|  |  |
| --- | --- |
| Name: | Arpan Nyati |
| Lab User ID: | 23SEK3324\_U04 |
| Date: | 10-01-24 |
| Application Name: | Wrong Secrets |

**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)

A screenshot of a computer

Description automatically generated

OWASP Wrong Secrets System Architecture

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)

OWASP (Open Web Application Security Project) Wrong Secret is effectively showcasing an insecure configuration such as storing sensitive information like API keys, passwords, and tokens in insecure manner, which makes the application vulnerable to attacks. It is widely used as a reference application during demonstrations or hands-on sessions about insecure configurations.

Normal behavior of the OWASP Wrong Secret application illustration includes:

1. The application usually contains sensitive secret keys that are incorrectly or insecurely stored in application files, version control systems (like Git), or even hard-coded in the application code itself.

2. The wrongly included secrets in the application are clearly visible and accessible, revealing sensitive information about the system, like database credentials, API keys, or private encryption keys.

3. The application may also present lack of proper encryption systems and practices, which means anyone with access to the application can view and misuse the secret keys.

4. The application might be set up in a way that allows extraction and misuse of these secrets, primarily via automated scripts or malicious actors.

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**

Ideally if a malicious user gains access to our system or application, our system will securely guard sensitive information and prevent unauthorized access or data breaches.

Assuming that OWASP Wrong Secret application already contains vulnerabilities due to exposure of sensitive information like API keys and passwords, if a user gains internal or external access to our system, they could exploit these secrets leading to unauthorized access or data breaches.

942236

**Unknown**

If a user gains access, they can exploit API keys and other secrets. Ideally in an application user should not be able to view or extract secrets.

Given that OWASP Wrong Secret application is designed with certain security flaws, we hypothesize that in a chaotic system with unpredictable failures (service outages, network issues, abrupt shutdowns), these failures could lead to additional unforeseen exploits or security loopholes being exposed

**Unknown**

**Known**

Experiment:

(Document your Preparation, Implementation, Observation and Analysis )

**Preparation:**

1. Creating an Instance on AWS: Utilizing Amazon's EC2 web service, you first decide on an instance type based on your budget and resource demands. The chosen instance type was t2.medium.

2. Updating the Instance: Once the instance is live, it's important to keep it updated with the latest security patches and bug fixes. This can be done by running commands such as sudo yum update (Amazon Linux) or sudo apt-get update && sudo apt-get upgrade (Ubuntu).

