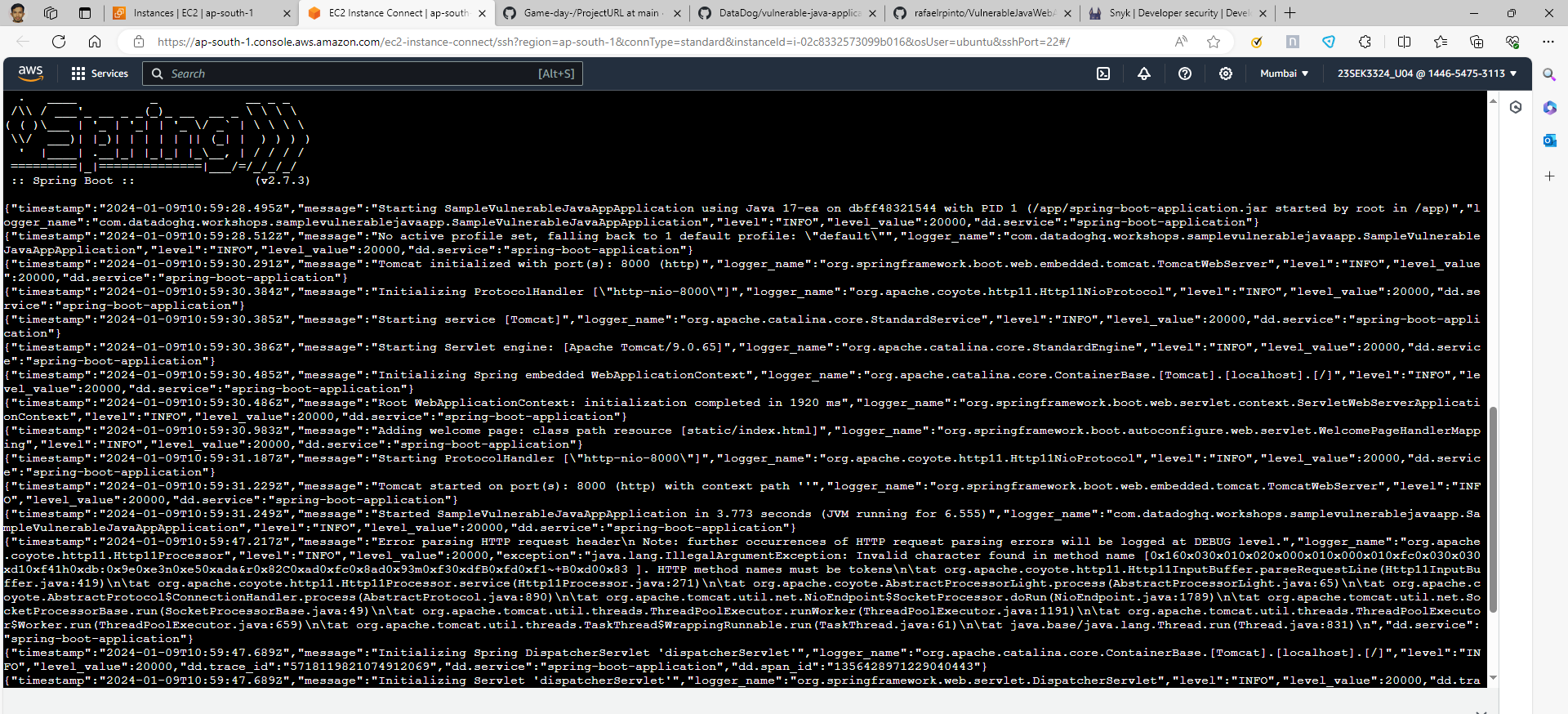
**Implementation**

1 . Updated the system using ‘sudo apt update’.

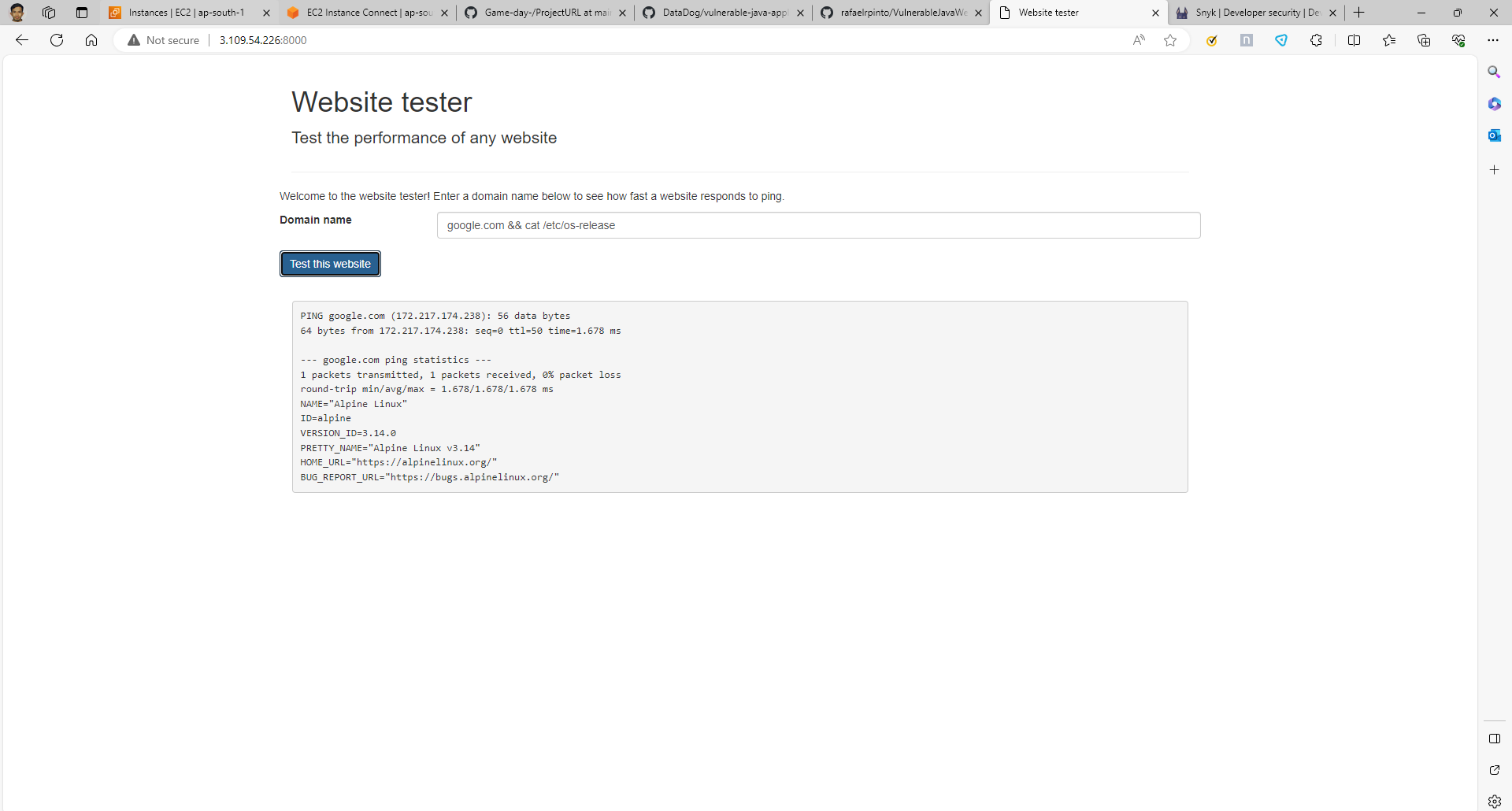
2. Install Docker: Run the installation commands for Docker.

