Web Application Vulnerability Assessment & Penetration Testing

Waisl Digital

Staging URL: https://waislstage.grapesmobile.com/

January 21st, 2026

Prepared By: Maneesh Kumar

Waisl Limited

1st floor, Wing-D Building No. 301,

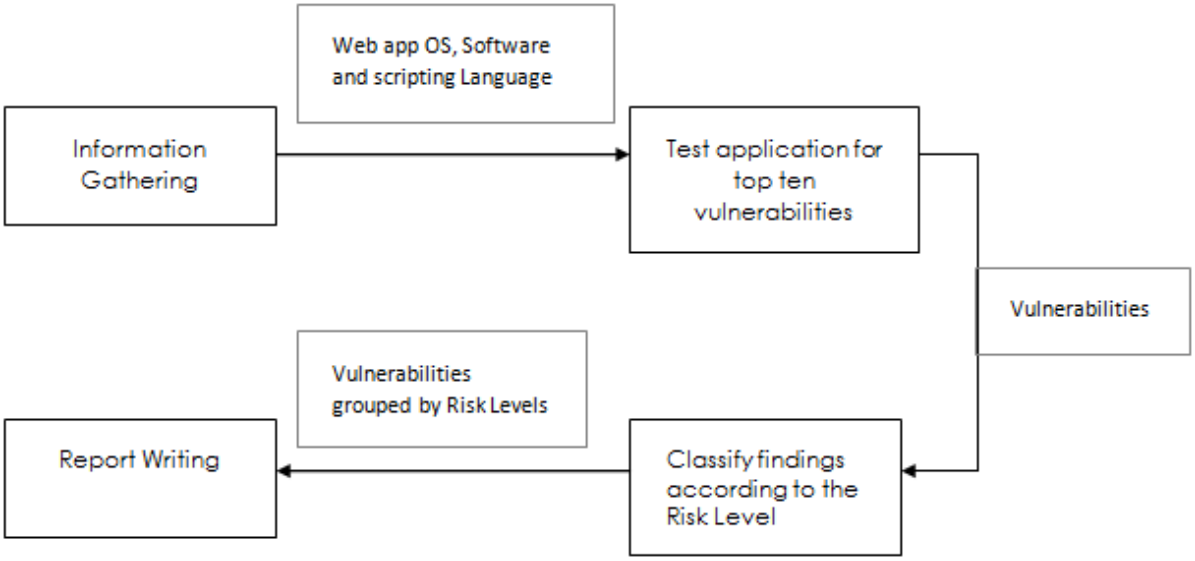
New Udaan Bhawan

Complex, opp. Terminal 3, New Delhi, Delhi 110037

Copyright Notice- Proprietary Information

Methodology

The methodology applied in Web Application Security Testing is explained in the diagram below:



Information Gathering: One of the first steps of this test is to identify the Web application environment, including the scripting language and Web server software in use, and the operating system of the target server. However, this step is generally omitted if the testing is limited to just the web application and not the host.

Test Application: While testing the application, we follow but are not limited to the OWASP standards. The top 10 vulnerabilities are tested for static and dynamic websites. Our testing is done manually as well as using tools. An indicative list of tools is given in the section below.

After an exhaustive testing, the findings are compiled and classified according to a Risk Level of High, Medium or Low depending on the harm they may cause to the Web Application, server or to the network.

A Report is created highlighting the findings together with details for each finding.

Tools Used

The following tools were used:

- Burp Suite
- Nessus
- Waisl Proprietary Checklists and Test Cases

Executive Summary

The significant issues are given in this section, the Executive summary. These list the security flaws that are of major concern. Vulnerabilities have been given a Severity rating of High, Medium or Low based on the risk they may pose. The basis of giving the severity rating is as described below:

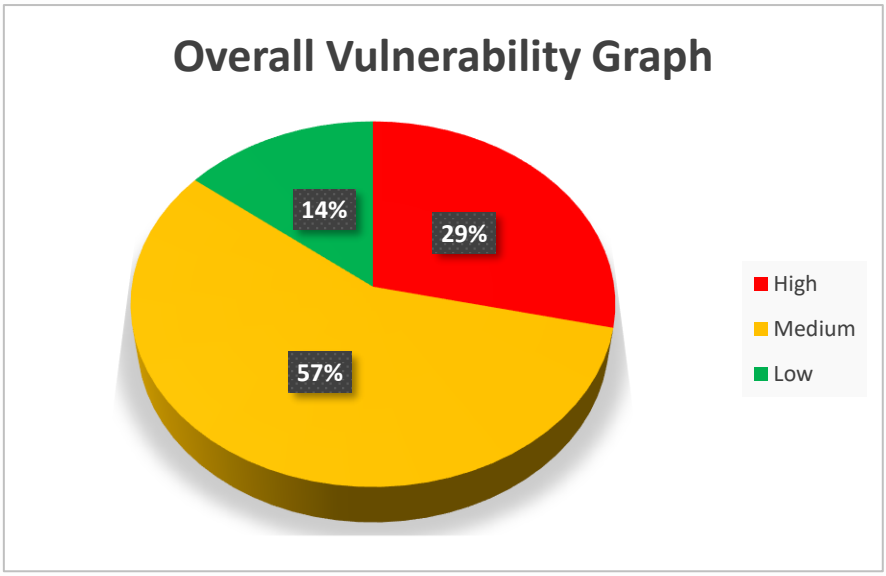
Table of Findings

The following key findings support our assessment of the weaknesses associated with the application.

S. No.	Severity	Vulnerability Description	Status	Remarks
1	**High**	Click jacking attack is possible in the application.	**Open**	-
2	**High**	Vulnerable React server components Detected.	**Open**	-
3	**Medium**	Vulnerable Apache HTTP Server Version Detected.	**Open**	-
4	**Medium**	Vulnerable Next.js Version Detected.	**Open**	-
5	**Medium**	An external service interaction (HTTP) was observed through Collaborator, indicating that the application initiated an outbound HTTP request to an external system.	**Open**	-
6	**Medium**	Security headers such as (X-Content-Type-Options, X-Frame-Options, Strict-Transport-Security, X-XSS, and Content-Security-Policy) are missing in Application.	**Open**	-
7	**Low**	Web Server Allows Password Auto-Completion.	**Open**	-

Overall Vulnerability Dashboard:

1	High	2
2	Medium	4
3	Low	1



Vulnerability Details:

Finding No. 1

Description: Click jacking attack is possible in the application.

Severity Level: High

Impact: Clickjacking can lead to unauthorized actions by users, data alteration, and potential exploitation of sensitive functionality.

Recommendation: Configure your web server to include an X-Frame-Options header

Example:

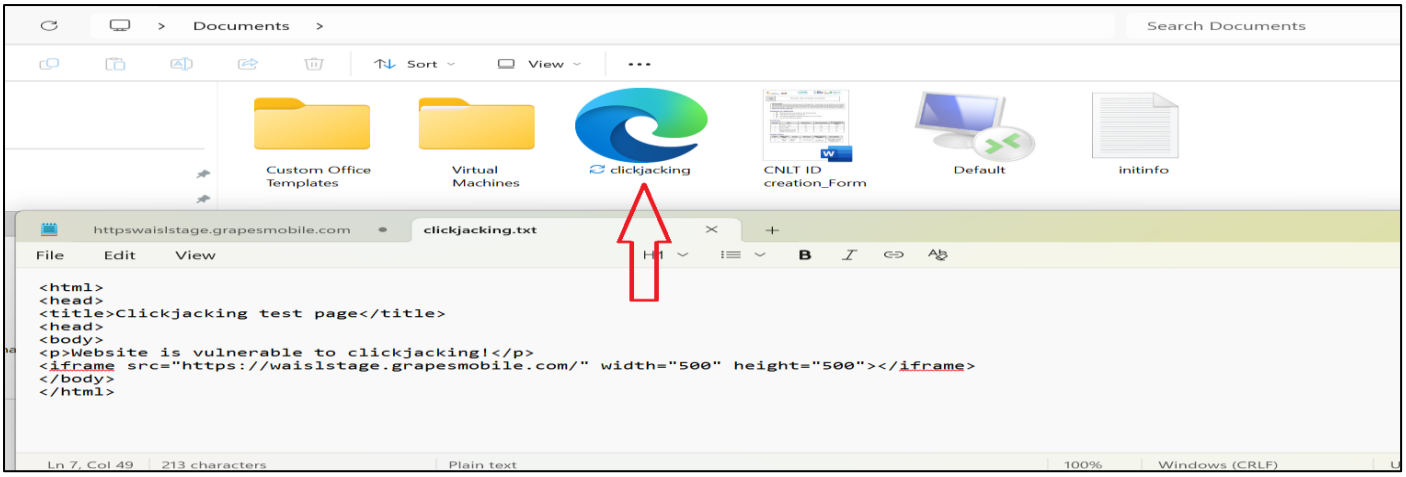
X-Frame-Options: deny

X-Frame-Options: sameorigin

Note: Fix this vulnerability throughout the application.

