Chaos Engineering – Hypothesis

Name:	GOWRAV KUMAR BAITHARU ARIPAKA
Lab User ID:	23SEK3324_U02
Date:	16-01-2024
Application Name:	OWASP Wrong Secrets

Follow the below guidelines:

Define Hypotheses and Scenarios:

- Similar to Chaos Engineering, start by defining hypotheses and scenarios related to potential threats.

Inject Controlled Failure:

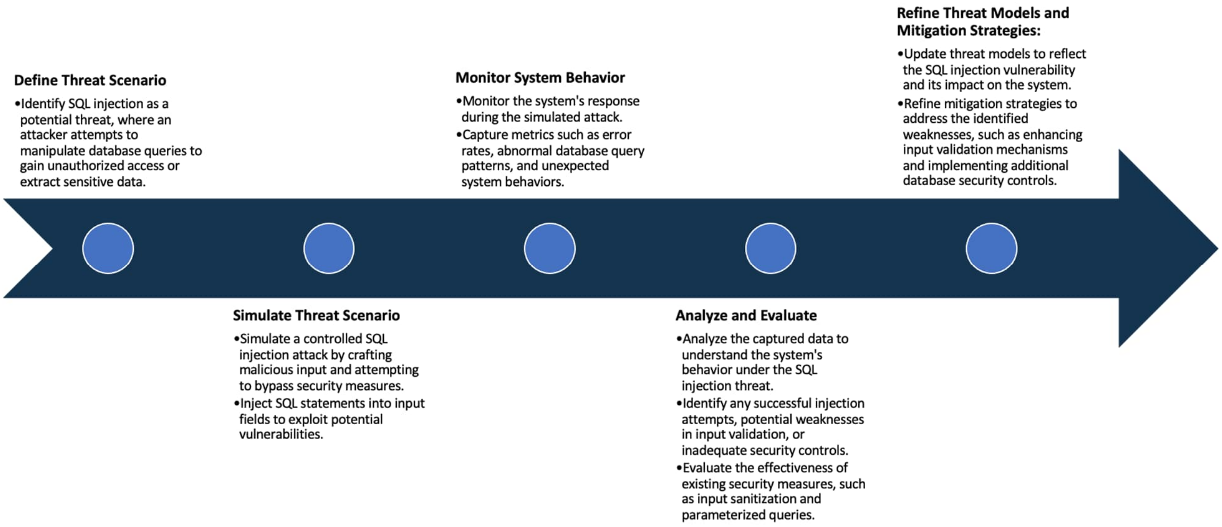
- Introduce controlled failure scenarios that mimic potential attack vectors or vulnerabilities identified in Threat Modeling.
- Simulate failure conditions, such as network disruptions, component failures, or data breaches, to observe the system's response.

Measure System Behavior:

- Capture and measure relevant system behavior metrics during the chaos experiments.
- Monitor the system's response to the injected failures, including performance metrics, error rates, and security-related indicators.
- Analyze and compare the observed behavior against the expected outcomes defined in the Threat Modeling process.

Learn and Iterate:

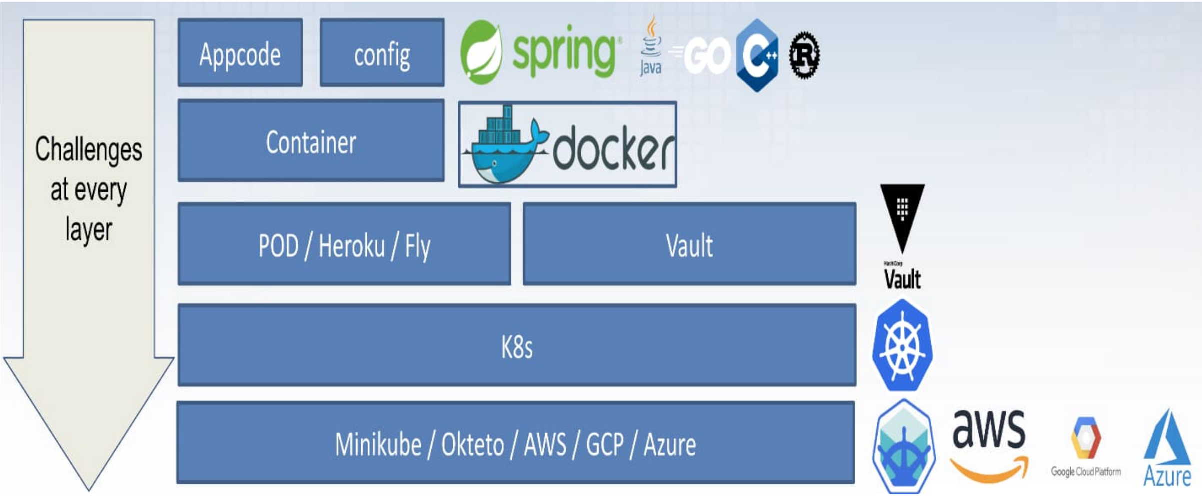
- Learn from the results of the chaos experiments and iterate on the Threat Modeling process.
- Analyze the observations and insights gained from the chaos experiments to refine the Threat Models. Update threat scenarios, adjust mitigation strategies, and improve security controls based on the lessons learned.



Chaos Engineering – Hypothesis

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)



OWASP Wrong Secrets System Architecture

Chaos Engineering – Hypothesis

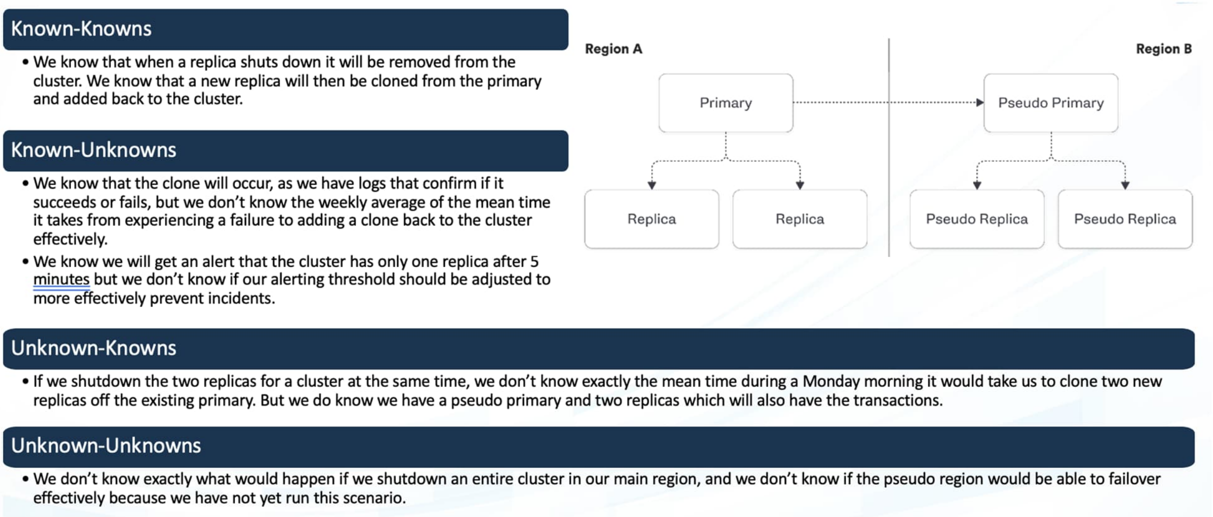
Define system’s normal behavior:

(Define the steady state of the system is defined, thereby defining some measurable outputs which can indicate the system’s normal behavior)

The web server initiates, listening on specified ports, like IP address: 8080. A user interacts with the hosted website through a web browser. The site comprises five components: Home, Challenges, GitHub, Stats, and About. The Challenges section displays various tasks, and upon user selection, redirects to dedicated challenge pages. These pages prompt users to submit solutions. If correct, the server reacts with a positive "Good Answer"; otherwise, a "Bad Luck, Try Again" message is displayed, accompanied by hints for assistance. This interactive platform engages users in problem-solving, fostering a dynamic and educational web experience. Additionally, users can explore GitHub for related resources and view stats related to challenges. Overall, the web server offers an interactive and educational environment with challenges, solutions, and community engagement elements.

