|  |  |
| --- | --- |
| Name: | Pratyusha Ampolu |
| Lab User ID: | 23SEK3324\_U09 |
| Date: | 09-01-2024 |
| Application Name: | Juice-shop |

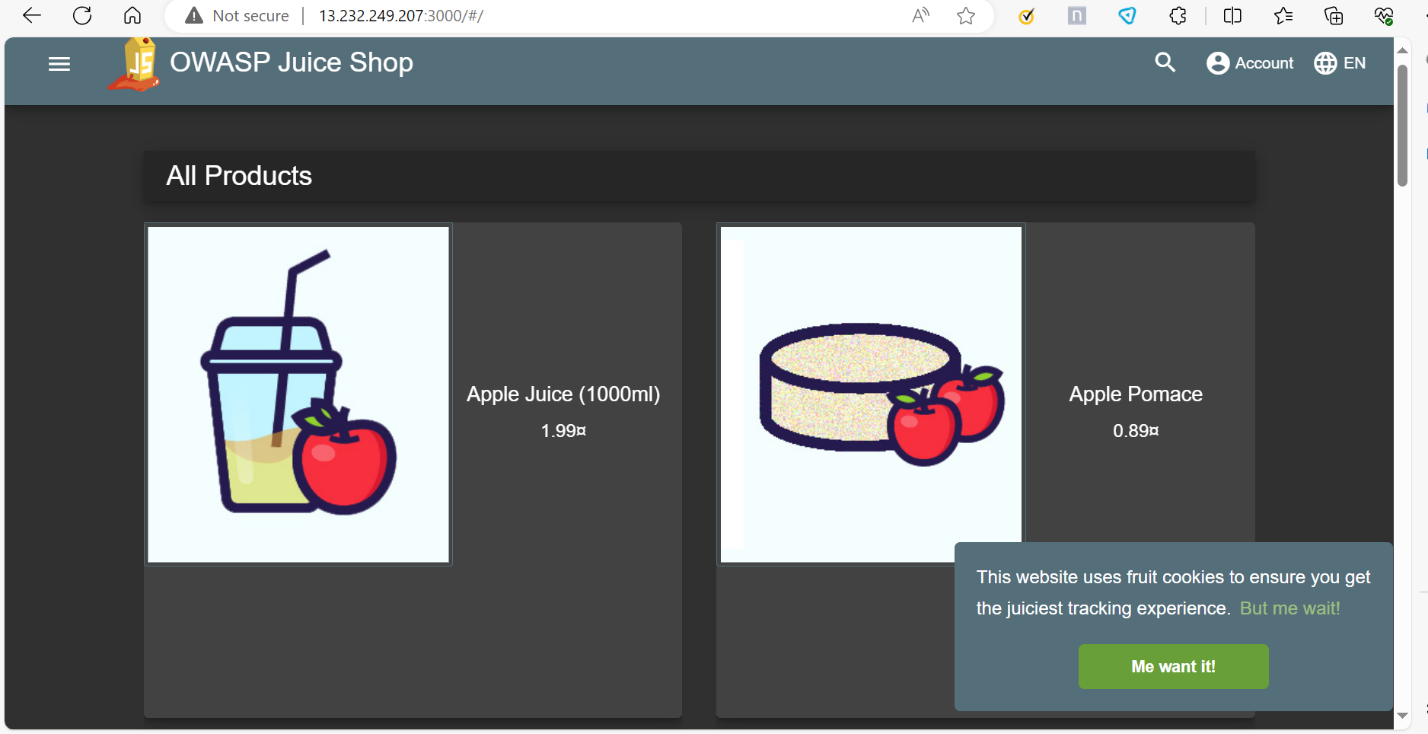
**Follow the below guidelines:**





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)



My vm

My docker

My Docker

Docker container

Docker container

Browser

13.232.249.207:3000

Docker image

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:3000.
* User can access the web site hosted on this server via web browser.
* Users can browse through a catalog of juices, view their details and add them to a virtual cart.
* Users can create accounts, log in and managing their profiles including reviewing products on the website.
* The application has several pages like contact , company and feedback.
* There are several APIs available for programmatic access to juice shop data and functionality.
* Administrators can manage users , products, orders and website settings.

