# Vulnerability Assessment and Penetration Testing (VAPT) Report

Organization: IIIT Kottayam

Faculty: Dr. A. Balu

Project: LMS (Library Management System)

Authored By: **K. Revanth**

# CONTENTS

# INTRODUCTION

This comprehensive report encapsulates the findings derived from the meticulous Vulnerability Assessment and Penetration Testing (VAPT) conducted on the Library Management System (LMS) web application. The LMS was developed by students of batch 2022 (group-2) as a crucial component of their academic project at the Indian Institute of Information Technology, Kottayam (IIIT Kottayam).

The LMS serves as a vital tool for managing the vast array of resources within the academic institution. From cataloging books to facilitating online reservations and managing user accounts, the LMS plays a pivotal role in streamlining library operations and enhancing the overall learning experience for students and faculty alike.

The primary objective of this assessment was to meticulously scrutinize, evaluate, and ultimately mitigate any security vulnerabilities inherent within the LMS application. The overarching goal is to fortify the application's security posture and safeguard the confidentiality, integrity, and availability of critical data and functionalities.

In today's interconnected digital landscape, the prevalence of cyber threats and malicious actors poses a significant risk to the security and integrity of web-based applications. Therefore, conducting a rigorous VAPT is paramount to identify and address any potential vulnerabilities that could be exploited by adversaries to compromise the system's security.

# SCOPE OF TESTING

The VAPT was conducted with a comprehensive scope, encompassing all facets of the LMS application, including but not limited to:

## Authentication and Authorization Mechanisms:

Authentication: This involves verifying the identity of users accessing the LMS application, typically through credentials such as usernames and passwords. The assessment examines the effectiveness of authentication mechanisms in preventing unauthorized access, including password policies, multi-factor authentication (MFA), and account lockout mechanisms.

Authorization: Authorization determines the actions and resources that authenticated users are permitted to access within the application. The assessment evaluates the implementation of role-based access control (RBAC), access control lists (ACLs), and other authorization mechanisms to ensure that users are granted appropriate permissions based on their roles and privileges.

## Input Validation and Data Sanitization Practices:

Input validation involves validating and sanitizing user-supplied data to prevent malicious input from compromising the security of the application. The assessment scrutinizes the effectiveness of input validation mechanisms in filtering and sanitizing input data to mitigate common vulnerabilities such as SQL injection, cross-site scripting (XSS), and command injection.

## Session Management and Cookie Handling:

Session management governs the creation, maintenance, and termination of user sessions within the application. The assessment evaluates the implementation of secure session management practices, including session tokens, session expiration, and session fixation prevention, to thwart session-related attacks such as session hijacking and session fixation.

Cookie handling involves managing cookies sent and received by the application to maintain user sessions and store user preferences. The assessment examines cookie attributes such as Secure, HttpOnly, and SameSite to ensure that cookies are transmitted securely and are not susceptible to tampering or exploitation.

## Database Security and Query Parameterization:

Database security entails protecting the integrity and confidentiality of data stored within the application's database. The assessment scrutinizes the database configuration, access controls, and encryption mechanisms to prevent unauthorized access, data leakage, and SQL injection attacks.

Query parameterization involves using parameterized queries or prepared statements to execute database queries securely, mitigating the risk of SQL injection vulnerabilities. The assessment evaluates the implementation of query parameterization techniques to ensure that user input is properly sanitized and validated before being incorporated into database queries.

## Secure Communication Protocols (HTTPS):

Secure communication protocols such as HTTPS (Hypertext Transfer Protocol Secure) encrypt data transmitted between the client and server, safeguarding it from eavesdropping and tampering by malicious actors. The assessment verifies the implementation of HTTPS to ensure that sensitive data, such as login credentials and personal information, is transmitted securely over the network.

## Error Handling and Exception Management:

Error handling involves gracefully handling and presenting errors and exceptions encountered during application execution. The assessment examines the adequacy of error handling mechanisms in providing informative yet secure error messages to users, mitigating the risk of information disclosure and aiding in the detection and mitigation of security incidents.

## Access Controls and Privilege Escalation:

Access controls restrict access to sensitive functionalities and resources based on user roles and permissions. The assessment evaluates the effectiveness of access control mechanisms in enforcing least privilege principles, preventing privilege escalation, and mitigating the risk of unauthorized access to critical functionalities and data.

