|  |  |
| --- | --- |
| Name: | Deekshith KR |
| Lab User ID: | 23SEK3324\_U08 |
| Date: | 10/01/2024 |
| Application Name: | [**Vulnerable Java Web Application**](https://github.com/D33ksh1th/VulnerableJavaWebApplication) – **o2** |

**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)

AWS

Ubuntu instance

Docker

Container

Web Server

WordPress server

MYSQL Database

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)

In this WordPress web application it is connected to MySQL database which ensures good connectivity and

High availability and is capable of taking user input and responds the actions .

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**

Things we are aware of but don’t understand.

Things we are aware of and understand.

942236

**Unknown**

**Unknown**

**Known**

Things we are neither aware of nor understand.

Things we understand but are not aware of.

Lets see how its done :

This application is deployed using AWS ubuntu instance .

First we are going create Dockerfile which is built using docker build -t where the image will be created.

Then the image is containerized using docker run command and exposed to the port.

We will have to create two container for WordPress and MYSQL database.

Once both are started we will link the MYSQL with WordPress and run it . Make sure the database has its credentials

All set up.

Once the container is ready run the ip in the web browser. The output is as follows:

A screenshot of a computer

Description automatically generated

The tests we are using here is :

A screenshot of a computer

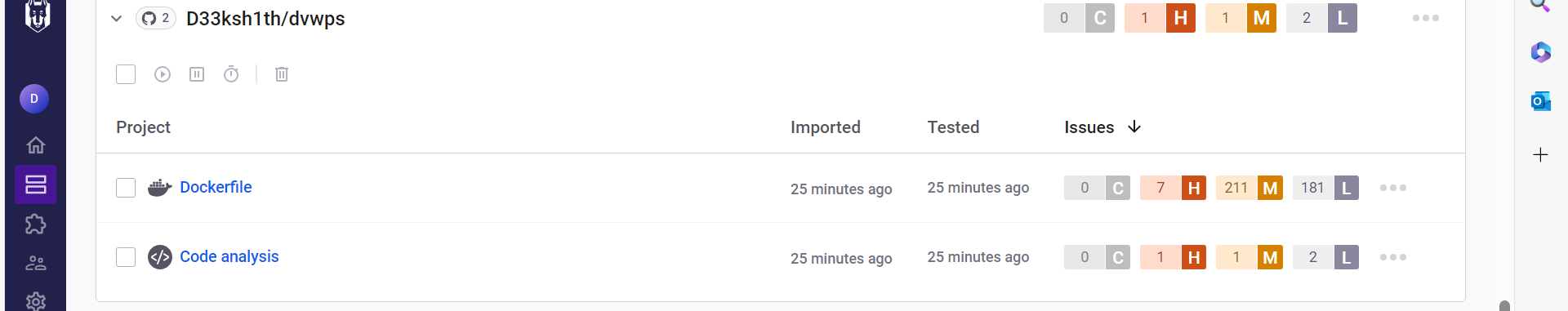
Description automatically generated

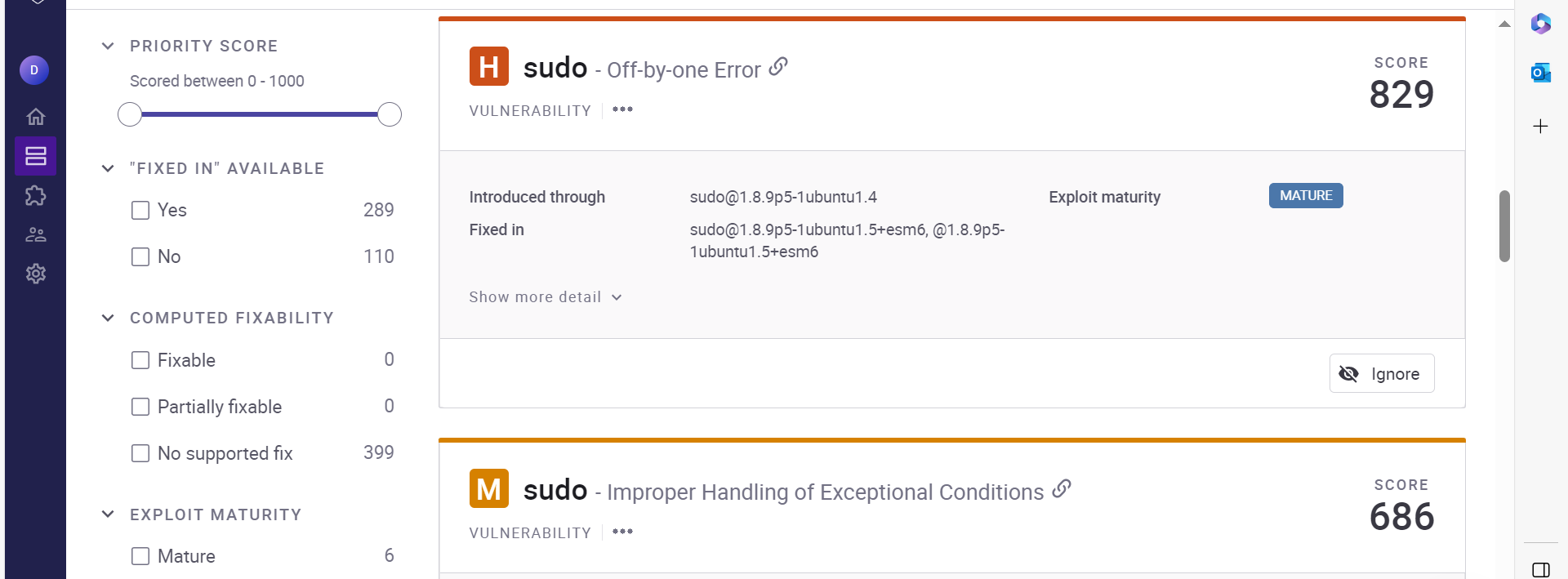
Since they are up and running now we can run the test , we are going to use these tools to test:

1. SNYK
2. ZAP

1 – Using SYNK

We can scan the vulnerability using SYNK by selecting the particular repository where the document lies





Some vulnerabilities are:

CWE-89: SQL Injection:

The vulnerability arises from the improper neutralization of special elements in SQL commands, exposing the application to potential SQL injection attacks.

Mitigation: Implement proper input validation and parameterized queries to neutralize special elements and prevent SQL injection.

CWE-755:

Improper Handling of Exceptional Conditions:

Improper handling of exceptional conditions can lead to unexpected issues or errors.

Mitigation: Implement proper handling mechanisms for exceptional conditions to ensure robustness and prevent unexpected behavior.

Code Issues:

CWE-532:

Insertion of Sensitive Information into Log File: Privacy Leak:

Inserting sensitive information into log files poses a privacy leak risk, potentially providing valuable guidance to attackers.

Mitigation: Avoid logging sensitive information or ensure proper access controls and encryption for log files to prevent privacy leaks.

Mitigation: Ensure that sensitive cookies in HTTPS sessions have the 'Secure' attribute set to prevent exposure over insecure channels.

By addressing these vulnerabilities and implementing the recommended mitigations, the application can enhance its overall security posture and minimize the risk of potential exploits. Regularly monitor and update security measures to stay resilient against evolving threats.

2 Using OWASP ZAP

We can use tool to check vulnerability in the web page simply by using the following command:

docker run -t ghcr.io/zaproxy/zaproxy:stable zap-baseline.py -t <http://localhost/>

the vulnerabilities found are:

Issue: Missing Strict-Transport-Security header.

Mitigation: Enable Strict-Transport-Security to force the use of secure connections.

Content-Security-Policy:

Issue: Missing Content-Security-Policy header.

Mitigation: Implement a Content-Security-Policy to mitigate various types of attacks, including XSS.

Cross-Origin-Embedder-Policy:

Issue: Missing Cross-Origin-Embedder-Policy header.

Mitigation: Add Cross-Origin-Embedder-Policy to control how a document loads cross-origin resources.

Issue: Missing Cross-Origin-Opener-Policy header.

Mitigation: Implement Cross-Origin-Opener-Policy for better control over cross-origin pages.

Issue: Missing Cross-Origin-Resource-Policy header.

Mitigation: Add Cross-Origin-Resource-Policy to control how resources can be shared cross-origin.

Issue: Missing X-Content-Type-Options header.

Mitigation: Enable X-Content-Type-Options to prevent browsers from interpreting files as a different MIME type.

Issue: Missing Clear-Site-Data header.

Mitigation: Implement Clear-Site-Data to clear browsing data for a given origin.

Issue: Mixed content found in scripts.

Mitigation: Ensure all script resources are loaded over HTTPS to prevent mixed content issues.

Issue: Mixed content found in link resources.

Mitigation: Load all link resources over HTTPS to avoid mixed content vulnerabilities

Experiment:

(Document your Preparation, Implementation, Observation and Analysis )