|  |  |
| --- | --- |
| Name: | Sandeep Chowdary |
| Lab User ID: | 23SEK3324\_U03 |
| Date: | 10-01-2024 |
| Application Name: | OWASP WrongSecrets |

**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)

AWS infra

user

Docker

http://<ip address>:8080

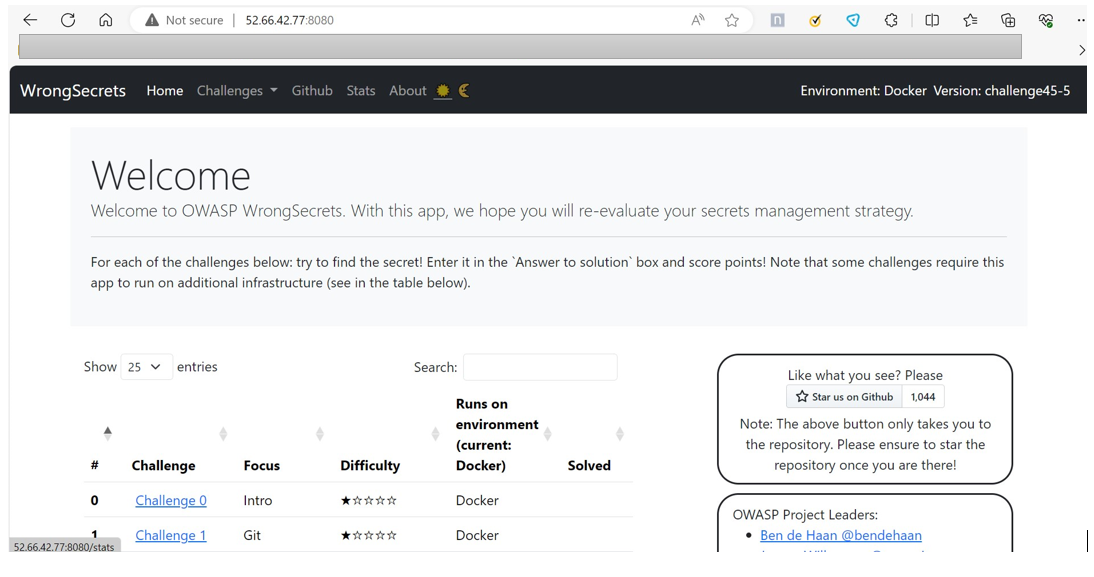
browser

container

Web app

browser

http://<ip address>:8080



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)

1)The name of the web application is OWASP wrong secrets. The website is packed with real life examples of how to

not store secrets in your software.

2) Each of these examples is captured in a challenge, which you need to solve using various tools and techniques.

Solving these challenges will help you recognize common mistakes & can help you to reflect on your own secrets

management strategy.

3) The UI of the application is user friendly.

4) The application is running on a single machine which has the following parameters:

Os: ubuntu, cpu: t2.medium, storage:15GB

5) The are more than one page and the application have various tabs to navigate between pages.

6)The application performance is good under normal load.

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**

While the system is designed to scale horizontally, the upper limits of scalability under various conditions, such as sudden spikes in user traffic, are not fully known or tested.

Under normal operating conditions, the system maintains a steady state of performance characterized by acceptable response times and resource utilization.

**Unknown**

**Unknown**

**Known**

Interactions between the system and dynamic cloud services may lead to unknown challenges or unexpected behaviors

While it is assumed that the system gracefully degrades under stress, there may be specific failure scenarios or combinations of failures that result in less-than-expected graceful degradation

1)Steady State of System Performance:

Under normal operating conditions, the system maintains a steady state of performance characterized by acceptable

response times and resource utilization.

2) Scaling Limits Under Load:

While the system is designed to scale horizontally, the upper limits of scalability under various conditions, such as

sudden spikes in user traffic, are not fully known or tested.

3) Assumed Graceful Degradation:

While it is assumed that the system gracefully degrades under stress, there may be specific failure scenarios or combinations of failures that result in less-than-expected graceful degradation.

4) Dynamic Cloud Service Interactions:

Interactions between the system and dynamic cloud services may lead to unknown challenges or

unexpected behaviors during chaos experiments.

