|  |  |
| --- | --- |
| Name: | Arpan Nyati |
| Lab User ID: | 23SEK3324\_U04 |
| Date: | 10-01-24 |
| Application Name: | OWASP JUICE SHOP |

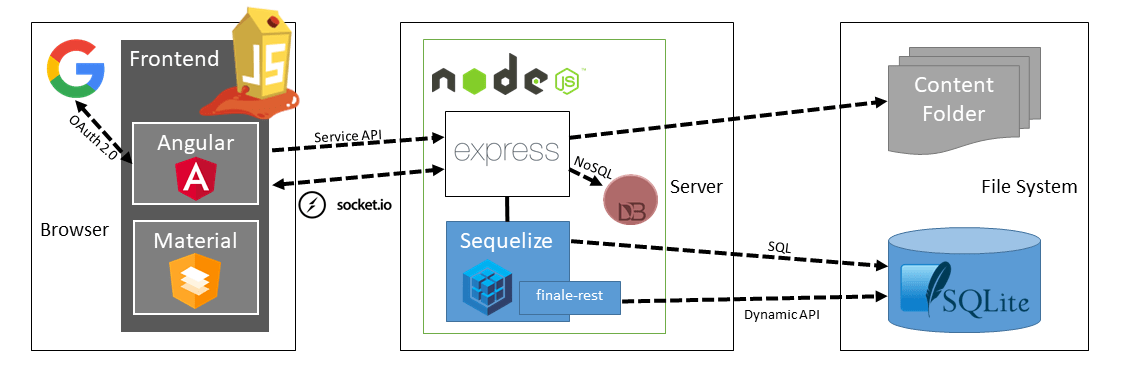
**Follow the below guidelines:**





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)



OWASP Juice Shop System Architecture

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)

OWASP Juice Shop is a web application developed by OWASP (Open Web Application Security Project) which acts as a platform for security testing and awareness-training. The normal behavior of this application mirrors a typical online shopping website or e-commerce platform but it is intentionally designed with security deficiencies for educational purposes.

Normal System Behavior:

1. User Registration: The system allows new users to sign up using an email and password.

2. User Login: Registered users can log in with their credentials.

3. Browse Products: Users can browse through the available products in the online store similar to any ecommerce platform.

4. Product Search: It has a search feature which allows users to locate specific products within the store.

5. View Product: Users can view a specific product to see more details including price, description, reviews and ratings.

6. Add to Basket: Users can add desired products to the shopping cart/basket.

7. Shopping Cart: Users can check, add, or remove items from their basket before proceeding to checkout.

8. Checkout: It allows users to make purchases for the items in their cart.

9. Feedback: Users can provide feedback or complaint about products or the store itself.

10. User Profile: It allows users to update or change their personal information tied to their account.

11. Score Board: This is a distinctive functionality of Juice Shop, where users can see the identified vulnerabilities that have been solved along with some hints about the unsolved ones.

12. Forgotten Password: The application provides a feature to reset the password if the user forgot their password.

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**

OWASP Juice Shop presents a range of security vulnerabilities, understanding both why these vulnerabilities exist and how they can be exploited and defended might not be immediately clear.

When inducing latency into the database, the OWASP Juice Shop application will continue to function and successfully serve users with acceptable performance.

942236

**Unknown**

The Juice Shop uses much different technology, like AngularJS, NodeJS, SQLite, and others. Some vulnerabilities in these tech stack yet to be ascertained.

With the introduction of random resource limitations like CPU and memory, the OWASP Juice Shop application will manage resource consumption efficiently, remain functional, and maintain performance.

**Unknown**

**Known**

Experiment:

(Document your Preparation, Implementation, Observation and Analysis )

**Preparation:**

1. An AWS account was set up with the necessary permissions to create and manage EC2 instances.

2. A t2.medium instance was selected due to its balance of computing power and cost, adequate for the task.

3. An AMI (Amazon Machine Image) with a popular Linux distribution was chosen as the operating system for the EC2 instance.

**Implementation:**

1. The t2.medium instance was launched with the chosen AMI on AWS console.

2. It was ensured that the instance is updated by running "sudo apt update" and "sudo apt upgrade" commands on the instance's terminal.

3. Docker was installed on the instance by using the commands: "sudo apt install docker.io" and then started using the command: "sudo systemctl start docker" and enabled to start at boot with "sudo systemctl enable docker".

