|  |  |
| --- | --- |
| Name: | BHUMIKA GUPTA |
| Lab User ID: | 23SEK3324\_U13 |
| Date: | January 10, 2024 |
| Application Name: | OWASP Juice Shop |

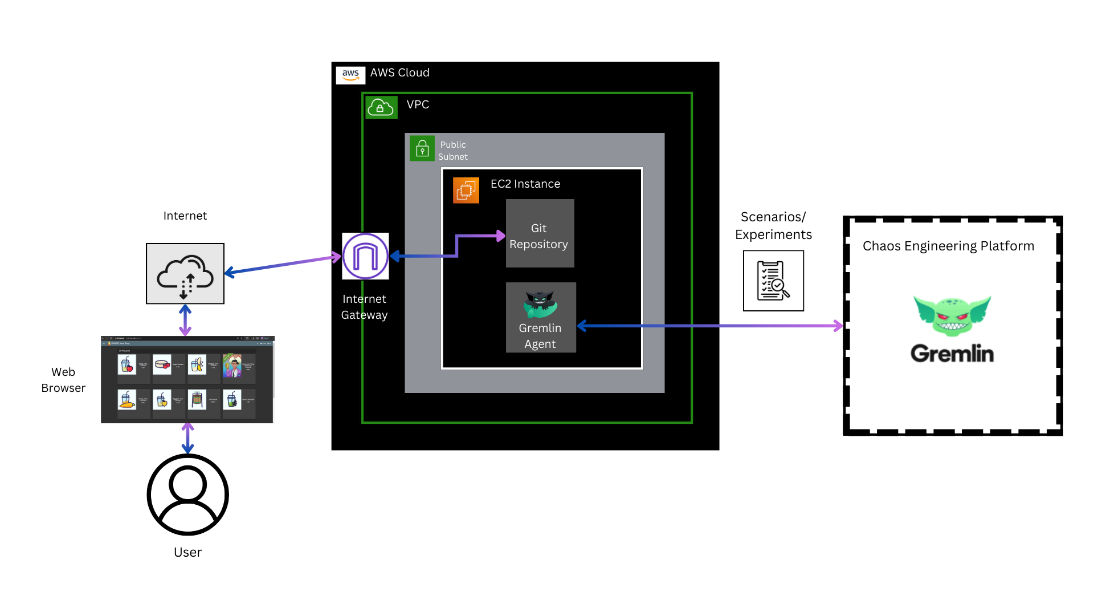
**Follow the below guidelines:**





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)



**Fig: System Architecture**

**System architecture** is a conceptual model that describes the structure and behavior of multiple components and subsystems like multiple software applications, network devices, hardware, and even other machinery of a system.

**Explanation of the above System Architecture-**

1. AWS Cloud: The entire infrastructure is hosted on AWS.
2. VPC (Virtual Private Cloud): The VPC serves as an isolated network within the AWS Cloud. It allows you to logically isolate resources and control network settings.
3. Public Subnet: Within the VPC, there is a public subnet. Public subnets are accessible from the internet and typically host resources like web servers.
4. EC2 Instance: An EC2 instance is launched in the public subnet.
   * + **Scenario 1:** *Web Application Deployment*
       - This instance hosts the web application.
       - Git Repository: This suggests that the source code for the OWASP Juice Shop is stored in a Git repository. Developers can collaborate on the code, and it can be version-controlled using Git.
       - Internet Gateway: An Internet Gateway allows communication between the VPC and the internet. In this case, it enables users to access the OWASP Juice Shop application from a web browser.
       - Web Browser (User): Users interact with the OWASP Juice Shop through a web browser, connecting to the public IP address or domain associated with the EC2 instance.
     + **Scenario 2:** *Gremlin Chaos Engineering Experiment*
       - This instance hosts the Gremlin Agent.
       - Gremlin Agent: The Gremlin Agent is a software component installed on the EC2 instance. It facilitates chaos engineering experiments by injecting faults into the system.
       - Gremlin Chaos Engineering Platform: The Gremlin Agent connects to the Gremlin Chaos Engineering Platform. This platform allows you to perform controlled experiments, such as shutting down the EC2 instance.