Experiment:

(Document your Preparation, Implementation, Observation and Analysis )

1)Overview of the application:

The OWASP WrongSecrets game website. The game is packed with real life examples of how to not store secrets

in your software. Each of these examples is captured in a challenge, which you need to solve using various tools and

techniques. Solving these challenges will help you recognize common mistakes & can help you to reflect on your own

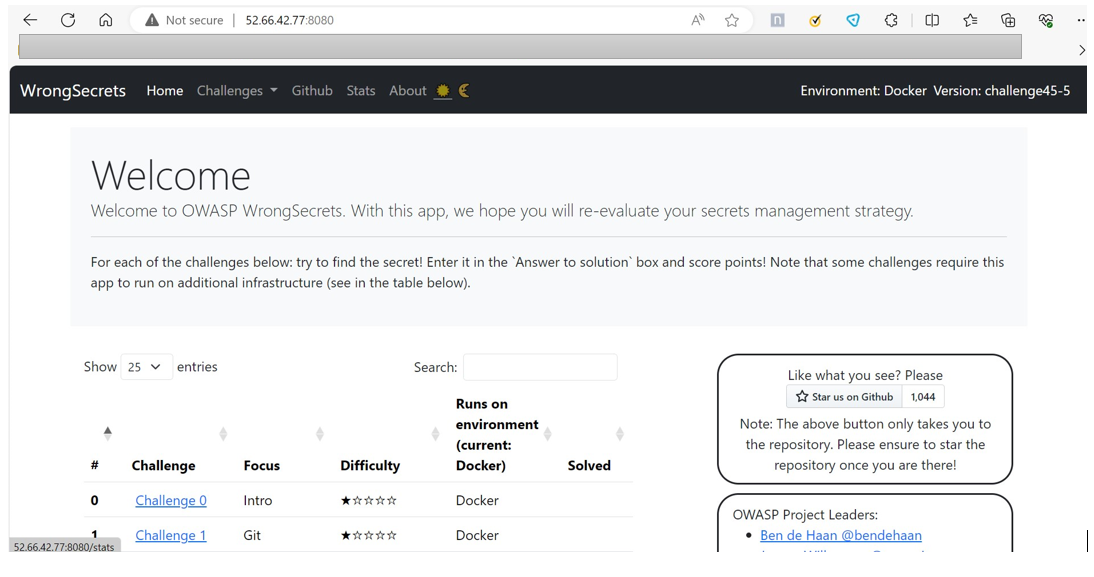
secrets management strategy.

2)Making the application live: Created an ec2 instance with ubuntu os.

The application is made live using the docker containers. Following is a docker Command:

docker run -p 8080:8080 jeroenwillemsen/wrongsecrets:latest-no-vault

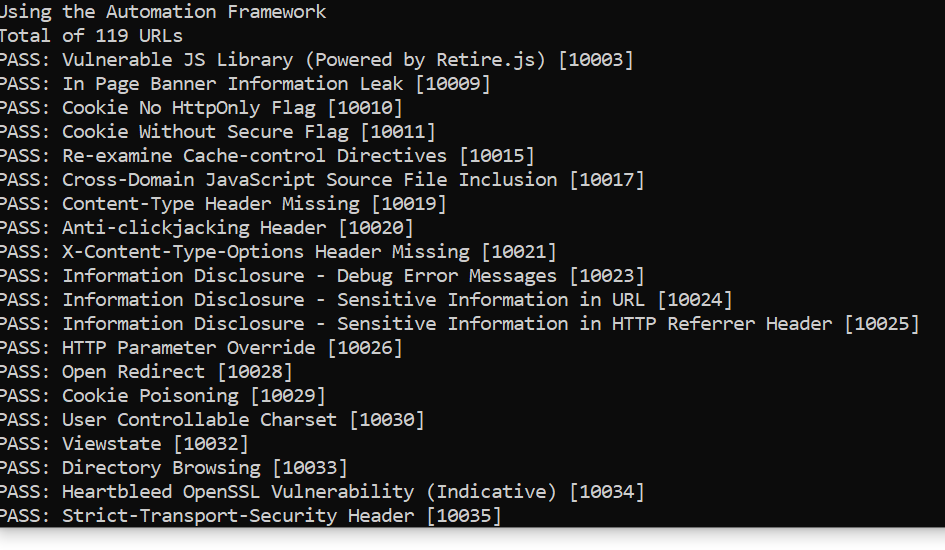
3)Checking the Application is live on port 8080:

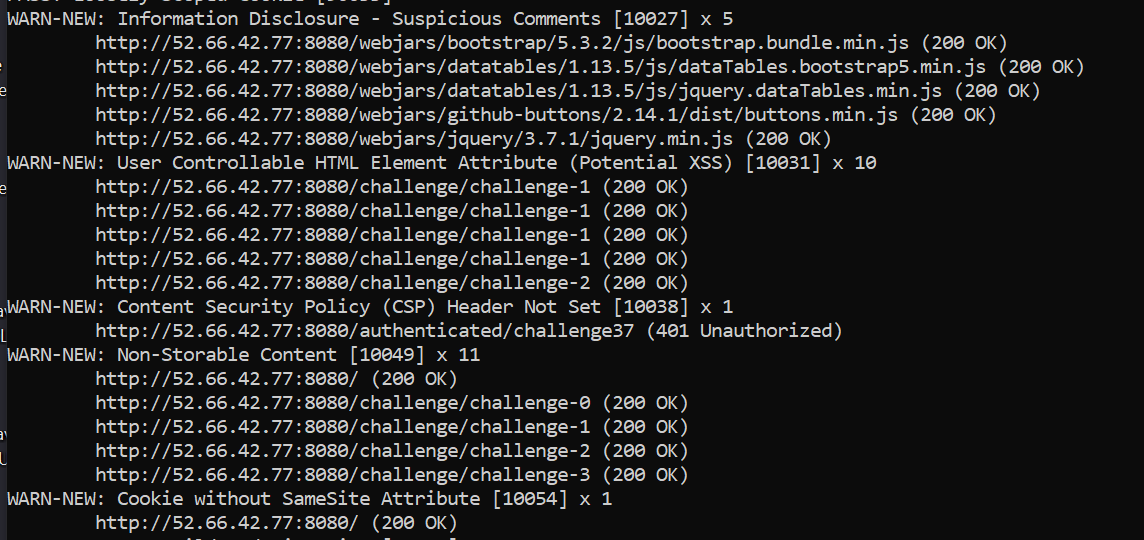


4)Scanning the vulnerabilities of the web application using Zap tool:

By using zap baseline scanner I scanned the web application with the following command:

docker run -t ghcr.io/zaproxy/zaproxy:stable zap-baseline.py -t http://52.66.42.77:8080







Vulnerabilities and solutions:

Alert Id 10027

Risk Informational

vul: The response appears to contain suspicious comments which may help an attacker. Note: Matches made within script blocks or files are against the entire content not only comments.

sol: Remove all comments that return information that may help an attacker and fix any underlying problems they refer to.

Alert Id 10031

vul: This check looks at user-supplied input in query string parameters and POST data to identify where certain HTML attribute values might be controlled. This provides hot-spot detection for XSS (cross-site scripting) that will require further review by a security analyst to determine exploitability.

sol: Validate all input and sanitize output it before writing to any HTML attributes.

Alert Id 10054

Risk Low

vul: A cookie has been set without the SameSite attribute, which means that the cookie can be sent as a result of a ‘cross-site’ request. The SameSite attribute is an effective counter measure to cross-site request forgery, cross-site script inclusion, and timing attacks.

sol: Ensure that the SameSite attribute is set to either 'lax' or ideally 'strict' for all cookies.

Alert Id 10055-4

Risk Medium

vul: Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks. Including (but not limited to) Cross Site Scripting (XSS), and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.

sol: Ensure that your web server, application server, load balancer, etc. is properly configured to set the Content-Security-Policy header.