Screenshots:

Step#1: In the given HTML code, enter the URL of website as a source of iframe tag. Now save it as a HTML file as shown below. After that, run the html file.



Step#2: As you can see, the website is able to open in frame which can also be open in any third-party website also.

Finding No. 2

Description: Vulnerable React server components Detected.

Severity Level: High

Impact: Exploiting the vulnerable React Server Components versions could allow attackers to execute arbitrary code, manipulate server-side logic, cause denial of service, or access sensitive application data.

Recommendation: Upgrade to React Server Components **19.0.3, 19.1.4, or 19.2.3**. Validate and sanitize all HTTP payloads, restrict access to Server Function endpoints, and monitor for suspicious activity.

Screenshots:

```
Nessus was able to detect the vulnerability by sending a specially crafted payload.
The remote host returned a React Server Component error digest when processing the
payload.
Action ID Used: x
Response Snippet:
0:{"a":"$@1","f":"","b":"OAwLQTbvJSuDn_kUY30dI"}
1:E{"digest":"1550024902"}
```

Finding No. 3

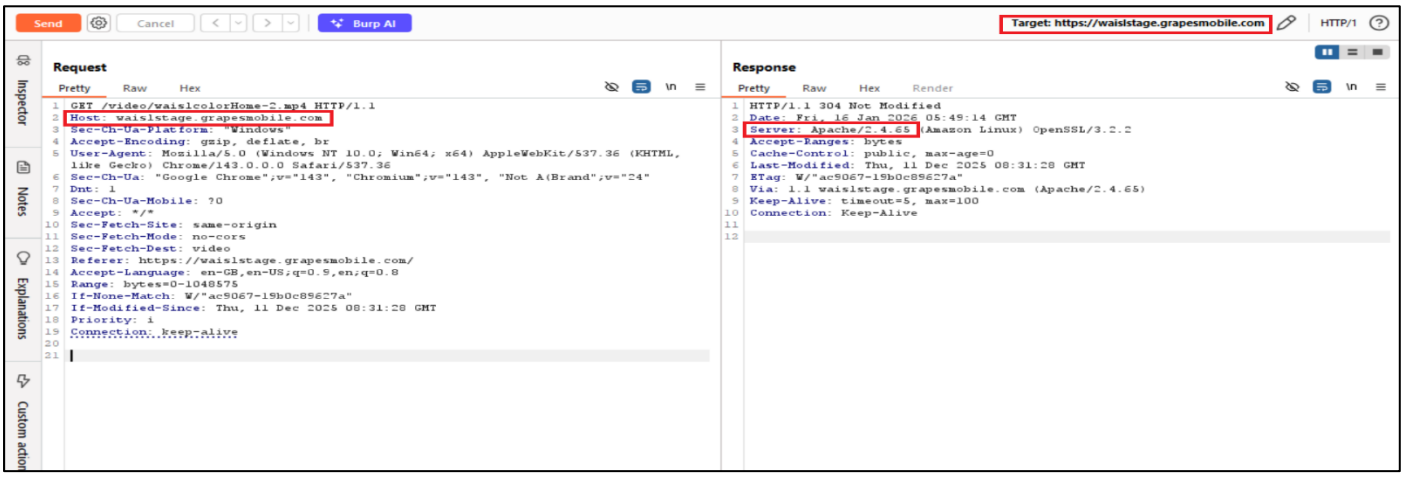
Description: Vulnerable Apache HTTP Server Version Detected.

Severity Level: Medium

Impact: Running a vulnerable Apache HTTP Server can allow attackers to execute code, access sensitive data, disrupt services, or compromise the website.