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)

**Steady State** – The steady state is the stable and expected state where the system functions smoothly without disruptions.

**System’s Normal Behavior -** It involves the consistent interactions between various components, resulting in reliable performance and desired outcomes. Measurable outputs and observed behaviors during steady state operations define the system's normal behavior

So the steady state and measurable outcome of our system is as follows:-

1. **Web Application Deployment:** Users can successfully access the OWASP Juice Shop web application hosted on the EC2 instance.
   * Measurable Outputs:

* Low latency in accessing the application.
* Consistent availability of the OWASP Juice Shop web interface.
* Expected and reliable responses to user interactions within the application.
* Stable and predictable resource utilization on the EC2 instance.

1. **Gremlin Chaos Engineering Experiment:** The Gremlin Agent operates without disruptions. Controlled chaos experiments, such as shutting down the EC2 instance, do not occur under normal circumstances.
   * Measurable Outputs:
     + Stable connectivity between the Gremlin Agent and the Gremlin Chaos Engineering Platform.
     + No unexpected interruptions or failures during Gremlin experiments.

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**

New security patch is applied to the EC2 instance

Regular increase in user traffic occurs

**Unknown**

**Unknown**

**Known**

An unforeseen and unprecedented external event

Git repository undergoes regular updates with the latest code changes

These hypothesis provide a structured approach to defining and understanding the expected and unexpected behaviors of the OWASP Juice Shop.

By categorizing scenarios into knowns and unknowns, and actions into intentional and unintentional, the hypothesis aid in identifying, responding to, and improving the security and robustness of the system.

**Knowns-Knowns Hypothesis:**

* If a regular increase in user traffic occurs, the system's scalability measures will handle the load, and users will experience no degradation in service.

**Known-Unknown Hypothesis:**

* If a new security patch is applied to the EC2 instance hosting the web application, it is expected to enhance system resilience. The specific impact on performance and security is not precisely known but is anticipated to be positive.

**Unknown-Known Hypothesis:**

* If the Git repository undergoes regular updates with the latest code changes, the system's stability will be maintained. The exact impact of each code change may not be known, but routine updates are expected to improve overall performance and security.

**Unknown-Unknown Hypothesis:**

* In the face of an unforeseen and unprecedented external event, such as a novel security vulnerability, the system may experience abnormal behavior. Measurable outputs would be unknown, and the hypothesis would focus on detecting anomalies and adapting the system to mitigate potential risks.

Experiment:

(Document your Preparation, Implementation, Observation and Analysis )

**Objective:**

The objective of this experiment is to evaluate the security posture of the OWASP Juice Shop, a deliberately insecure web application, and to assess its resilience to chaos by simulating controlled failures using the Gremlin Chaos Engineering platform. Additionally, the experiment aims to identify vulnerabilities in the OWASP Juice Shop by leveraging Trivy, a vulnerability scanner, installed on an EC2 instance.

