|  |  |
| --- | --- |
| Vulnerability Number -01 | Target: **testphp.vulnweb.com** |

|  |  |  |
| --- | --- | --- |
| Vulnerability Number | Vulnerability Name | Severity |
| V01 | Information Leakage through Search Engine Indexing | Low |

General Description:

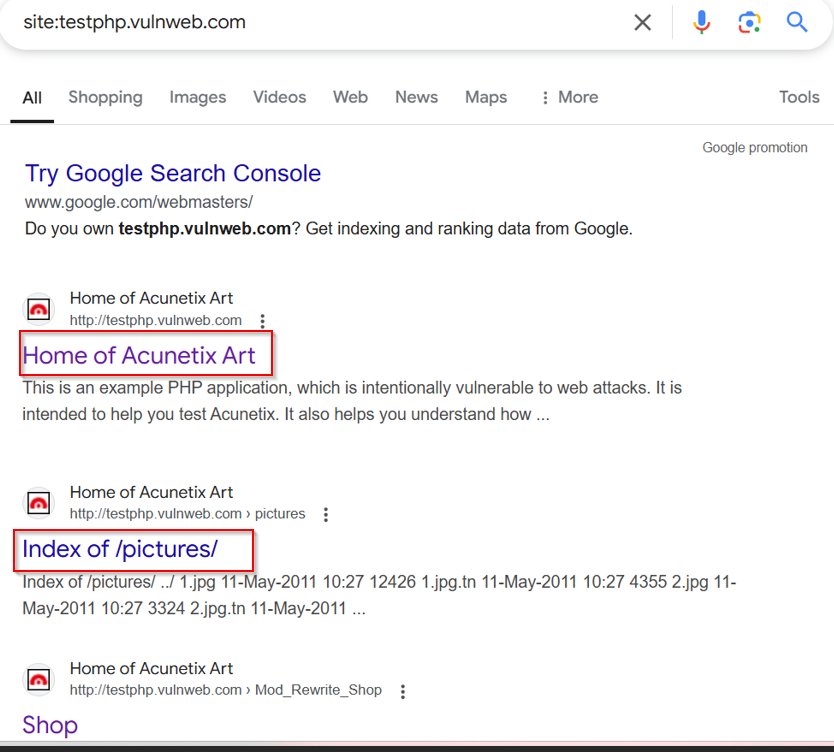
During penetration testing, we observe that the **Search Engine Discovery and Reconnaissance for Information Leakage** is a method used by security professionals, penetration testers, and threat hunters to identify unintentional exposure of sensitive or private information via search engines.

Vulnerability Resources:

The vulnerability has been identified on the following IP:

Affected Target:

Test Performed Evidence:



Remediation:

1. Exposed Sensitive Files (Documents, Contracts, Financials, etc.)
2. Exposed Backup Files (Database Dumps, Old Code, Configurations)
3. Exposed Credentials or API Keys
4. Exposed Administrative Interfaces (Admin Pages, Login Forms)

Reference:

[WSTG - Stable | OWASP Foundation](https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/01-Information_Gathering/01-Conduct_Search_Engine_Discovery_Reconnaissance_for_Information_Leakage)

|  |  |
| --- | --- |
| Vulnerability Number -02 | Target:google.com |

|  |  |  |
| --- | --- | --- |
| Vulnerability Number | Vulnerability Name | Severity |
| V02 | Exposed Sensitive Resources via robots.txt | Low |

General Description:

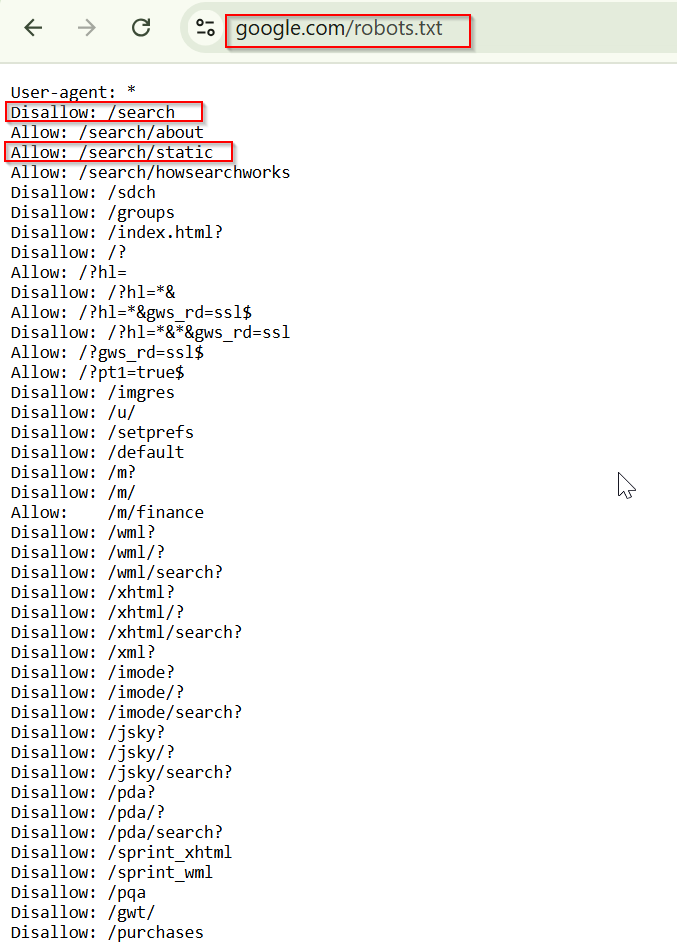
During penetration testing, we discovered that the /robots.txt file guides web crawlers on which parts of the site to index. However, it may inadvertently expose sensitive or resticted areas to attackers.

