|  |  |
| --- | --- |
| Name: | Sandeep Chowdary |
| Lab User ID: | 23SEK3324\_U03 |
| Date: | 10-01-2024 |
| Application Name: | Website tester (Vulnerable Java application) |

**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>:8000

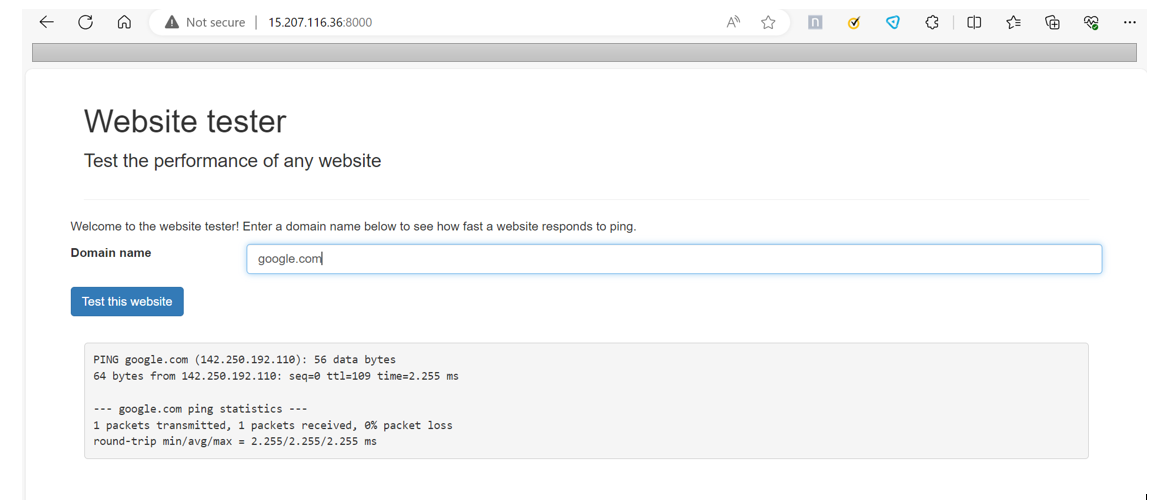
browser

container

Web app

browser

http://<ip address>:8000



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 website tester is working on port number 8000. Basic functionality of the application is to ping the domain

given by the user in the input box.

2) It has only one page. The user can only interact with this single page.

3) The website tester has one input box where a user can input any domain name to check whether the domain is

Listening or not.

4) The website tester using ping command to test the given website.

5) There is a button which after clicking will generate the ping command output on the screen.

6) The response time of the application is very quick.

7) It takes very less memory as it’s a simple web application.

8) There is not much functionality in the application so the cpu utilization is normal.

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

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

10) The overall working of the application is fine under normal conditions.

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.

Users can interact with the system through standard workflows, such as browsing, giving the domain name as input without encountering unexpected errors or disruptions.

**Unknown**

**Unknown**

**Known**

There may exist systemic failure modes or combinations of events that lead to novel and unpredictable issues within the system that have not been considered or documented

Stress testing is conducted based on assumed stress scenarios, but there may be stress conditions or combinations of stress factors that have not been explicitly considered.

1)Novel Systemic Failure Modes:There may exist systemic failure modes or combinations of events that lead to novel

and unpredictable issues within the system that have not been considered or documented.

2) Stress Testing Assumptions: Stress testing is conducted based on assumed stress scenarios, but there may be stress

conditions or combinations of stress factors that have not been explicitly considered.

3) 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.

4) Users can interact with the system through standard workflows, such as browsing, giving the domain name as

input without encountering unexpected errors or disruptions.

Experiment:

(Document your Preparation, Implementation, Observation and Analysis )

1)Overview of the application:

The website tester is web application which is developed using java. The basic functionality of the website is to test

Whether a website is listening to the requests. The website tester takes domain name as input and run’s a ping

Command to check the site is up or not. It give the output of the ping statistic. By using this application we can simple