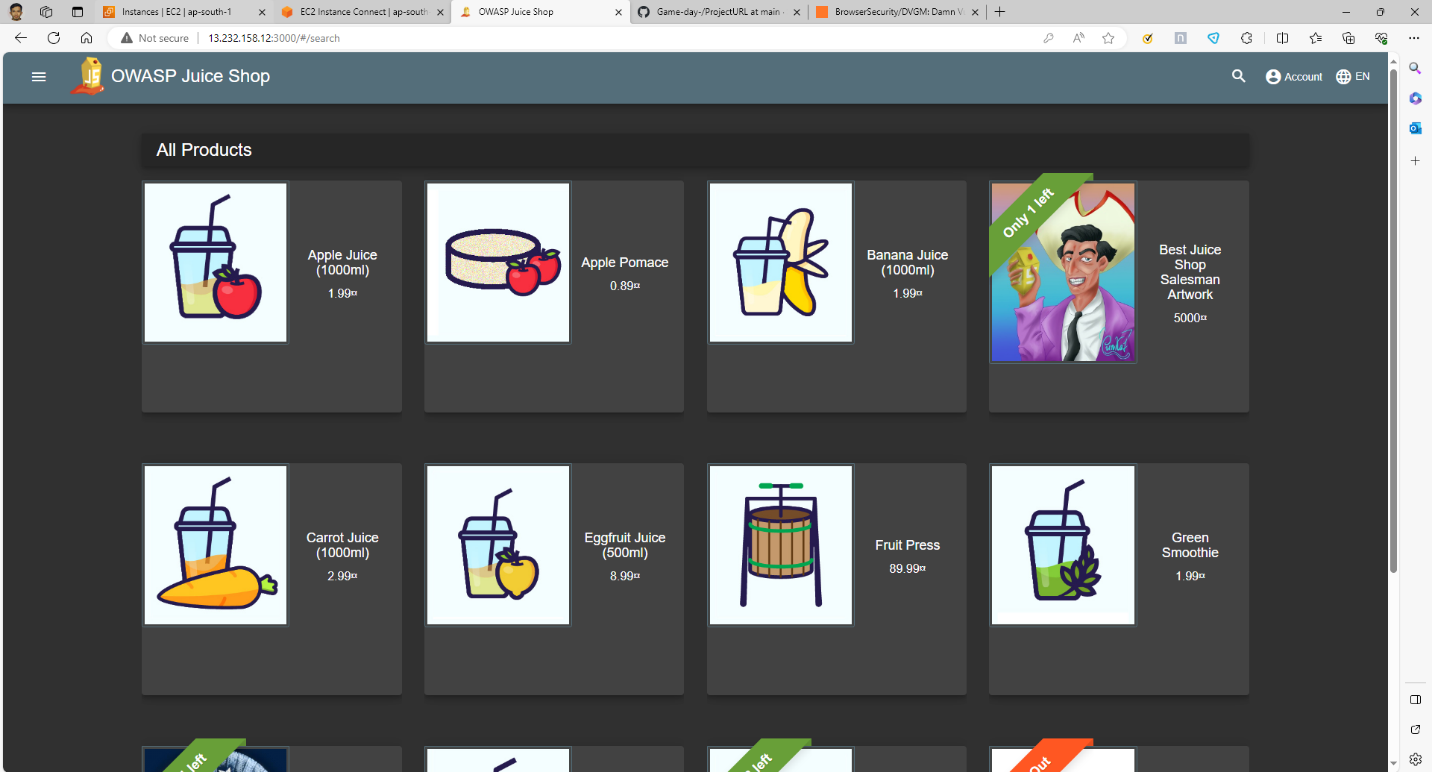
4. Docker's successful installation was confirmed by running the "docker --version" command.

5. OWASP Juice Shop, an intentionally insecure application, was pulled from Docker Hub using the command: "docker pull bkimminich/juice-shop".

6. The docker instance of the OWASP Juice Shop was run using the command: "docker run -d -p 80:3000 bkimminich/juice-shop".

This made the local server live by mapping the container's port 3000 to the instance's port 80.