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.")



Chaos Engineering – Hypothesis

	Known	Unknown
Known	If we induce a CPU outage by simulating high load, then the system will automatically scale up to handle increased demand, maintaining performance and availability.	If the application experiences any kind of external attack, then its defense mechanisms will effectively mitigate the threat, preventing unauthorized access or data compromise.
Unknown	Through load testing, engineers discover a particular microservice experiences significant latency only when multiple users concurrently access a specific feature.	If we shutdown the replica nodes then we don't know whether the pseudo primary nodes and it's replicas are not able to take over the transactions effectively

Chaos Engineering – Hypothesis

Experiment:

(Document your Preparation, Implementation, Observation and Analysis)

1)Overview of the application:

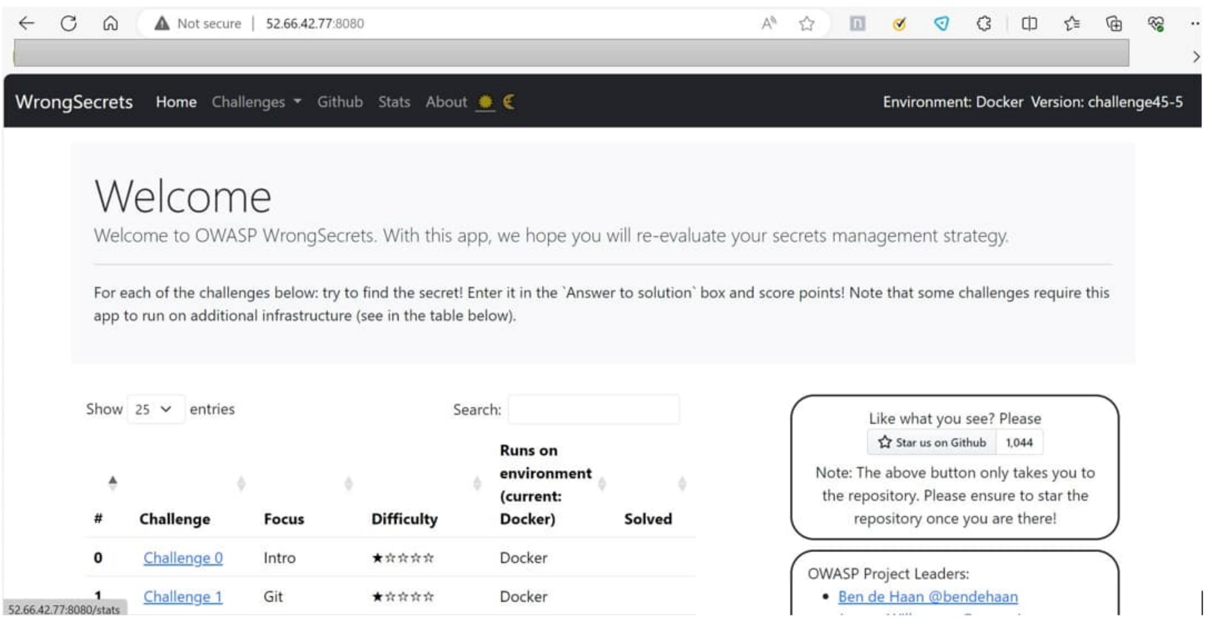
The OWASP WrongSecrets game website. The game is packed with real life examples of how to not store secrets in your software. Each of these examples is captured in a challenge, which you need to solve using various tools and techniques. Solving these challenges will help you recognize common mistakes & 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

```
Using the Automation Framework
Total of 119 URLs
PASS: Vulnerable JS Library (Powered by Retire.js) [10003]
PASS: In Page Banner Information Leak [10009]
PASS: Cookie No HttpOnly Flag [10010]
PASS: Cookie Without Secure Flag [10011]
PASS: Re-examine Cache-control Directives [10015]
PASS: Cross-Domain JavaScript Source File Inclusion [10017]
PASS: Content-Type Header Missing [10019]
PASS: Anti-clickjacking Header [10020]
PASS: X-Content-Type-Options Header Missing [10021]
PASS: Information Disclosure - Debug Error Messages [10023]
PASS: Information Disclosure - Sensitive Information in URL [10024]
PASS: Information Disclosure - Sensitive Information in HTTP Referrer Header [10025]
PASS: HTTP Parameter Override [10026]
PASS: Open Redirect [10028]
PASS: Cookie Poisoning [10029]
PASS: User Controllable Charset [10030]
PASS: Viewstate [10032]
PASS: Directory Browsing [10033]
PASS: Heartbleed OpenSSL Vulnerability (Indicative) [10034]
PASS: Strict-Transport-Security Header [10035]
```

```
WARN-NEW: Information Disclosure - Suspicious Comments [10027] x 5
        http://52.66.42.77:8080/webjars/bootstrap/5.3.2/js/bootstrap.bundle.min.js (200 OK)
        http://52.66.42.77:8080/webjars/datatables/1.13.5/js/dataTables.bootstrap5.min.js (200 OK)
        http://52.66.42.77:8080/webjars/datatables/1.13.5/js/jquery.dataTables.min.js (200 OK)
        http://52.66.42.77:8080/webjars/github-buttons/2.14.1/dist/buttons.min.js (200 OK)
        http://52.66.42.77:8080/webjars/jquery/3.7.1/jquery.min.js (200 OK)
WARN-NEW: User Controllable HTML Element Attribute (Potential XSS) [10031] x 10
        http://52.66.42.77:8080/challenge/challenge-1 (200 OK)
        http://52.66.42.77:8080/challenge/challenge-1 (200 OK)
        http://52.66.42.77:8080/challenge/challenge-1 (200 OK)
        http://52.66.42.77:8080/challenge/challenge-1 (200 OK)
        http://52.66.42.77:8080/challenge/challenge-2 (200 OK)
WARN-NEW: Content Security Policy (CSP) Header Not Set [10038] x 1
        http://52.66.42.77:8080/authenticated/challenge37 (401 Unauthorized)
WARN-NEW: Non-Storable Content [10049] x 11
        http://52.66.42.77:8080/ (200 OK)
        http://52.66.42.77:8080/challenge/challenge-0 (200 OK)
        http://52.66.42.77:8080/challenge/challenge-1 (200 OK)
        http://52.66.42.77:8080/challenge/challenge-2 (200 OK)
        http://52.66.42.77:8080/challenge/challenge-3 (200 OK)
WARN-NEW: Cookie without SameSite Attribute [10054] x 1
        http://52.66.42.77:8080/ (200 OK)
```

```
FAIL-NEW: 0    FAIL-INPROG: 0  WARN-NEW: 11    WARN-INPROG: 0  INFO: 0 IGNORE: 0       PASS: 54
```