3. Confirm Docker has been installed correctly by running: `sudo docker run hello-world`. This command downloads a test image and runs it in a container.

4. Build and Run your server: Build your Docker image and run it accordingly.



Live Server on AWS EC2



Live Web APP of Vulnerable Java Application.

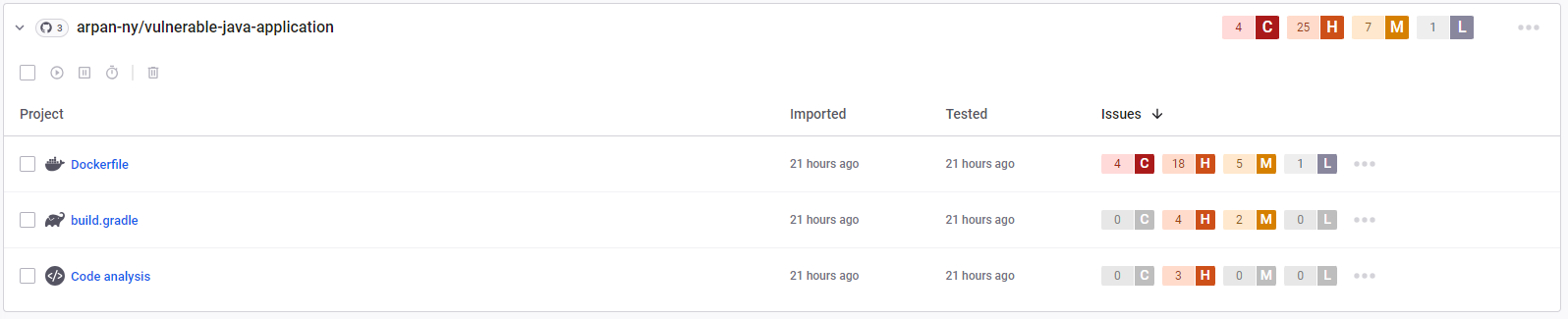
**Observation**

1. Install Snyk: Install the Snyk CLI tool, which you can use to run a security vulnerability test on your machine. Here, you can install it using npm by running `npm install -g snyk`.

2. Authenticate your session: Use `snyk auth` command, it will guide you to the Snyk website to obtain a token. After that Snyk CLI will be connected to your account.

3. Run a Test: Use the `snyk test` command to start the vulnerability scan. This will analyze your project’s dependencies and report any known vulnerabilities.

4. Interpret the Results: The snyk test command will provide a report of vulnerabilities it encountered, their severity level, and even possible remediation methods.



**Analysis**

There are over 4 critical, 25 high, 7 medium and 1 low vulnerabilities in the vulnerable java application.

Some of them are explained below.

1. CVE-2021-3711:

- Impact: This vulnerability affects OpenSSL, an open-source tool widely used for secure communications. It can lead to a high-severity issue in which an attacker could exploit this vulnerability to trigger a buffer overflow with potential for arbitrary code execution.

- Mitigation: Updating OpenSSL to versions 1.1.1l, 1.0.2za or newer, which contain the patch for this vulnerability.

2. CVE-2021-36159:

- Impact: This vulnerability is present in the polkit's pkexec utility in polkit before 0.119. An attacker could install a setuid-root program that could lead to privilege escalation.

- Mitigation: Updating to the latest version of polkit (0.120 or newer), which contains the patch for this vulnerability, is recommended.

3. CVE-2021-46848:

- Impact: This vulnerability is found in the ImageIO framework of Apple macOS. Successful exploitation of this vulnerability could lead to arbitrary code execution.

- Mitigation: Updating to Apple macOS Big Sur 11.6, which contains the patch for this vulnerability, is recommended.

4. CVE-2018-25032:

- Impact: I must apologize as this CVE number either doesn't exist or has not been published publicly yet. I'm unable to provide details on it.