7. Once the server was live, it was confirmed by visiting the public IP address of the EC2 instance on a web browser.



Web Page of OWASP Juiceshop

A screenshot of a computer

Description automatically generated

Live Server on AWS EC2

**Observations:**

1. The server was running successfully, presenting the OWASP Juice Shop's welcome page.

2. Snyk, a vulnerability scanner, was implemented at this stage to scan the server application.

3. The Snyk scan was performed and it identified various vulnerabilities that exist within the OWASP Juice Shop application. The vulnerability details like CVE references, severity, exploit maturity, and possible remediation steps were provided by Snyk.

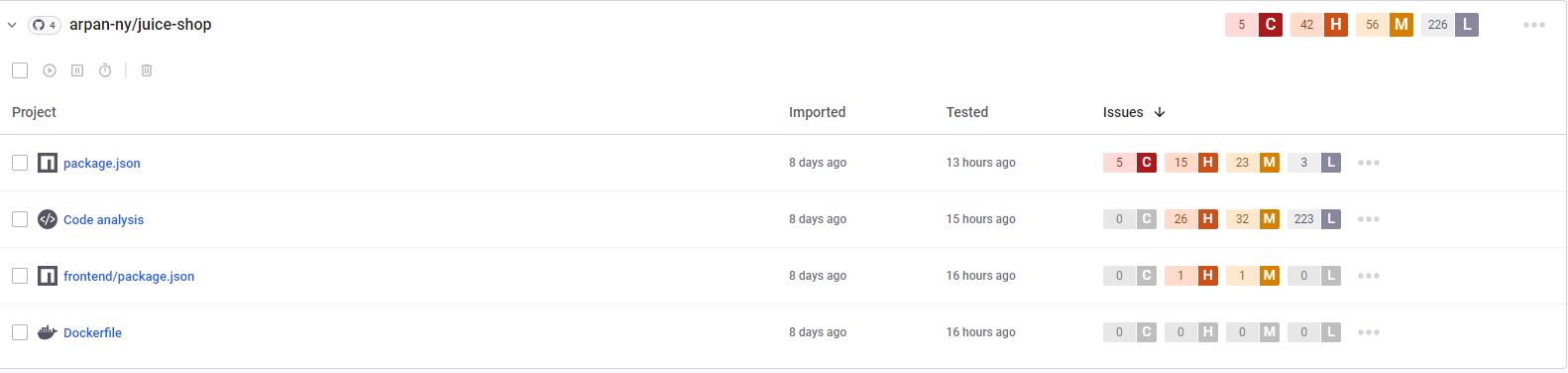
4. It was observed that while some vulnerabilities were more severe than others, all of them still require attention. Patches offered were mainly version updates, showing the importance of regularly updating applications.

5. Overall, the scan's findings emphasized the importance of vulnerability scanning as part of a security routine.

This was a valuable exercise for understanding docker container usage, application deployment, and the criticality of scanning for vulnerabilities that an application may be exposed to.

**Analysis**

There are over 5 critical, 42 high, 56 medium and 226 low vulnerability in OWASP Juice shop.



Below are the 4 high vulnerability.

1. CVE-2021-3807

This is a vulnerability related to the anvaka/npmgraph.an library for JavaScript and Node.js. The flaw lies in insufficient validation of user input when processing package details. It creates a potential for Remote Code Execution (RCE) when a user visits a malicious package website. The vulnerability has a high severity rating, and users are highly recommended to uninstall any versions before 1.1.30. As of now, the vendor hasn't provided any fixes for versions older than 1.1.29.

2. CVE-2015-9235

CVE-2015-9235 is a vulnerability in the jQuery package, versions prior to 3.0.0. An issue in the parseHTML method could allow attackers to execute arbitrary JavaScript code via a cross-site scripting (XSS) attack. It is a medium-severity issue, and an update addressing the vulnerability has been released. It is recommended to update to version 3.0.0 or later to resolve this problem.

3. CVE-2019-10744

A high severity vulnerability identified in open-source project lodash, specifically impacting versions before 4.17.13. An exploit could allow an attacker to execute arbitrary commands due to Prototype Pollution. Developers using lodash in their projects are strongly recommended to update to version 4.17.15 or later to mitigate the risk associated with this vulnerability.

4. CVE-2016-1000223

This vulnerability affects the Plack::Middleware::Debug module in the Perl language. It allows remote attackers to execute arbitrary Perl code by leveraging a failure to block certain strings that start a subroutine reference. Details about the vulnerability's severity are not disclosed publicly yet, but updating to a newer version is usually a good strategy to mitigate the risk.