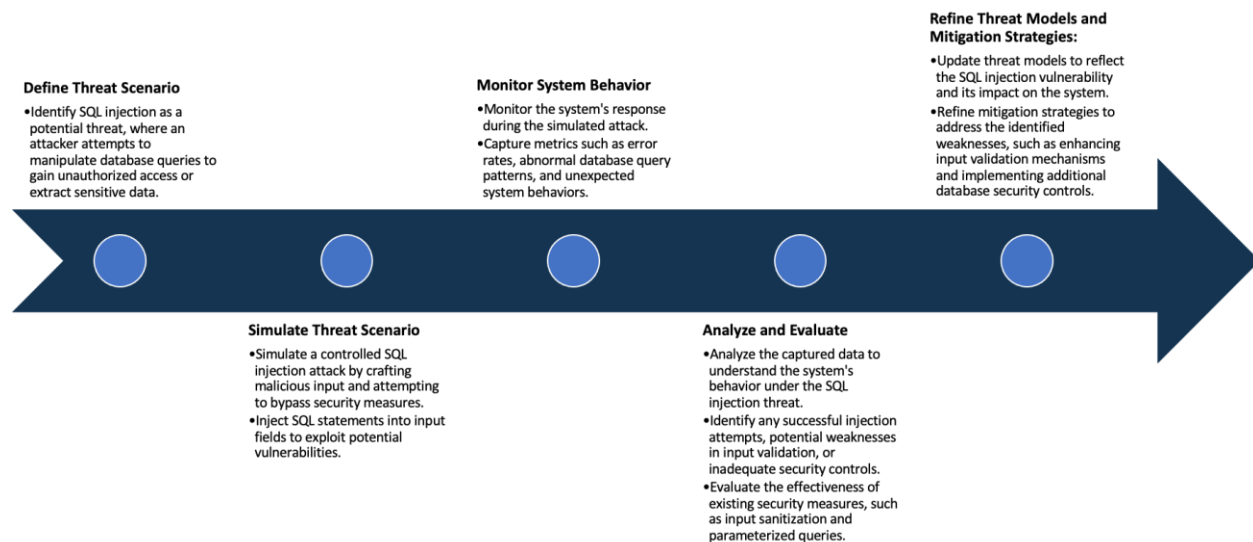


# Chaos Engineering - Report

Name:	Sunidhi Agrawal
Lab User ID:	23SEK3324_U11
Date:	09.01.2024
Application Name:	OWASP WrongSecrets

## Follow the below guidelines:

Define Hypotheses and Scenarios:	Inject Controlled Failure:	Measure System Behavior:	Learn and Iterate:
<ul style="list-style-type: none"> <li>Similar to Chaos Engineering, start by defining hypotheses and scenarios related to potential threats.</li> </ul>	<ul style="list-style-type: none"> <li>Introduce controlled failure scenarios that mimic potential attack vectors or vulnerabilities identified in Threat Modeling.</li> <li>Simulate failure conditions, such as network disruptions, component failures, or data breaches, to observe the system's response.</li> </ul>	<ul style="list-style-type: none"> <li>Capture and measure relevant system behavior metrics during the chaos experiments.</li> <li>Monitor the system's response to the injected failures, including performance metrics, error rates, and security-related indicators.</li> <li>Analyze and compare the observed behavior against the expected outcomes defined in the Threat Modeling process.</li> </ul>	<ul style="list-style-type: none"> <li>Learn from the results of the chaos experiments and iterate on the Threat Modeling process.</li> <li>Analyze the observations and insights gained from the chaos experiments to refine the Threat Models. Update threat scenarios, adjust mitigation strategies, and improve security controls based on the lessons learned.</li> </ul>