Vulnerability Resources:

The vulnerability has been identified on the following IP:

Affected Target: google.com

Test Performed Evidence:



Remediation:

The robots.txt file is not itself a security threat, and its correct use can represent good practice for non-security reasons. You should not assume that all web robots will honor the file's instructions. Rather, assume that attackers will pay close attention to any locations identified in the file. Do not replay on robots.txt to provide any kind of protection over unauthorized access.

Refenrence:

<https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/01-Information_Gathering/03-Review_Webserver_Metafiles_for_Information_Leakage>

|  |  |
| --- | --- |
| Vulnerability Number -03 | Target: |

|  |  |  |
| --- | --- | --- |
| Vulnerability Number | Vulnerability Name | Severity |
| V03 | META Tag | Low |

General Description:

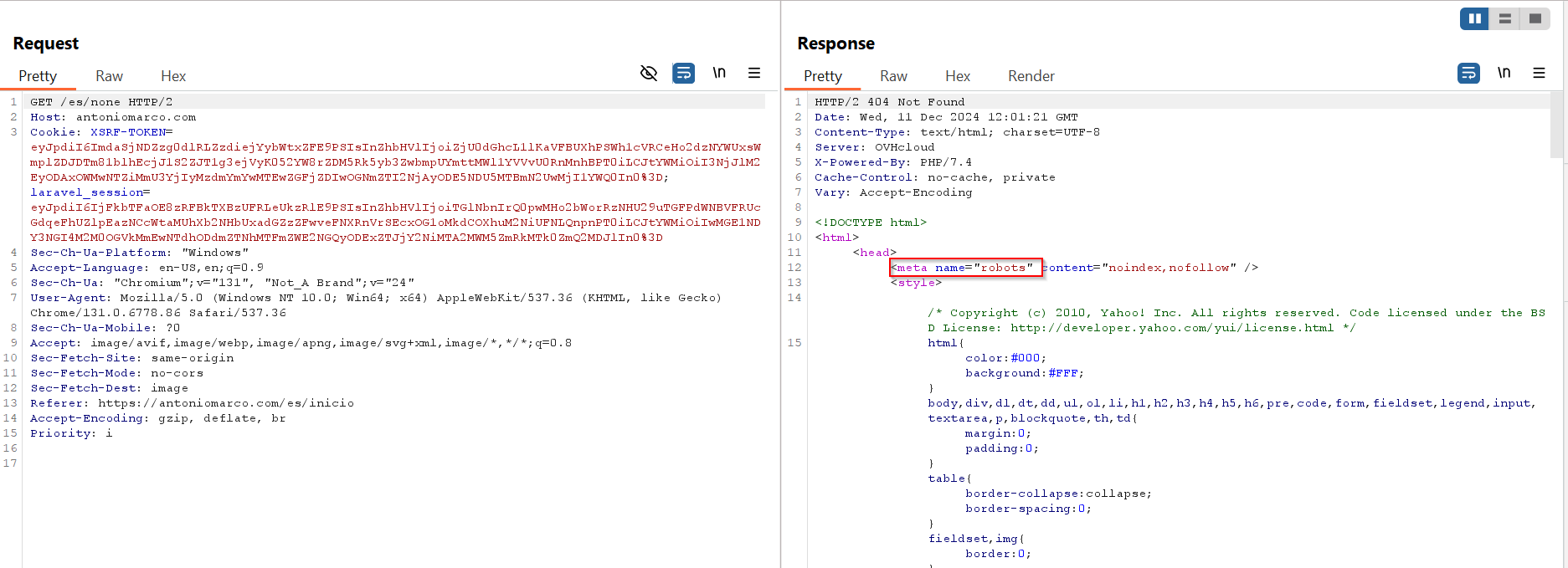
Attackers can inject malicious or unintended values into HTTP headers through improperly validated meta tags.This could allow attackers to set cookies or modify headers if a browser or application interprets this meta tag improperly.

Vulnerability Resources:

The vulnerability has been identified on the following IP:

Affected Target:

Test Performed Evidence:



Remediation:

1. Validate and Sanitize Input

2. Avoid Dynamic Meta Tags

3. Disable Meta Refresh

4. Enforce Strong CSP

5. Set Security Headers Properly

Refenrence:

https://owasp.org/www-project-web-security-testing-guide/latest/4-Web\_Application\_Security\_Testing/01-Information\_Gathering/03-Review\_Webserver\_Metafiles\_for\_Information\_Leakage

|  |  |
| --- | --- |
| Vulnerability Number -04 | Target: 208.91.198.72 |

|  |  |  |  |
| --- | --- | --- | --- |
| Vulnerability Number | Vulnerability Name | Severity | |
| V04 | Enumerate Applications on Web server | | Low |

General Description:

Application enumeration on a web server refers to the process of identifying web applications, services, and resources hosted on a target server. This is a critical step in penetration testing and reconnaissance, as it provides insights into the attack surface and potential vulnerabilities.

Vulnerability Resources:

The vulnerability has been identified on the following IP:

Affected Target:

Test Performed Evidence:



Remediation:

1. Remove or obfuscate server banners and response headers.

2. Disable directory listing.

3. Protect sensitive directories with authentication.

4. Keep applications and servers updated to avoid exploits on outdated versions.

Refenrence:

https://owasp.org/www-project-web-security-testing-guide/latest/4-Web\_Application\_Security\_Testing/01-Information\_Gathering/04-Enumerate\_Applications\_on\_Webserver

|  |  |
| --- | --- |
| Vulnerability Number -05 | Target: Any browser |

|  |  |  |  |
| --- | --- | --- | --- |
| Vulnerability Number | Vulnerability Name | Severity | |
| V05 | Web Page & Comments & meta data infomation | | Low |

General Description:

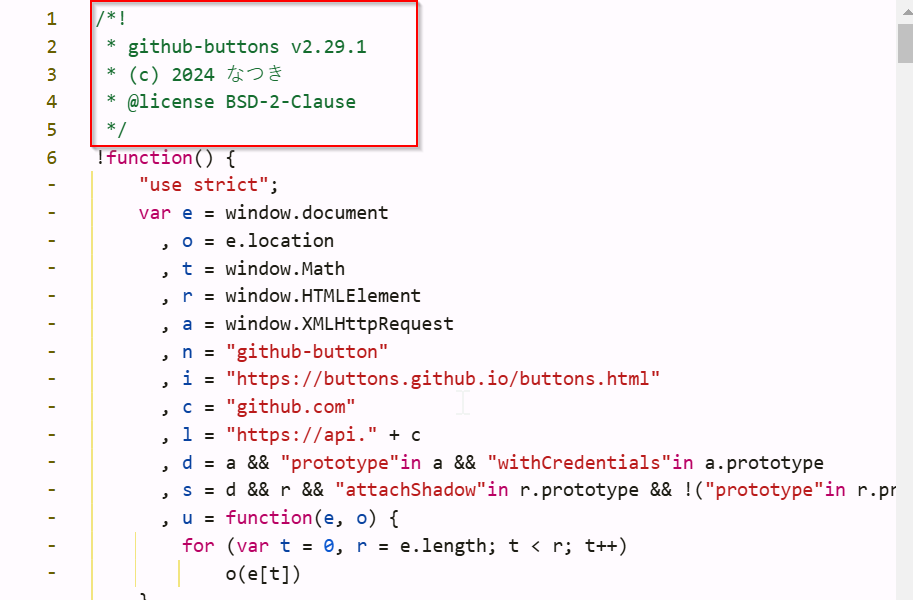
It is very common, and even recommended, for programmers to include detailed comments and metadata on their source code. However, comments and metadata included into the HTML code might reveal internal information that should not be available to potential attackers. Comments and metadata review should be done in order to determine if any information is being leaked.

Vulnerability Resources:

The vulnerability has been identified on the following IP:

Affected Target: Any browser

Test Performed Evidence:



Remediation:

1. Ensure encryption in transit (HTTPS/TLS).

2. Avoid exposing sensitive business logic in the frontend.

3. Do not include sensitive or internal details in comments.

4. Use a code review process to identify and remove insecure comments.

5. Review metadata values to ensure no sensitive URLs or data are exposed.

Refenrence:

<https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/01-Information_Gathering/03-Review_Webserver_Metafiles_for_Information_Leakage>

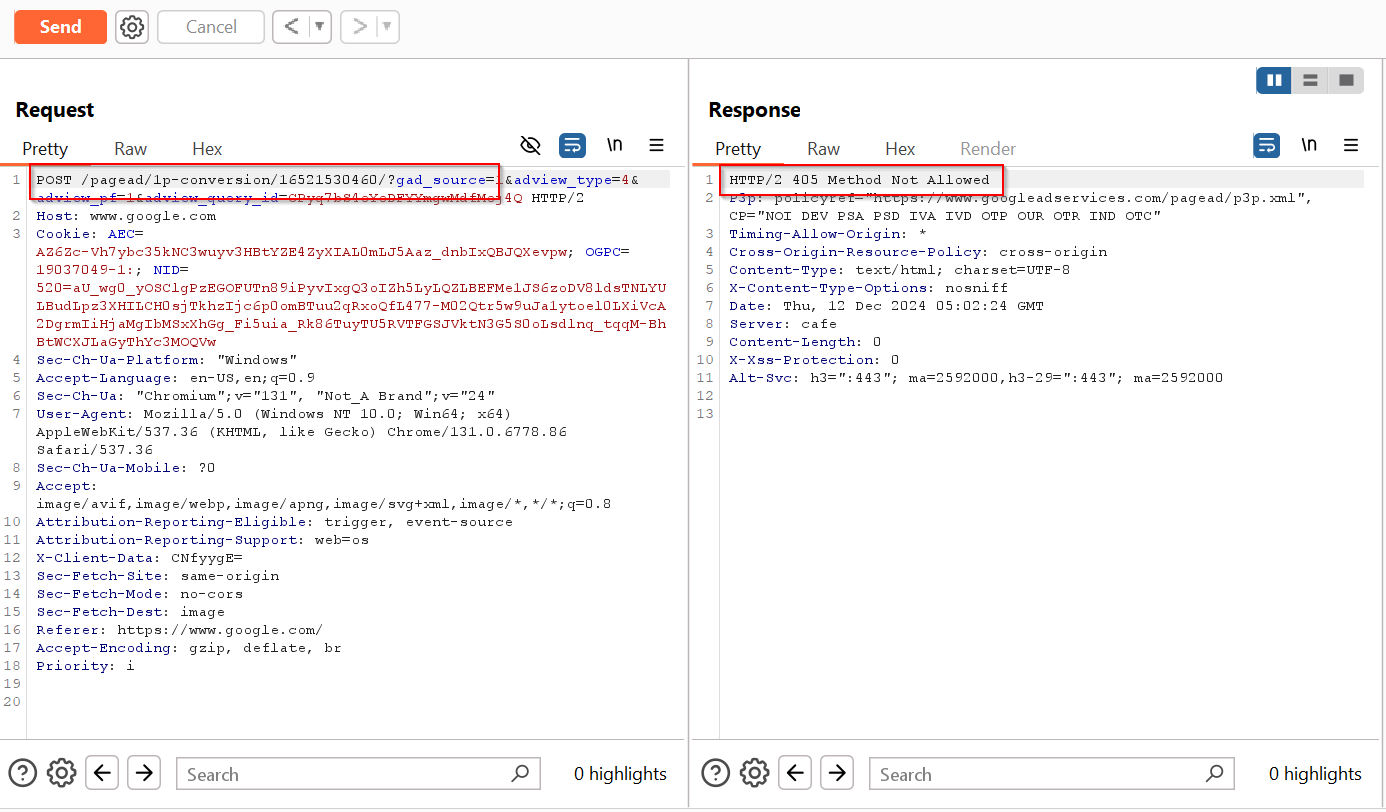
|  |  |
| --- | --- |
| Vulnerability Number -06 | Target: |

|  |  |  |  |
| --- | --- | --- | --- |
| Vulnerability Number | Vulnerability Name | Severity | |
| V06 | Host Header injection Detected | | Medium |