Recommendation: Update the Apache HTTP Server to the latest secure version and apply all relevant security patches.

Screenshots:



Finding No. 4

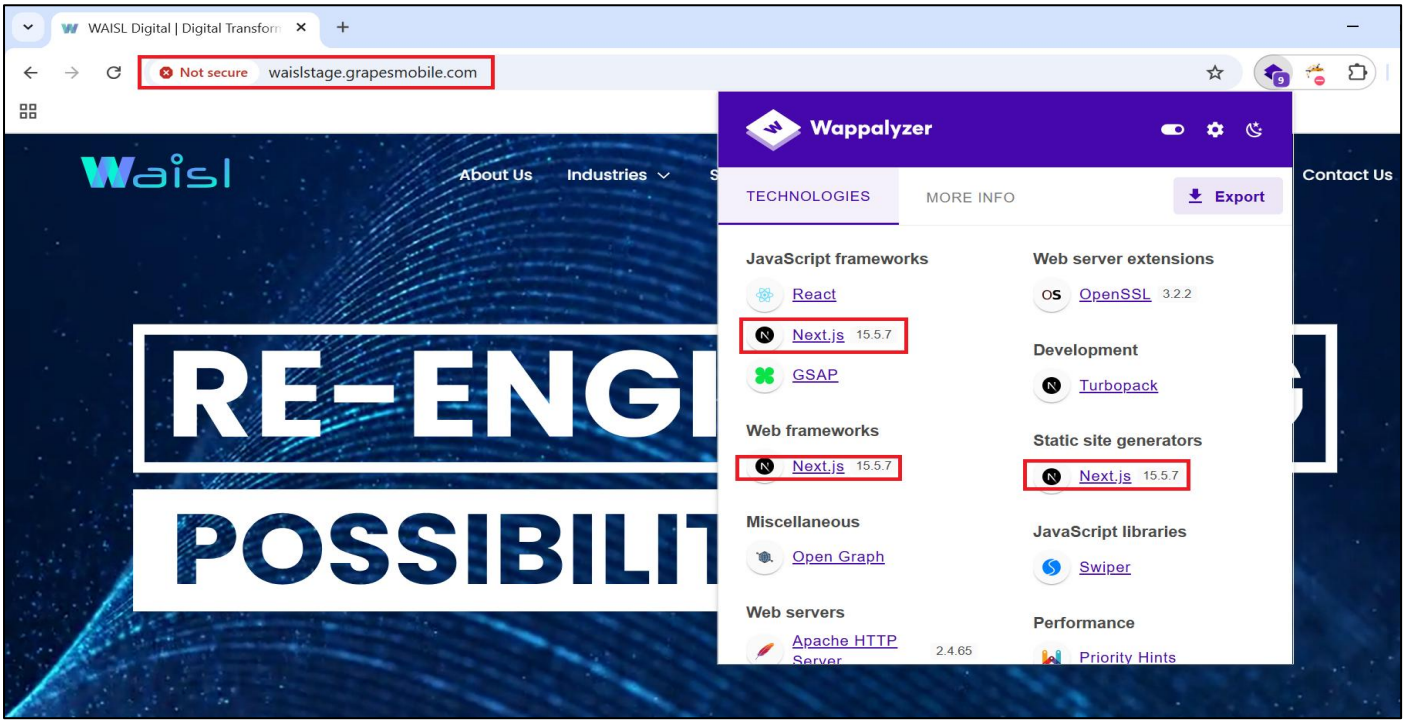
Description: Vulnerable Next.js Version Detected.

Severity Level: Medium

Impact: Attackers could exploit the vulnerable Next.js version to execute code, access sensitive data, or compromise the application.

Recommendation: Upgrade to version 15.5.9 or later to ensure all security patches are applied.

Screenshots:



Finding No. 5

Description: An external service interaction (HTTP) was observed through Collaborator, indicating that the application initiated an outbound HTTP request to an external system.

