|  |  |
| --- | --- |
| Name: | Ajay Jain |
| Lab User ID: | 23SEK3324\_U33 |
| Date: | 10/01/2024 |
| 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)



Juice shop deployment

Master

Application hosted on localhost: 30377

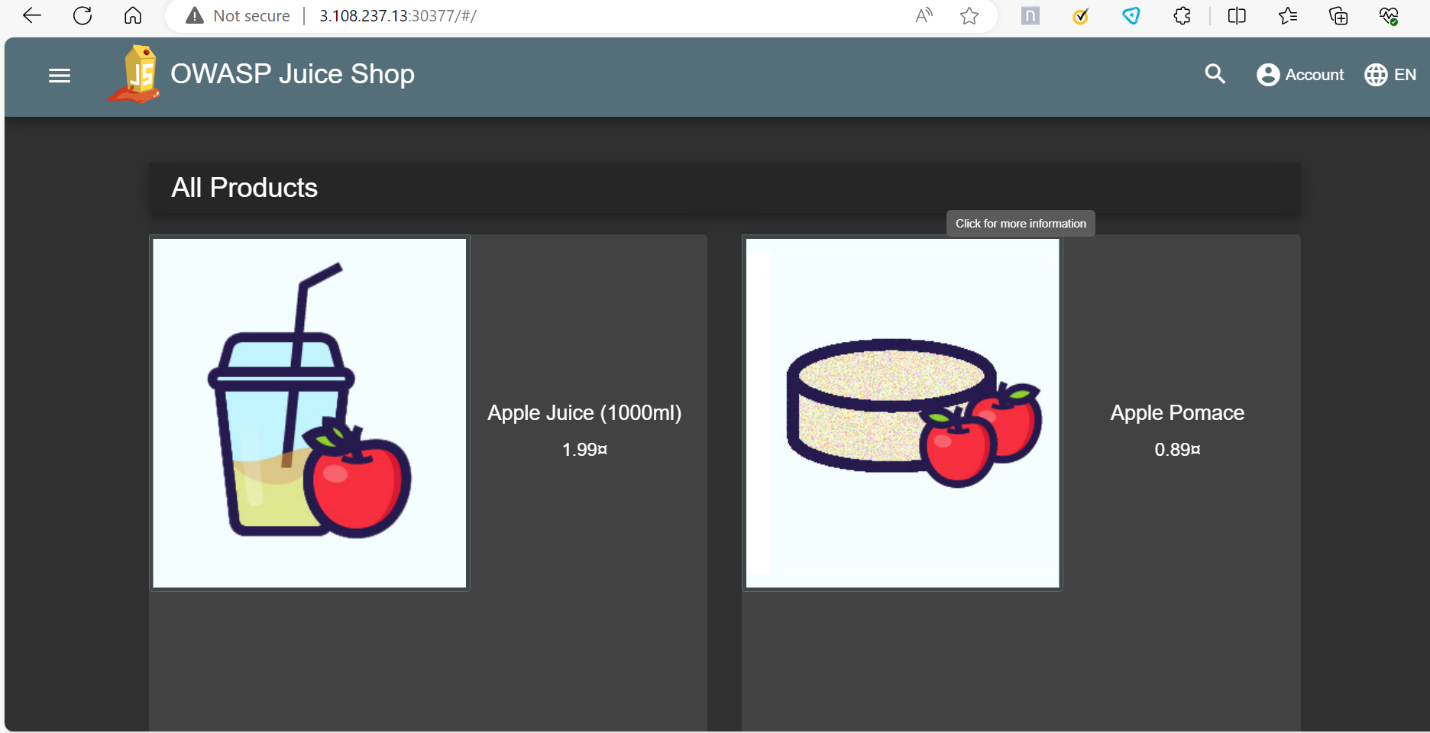
Juice shop service

Worker-1



Gremlin for chaos Engineering

Worker-2



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 application responds to HTTP requests within a reasonable time frame, such as less than 5 seconds.

The application does not leak any sensitive information, such as cookies, session tokens, or encryption keys.

The application does not have any SQL injection, cross-site scripting (XSS), or other common web application attacks.

The application does not have any misconfigurations, such as insecure file permissions, default credentials, or unnecessary services.

The application does not have any performance issues, such as high CPU or memory usage, slow response time, or frequent crashes.

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**

In case a node fails we know that application will run on the second node but we don’t know the time taken for it to run on that node

If a node fails then we know then the replicas will ensure that the application runs perfectly on the Active Node.

**Unknown**

If we shutdown the replica nodes then we don’t know whether the pseudo primary nodes and it’s replicas are not able to take over the transactions effectively

If we shut down all the replicas then we don’t know the time taken to restore the service. But we do know that the pseudo primary and its replicas can carry out the transactions.

**Unknown**

**Known**

Experiment:

(Document your Preparation, Implementation, Observation and Analysis )

**Implementation:**

1. Firstly, Set up the Kubernetes cluster with 1 master and 2 worker nodes
2. After setting up the Kubernetes cluster Create a Deployment and Service and run them
3. After deploying the application make use of localhost IP with port no to view the live application
4. Upload the repository on Snyk to view the vulnerabilities in the webpage.

**Observation and Results:**

1. Upon using Snyk tool we find several Vulnerabilities. Some vulnerabilities are listed below.

**Vulnerabilities Found:**

1. **sanitize-html-Arbitrary Code Execution:** An arbitrary code execution vulnerability is a security flaw in software or hardware allowing an attacker to run any commands or code of their choice on a target machine or in a target process.

**Fix:** Avoid the use of dangerous functions, Reconsider the need for dynamic code execution, Lock down the interpreter, Utilize a static analysis tool, Use a security linter.

1. **Remote Code Execution:** Remote**code execution (RCE) is a type of security vulnerability that allows attackers to run arbitrary code on a remote machine, connecting to it over public or private networks.**

**Fix:** Avoid the use of dangerous functions, Reconsider the need for dynamic code execution, Lock down the interpreter, Utilize a static analysis tool, Use a security linter.

1. **Sandbox Bypass:** This vulnerability allows attackers with permission to define and run sandboxed scripts, including Pipelines, to bypass the sandbox protection and execute arbitrary code in the context of the Jenkins controller JVM.

**Fix:** Encrypting the data, Activity monitoring for checking legitimate user activity, AI algorithm to detect anomalies and pattern.

1. **marsdb Arbitrary Code Injection:** Code Injection vulnerability occurs when an application does not properly validate and sanitize user input, allowing an attacker to inject malicious code into the application's codebase. This vulnerability is often characterized by the ability to execute arbitrary commands or code within the application's context.

### **Fix:** Avoid the use of dangerous functions, Reconsider the need for dynamic code execution, Lock down the interpreter, Utilize a static analysis tool, Use a security linter.