7) **Performing Chaos engineering by using Gremlin:**

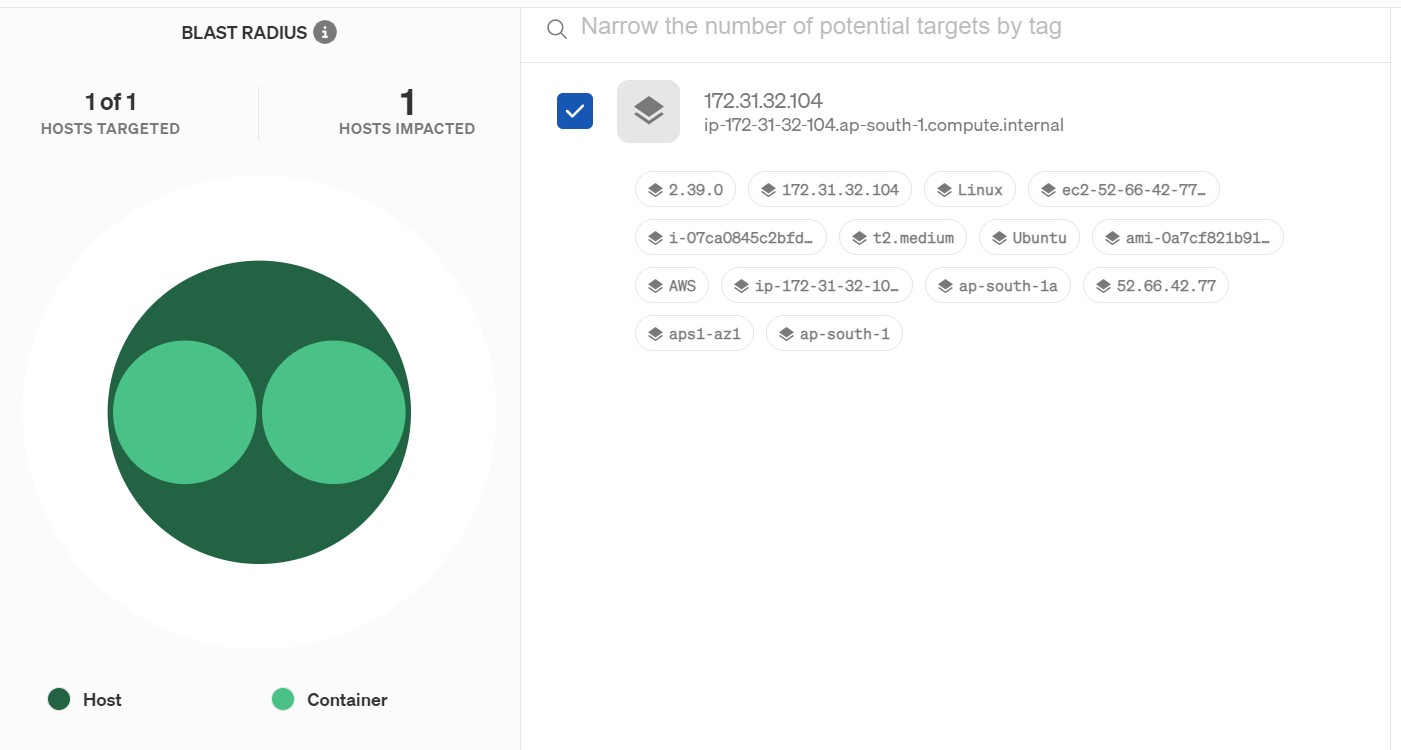
With the help of gremlin UI based web application I have performed a simple experiment to check the behavior of

the system in a disruptive environment.

For this I logged in to my gremlin account.

I installed the gremlin agent over the host machine and attached the machine to the gremlin and performed the

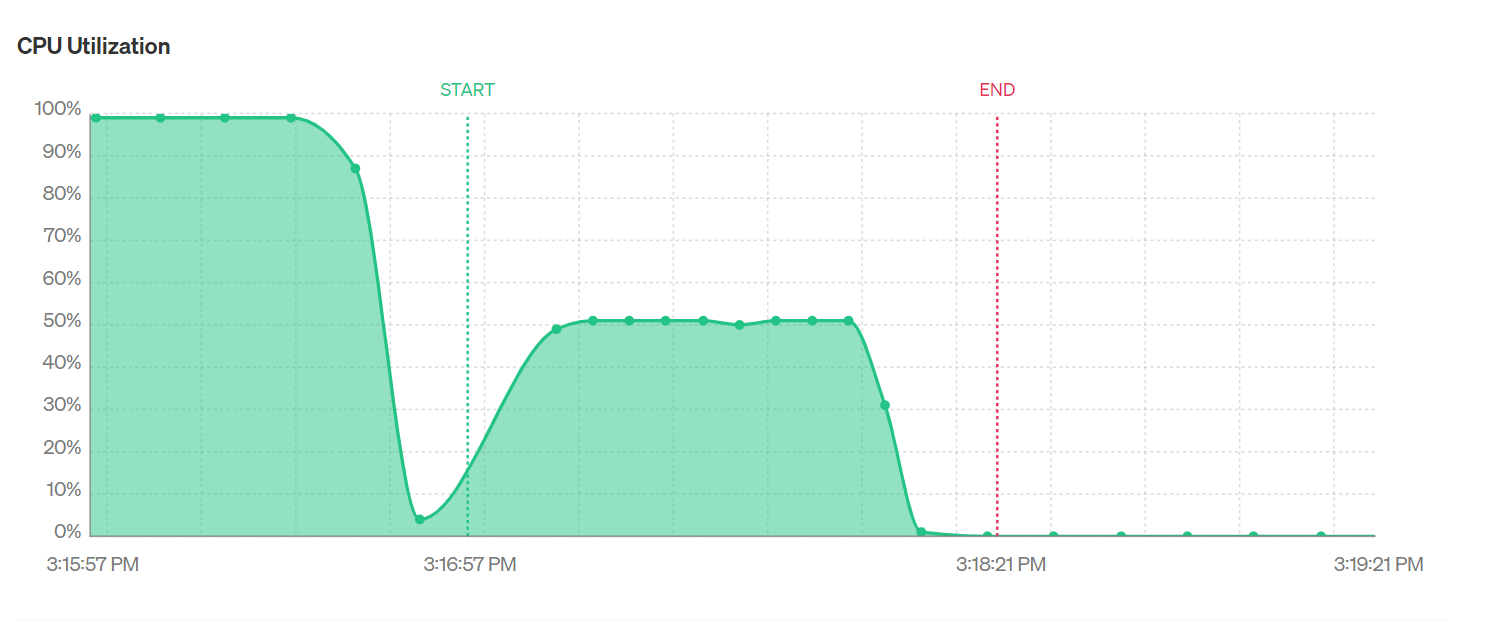
the following experiments.