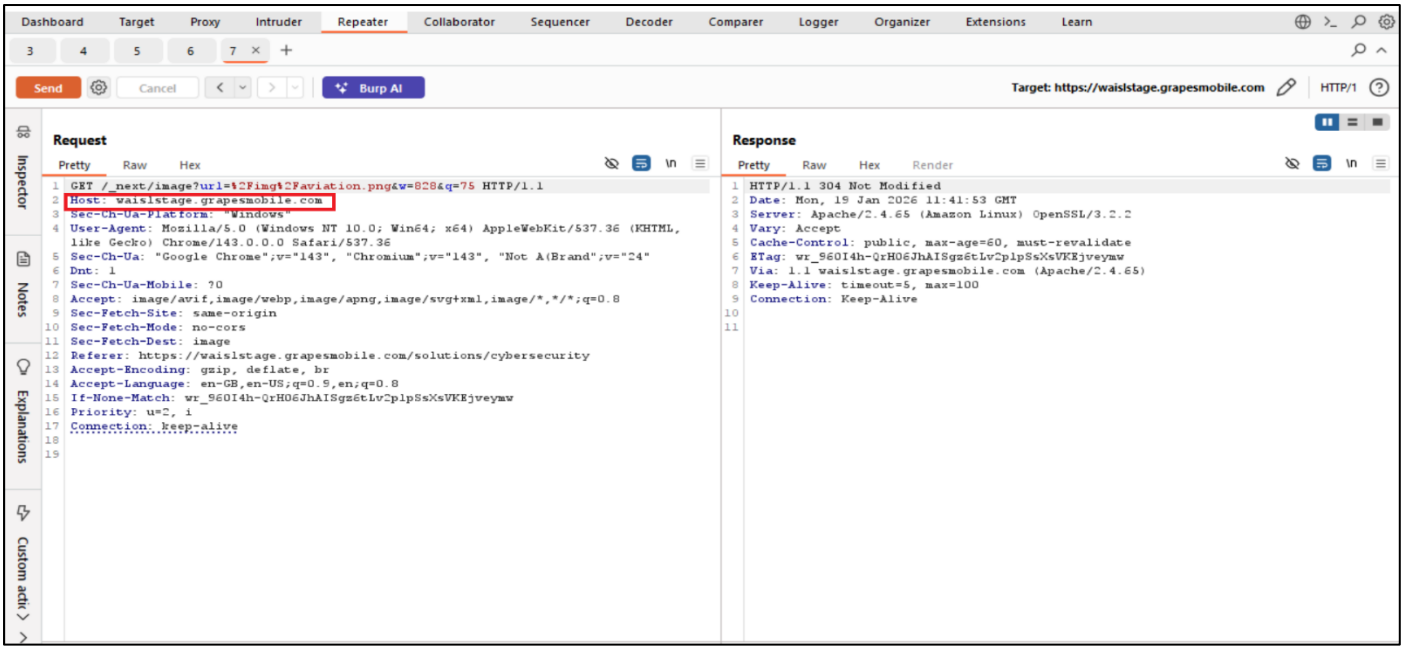
Severity Level: Medium

Impact: This behavior may indicate a security weakness that could allow data leakage, unauthorized external communication, or potential command-and-control activity if exploited.

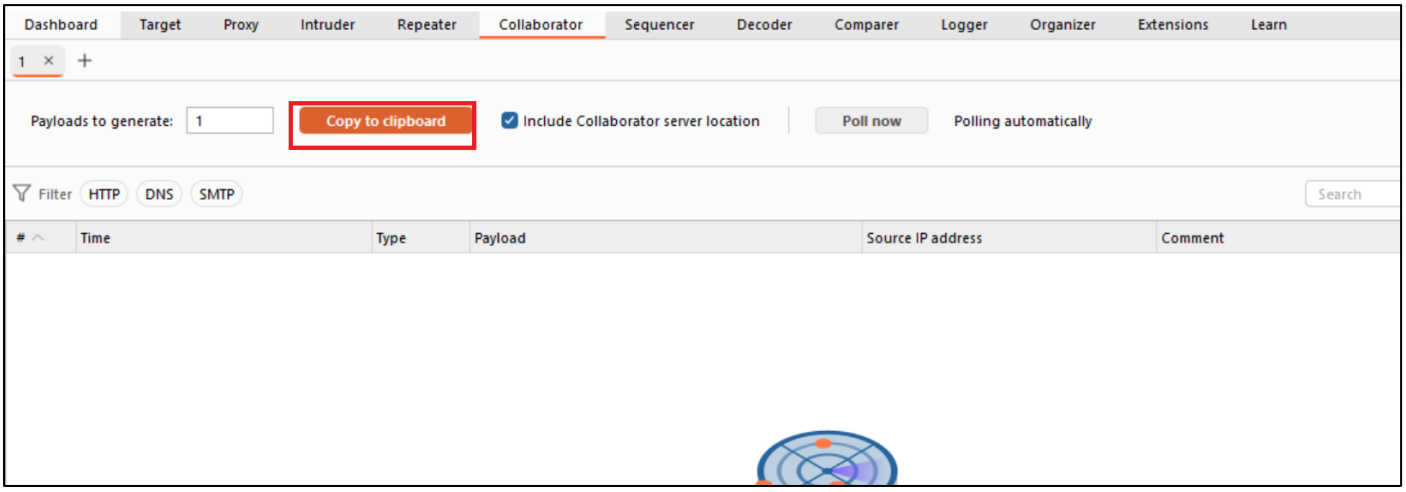
Recommendation: Confirm if external service interaction is expected. If required, secure and restrict outbound connections. If not required, block all external requests except trusted destinations.