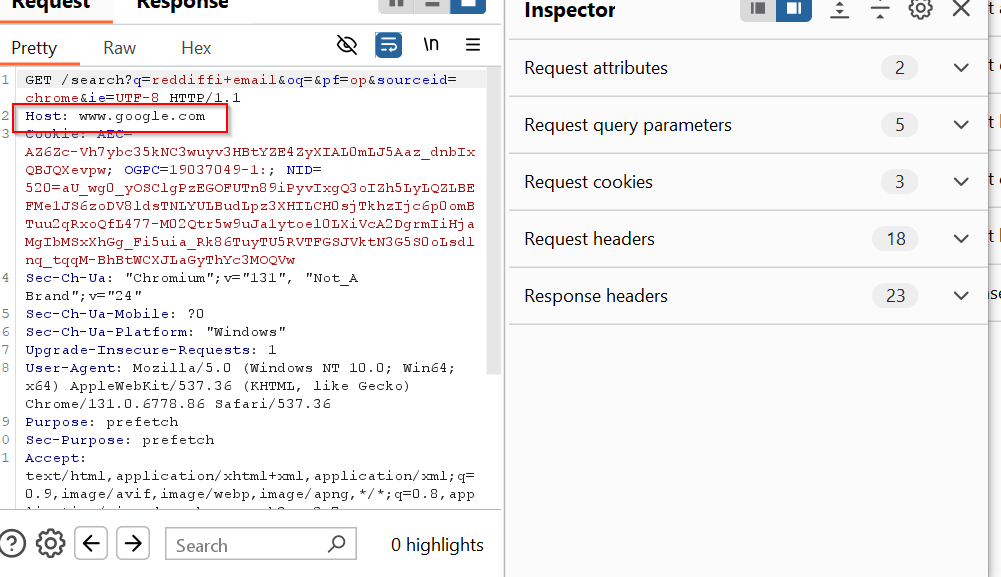
General Description:

During penetration testing, we observed that manipulating the host header in the request, the application responded with a 200 OK status, indicating that it processed the request without validating the host header properly. This behaviour may expose the application to various attacks, such as server-side request forgery (SSRF), cache poisoning.

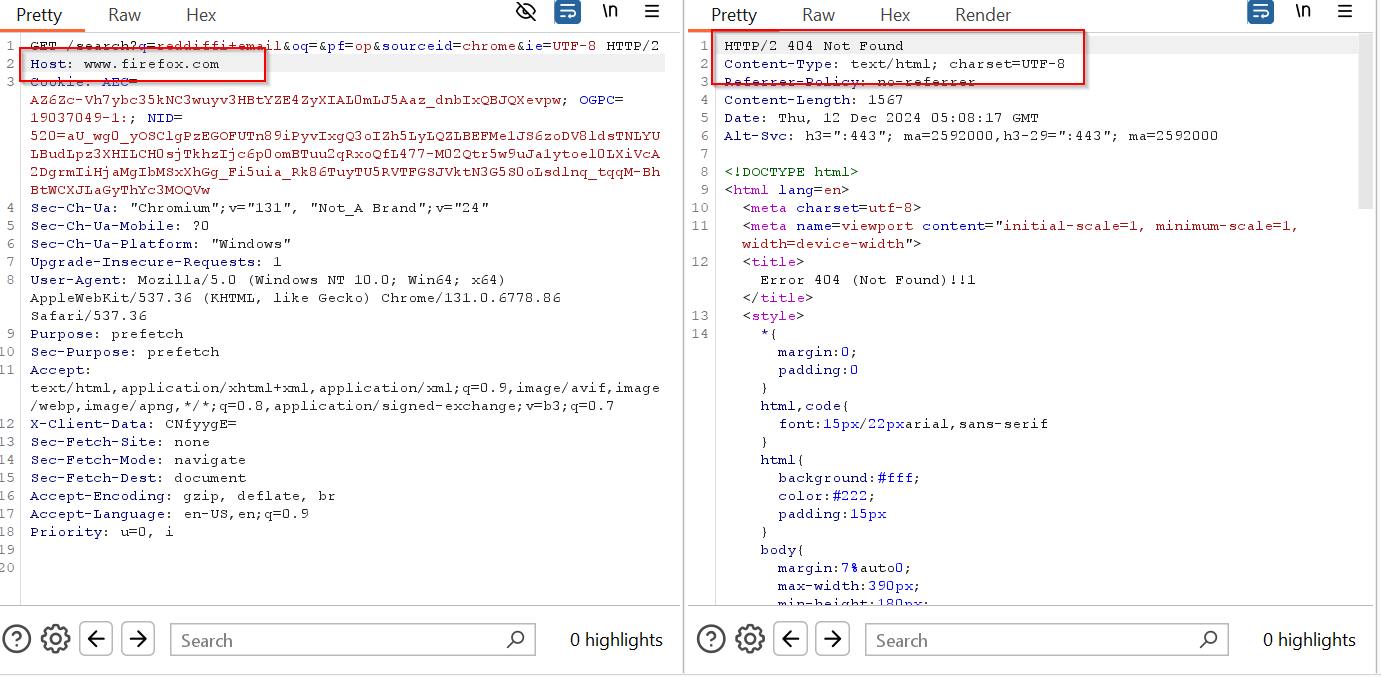
Test Performed Evidence:







Now changed Host name from **Google.com to Firefox.com**



Remediation:

1. Implement Host Header Validation: Implement a strict validation process for the Host header. The value of the host header should be checked against a predefined list of allowed domain names or IP addresses. Only requests with valid host headers should be processed.

