|  |  |
| --- | --- |
| Name: | BHUMIKA GUPTA |
| Lab User ID: | 23SEK3324\_U13 |
| Date: | January 10, 2024 |
| Application Name: | Damn-vulnerable-bank |

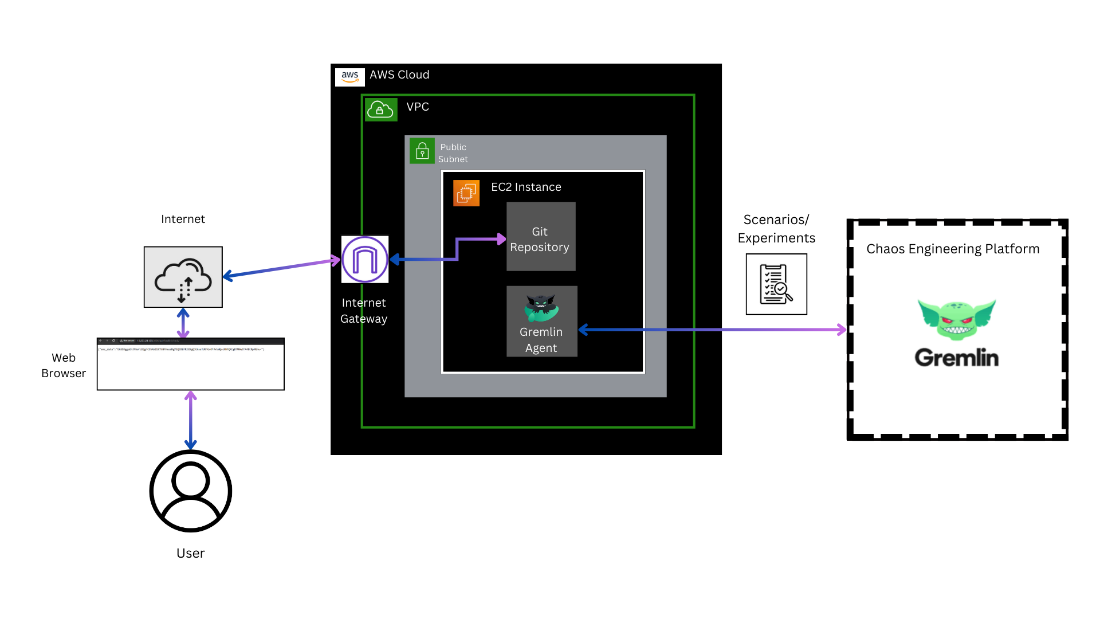
**Follow the below guidelines:**





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: The EC2 instance contains a Git repository with the web application's source code. The web application can be accessed and served from this Git repository.
       - Internet Gateway: The VPC is connected to an Internet Gateway. This allows the EC2 instance to communicate with the internet, enabling users to access the web application.
       - Web Browser (User): Users, through their web browsers, access the web application hosted on the EC2 instance. The internet gateway facilitates the communication between the EC2 instance and the users.
     + **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 web application hosted on the EC2 instance.
   * Measurable Outputs:
     + Successful HTTP requests.
     + Low latency in serving web pages.
     + Minimal error rates during user interactions.
2. **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**

Users performing standard operations like logging in, logging out, checking balances, and making transactions.

Unknown users attempting successful logins or users trying unauthorized actions.

**Unknown**

**Unknown**

**Known**

Changes made to backend URLs or security configurations.

Application experiences unexpected crashes or errors.

These hypothesis provide a structured approach to defining and understanding the expected and unexpected behaviors of the Damn Vulnerable Bank application.

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

**Knowns-Knowns Hypothesis:**

* Users performing standard operations like logging in, logging out, checking balances, and making transactions.
* This hypothesis helps establish a baseline for normal user interactions. Deviations from this known behavior may indicate potential issues or anomalies.

**Known-Unknown Hypothesis:**

* Unknown users attempting successful logins or users trying unauthorized actions.
* This hypothesis focuses on detecting unexpected behaviors or security breaches.
* Deviations from known user patterns may signal security risks or attempted exploits.

**Unknown-Known Hypothesis:**

* Changes made to backend URLs or security configurations.
* Assumes that certain modifications are expected and should not compromise security.
* The system should be resilient to planned changes without adverse effects.