Performed two experiments on the host machine:

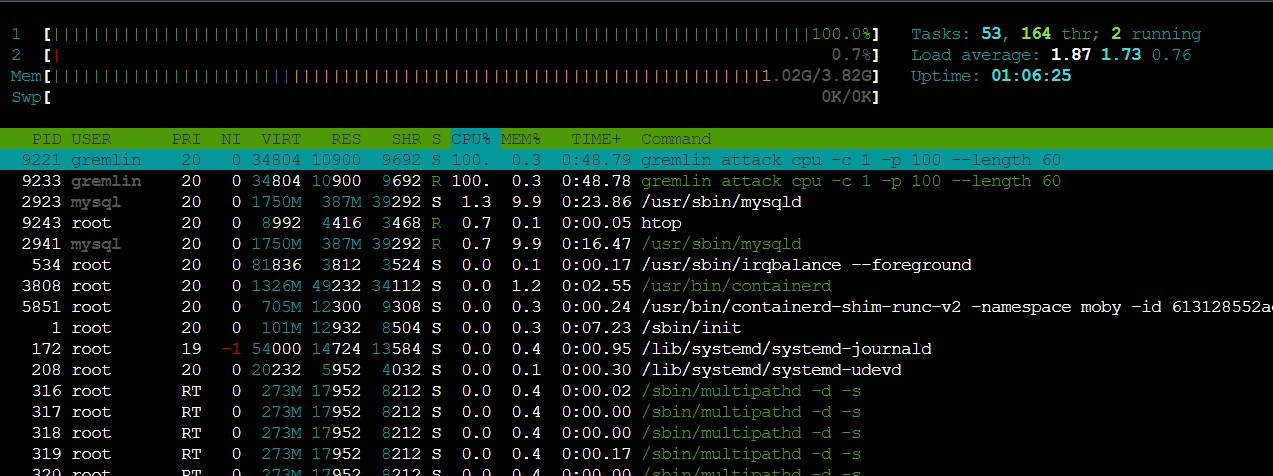
One experiment was to check what happens if the cpu utiliztions is 100%.

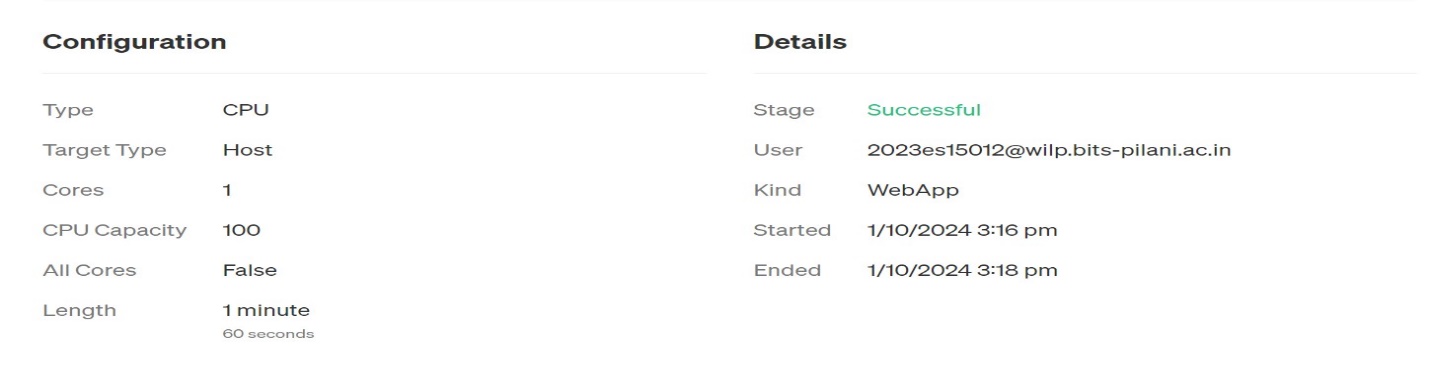
For another experiment I used a scenario that is present in the gremlin which attacks the memory of the host.