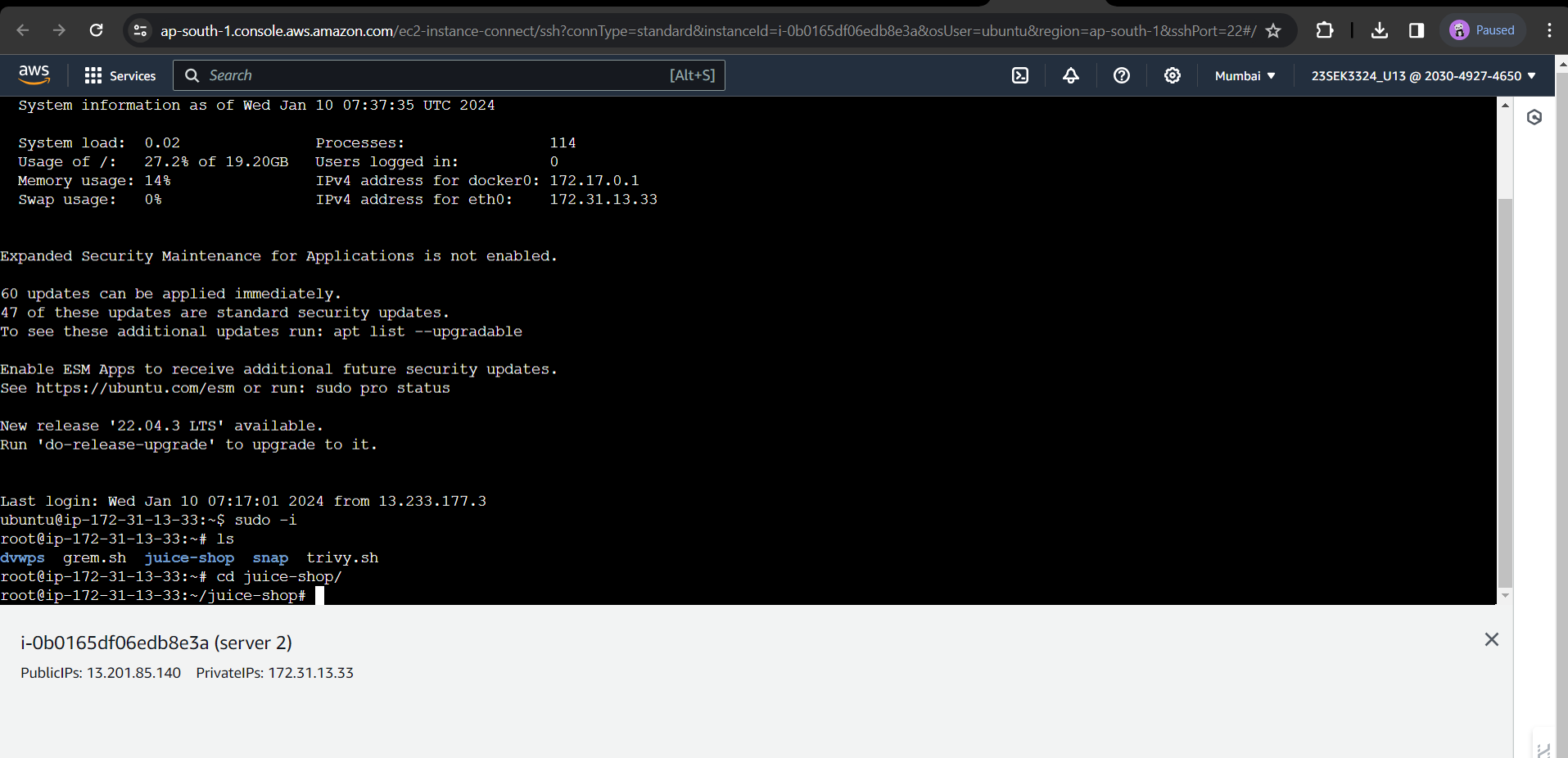
**Context:**

The OWASP Juice Shop, intentionally designed with vulnerabilities, serves as an ideal candidate for security training and testing. This experiment aims to comprehensively assess the OWASP Juice Shop's security and resilience through controlled chaos experiments using Gremlin and vulnerability scanning with Trivy. The infrastructure, deployed on AWS, includes EC2 instances hosting the Juice Shop and the Gremlin agent, providing a realistic environment for testing and analysis.