Screenshots:

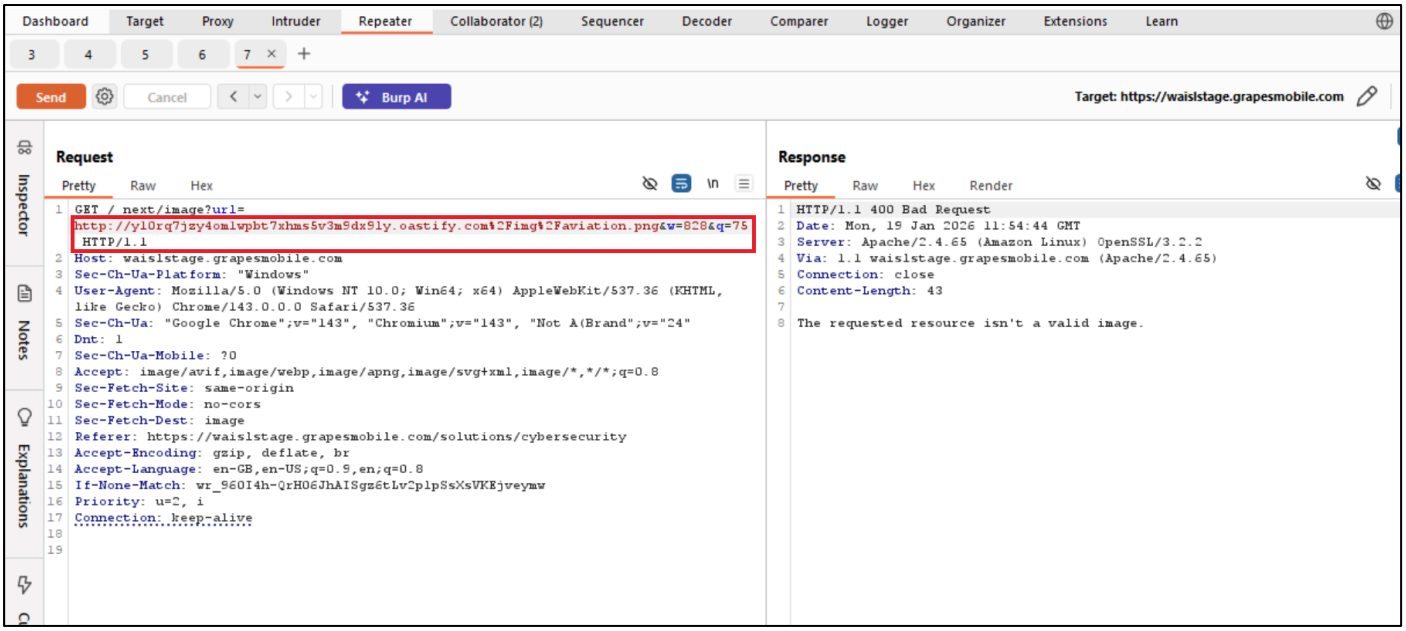
Step 1#: Capture the request and send it to Repeater.