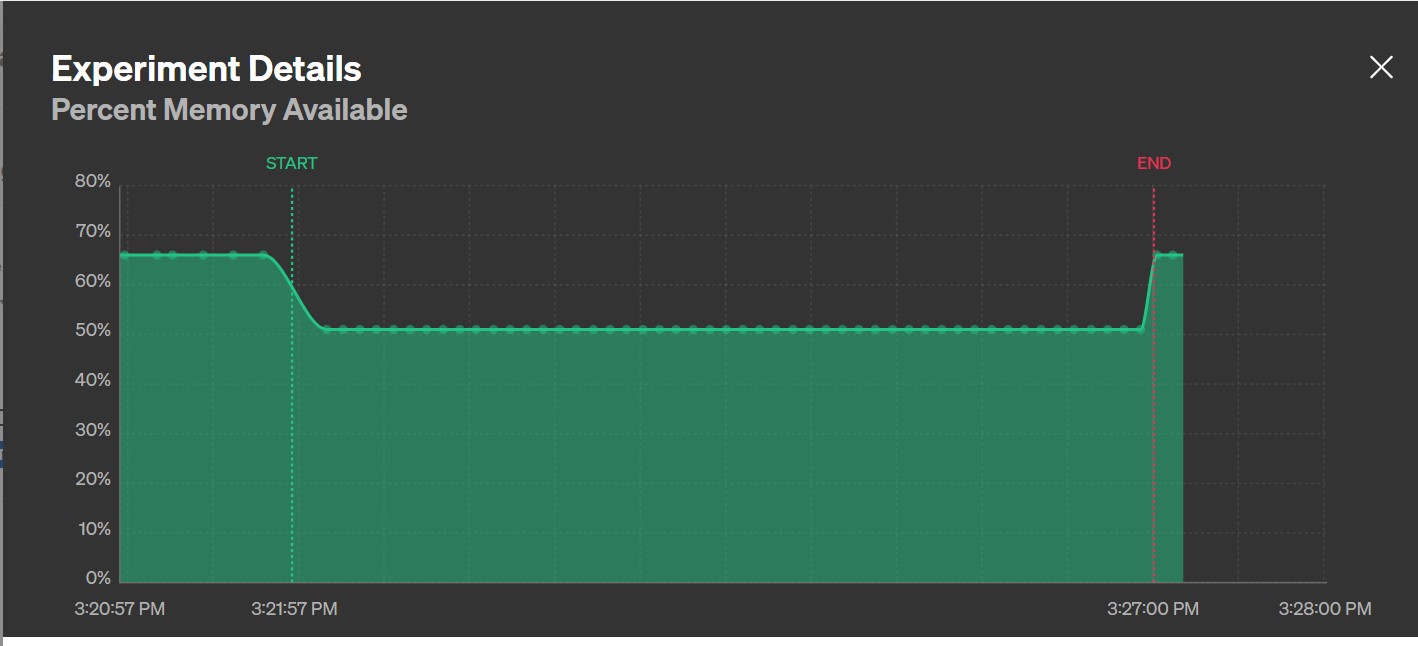
The cpu of the host machine was made 100% with this experiment.