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.	

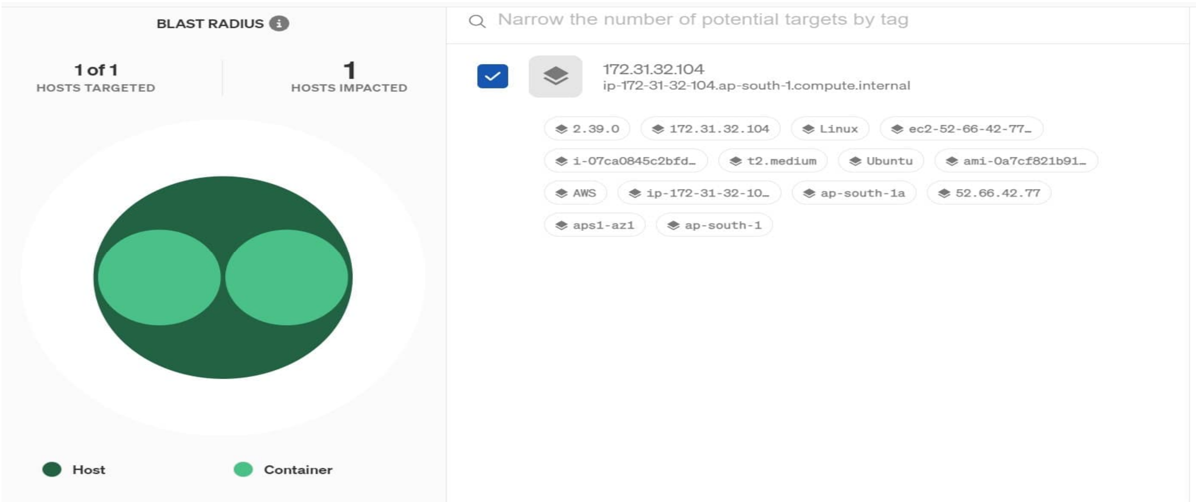
Alert Id	10055-4
Risk	Medium
vul: Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks. Including (but not limited to) Cross Site Scripting (XSS), and data injection attacks. These attacks are used for everything from data theft to site 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.