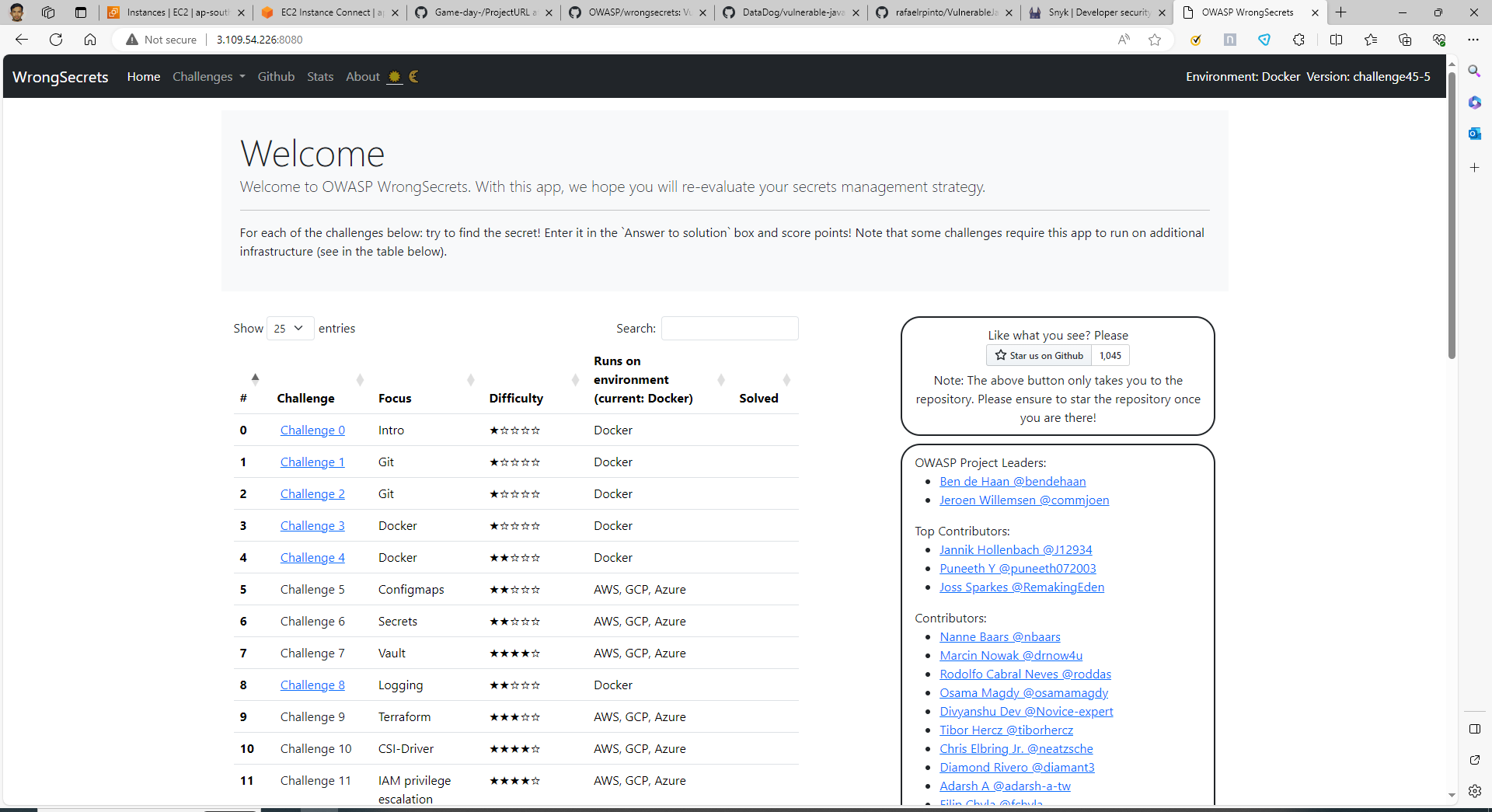
3. Installing Docker: Docker is a platform that allows you to automate the deployment, scaling, and isolation of applications. It uses containerization to package up an application with all of the parts it needs, such as libraries and other dependencies and ships it as one package.

**Implementation:**

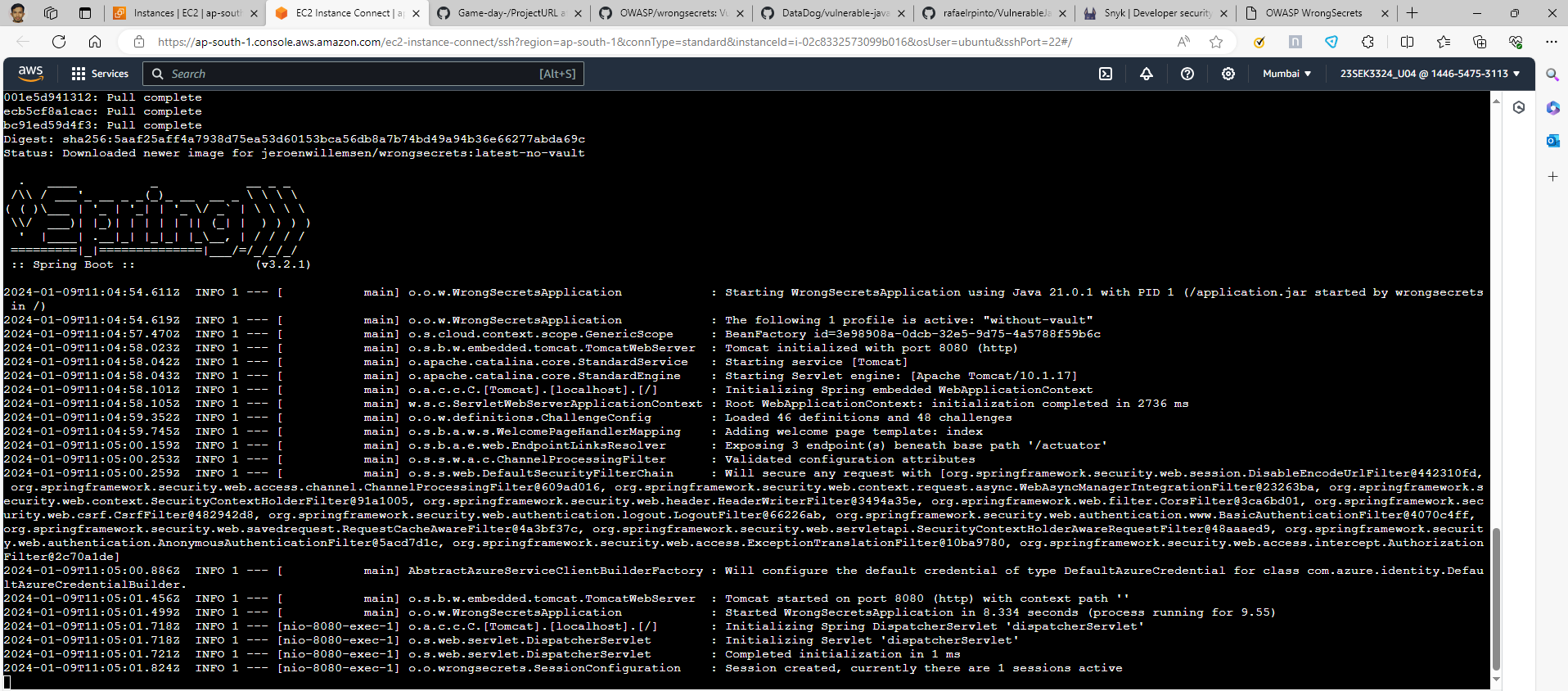
1. Making a server live: A server was developed and deployed using Docker. This could involve setting up a Docker file, building the Docker image, and then running a container from that image.