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.")



* Current Docker Configuration
* Historical System stability data
* Optimal configuration.
* Impact of docker on system reliability.

**Known**

Things we are aware of but don’t understand.

Things we are aware of and understand.

* Compatibility with future Docker versions
* Employee adaption to docker usage.
* Unforeseen bugs in docker configuration.
* Unpredicted events affecting stability.

**Unknown**

**Unknown**

**Known**

Things we are neither aware of nor understand.

Things we understand but are not aware of.

Overview: OWASP juice-shop is an intentionally vulnerable web application developed by the Open Web Application Security Project (OWASP). Now we are performing some security tools against the application to check the vulnerabilities.

Step-1: Installing the Docker

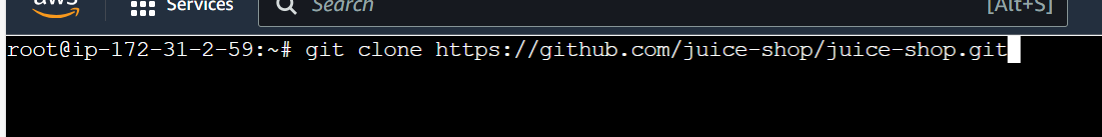
To install the docker we can use the below command.

A black and white screen with white text

Description automatically generated

Step-2: Clone the juice-shop repository from the git.

To clone the juice-shop we can use



Step-3: Creating an image/pulling an image for the juice-shop repository.