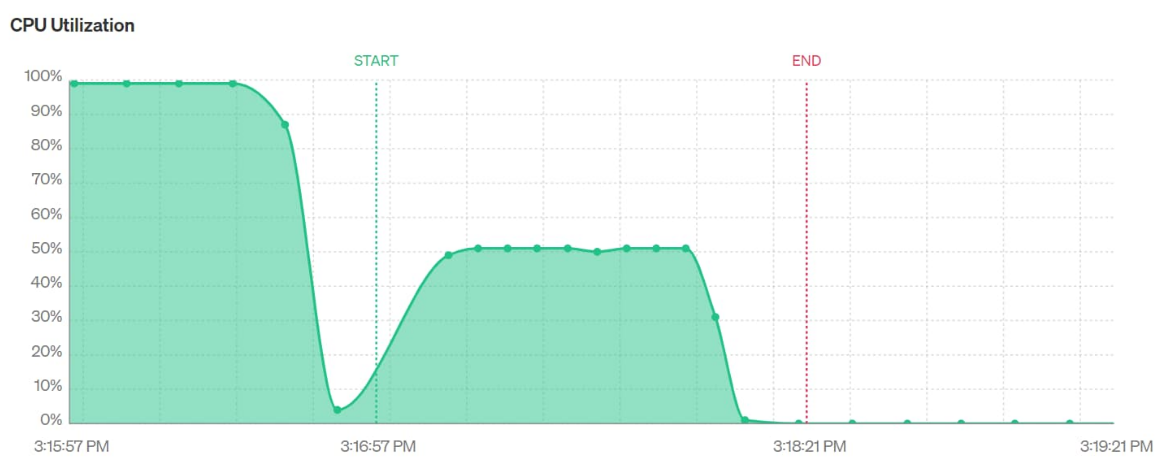
I installed the gremlin agent over the host machine and attached the machine to the gremlin and performed the the following experiments.



Performed two experiments on the host machine:

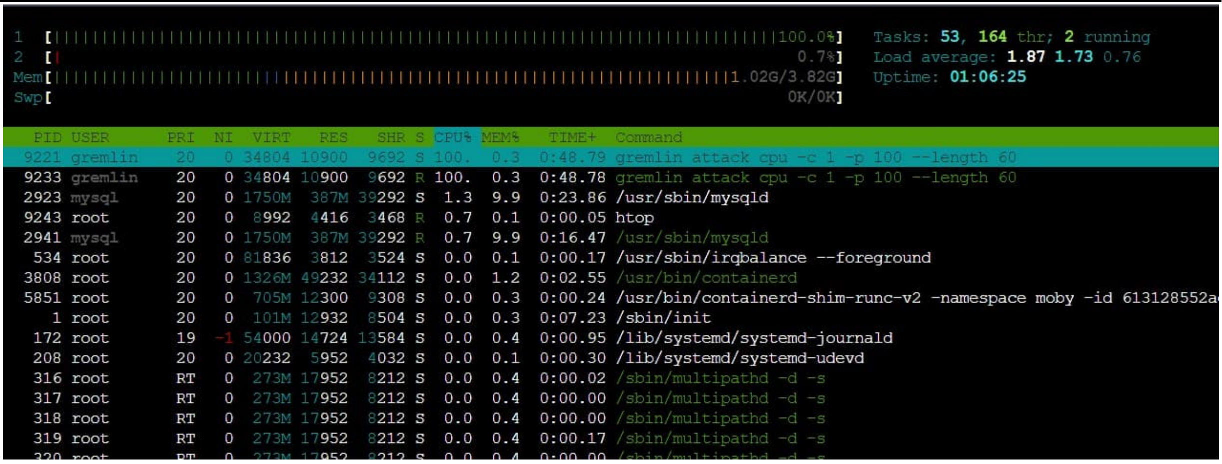
One experiment was to check what happens if the cpu utiliztions is 100%.