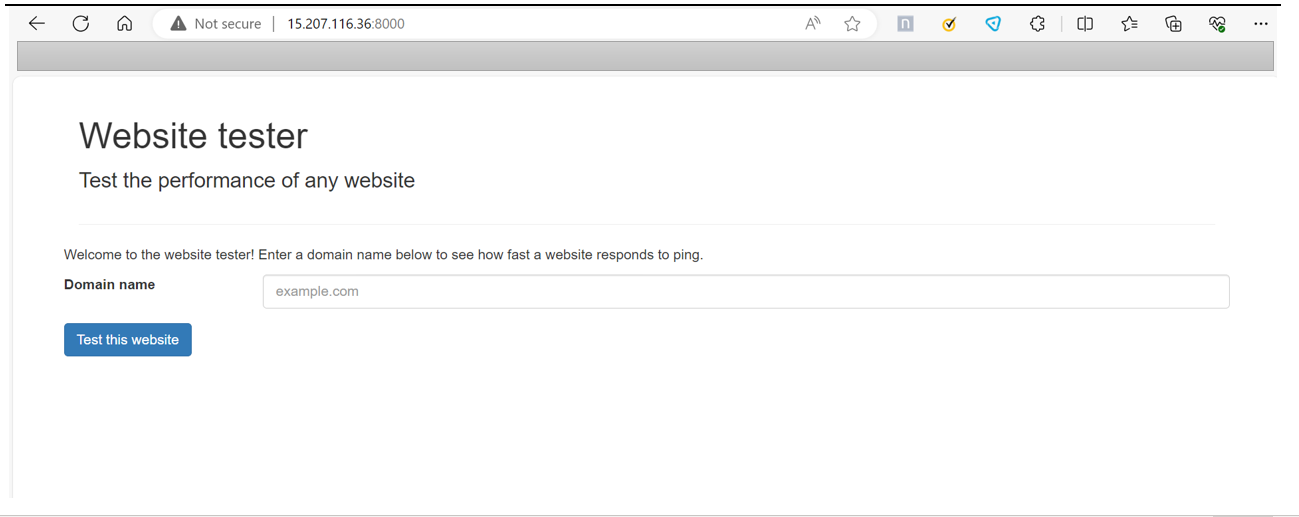
Check the availability of the website by using its domain name.

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 --rm -p 8000:8000 ghcr.io/datadog/vulnerable-java-application

3)Checking the Application is live on port 8000:



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

I installed the horusec using the following command:

>> curl -fsSL https://raw.githubusercontent.com/ZupIT/horusec/main/deployments/scripts/install.sh | bash -s latest

After installing the horusec with the I have scanned the repo with the following command:

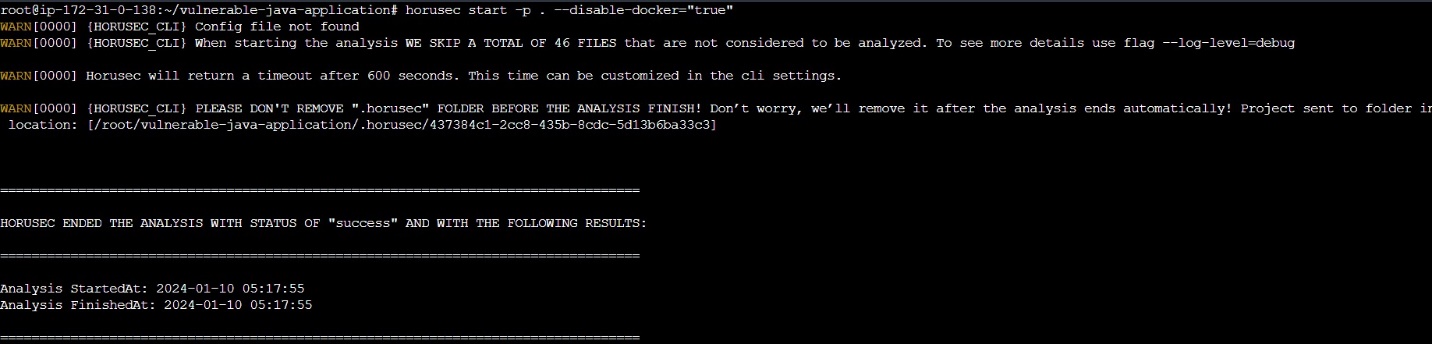


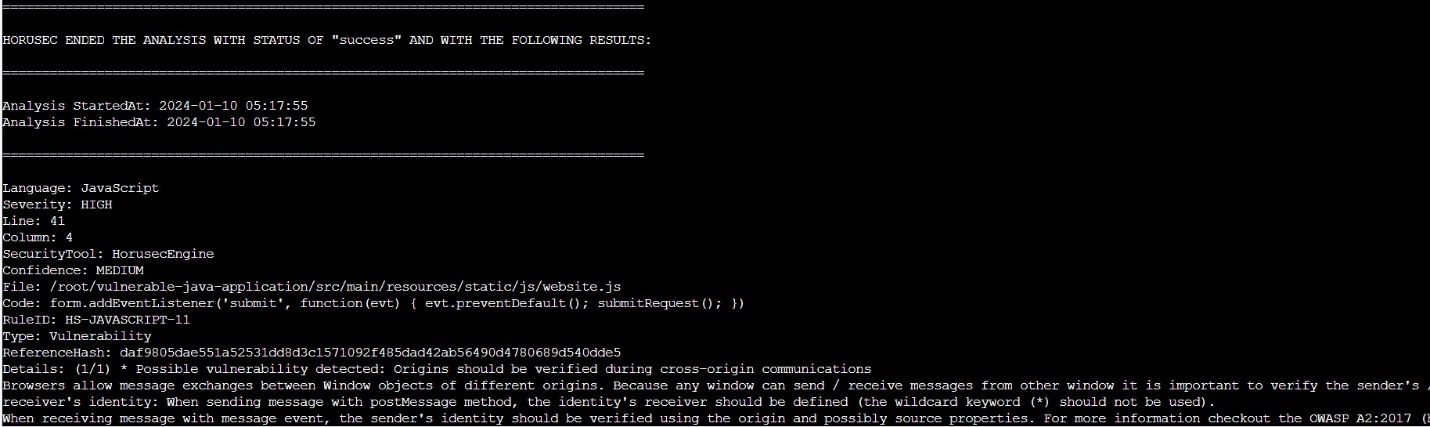
The following is the output report from horusec scan:

In this analysis, a total of 5 possible vulnerabilities were found and we classified them into:

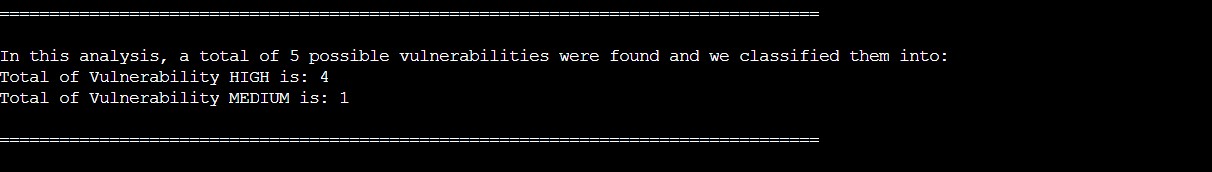
Total of Vulnerability HIGH is: 4

Total of Vulnerability MEDIUM is: 1









5) Scanning vulnerabilities using nuclei tool:

Installed the nuclei:



Output after the scanning:

[http-missing-security-headers:referrer-policy] [http] [info] http://15.207.116.36:8000/

[http-missing-security-headers:cross-origin-opener-policy] [http] [info] http://15.207.116.36:8000/

[http-missing-security-headers:x-frame-options] [http] [info] http:// 15.207.116.36:8000/

[http-missing-security-headers:content-security-policy] [http] [info] http:// 15.207.116.36:8000/

[http-missing-security-headers:permissions-policy] [http] [info] http:// 15.207.116.36:8000/

[http-missing-security-headers:x-content-type-options] [http] [info] http:// 15.207.116.36:8000/

[http-missing-security-headers:x-permitted-cross-domain-policies] [http] [info] http:// 15.207.116.36:8000/

[http-missing-security-headers:clear-site-data] [http] [info] http:// 15.207.116.36:8000/

[http-missing-security-headers:cross-origin-embedder-policy] [http] [info] http:// 15.207.116.36:8000/

[http-missing-security-headers:cross-origin-resource-policy] [http] [info] http:// 15.207.116.36:8000/

[http-missing-security-headers:strict-transport-security] [http] [info] http:// 15.207.116.36:8000/

[missing-sri] [http] [info] http:// 15.207.116.36:8000/

[spring-detect] [http] [info] http:// 15.207.116.36:8000/error

Vulnerability and Solution:

5.1) vul: nvicti detected that no Referrer-Policy header implemented.Referrer-Policy is a security header designed to prevent cross-domain Referer leakage

sol:Please implement a Referrer-Policy by using the Referrer-Policy response header or by declaring it in the meta tags. It’s also possible to control referrer information over an HTML-element by using the rel attribute.

5.2) vul:Subresource Integrity (SRI) provides a mechanism to check integrity of the resource hosted by third parties like Content Delivery Networks (CDNs) and verifies that the fetched resource has been delivered without unexpected manipulation.

solu: Using Subresource Integrity is simply to add integrity attribute to the script tag along with a base64 encoded cryptographic hash value.