# Chaos Engineering - Report

## 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)

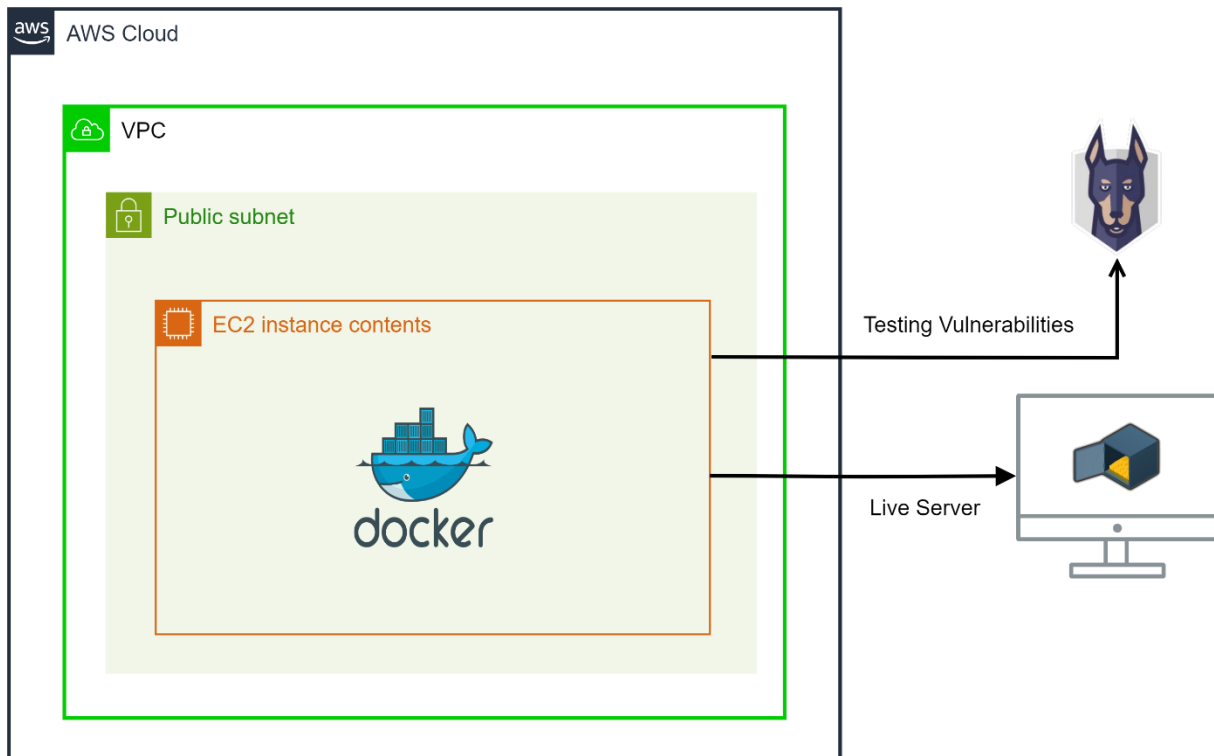
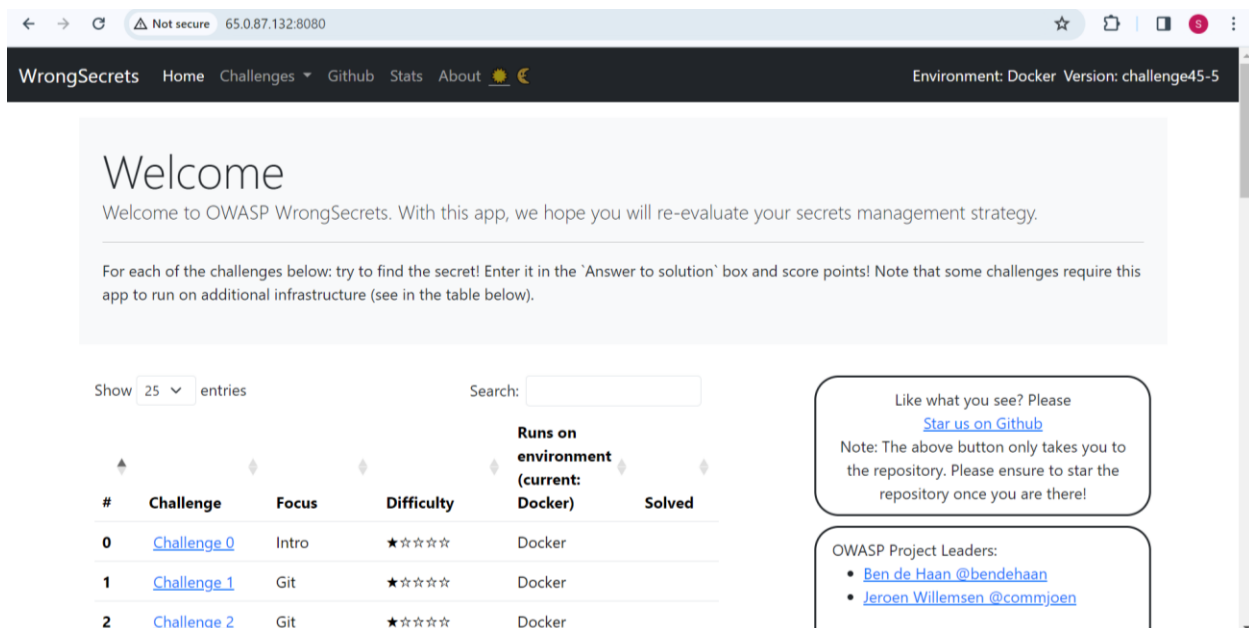