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.



```
1  [||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||100.0%]   Tasks: 53, 164 thr; 2 running
2  [|                                                                         0.7%]   Load average: 1.87 1.73 0.76
Mem[|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||1.02G/3.82G]   Uptime: 01:06:25
Swp[                                                                        0K/0K]

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 9221 gremlin    20   0 34804 10900  9692 S 100.  0.3  0:48.79 gremlin attack cpu -c 1 -p 100 --length 60
 9233 gremlin    20   0 34804 10900  9692 R 100.  0.3  0:48.78 gremlin attack cpu -c 1 -p 100 --length 60
 2923 mysql      20   0 1750M  387M 39292 S  1.3  9.9  0:23.86 /usr/sbin/mysqld
 9243 root       20   0  8992  4416  3468 R  0.7  0.1  0:00.05 htop
 2941 mysql      20   0 1750M  387M 39292 R  0.7  9.9  0:16.47 /usr/sbin/mysqld
  534 root       20   0 81836  3812  3524 S  0.0  0.1  0:00.17 /usr/sbin/irqbalance --foreground
 3808 root       20   0 1326M 49232 34112 S  0.0  1.2  0:02.55 /usr/bin/containerd
 5851 root       20   0  705M 12300  9308 S  0.0  0.3  0:00.24 /usr/bin/containerd-shim-runc-v2 -namespace moby -id 613128552a
    1 root       20   0  101M 12932  8504 S  0.0  0.3  0:07.23 /sbin/init
  172 root       19  -1 54000 14724 13584 S  0.0  0.4  0:00.95 /lib/systemd/systemd-journald
  208 root       20   0 20232  5952  4032 S  0.0  0.1  0:00.30 /lib/systemd/systemd-udevd
  316 root       RT   0  273M 17952  8212 S  0.0  0.4  0:00.02 /sbin/multipathd -d -s
  317 root       RT   0  273M 17952  8212 S  0.0  0.4  0:00.00 /sbin/multipathd -d -s
  318 root       RT   0  273M 17952  8212 S  0.0  0.4  0:00.00 /sbin/multipathd -d -s
  319 root       RT   0  273M 17952  8212 S  0.0  0.4  0:00.17 /sbin/multipathd -d -s
```