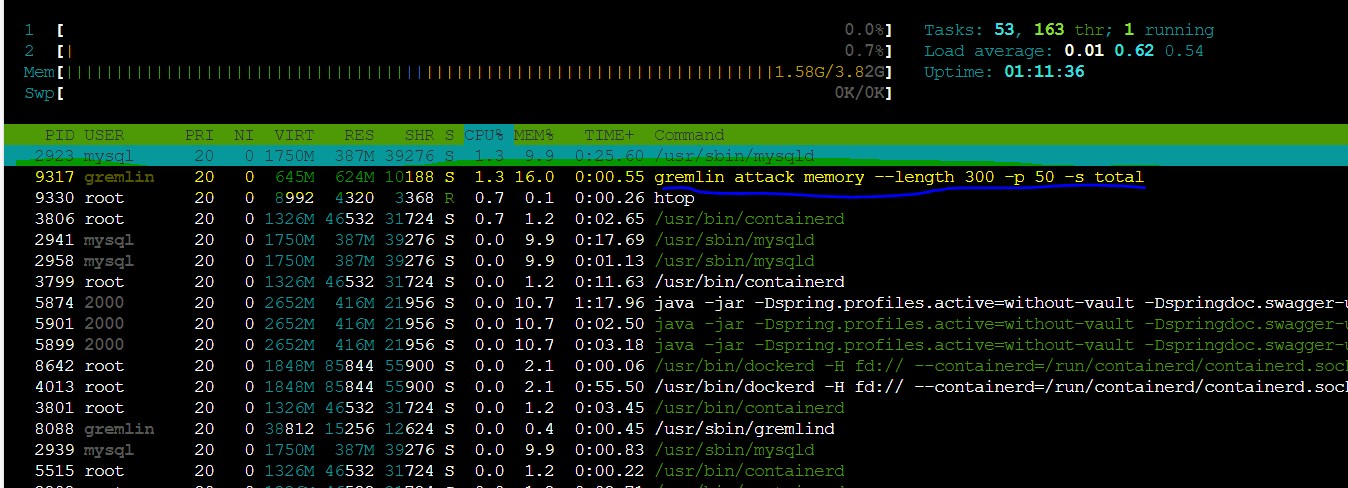
Below is the result.