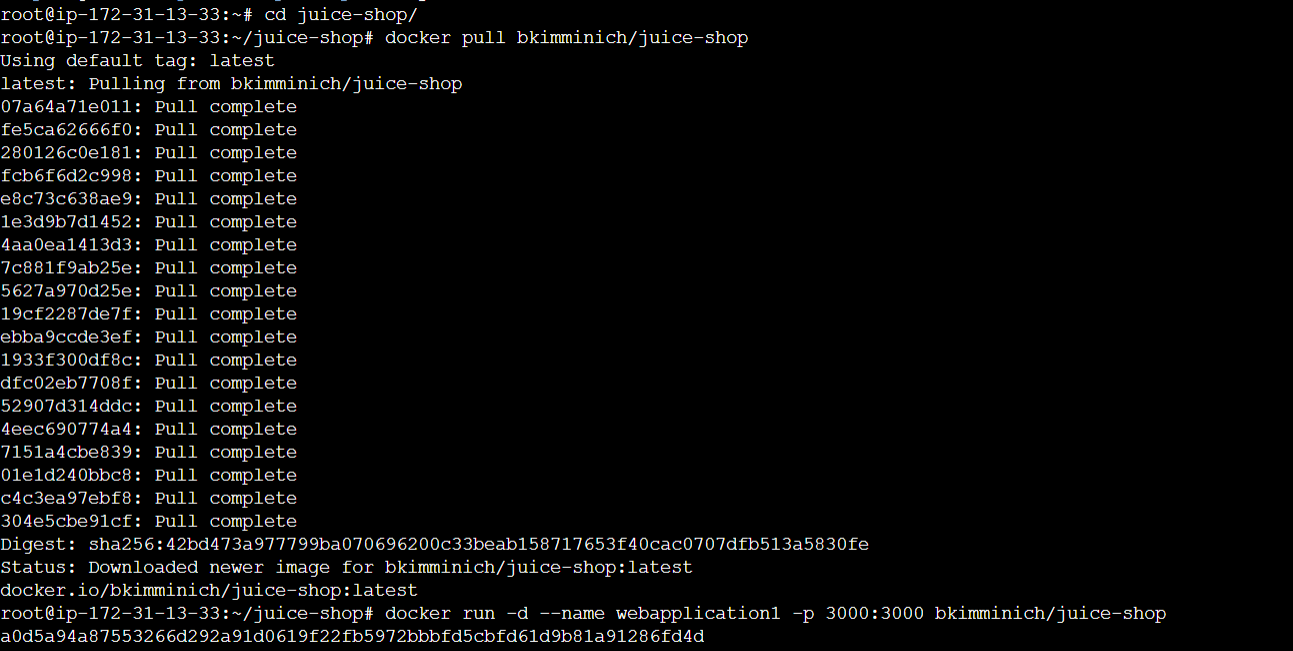
This experiment is organized into four main sections:

**Preparation:**

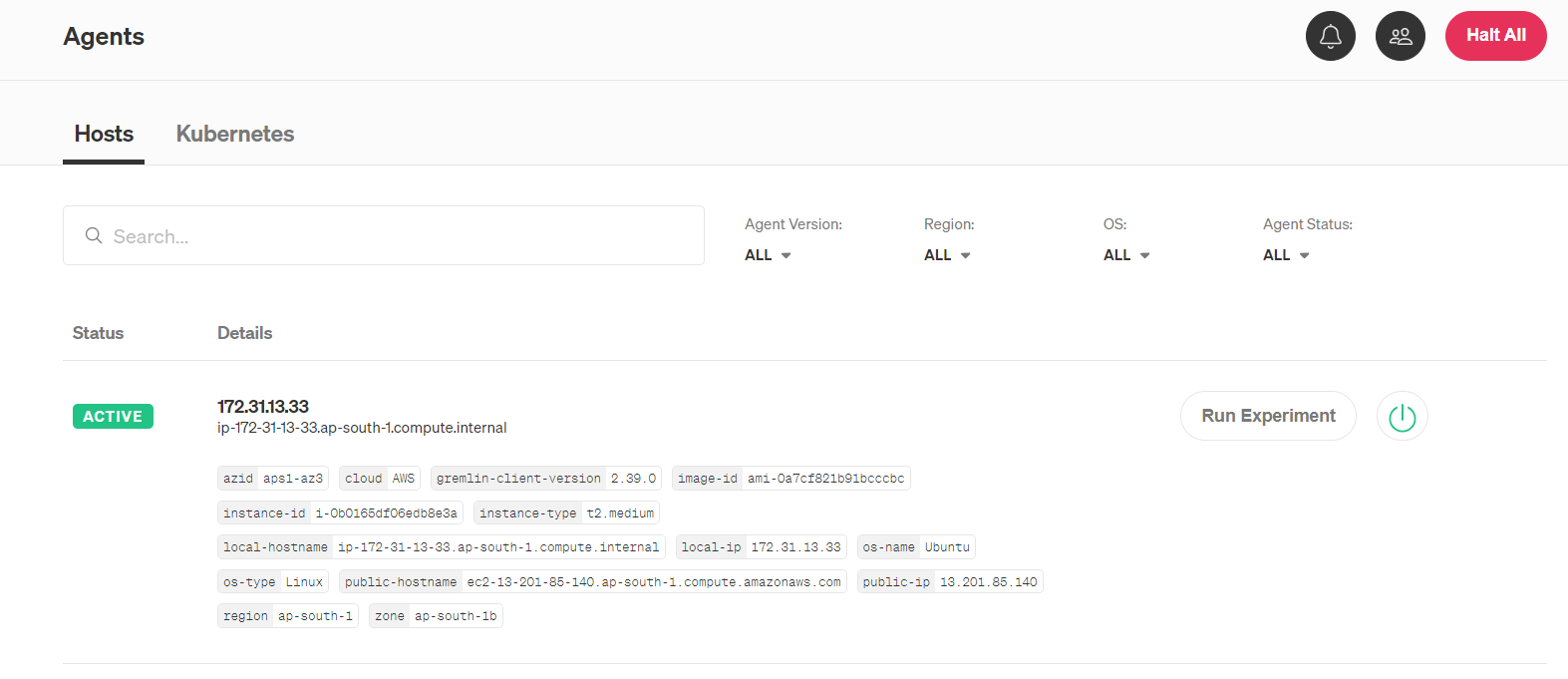
1. **AWS Infrastructure Setup:**
   * Launched an EC2 instance within an AWS Virtual Private Cloud (VPC).
   * Configured a public subnet to enable internet connectivity.
   * Connected the VPC to an Internet Gateway, allowing communication with the internet.



1. **Application Deployment:**
   * Installed a Git repository containing the OWASP Juice Shop web application on the EC2 instance.
   * Ensured the web application is accessible via a web browser over the internet.



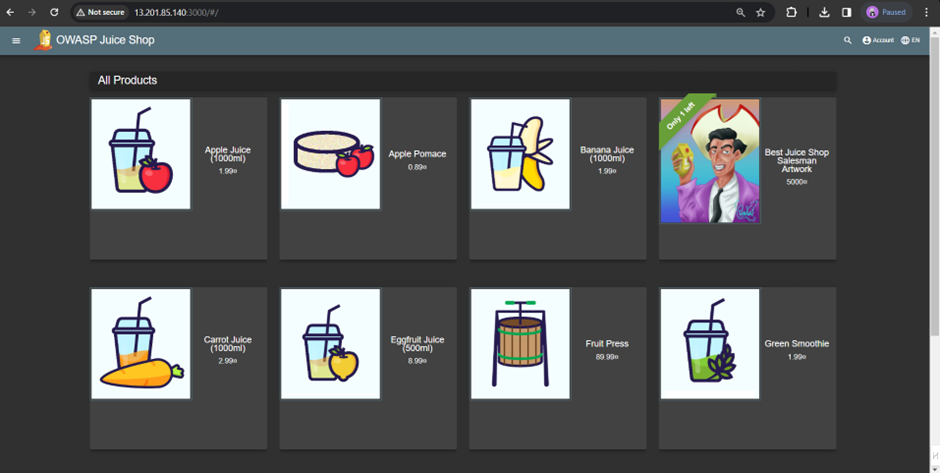
1. **Chaos Engineering Setup:**
   * Deployed a Gremlin Agent on a separate EC2 instance within the same public subnet.
   * Established connectivity between the Gremlin Agent and the Gremlin Chaos Engineering Platform.