Figure 1: System Architecture



The screenshot shows the OWASP WrognSecrets application interface. The header includes navigation links (Home, Challenges, Github, Stats, About) and the environment information (Environment: Docker, Version: challenge45-5). The main content area displays a 'Welcome' message and a list of challenges. The challenges are listed in a table with columns for #, Challenge, Focus, Difficulty, Runs on environment (current: Docker), and Solved.

#	Challenge	Focus	Difficulty	Runs on environment (current: Docker)	Solved
0	<a href="#">Challenge 0</a>	Intro	★☆☆☆☆	Docker	
1	<a href="#">Challenge 1</a>	Git	★☆☆☆☆	Docker	
2	<a href="#">Challenge 2</a>	Git	★☆☆☆☆	Docker	

On the right side of the interface, there is a call to action to 'Star us on Github' and a note about the repository. Below this, the OWASP Project Leaders are listed: Ben de Haan (@bendeHaan) and Jeroen Willemsen (@commjoen).

Figure 2: OWASP WrognSecrets

# Chaos Engineering - Report

## 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 normal behavior of a system refers to its typical or expected state when operating under standard conditions. In the context of the OWASP WrongSecrets game, defining the steady state involves understanding the normal and expected behavior of the web application under typical operating conditions. Since the WrongSecrets game is designed to highlight common mistakes in storing secrets, the steady state would encompass the following aspects:

---

### **1. Functionalities:**

- Understand the normal functioning of the WrongSecrets web application. This includes user interactions, the flow of the application, and expected outcomes.

### **2. Logging and Monitoring:**

- Establish the standard logs and monitoring mechanisms in use. Identify the types of events that are logged and monitored during regular operations.

### **3. Secrets Storage and Retrieval:**

- Identify the standard procedures for storing and retrieving secrets within the application. This includes the expected methods for handling sensitive information, such as passwords or API keys.

### **4. User Authentication:**

- Define the usual pattern of user authentication. This involves recognizing the standard process for user login, password reset, and account management.

### **5. Access Controls:**

- Understand the access controls in place, including user roles and permissions. Recognize the normal behavior of users based on their roles within the application.

# Chaos Engineering - Report

## Hypothesis:

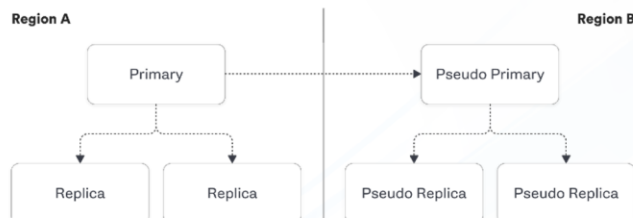
(During an experiment, we need a hypothesis for comparing to a stable control group, and the same applies here too. If there is a reasonable expectation for a particular action according to which we will change the steady state of a system, then the first thing to do is to fix the system so that we accommodate 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-Knowns

- We know that when a replica shuts down it will be removed from the cluster. We know that a new replica will then be cloned from the primary and added back to the cluster.

### Known-Unknowns

- We know that the clone will occur, as we have logs that confirm if it succeeds or fails, but we don't know the weekly average of the mean time it takes from experiencing a failure to adding a clone back to the cluster effectively.
- We know we will get an alert that the cluster has only one replica after 5 minutes but we don't know if our alerting threshold should be adjusted to more effectively prevent incidents.



### Unknown-Knowns

- If we shutdown the two replicas for a cluster at the same time, we don't know exactly the mean time during a Monday morning it would take us to clone two new replicas off the existing primary. But we do know we have a pseudo primary and two replicas which will also have the transactions.