<scriptsrc="https://code.jquery.com/jquery-2.1.4.min.js"integrity="sha384 R4/ztc4ZlRqWjqIuvf6RX5yb/v90qNGx6fS48N0tRxiGkqveZETq72KgDVJCp2TC" crossorigin="anonymous"></script>

5.3) vul: Permissions Policy Header is an added layer of security that helps to restrict from unauthorized access or usage of browser/client features by web resources. This policy ensures the user privacy by limiting or specifying the features of the browsers can be used by the web resources. Permissions Policy provides a set of standard HTTP headers that allow website owners to limit which features of browsers can be used by the page such as camera, microphone, location, full screen etc.

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

5.4) vul:The issue with CVE-2022-22963 is that it permits using HTTP request header spring.cloud.function.routing-expression parameter and SpEL expression to be injected and executed through StandardEvaluationContext.According to the CVSS system, it scores 9.8 as HIGH severity.

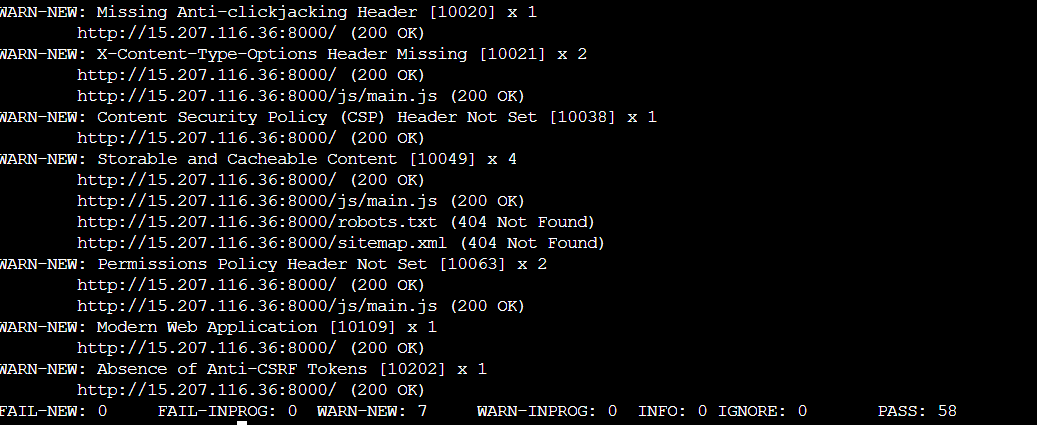
sol: The best defense for this type of vulnerability is to patch it as soon as possible. Having a clear understanding of the packages being used in your environment is a must in today’s world. Using modern tools, such as SCA, can help accomplish this goal and prioritize systems appropriately.

5.5) vul: The 'X-Content-Type-Options Header Missing' vulnerability is a common security issue in web applications. This vulnerability arises when a web server doesn't set the 'X-Content-Type-Options' header in its response, allowing attackers to perform content-type sniffing attacks.

Sol: When serving resources, make sure you send the content-type header to appropriately match the type of the resource being served. For example, if you are serving an HTML page, you should send the HTTP header:

Content-Type: text/html

6)Vulnerability scanning using ZAP:



Vulnerabilities & Solutions:

6.1) Alert Id 10020-1 Medium

vul: The response does not include either Content-Security-Policy with ‘frame-ancestors’ directive or X-Frame-Options to protect against ‘ClickJacking’ attacks.

Sol: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. If you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

6.2) Alert Id 10021

vul: The Anti-MIME-Sniffing header X-Content-Type-Options was not set to ’nosniff’. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.

sol:Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.

6.3)Alert Id 10038-1

vul: Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including 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 configured to set the Content-Security-Policy header.

6.4) Alert Id 10049-3

vul : The response contents are storable by caching components such as proxy servers, and may be retrieved directly from the cache, rather than from the origin server by the caching servers, in response to similar requests from other users. If the response data is sensitive, personal or user-specific, this may result in sensitive information being leaked. In some cases, this may even result in a user gaining complete control of the session of another user, depending on the configuration of the caching components in use in their environment. This is primarily an issue where “shared” caching servers such as “proxy” caches are configured on the local network. This configuration is typically found in corporate or educational environments, for instance.

sol: Validate that the response does not contain sensitive, personal or user-specific information. If it does, consider the use of the following HTTP response headers, to limit, or prevent the content being stored and retrieved from the cache by another user: Cache-Control: no-cache, no-store, must-revalidate, private Pragma: no-cache Expires: 0 This configuration directs both HTTP 1.0 and HTTP 1.1 compliant caching servers to not store the response, and to not retrieve the response (without validation) from the cache, in response to a similar request.