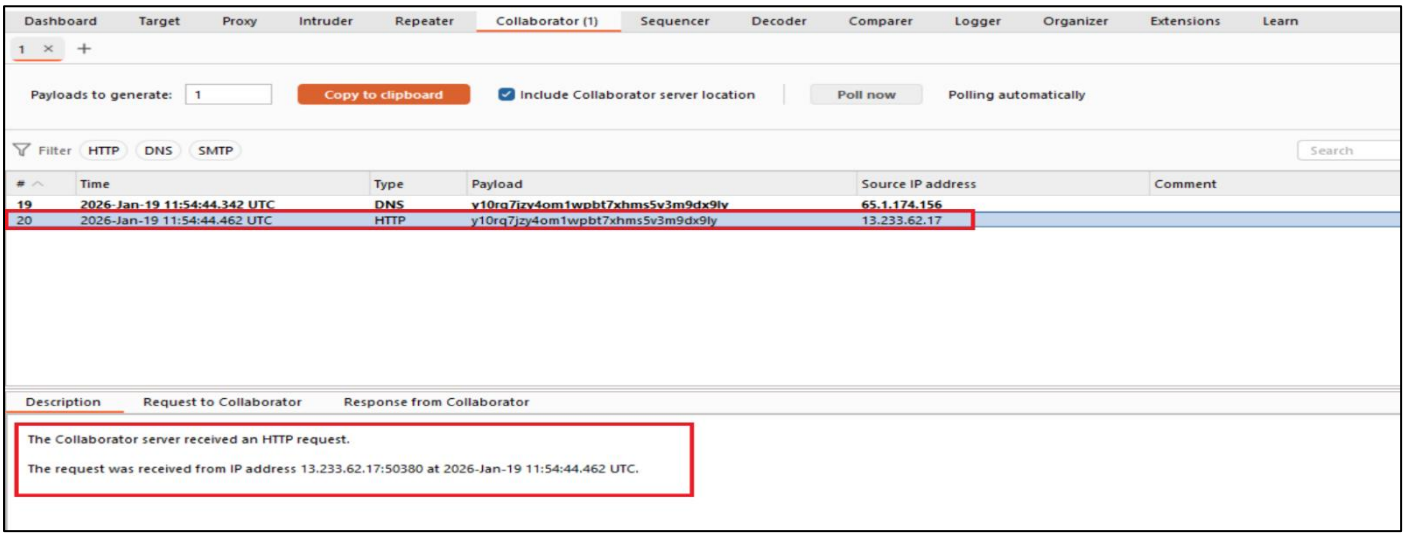
Step 2#: Then go to the Collaborator tab and copy it to the clipboard.



Step 3#: Once copied to the clipboard, paste it into the URL field.



Step 4#: An HTTP-based external service interaction was detected via Collaborator.



Finding No. 6

Description: Security headers such as (X-Content-Type-Options, X-Frame-Options, Strict-Transport-Security, X-XSS, and Content-Security-Policy) are missing in Application.

Severity Level: Medium

Impact: Missing security headers leave the application exposed to multiple client-side attacks, including XSS, clickjacking, and MITM exploits.

Recommendation: Following HTTP response security headers must be implemented:

Header	Rationale
Content-Security-Policy	The majority of CSP functionality only affects pages rendered as HTML.
HTTP Strict-Transport-Security	To require connections over HTTPS and to protect against spoofed certificates.
X-Content-Type-Options	To prevent browsers from performing MIME sniffing, and inappropriately interpreting responses as HTML.
X-Frame-Options	To protect against drag-and-drop style clickjacking attacks.
X-XSS-Protection	X-XSS header protects against Cross-Site Scripting attacks.

Screenshots:

Finding No. 7

Description: Web Server Allows Password Auto-Completion.

Severity Level: Low

Impact: Enabling password auto-completion may expose user credentials on shared or compromised systems, increasing the risk of unauthorized account access and weakening authentication security.

Recommendation: Add the attribute 'autocomplete=off' to these fields to prevent browsers from caching credentials.

Screenshots:

```
Page : /login
Destination Page: /login

Page : /register
Destination Page: /register
```

To see debug logs, please visit individual host

Port	Hosts
3000 / tcp / www	13.233.62.17

```
Page : /admin_cmsdhye66363/login
Destination Page: /admin_cmsdhye66363/login
```

To see debug logs, please visit individual host

Port	Hosts
4001 / tcp / www	13.233.62.17