### Unknown-Unknowns

- We don't know exactly what would happen if we shutdown an entire cluster in our main region, and we don't know if the pseudo region would be able to failover effectively because we have not yet run this scenario.

Known	<p><b>- Web Server Vulnerabilities:</b> Known vulnerabilities in the OWASP WrongSecrets server identified through prior assessments.</p> <p><b>- Chaos Engineering Tools:</b> Familiarity with the chaos engineering tools and frameworks to be used, such as Chaos Monkey or Gremlin.</p> <p><b>- Deployment Environment:</b> Knowledge of the Dockerized deployment on an Ubuntu EC2 machine.</p>	<p><b>- Impact of Chaos Experiments:</b> The precise impact of chaos experiments on the OWASP WrongSecrets server is known to some extent but may vary based on the specific experiment.</p> <p><b>- Response to Specific Scenarios:</b> While the vulnerabilities are known, the exact system response to simulated chaos scenarios is not entirely predicted.</p>
Unknown	<p><b>- Hidden Security Measures:</b> The existence of any unintentional or undocumented security measures within the OWASP WrongSecrets server.</p>	<p><b>- Emergent Vulnerabilities:</b> Unanticipated vulnerabilities that may surface because of chaos experiments.</p> <p><b>- System Behaviors Under Extreme Stress:</b> Unknown reactions and behaviors of the system under extreme stress or unexpected conditions.</p> <p><b>- Unexplored Attack Vectors:</b> Potential attack vectors that were not considered during the vulnerability assessment and may become apparent during chaos experiments.</p>

## Experiment:

(Document your Preparation, Implementation, Observation and Analysis)

### **Project Overview:**

The main objective of this project is to deploy the OWASP WrongSecrets server on a Docker container running on an Ubuntu EC2 machine. WrongSecrets is a deliberately insecure web application designed for security testing and training purposes. By deploying it on Docker, we aim to create an isolated and reproducible environment for testing various security vulnerabilities.

### **Tools Used:**

#### Docker:

- Purpose: Containerization tool used to package the OWASP WrongSecrets and its dependencies.
- Explanation: Docker enables the creation of isolated containers, ensuring consistency across different environments.

#### Snyk:

- Purpose: Snyk is employed in this project to address security concerns by identifying and mitigating vulnerabilities in both the Docker container images and project dependencies.
- Explanation: The tool contributes to the overall goal of creating a secure and resilient environment for the Damn Vulnerable WordPress site and its supporting infrastructure.

#### Chaos Engineering:

- Purpose: To introduce controlled chaos and observe system behavior.
- Explanation: Chaos experiments are designed to uncover vulnerabilities, improve system resilience, and enhance overall reliability.

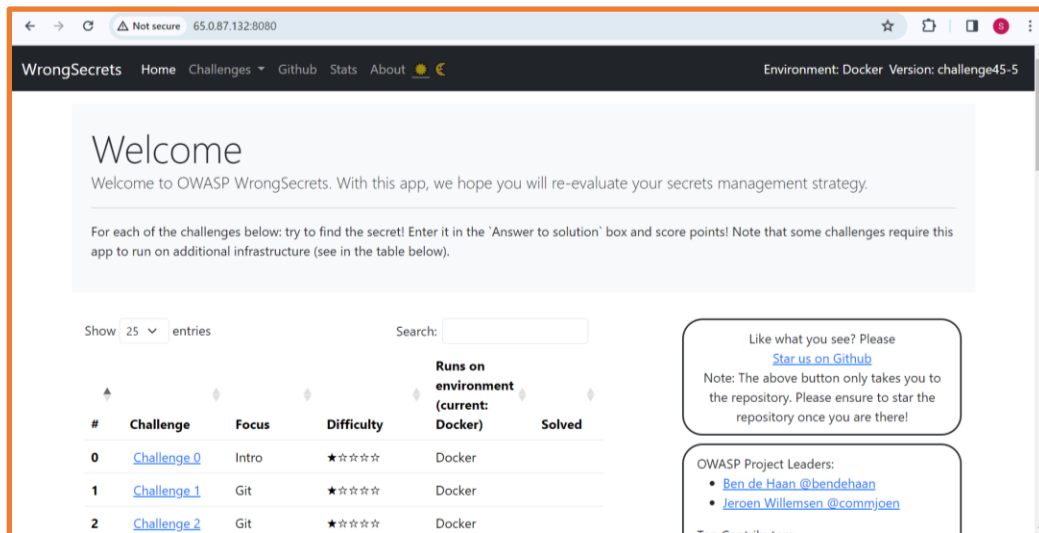
# Chaos Engineering - Report

## Implementation:

### 1. Pulling WrongSecrets Docker Image::

- **Command:**  

```
docker run -p 8080:8080 jeroenwillemsen/wrongsecrets:latest-no-vault
```
- **Explanation:** The -p flag maps port 8080 from the host to port 8080 in the container.