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

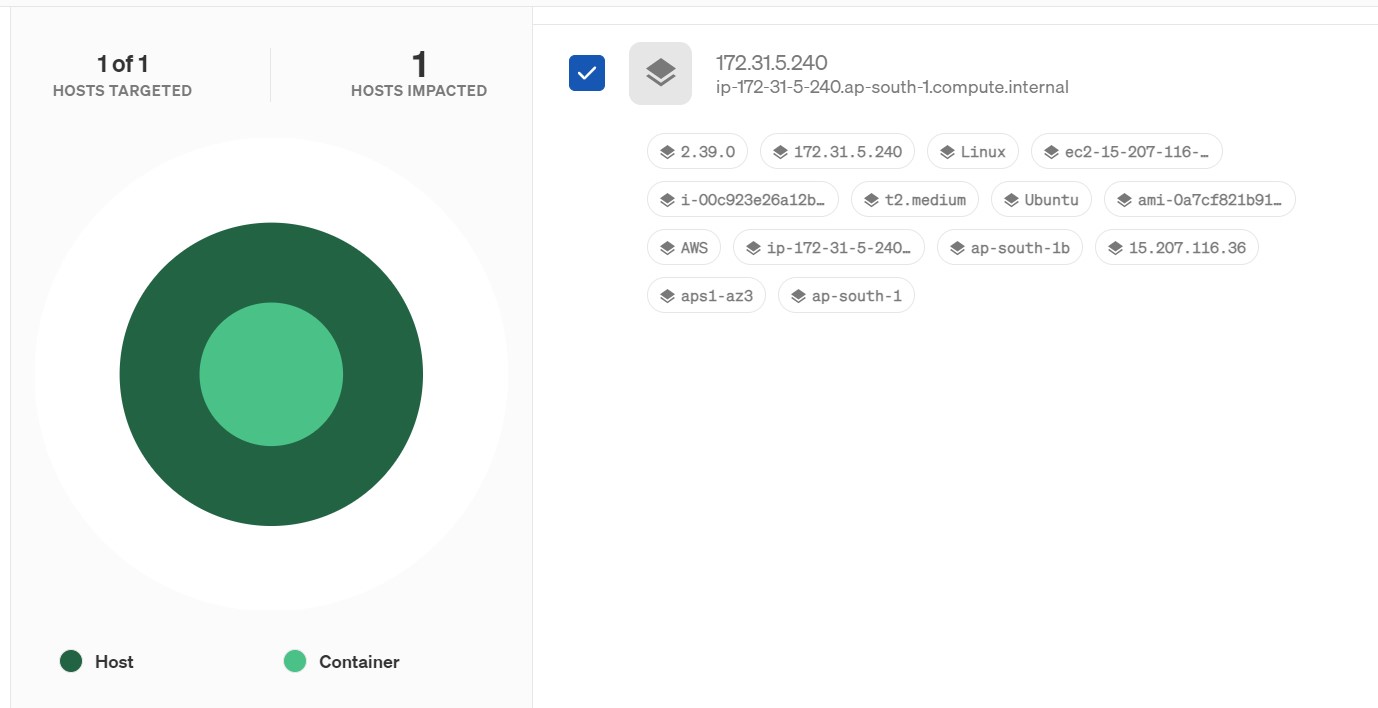
With the help of gremlin UI based web application I have performed a simple experiment to check the behavior of

the system in a disruptive environment.

For this I logged in to my gremlin account.