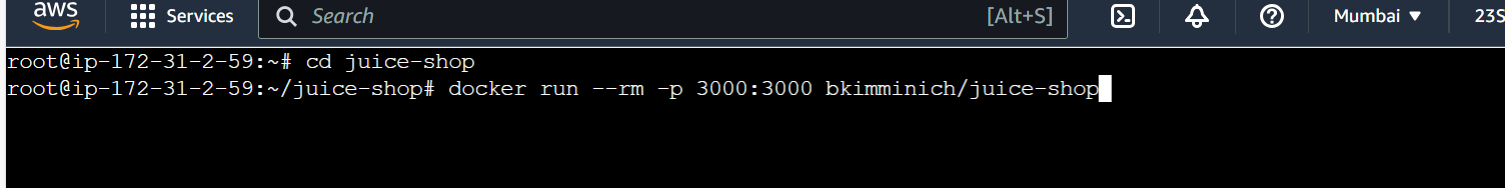
To pull the image we can use

A screen shot of a computer

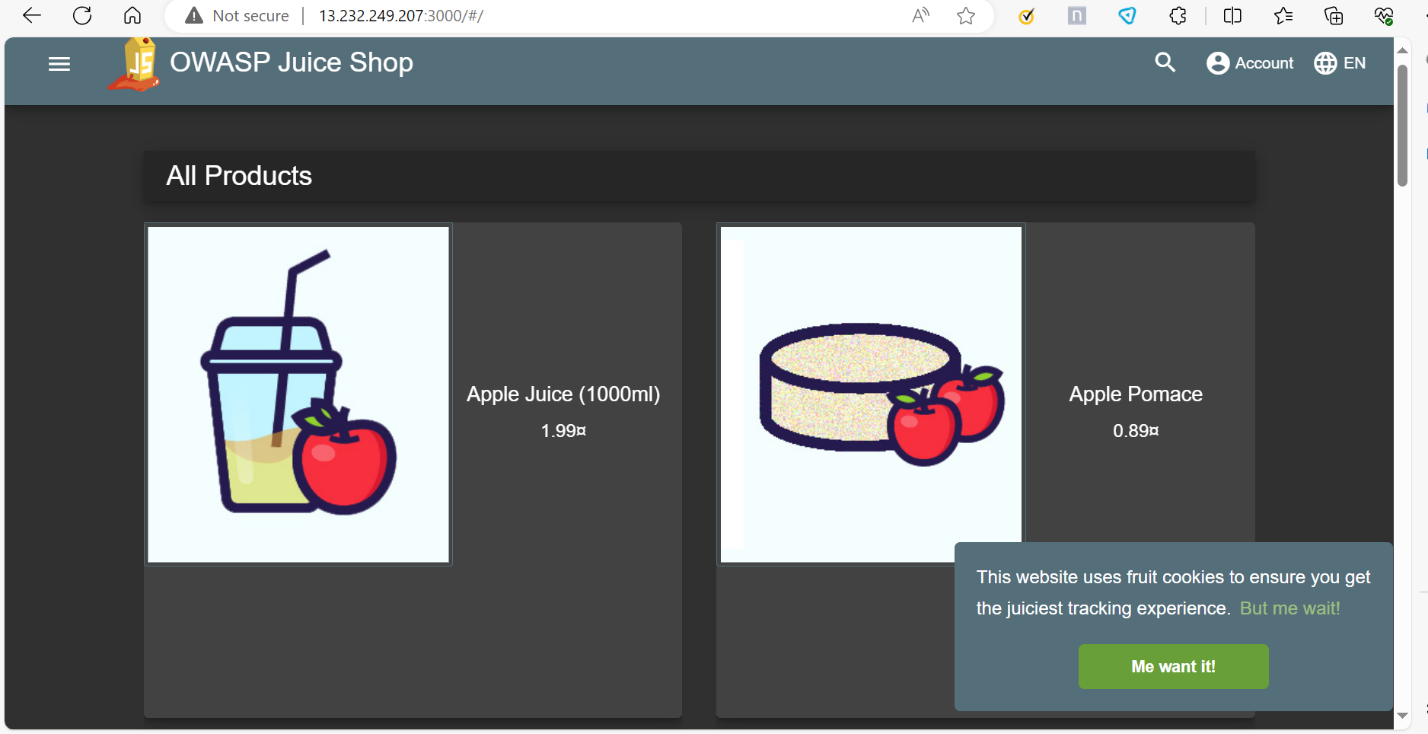
Description automatically generated

Step-4: we have to create the container using port 3000 and with the image created/pulled before.

To create the container, we use



Then, check using ip:3000 to see the live webpage. The webpage will be like



Tools:

The following tools are used in this project to do vulnerability analysis.

1. OWASP ZEP

2. SNYK

Observation:

1.OWSAP ZEP:

Using this tool we can perform the security analysis .

The following are some vulnerabilities and solutions:

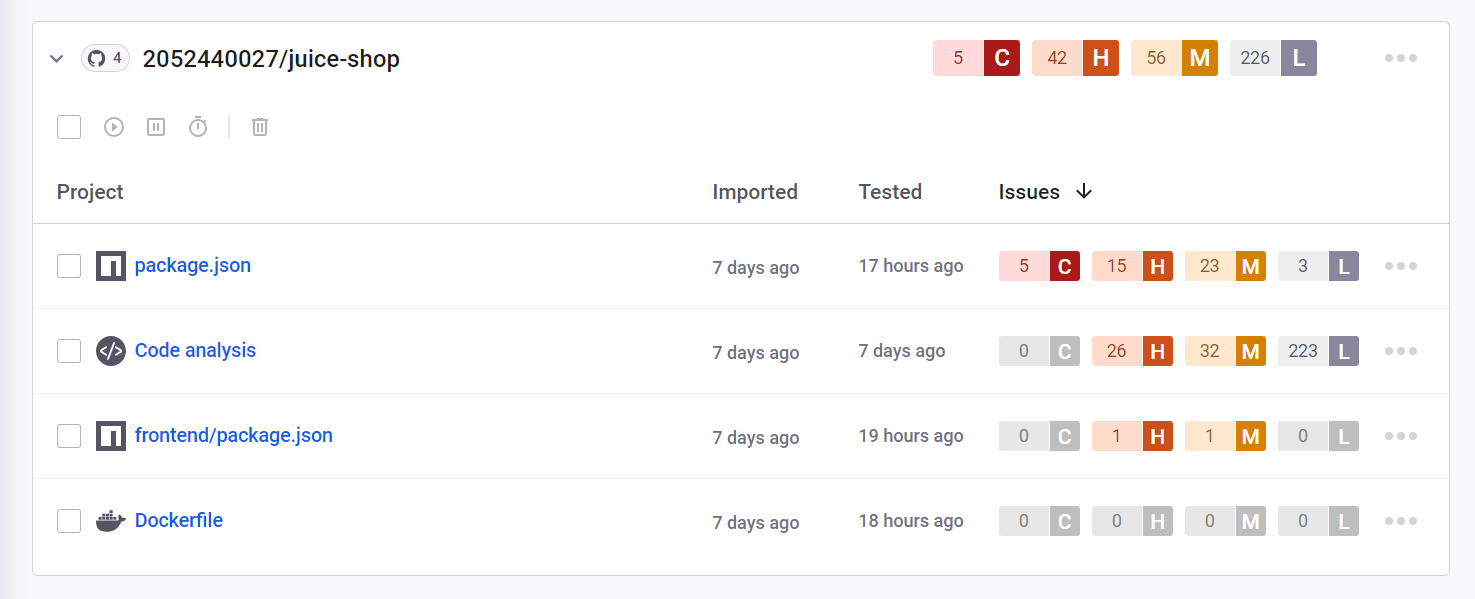
**Content security policy**: Content Security Policy is an added layer of security that helps to detect and mitigate certain types of attacks, including cross site scripting and data injection attack.

**Solution**: Ensure that your web server, application server, load balancer etc is configured to set the content-security -policy header.

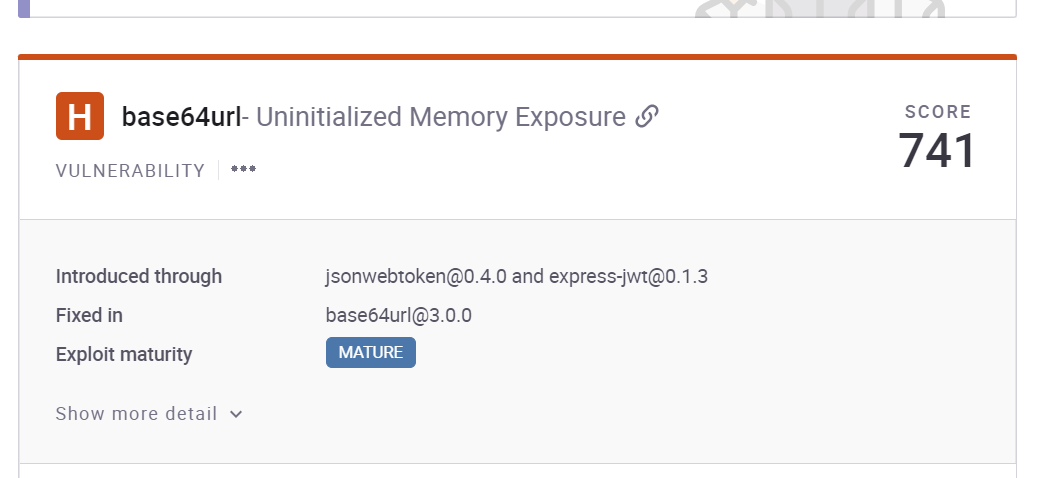
**Dangerous JS Function**: A dangerous JS function seems to be in use that would leave the site vulnerable.