### 2. Security Analysis with Snyk:

- **Explanation:** Utilize Snyk to scan project dependencies and Docker container images for known vulnerabilities. Receive continuous monitoring alerts and remediation guidance to enhance security.

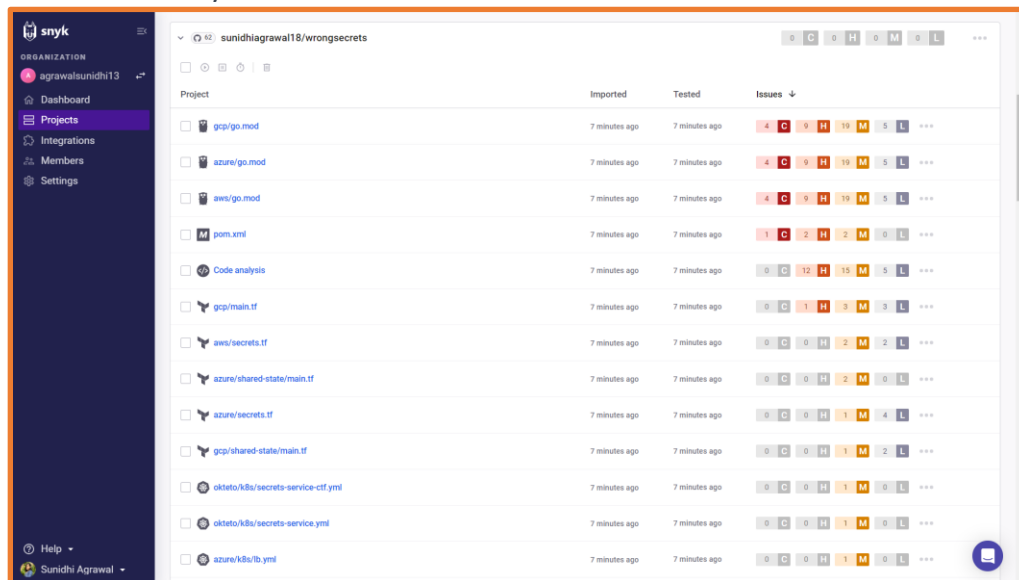


Figure 3: Snyk Vulnerability scan of OWASP WrongSecrets

# Chaos Engineering - Report

## Observation and Analysis:

### Vulnerabilities Identified in the Project:

#### - CVE-2022-23555: Improper Authentication in authentik:

- **Description:** authentik is an open-source Identity Provider focused on flexibility and versatility. Versions prior to 2022.11.4 and 2022.10.4 are vulnerable to Improper Authentication.

#### - CVE-2018-10054: Remote Code Execution (RCE) in com.h2database:h2:

- **Description:** H2 1.4.197, as used in Datomic before 0.9.5697 and other products, allows remote code execution because CREATE ALIAS can execute arbitrary Java code.

#### - Hardcoded Secret

- **Description:** The product uses hard-coded constants instead of symbolic names for security-critical values, which increases the likelihood of mistakes during code maintenance or security policy change.

### In response to these vulnerabilities:

- Regularly monitor and apply security updates for all software components.
- Conduct thorough security assessments and penetration testing to identify and address potential vulnerabilities.
- Implement strong access controls, including proper authentication mechanisms and least privilege principles.
- Educate development and operational teams on secure coding practices and security best practices.
- Establish a process for handling and responding to security incidents promptly.

### Conclusion:

The deployment of the OWASP WrongSecrets server on Docker provides a controlled environment for security testing and training purposes. The use of Docker ensures consistency across deployments, making it easy to scale and reproduce the environment. Regular monitoring of container logs and proper networking configuration are essential for maintaining a secure and functional deployment.