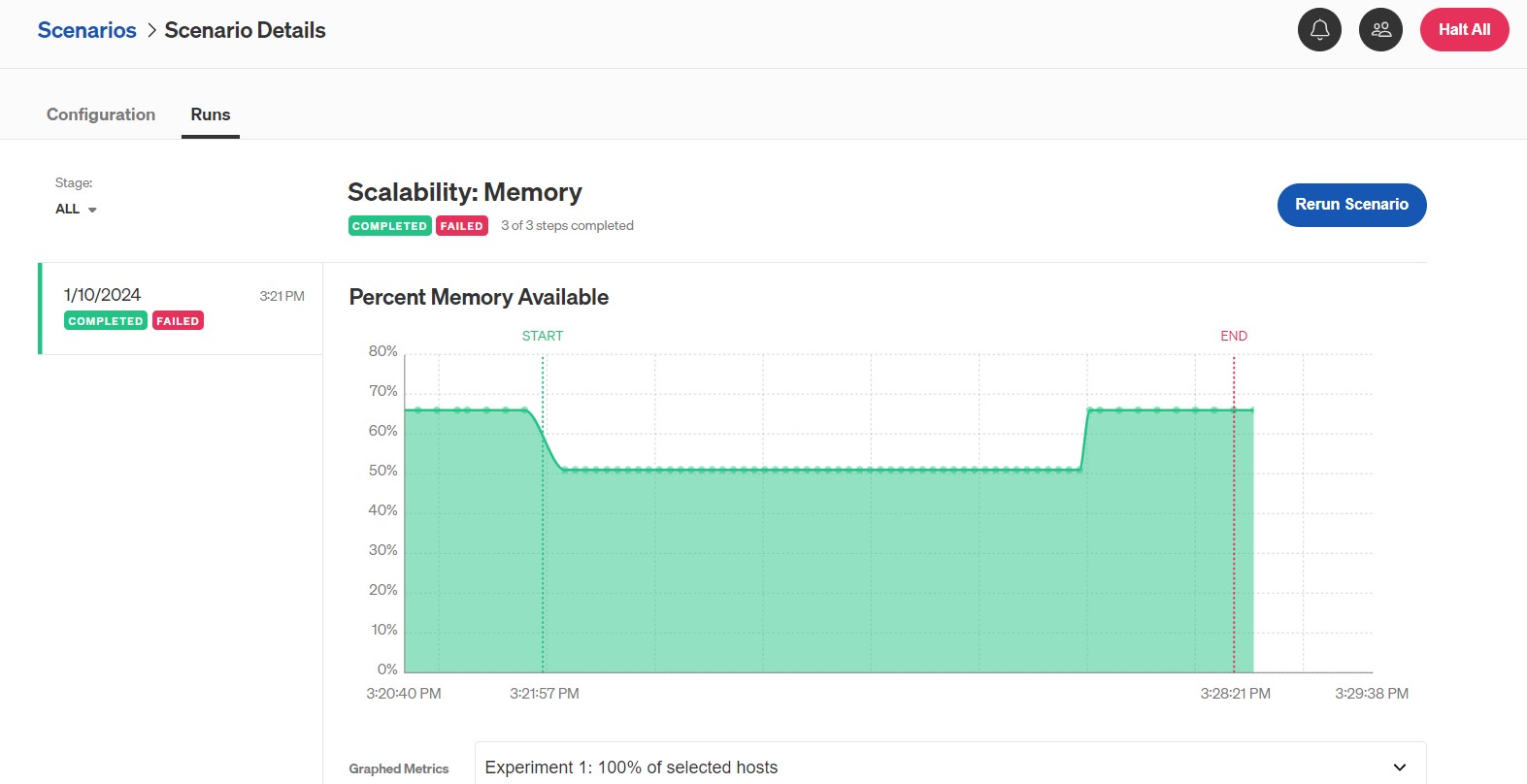


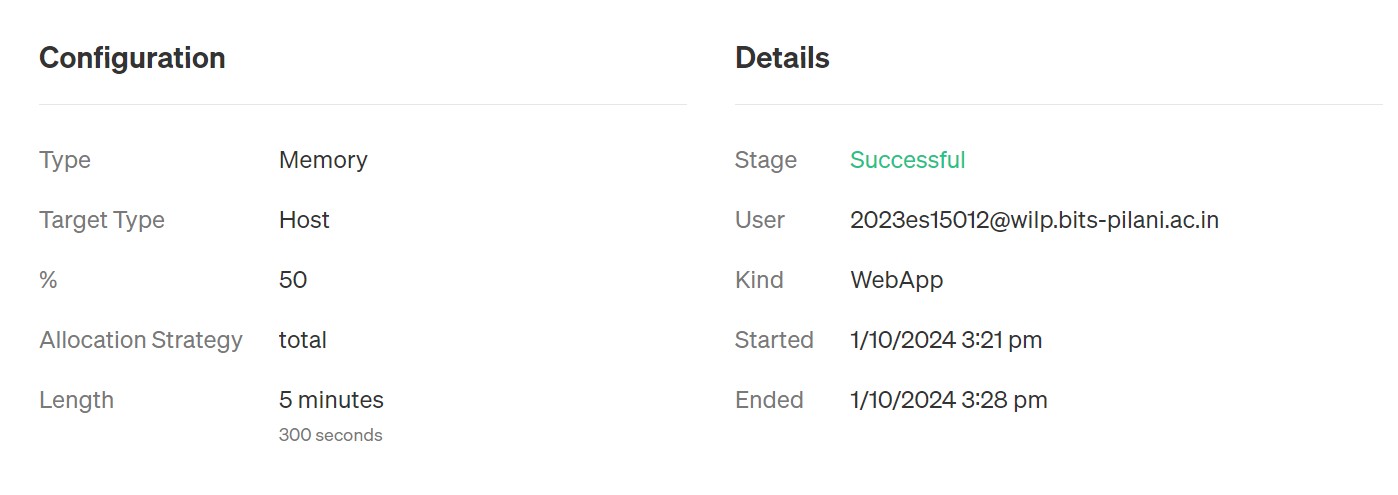


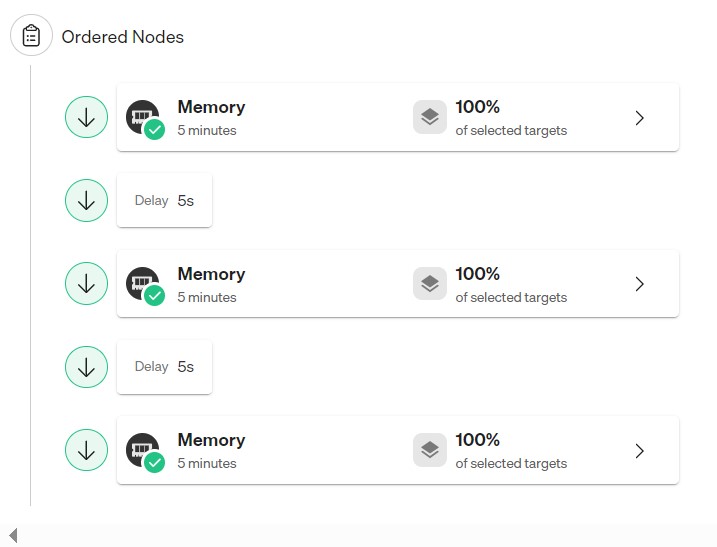
Attacking Memory:











Conclusion:

For the first experiment the cpu was attacked with the help of gremlin agent over the host machine. The experiment

was performed for 60 seconds. In that experiment the website was not responding properly as the cpu was consumed

100% by the gremlin agent. The system did not auto scale resources which caused the application down time.

For the second experiment the memory was attacked and the website was not responding for multiple requests at

a time. The system in this scenario didn’t scale up and application was in downtime.