2. Sanitize User Inputs: Ensure that all user inputs, including the host header, are properly sanitized and validated. Avoid passing user-controlled values directly into critical areas of the application like URLs, API calls, or database queries.

3. Use Secure HTTP Headers: Ensure the application uses security headers like Strict-Transport-Security (HSTS), Content-Security-Policy (CSP), and X-Frame-Options to mitigate potential risks and attacks resulting from compromised headers.

Refenrence:

<https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/17-Testing_for_Host_Header_Injection>

|  |  |
| --- | --- |
| Vulnerability Number -07 | Target: |

|  |  |  |
| --- | --- | --- |
| Vulnerability Number | Vulnerability Name | Severity |
| V07 | Map Execution paths through an Application | Medium |

General Description:

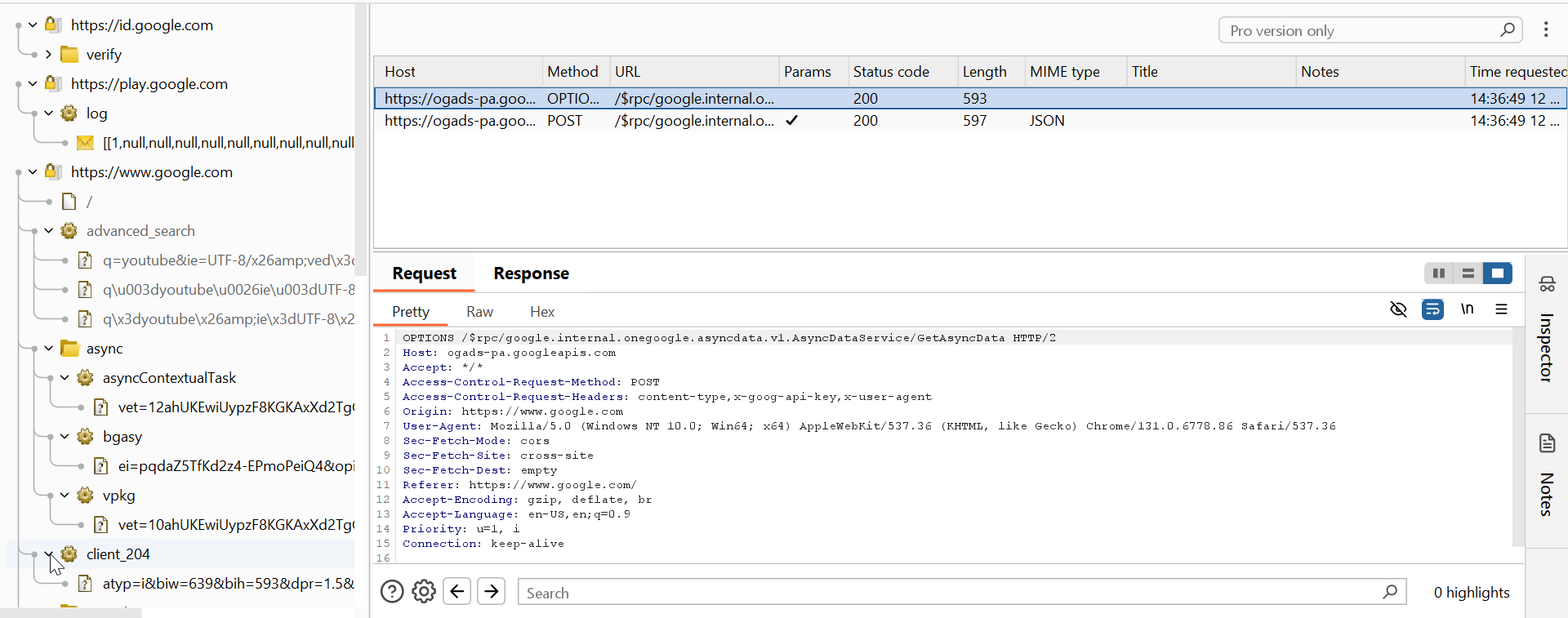
Mapping execution paths through an application involves tracing how data and user inputs flow through the application from entry points (like a user submitting a form or making a request) to their destination (like a database, file system, or API response). This process is critical for understanding the application's behavior and identifying vulnerabilities in its logic or implementation.

Vulnerability Resources:

The vulnerability has been identified on the following IP:

Affected Target:

Test Performed Evidence:



Remediation:

1. Review all user inputs (e.g., forms, APIs, file uploads) and validate them with proper input sanitization and encoding.

2. Implement strong logging and tracing mechanisms to monitor data movement through controllers, services, and databases.

3. Sanitize and encode all outputs to prevent injection (e.g., XSS) or unintended data exposure.

Refenrence:

<https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/01-Information_Gathering/07-Map_Execution_Paths_Through_Application>

|  |  |
| --- | --- |
| Vulnerability Number -08 | Target: |

|  |  |  |  |
| --- | --- | --- | --- |
| Vulnerability Number | Vulnerability Name | Severity | |
| V08 | Finger Print Web Application & Web Application Frame work | | Medium |

General Description:

Fingerprinting is the process of gathering information about a web application and its framework to understand its structure, technologies, and potential vulnerabilities. This is a crucial step in penetration testing and reconnaissance during cybersecurity assessments.