**Unknown-Unknown Hypothesis:**

* Application experiences unexpected crashes or errors.
* This hypothesis emphasizes the need for continuous improvement.
* Unexpected issues prompt investigation and refinement of the system to enhance its reliability and stability.

Experiment:

(Document your Preparation, Implementation, Observation and Analysis )

Objective:

The purpose of this report is to document the preparation, observation, and analysis conducted during the deployment, chaos engineering experiments, and vulnerability assessment of the Damn Vulnerable Bank application within an AWS environment.

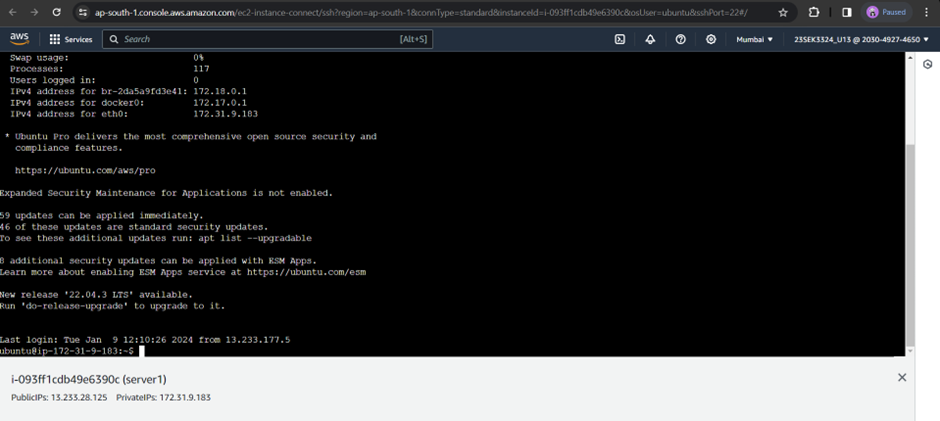
Context:

In order to assess the security posture and operational resilience of the Damn Vulnerable Bank application, we undertook a series of activities ranging from the deployment of the application in AWS to conducting controlled chaos experiments using Gremlin. Additionally, vulnerability scanning using Trivy was implemented to identify potential security vulnerabilities within the application.

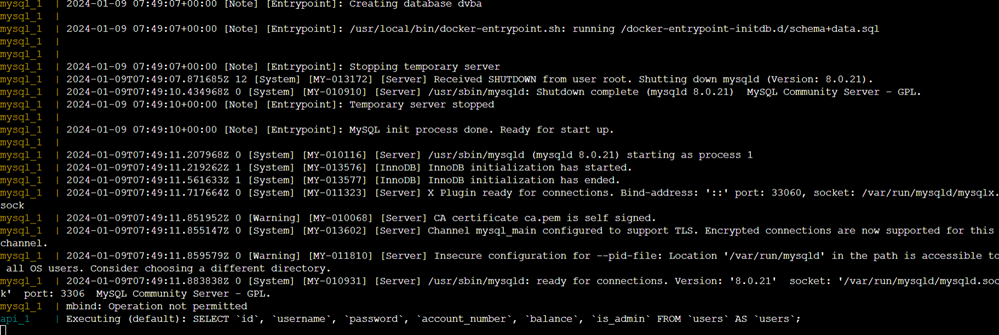
This experiment is organized into four main sections:

**Preparation:**

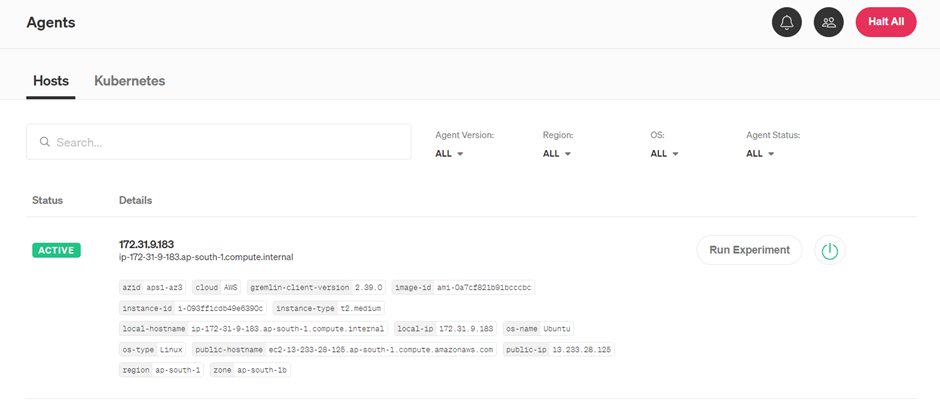
1. **AWS Infrastructure Setup:**
   * Launched an EC2 instance within an AWS Virtual Private Cloud (VPC).
   * Configured a public subnet to enable internet connectivity.
   * Connected the VPC to an Internet Gateway, allowing communication with the internet.