## Third-Party Integrations and Dependencies:

Third-party integrations and dependencies encompass external libraries, frameworks, and services utilized by the LMS application. The assessment scrutinizes the security posture of third-party components, including their vulnerability status, security patches, and adherence to secure coding practices, to mitigate the risk of supply chain attacks and dependency vulnerabilities.

By comprehensively evaluating each facet of the LMS application's security posture, the VAPT aims to identify, prioritize, and remediate security vulnerabilities to safeguard the confidentiality, integrity, and availability of the system and its data.

# METHODOLOGY

## Systematic and Methodical Approach:

The VAPT was conducted following a systematic and methodical approach to ensure thorough coverage and accurate assessment of the LMS application's security posture. This approach involved defining clear objectives, establishing testing criteria, and adhering to established methodologies throughout the assessment process.

## Combination of Tools and Techniques:

The assessment utilized a combination of automated scanning tools, manual penetration testing techniques, and ethical hacking methodologies to identify and evaluate security vulnerabilities within the LMS application. This multi-faceted approach allowed for comprehensive coverage and in-depth analysis of potential security risks.

## Automated Scanning Tools:

Automated scanning tools were employed to conduct initial reconnaissance and identify common security vulnerabilities within the LMS application. These tools, such as OWASP ZAP (Zed Attack Proxy), facilitate the automated discovery of vulnerabilities such as SQL injection, cross-site scripting (XSS), insecure configuration, and more. Automated scans help expedite the identification process and provide a baseline assessment of the application's security posture.

## Manual Penetration Testing Techniques:

Manual penetration testing techniques were employed to supplement automated scans and validate the findings, as well as to identify complex or nuanced security vulnerabilities that may evade automated detection. Manual testing involves skilled security professionals actively probing the application for weaknesses, exploiting vulnerabilities, and assessing the potential impact of security flaws. Techniques such as parameter manipulation, session manipulation, and authentication bypass were

applied to simulate real-world attack scenarios and uncover vulnerabilities that may not be detected by automated tools alone.

## Ethical Hacking Methodologies:

Ethical hacking methodologies were employed to conduct controlled and authorized attacks on the LMS application, mimicking the tactics, techniques, and procedures (TTPs) used by malicious actors. Ethical hackers, also known as white-hat hackers, employ a structured and disciplined approach to identify and exploit security vulnerabilities within the application, all while adhering to ethical standards and guidelines. By adopting an adversarial mindset, ethical hackers can uncover hidden security risks and provide actionable recommendations for remediation.

## Primary Tool: OWASP ZAP (Zed Attack Proxy):

OWASP ZAP was selected as the primary tool for conducting the VAPT due to its robust feature set, extensive scanning capabilities, and open-source nature. As a widely-used security testing tool, OWASP ZAP offers a comprehensive suite of functionalities for identifying and assessing security vulnerabilities within web applications. These functionalities include passive and active scanning, spidering, fuzzing, scripting, and more. OWASP ZAP enables security professionals to conduct detailed analysis, generate actionable reports, and facilitate remediation efforts to enhance the security posture of web applications.

By leveraging a systematic approach and employing a combination of automated tools, manual techniques, and ethical hacking methodologies, the VAPT aimed to provide a comprehensive assessment of the LMS application's security vulnerabilities. This approach ensured thorough coverage, accurate identification, and effective remediation of security risks, ultimately enhancing the overall security posture of the LMS application.

# VULNERABILITY ASSESSMENT FINDINGS & REMEDIATION RECOMMENDATIONS

This section presents the identified security weaknesses within the application and provides actionable steps to address and mitigate those vulnerabilities effectively. By categorizing vulnerabilities based on their severity and potential impact, organizations can prioritize remediation efforts and allocate resources effectively to address the most critical security risks first. Vulnerabilities can be broadly classified as following:

## CRITICAL VULNERABILITIES:

Critical vulnerabilities are the most severe and pose a significant risk to the security of the application. They can potentially lead to complete compromise of the system, unauthorized access to sensitive data, or denial of service.