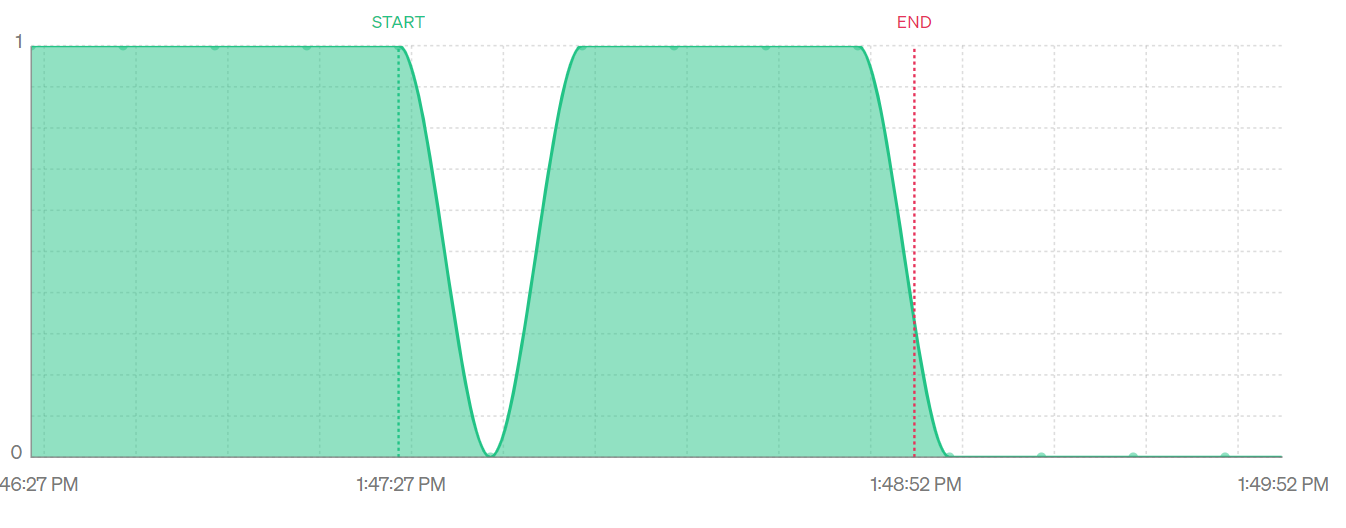
I installed the gremlin agent over the host machine and attached the machine to the gremlin and performed the

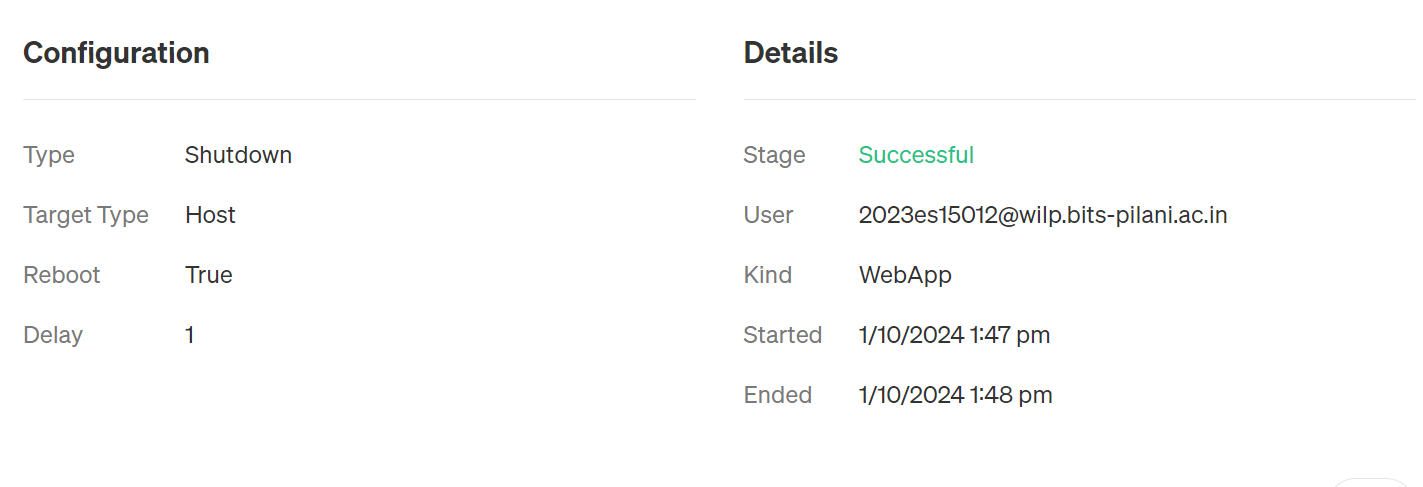
the following experiments.



Performed an experiment on the host machine.

The experiment was to shutdown the machine for 1min and check how the system behaves.





Result:

The website didn’t respond after when the experiment was performed.

The website was not responding on the same IP address. The IP address of the website is changed after the

experiment is perfomed.

The system needs to have an elastic ip .

The system doesn’t scale up automatically.

The system doesn’t have a backup server.