**Solution:** see the references for security advice on the use of these functions.

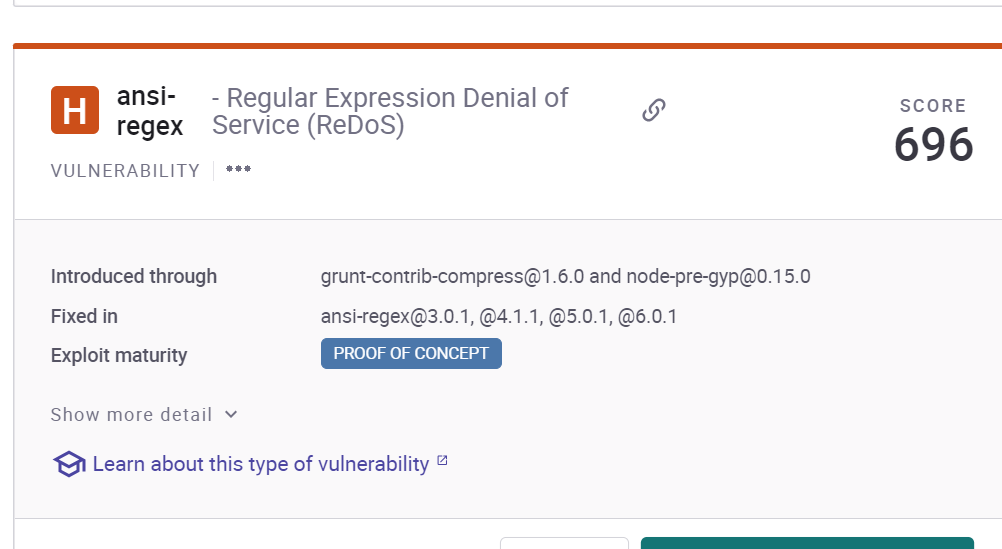
2.Using SNYK:



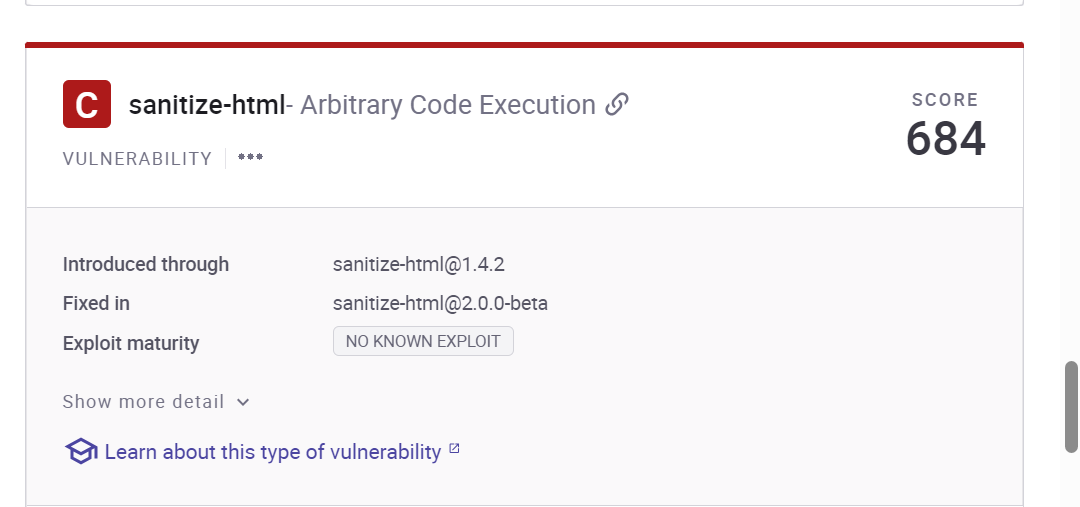
Uninitialized memory exposure:



Solution: Ensure that any possibly sensitive data specified in the requirements is verified with designers to ensure that is calculated risk or mitigated elsewhere.



Solution: Ensure that protocols have specific limits of scale placed on them. Ensure that all failures in resource allocation place the system into a safe posture.



Solution: Use dynamic tools and techniques that interact with he product using large test suites with many diverse inputs, such as fuzz testing(fuzzing), robustness testing and fault injection.