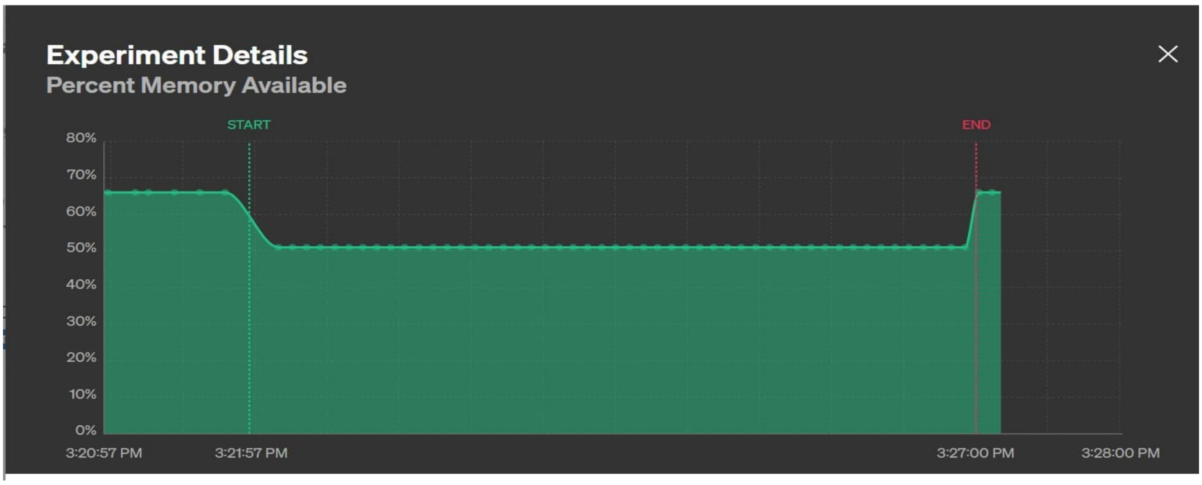
Configuration

Type	CPU
Target Type	Host
Cores	1
CPU Capacity	100
All Cores	False
Length	1 minute 60 seconds