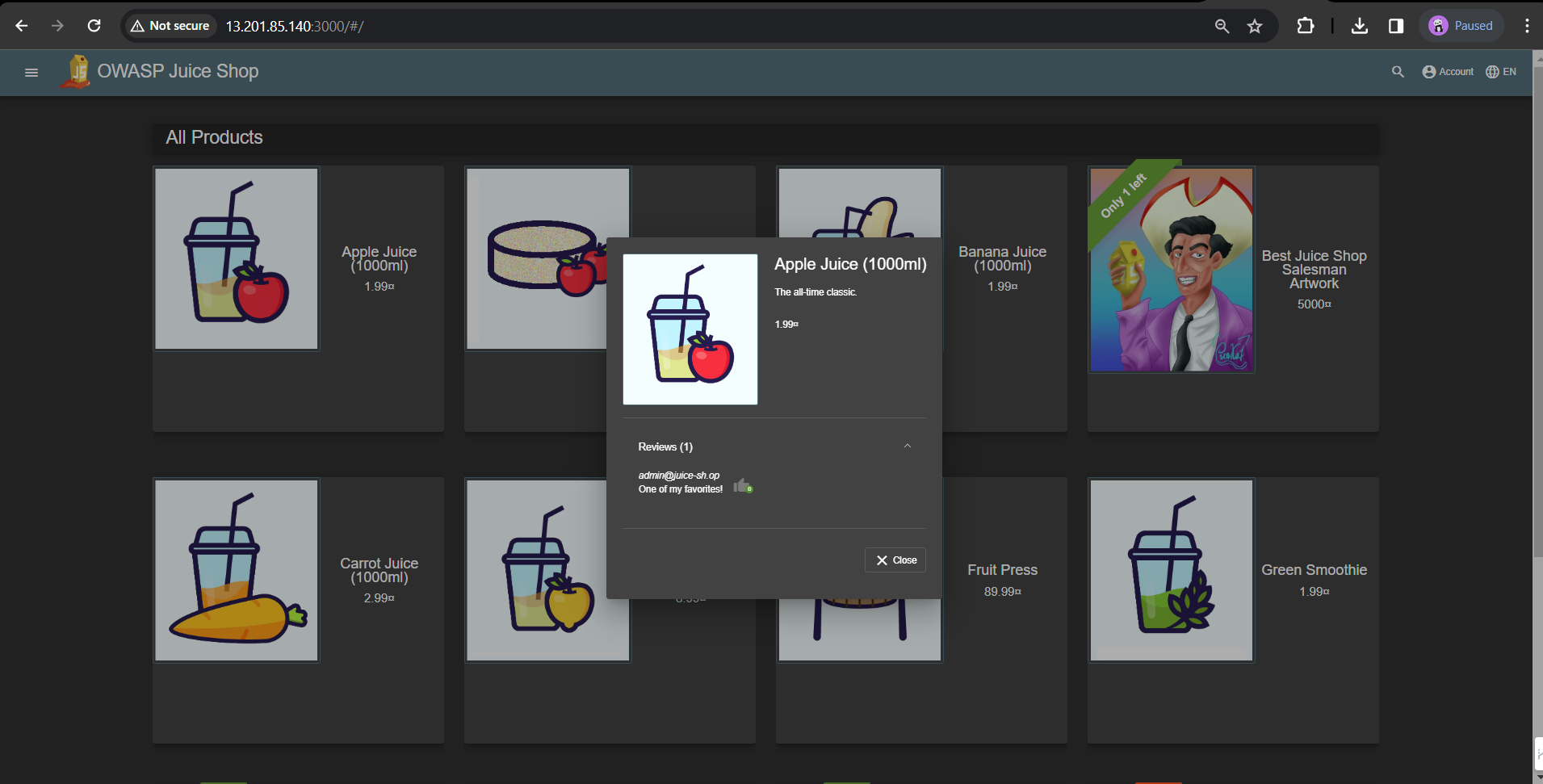


1. **Vulnerability Scanning:**
   * Install Trivy on the EC2 instance hosting the vulnerable OWASP Juice Shop.
   * Configure Trivy to scan the Docker image regularly for vulnerabilities.