2. OWASP ZAP: The Zed Attack Proxy (ZAP) is one of the world’s most popular free security tools and is actively maintained by a dedicated international team of volunteers. It can be used for both automated and manual security testing.

3. Snyk: An open-source security platform designed to help software-driven businesses enhance developer security. Snyk made sure that the server and its dependencies are free of known vulnerabilities (as listed in its vulnerability database).



Live Web Page of OWASP Wrong Secrets



Live OWASP Wrong Secrets server on AWS EC2

**Observation:**

After the server was live, a scan was performed by using the Snyk tool. This tool analyzed the server's code for known vulnerabilities like the OWASP Wrong Secret. During this examination, Snyk compared the services running on the Docker container against its comprehensive database of known security vulnerabilities. Any potential security issues being identified would suggest that respective countermeasures should be employed to mitigate such vulnerabilities.

Note: Any discovered vulnerabilities, their potential risks, and recommended solutions will mainly depend on the Snyk output results. It's recommended that regular scans are made part of the deployment process, to keep security checks on-going and up-to-date.

A screenshot of a computer

Description automatically generated

Vulnerability scan results using Snyk

**Analysis**

There are over 13 critical, 42 high, 86 medium and 48 low vulnerabilities in the OWASP Wrong Secrets Application.

Some of them are explained below.

1. CVE-2020-15113:

CVE-2020-15113 refers to a vulnerability in the command "pomerium routes". This issue affects Pomerium up to and including v0.10.0. In certain circumstances, an attacker could use this vulnerability to gain unauthorized access. When confronted with certain URLs, the route rewriter can cause Pomerium to incorrectly route traffic (including directing traffic to unexpected routes). To mitigate the vulnerability, it is recommended to upgrade to Pomerium version 0.10.0-rc2 or later.

2. CVE-2018-1098:

This vulnerability involves "etcd" prior to version 3.2.26 and prior to version 3.3. Your system could be vulnerable to a Remote Denial of Service attack when an etcd server is directly connected to the internet. A malicious remote attacker could leverage this vulnerability using a specially crafted request causing the etcd to consume excessive resources, leading to a denial of service. Upgrading to etcd 3.2.26+/3.3.3+ or using a reverse proxy between etcd server and the public internet can mitigate this vulnerability.

3. CVE-2018-1099:

This is a high-severity security flaw found in the Dynamic Host Configuration Protocol (DHCP) client packages as supplied in Red Hat Enterprise Linux. It allows for arbitrary command execution with root privileges via a malicious DHCP server. Affects dhcp-4.1.1-53.P1.el5 and before version dhcp-4.1.1-58.P1.el5. Users are advised to update to dhcp-client-4.2.5-68.el7 to resolve the issue and avoid potential system compromise.

4. CVE-2020-15112:

CVE-2020-15112 refers to a vulnerability in StreamPipes versions prior to 0.66.0. The installer does not enforce the creation of unique credentials or restrict the IP range behind a firewall by default. This vulnerability exposes sensitive information and allows unauthorized access. By updating StreamPipes to version 0.66.0 or later, this issue can be mitigated.