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



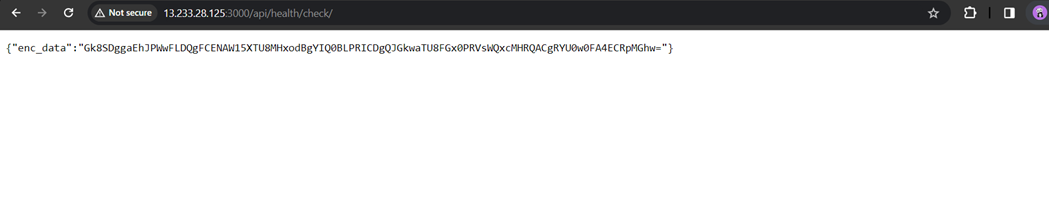
1. **Chaos Engineering Setup:**
   * Deployed a Gremlin Agent on a separate EC2 instance within the same public subnet.
   * Established connectivity between the Gremlin Agent and the Gremlin Chaos Engineering Platform.



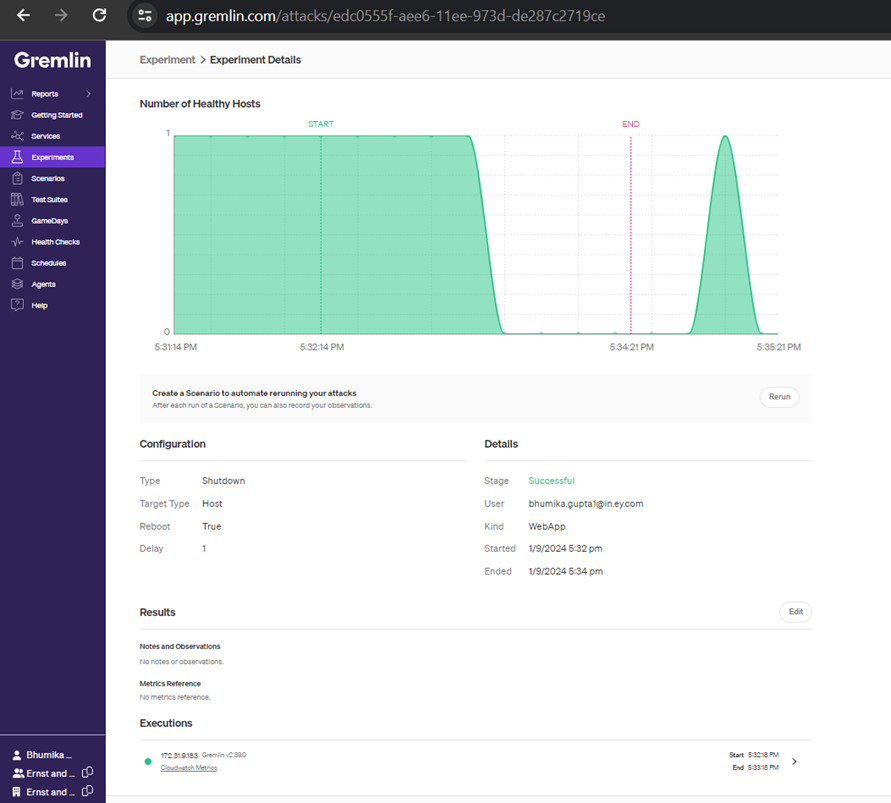
1. **Vulnerability Scanning:**
   * Installed Trivy on the EC2 instance hosting the Damn Vulnerable Bank application.
   * Utilized Trivy to scan the application for vulnerabilities.