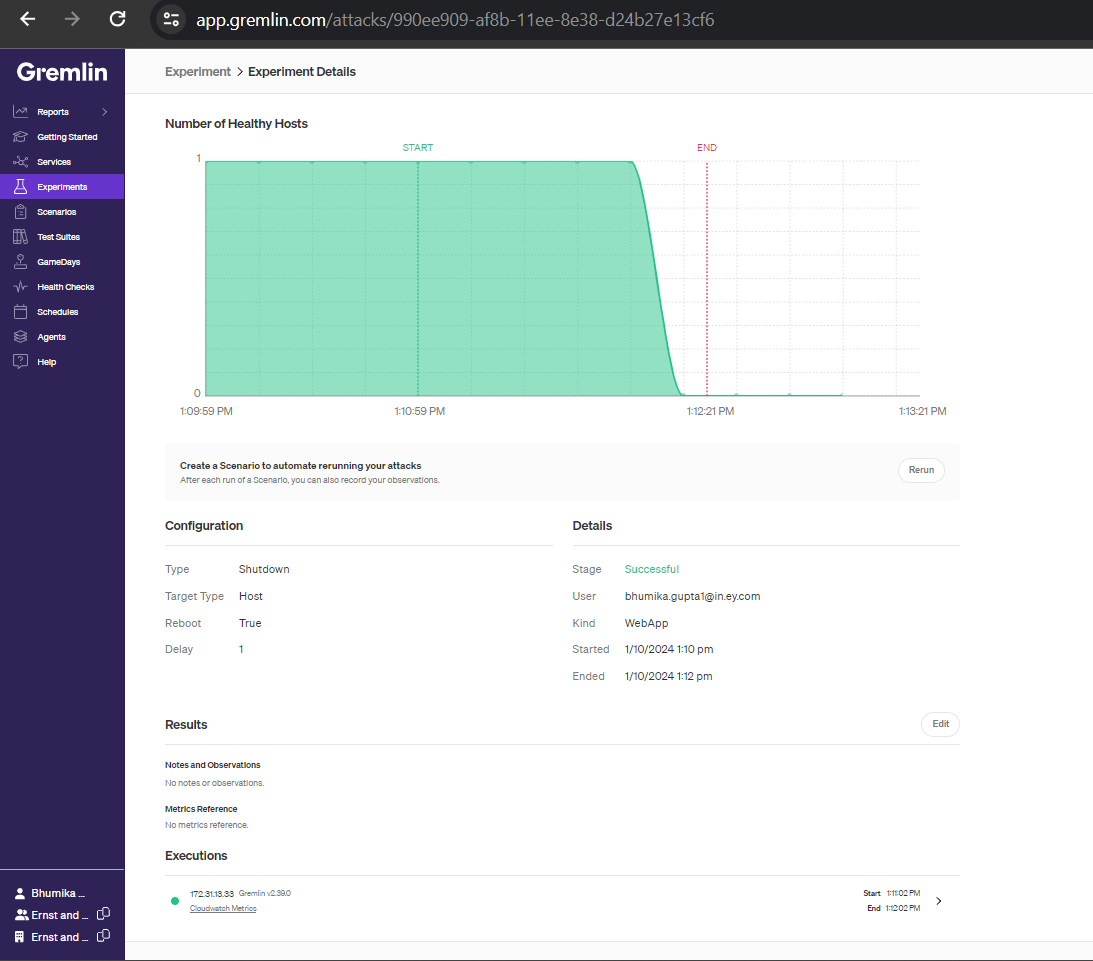
**Implementation:**

1. **Web Application Usage:**
   * Ensure the Docker container with the vulnerable OWASP Juice Shop is running on the EC2 instance.
   * Monitor the Git repository for version control and manage changes to the OWASP Juice Shop and configurations.