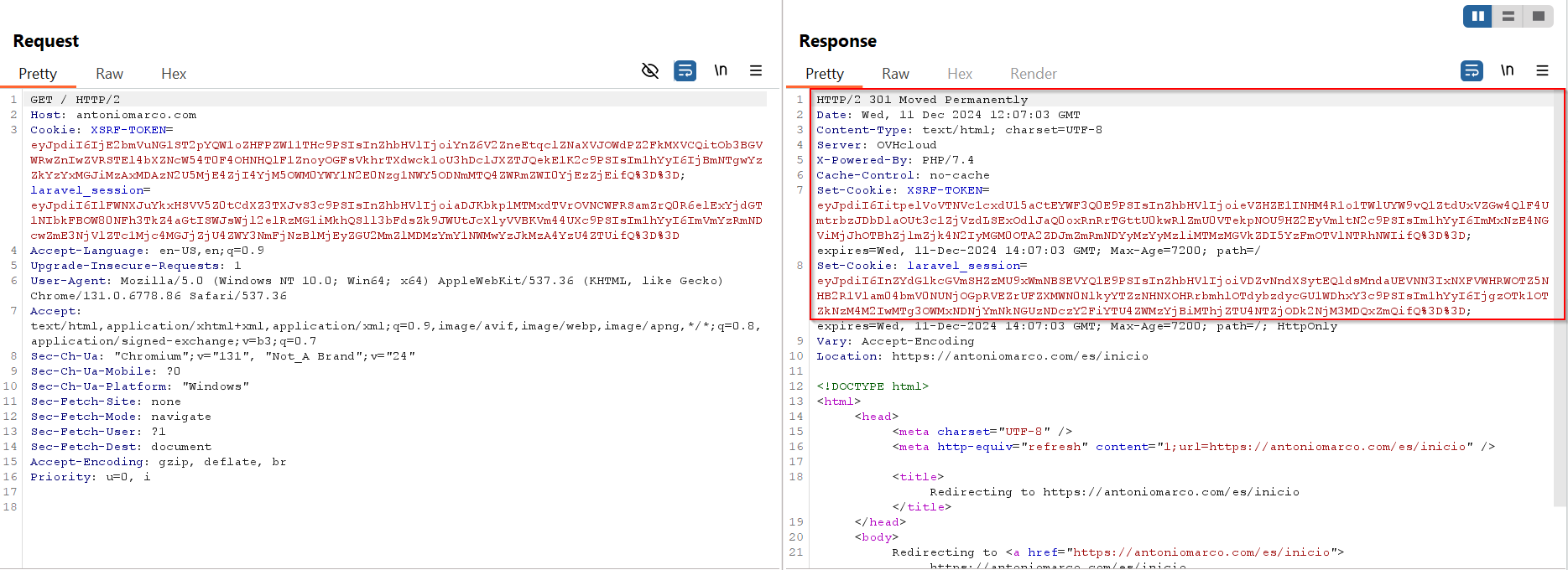
Vulnerability Resources:

The vulnerability has been identified on the following IP:

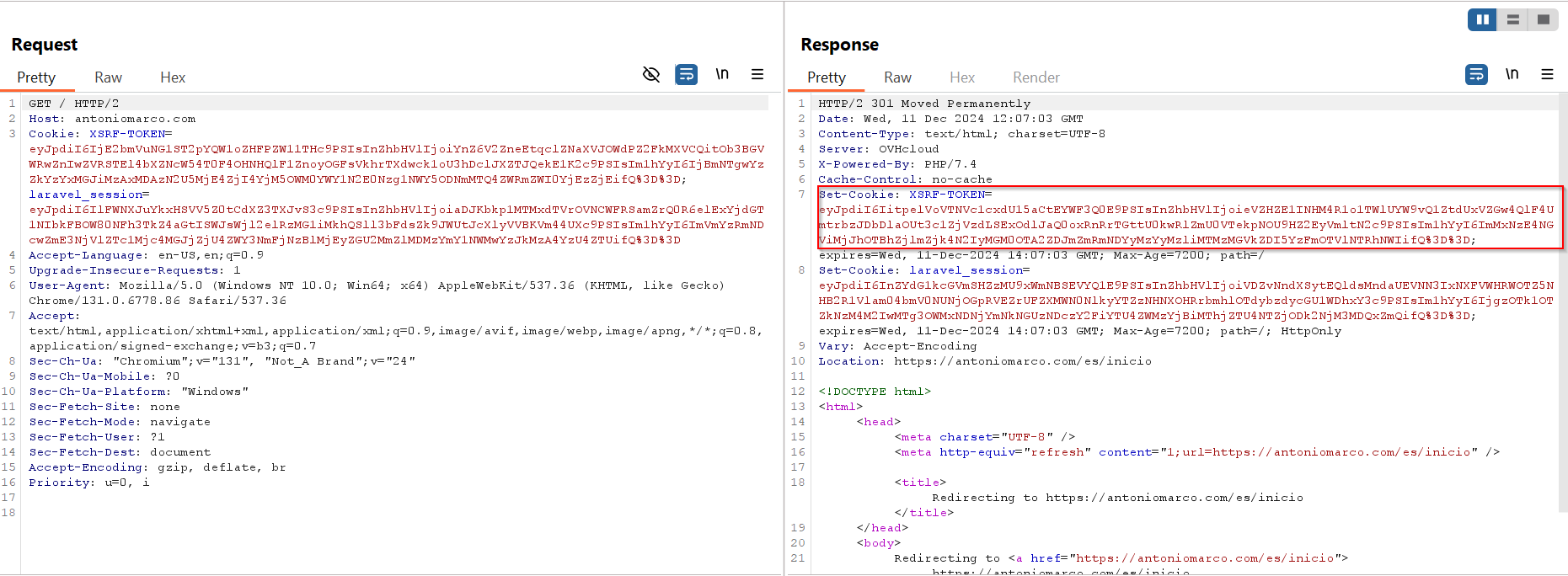
Affected Target:

Test Performed Evidence:

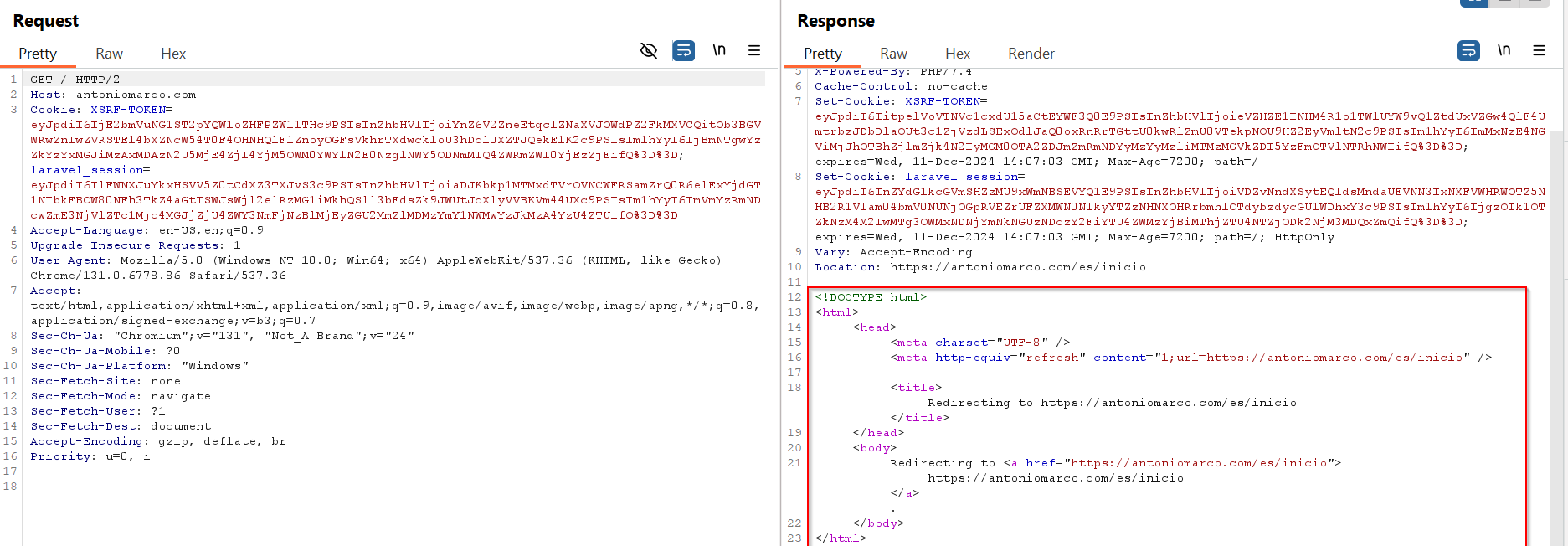
HTTP headers



Cookies:



HTML source code



Specific files and folders

Remediation:

1. Remove or mask headers like X-Powered-By, Server, or X-AspNet-Version.

2. Turn off debug mode and prevent error messages from exposing stack traces or sensitive details.

3. Remove or restrict access to default pages (e.g., /admin, /install, /config).

4. Secure cookies with flags like HttpOnly, Secure, and SameSite.

5. Use Web Application Firewalls to block suspicious requests.

Refenrence:

https://owasp.org/www-project-web-security-testing-guide/latest/4-Web\_Application\_Security\_Testing/01-Information\_Gathering/08-Fingerprint\_Web\_Application\_Framework

|  |  |
| --- | --- |
| Vulnerability Number -09 | Target: |

|  |  |  |  |
| --- | --- | --- | --- |
| Vulnerability Number | Vulnerability Name | Severity | |
| V09 | HTTP METHODS {get,put,post,delete,options,head,trace} | | High |

General Description:

During penetration testing, we observed that the application is utilizing various HTTP methods**.**

Vulnerability Resources:

The vulnerability has been identified on the following IP:

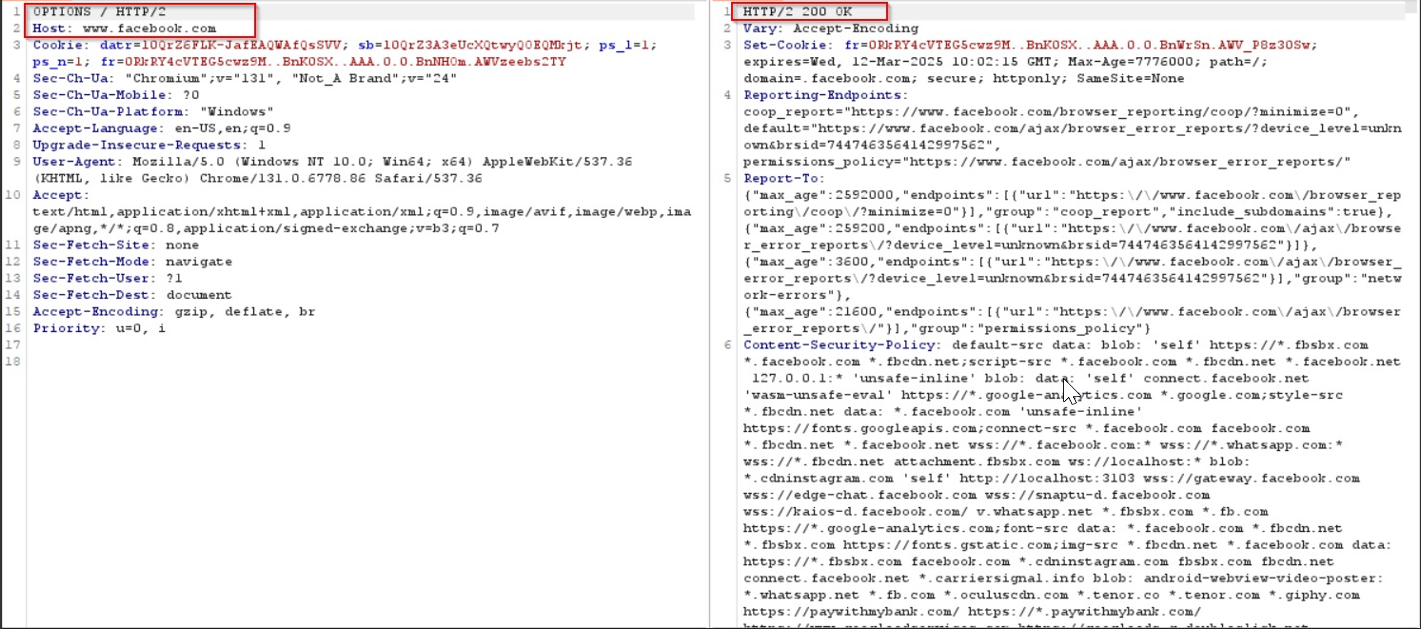
Affected Target:

Test Performed Evidence:









Remediation:

1. Restrict HTTP Methods: Limit allowed HTTP methods to only those required for the application, and disable unnecessary methods (e.g., PUT, DELETE, TRACE, or OPTIONS).

2. Use Security Headers: Implement headers such as Access-Control-Allow-Methods to control which methods are allowed for cross-origin requests.

Refenrence:

<https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/06-Test_HTTP_Methods>

|  |  |
| --- | --- |
| Vulnerability Number -10 | Target: |

|  |  |  |  |
| --- | --- | --- | --- |
| Vulnerability Number | Vulnerability Name | Severity | |
| V10 | Test HTTP Strict Transport Security | | Medium |

General Description:

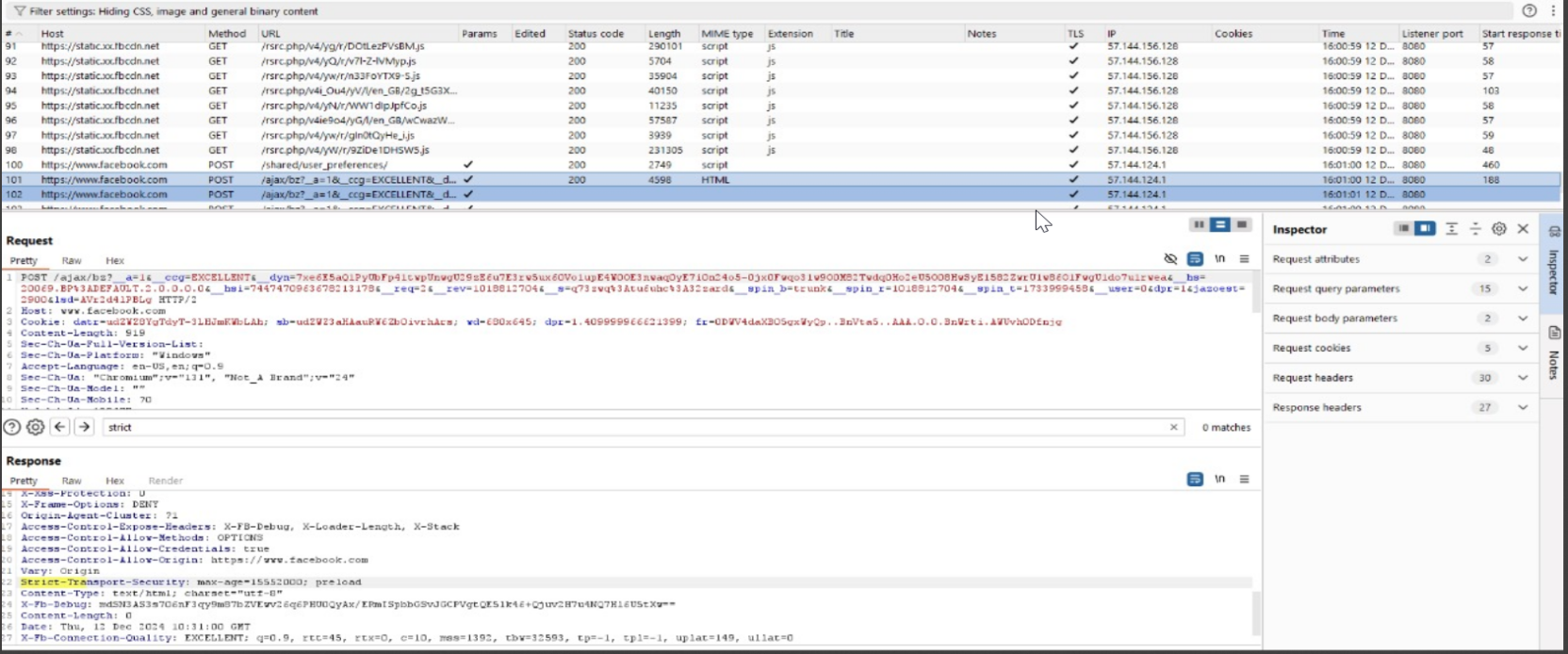
The HTTP Strict Transport Security (HSTS) feature enables a web server to inform the user’s browser, via a special response header, that it should never establish an unencrypted HTTP connection to the specified domain servers. Instead, it should automatically establish all connection requests to access the site through HTTPS. This also prevents users from overriding certificate errors

Vulnerability Resources:

The vulnerability has been identified on the following IP:

Affected Target:

Test Performed Evidence:



Remediation:

1. Add Strict-Transport-Security with appropriate attributes in the server configuration.

2. Ensure all resources (pages, APIs, assets) are served over HTTPS.

3. Implement 301 redirects for all HTTP requests to HTTPS.

4. Use a long duration (e.g., 1 year = 31536000 seconds) to maintain HSTS policy persistence.

Refenrence:

<https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/07-Test_HTTP_Strict_Transport_Security>