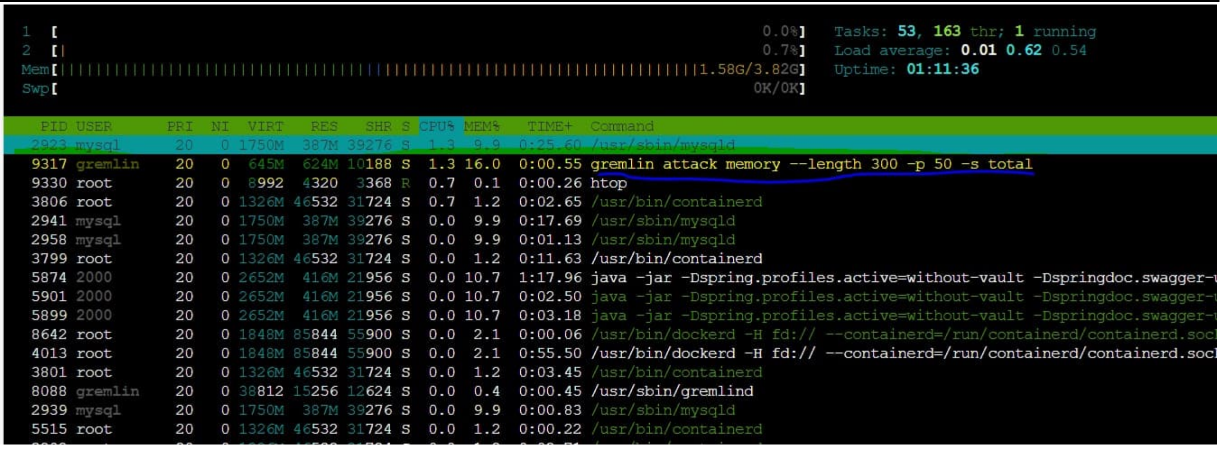
Details

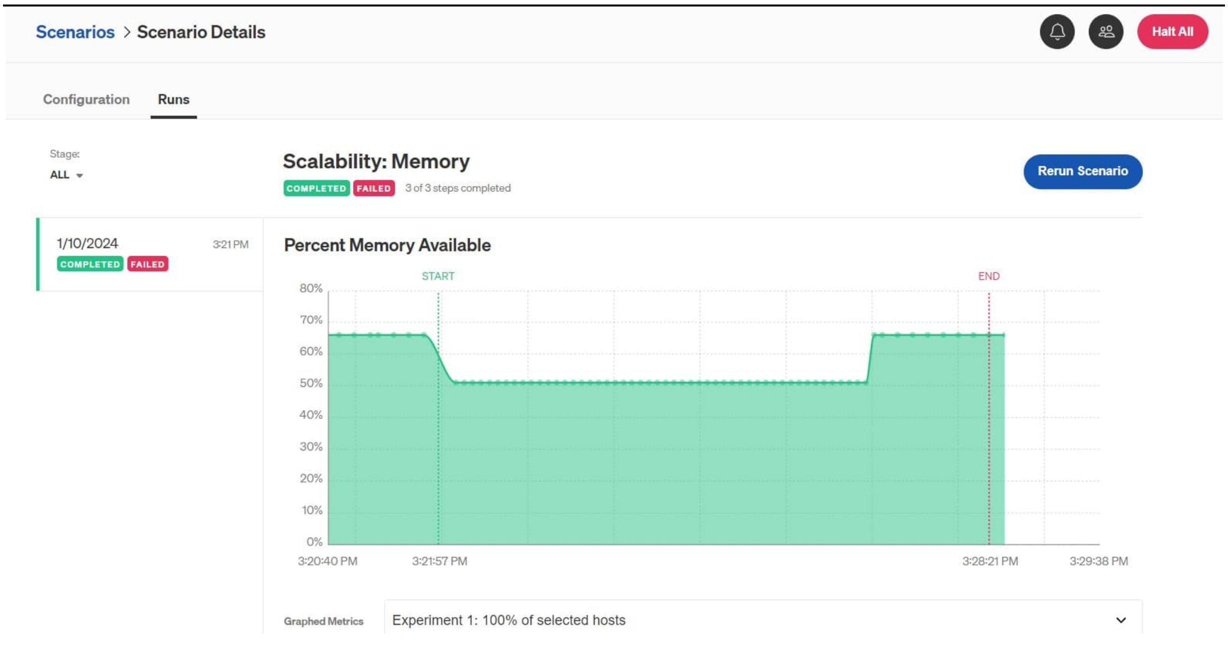
Stage	Successful
User	2023es15012@wilp.bits-pilani.ac.in
Kind	WebApp
Started	1/10/2024 3:16 pm
Ended	1/10/2024 3:18 pm

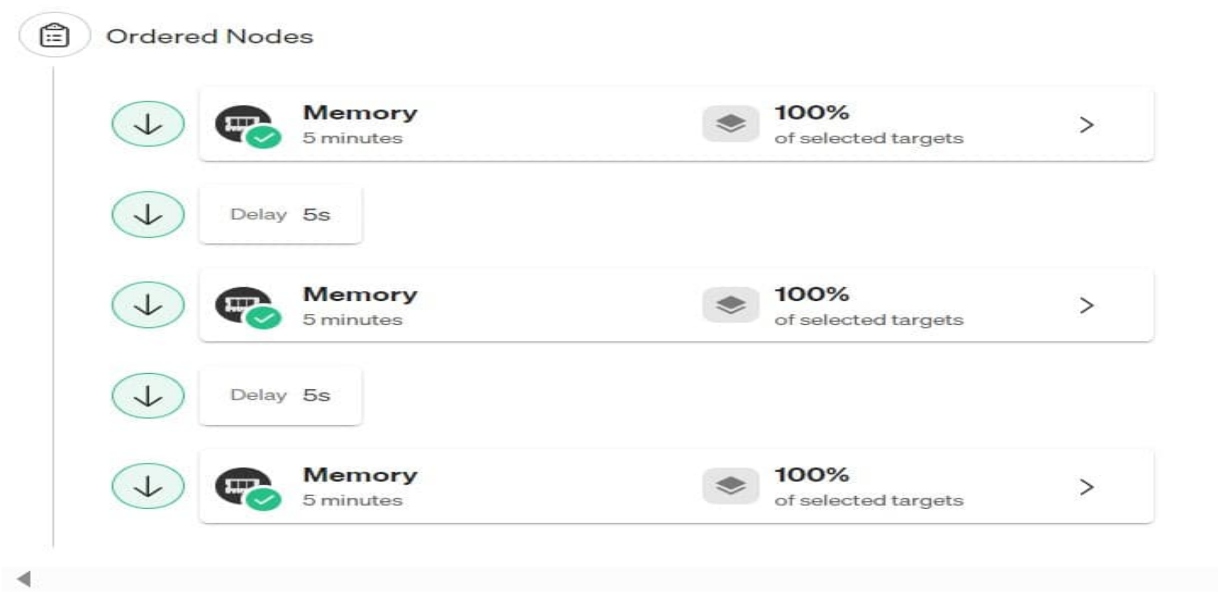
Attacking Memory:



Chaos Engineering – Hypothesis







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.