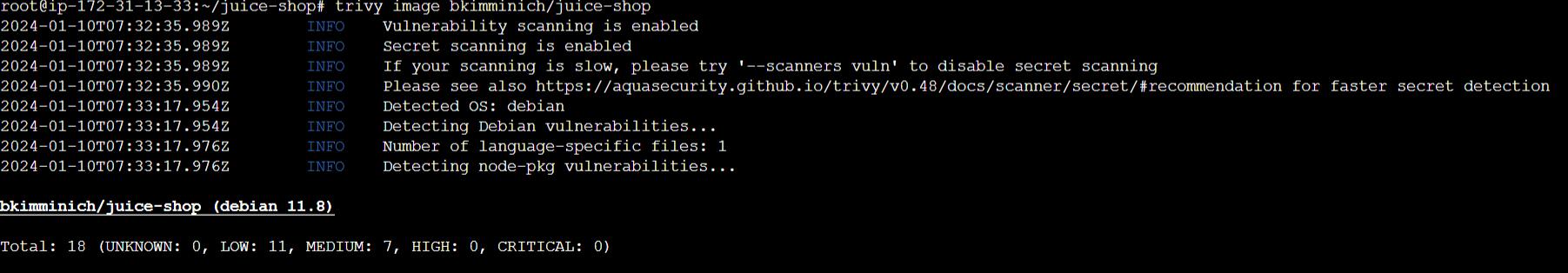




1. **Chaos Engineering Experiment:**
   * Conducted controlled chaos experiments using Gremlin, such as simulating a shutdown scenario, to evaluate system resilience.
   * Monitored system behaviour during Gremlin experiments to observe how the OWASP Juice Shop handles disruptions.



1. **Vulnerability Mitigation:**
   * Utilized Trivy to scan the OWASP Juice Shop codebase for vulnerabilities.
   * Analyzed Trivy scan results to identify and understand potential security flaws within the application.



|  |  |  |
| --- | --- | --- |
| **Vulnerability** | **Severity** | **Weakness** |
| CVE-2023-4806 | MEDIUM | Referencing memory after it has been freed can cause a program to crash, use unexpected values, or execute code. |
| CVE-2019-1010022 | LOW | The product performs operations on a memory buffer, but it can read from or write to a memory location that is outside of the intended boundary of the buffer. |
| CVE-2023-4813 | MEDIUM | Referencing memory after it has been freed can cause a program to crash, use unexpected values, or execute code. |
| CVE-2018-20796 | LOW | The product does not properly control the amount of recursion that takes place, consuming excessive resources, such as allocated memory or the program stack. |

**Observation:**

1. **Web Application Deployment:**
   * Observe user access to the vulnerable OWASP Juice Shop through a web browser.
   * Monitor the EC2 instance for stability and resource utilization during normal usage.
2. **Chaos Engineering Configuration:**
   * During Gremlin shutdown experiments, observed the system's behavior, focusing on downtime, recovery time, and impact on user access.
   * Documented any unexpected behaviour or failures during the chaos experiments.
3. **Vulnerability Scanning:**
   * Trivy scan results revealed vulnerabilities present in the OWASP Juice Shop codebase.
   * Categorized vulnerabilities based on severity levels and identified potential areas of improvement.

**Analysis:**

1. **Web Application Stability:**
   * Observe user access to the vulnerable OWASP Juice Shop through a web browser.
   * Monitor the EC2 instance for stability and resource utilization during normal usage.
2. **Chaos Experiment Outcomes:**
   * Analysed the impact of simulated shutdowns on the OWASP Juice Shop.
   * Assessed the system's ability to recover and maintain functionality after the chaos experiment.
3. **Vulnerability Analysis:**
   * Evaluated the severity of vulnerabilities found by Trivy.
   * Provided recommendations for mitigating identified vulnerabilities and enhancing overall security.