**Implementation:**

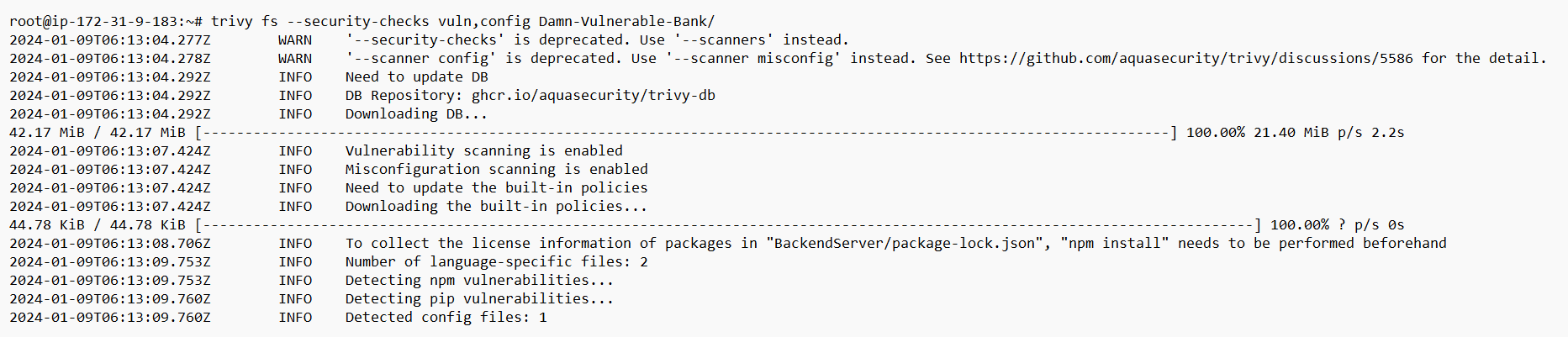
1. **Web Application Usage:**
   * Accessed the Damn Vulnerable Bank web application through a web browser.
   * Performed typical user actions like logging in, viewing balances, and conducting transactions.
   * This is mobile application so only enc data is available here so we are trying to check th health check data here.



1. **Chaos Engineering Experiment:**
   * Initiated a controlled chaos experiment using the Gremlin Chaos Engineering Platform.
   * Experiment focused on inducing a shutdown scenario to assess the system's resilience.



1. **Vulnerability Mitigation:**
   * Analyzed Trivy scan results to identify vulnerabilities in the Damn Vulnerable Bank application.
   * Implemented mitigation strategies, such as updating dependencies or applying patches, to address the identified vulnerabilities.



|  |  |  |
| --- | --- | --- |
| **Vulnerability** | **Severity** | **Weakness** |
| CVE-2021-43138 | HIGH | The product receives input from an upstream component that specifies attributes that are to be initialized or updated in an object, but it does not properly control modifications of attributes of the object prototype. |
| CVE-2017-20165 | MEDIUM | The product uses a regular expression with an inefficient, possibly exponential worst-case computational complexity that consumes excessive CPU cycles. |
| CVE-2017-16137 | MEDIUM | The product does not properly control the allocation and maintenance of a limited resource, thereby enabling an actor to influence the number of resources consumed, eventually leading to the exhaustion of available resources. |
| CVE-2022-38900 | HIGH | The product receives input or data, but it does not validate or incorrectly validates that the input has the properties that are required to process the data safely and correctly. |

**Observation:**

1. **Web Application Deployment:**
   * Observed successful deployment of the Damn Vulnerable Bank web application on the EC2 instance.
   * Verified internet connectivity and accessibility through a web browser.
2. **Chaos Engineering Configuration:**
   * Confirmed proper installation and functioning of the Gremlin Agent on the designated EC2 instance.
   * Verified connectivity between the Gremlin Agent and the Gremlin Chaos Engineering Platform.
3. **Vulnerability Scanning:**
   * Executed Trivy to scan the Damn Vulnerable Bank application for vulnerabilities.
   * Observed Trivy's output, identifying potential security issues within the application.

**Analysis:**

1. **Web Application Stability:**
   * Monitored the stability of the Damn Vulnerable Bank web application during user interactions.
   * Analysed logs and performance metrics to ensure no unexpected behaviours or errors.
2. **Chaos Experiment Outcomes:**
   * Assessed the outcome of the chaos experiment induced by the Gremlin Chaos Engineering Platform.
   * Observed how the system responded to a controlled shutdown scenario.
3. **Vulnerability Analysis:**
   * Reviewed Trivy scan results to understand the vulnerabilities present in the Damn Vulnerable Bank application.
   * Prioritized vulnerabilities based on severity and potential impact.