The critical vulnerability found in this VAPT is as follows:

### ***Cloud Metadata Potentially Exposed***

**URL:** http://localhost:5173/latest/meta-data/

**Risk:** Critical

**Confidence:** Low

**Parameter:**

**Attack:** 169.254.169.254

**Evidence:**

**CWE ID:** 0

**WASC ID:** 0

**Source:** Active (90034 - Cloud Metadata Potentially Exposed)

**Input Vector:**

Description:

The Cloud Metadata Attack attempts to abuse a misconfigured NGINX server in order to access the instance metadata maintained by cloud service providers such as AWS, GCP and Azure.

All of these providers provide metadata via an internal unrouteable IP address '169.254.169.254' - this can be exposed by incorrectly configured NGINX servers and accessed by using this IP address in the Host header field,

Other Info:

Based on the successful response status code cloud metadata may have been returned in the response. Check the response data to see if any cloud metadata has been returned.

The meta data returned can include information that would allow an attacker to completely compromise the system.

**Solution:**

Do not trust any user data in NGINX configs, in this case it is probably the use of the $host variable which is set from the 'Host' header and can be controlled by an attacker.

**Reference:**

https ://www.nginx.com/blog/tru st-no-one-perils-of-tnu sting-:u ser-inpu t/

**Alert Tags:**

| Key | Value |
|---|---|
| OWASP_2021_A05 | https://owasp.org/Top10/A05_2021-Security_Misconfiguration/ |
| OWASP_2017_A06 | https://owasp.org/www-project-top-ten/2017/A6_2017-Security_Misconfiguration.html |

# HIGH-RISK VULNERABILITIES:

High-risk vulnerabilities are significant and can lead to serious security breaches if exploited. While they may not pose an immediate threat as critical vulnerabilities, they still require prompt attention and remediation.

The high-risk vulnerabilities found in this VAPT are as follows:

## ***Content Security Policy (CSP) Header Not Set***

**URL:** http://localhost:5173/

**Risk:** High

**Confidence:** High

**Parameter:**

**Attack:**

**Evidence:**

**CWE ID:** 693

**WASC ID:** 15

**Source:** Passive (10038 - Content Security Policy (CSP) Header Not Set)

**Alert Reference:** 10038-1

**Input Vector:**

Description:

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.

**Solution:**

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

**Reference:**

https://developer.mozilla.org/en-US/docs/Web/Security/CSP/Introducing_Content Security_Policy

https://cheatsheetseries.owasp.org/cheatsheets/Content_Security Policy Cheat Sheet,html

http://www.w3.org/TR/CSP/

**Alert Tags:**

| Key | Value |
|---|---|
| OWASP_2021_A05 | https://owasp.org/Top10/A05_2021-Security_Misconfiguration/ |
| OWASP_2017_A06 | https://owasp.org/www-project-top-ten/2017/A6_2017-Security_Misconfiguration.html |

## ***Cross-Domain Misconfiguration***

**URL:** http://localhost:5173/

**Risk:** High

**Confidence:** Medium

**Parameter:**

**Attack:**

**Evidence:** Access-Control-Allow-Origin: *

**CWE ID:** 264

**WASC ID:** 14

**Source:** Passive (10098 - Cross-Domain Misconfiguration)

**Input Vector:**

Description:

Web browser data loading may be possible, due to a Cross Origin Resource Sharing (CORS) misconfiguration on the web server

Other Info:

The CORS misconfiguration on the web server permits cross-domain read requests from arbitrary third party domains, using unauthenticated APIs on this domain, Web browser implementations do not permit arbitrary third parties to read the response from authenticated APIs, however, This reduces the risk somewhat, This misconfiguration could be used by an attacker to access data that is available in an unauthenticated manner, but which uses some other form of security, such as IP address white-listing,

**Solution:**

Ensure that sensitive data is not available in an unauthenticated manner (using IP address white-listing, for instance).

Configure the "Access-Control-Allow-Origin" HTTP header to a more restrictive set of domains, or remove all CORS headers entirely, to allow the web browser to enforce the Same Origin

Policy (SOP) in a more restrictive manner,

**Reference:**

https://wlncat.fortify.com/en/detail?id=desc.config.dotnet.html5_overly_permissive_cors_policy

**Alert Tags:**

| Key | Value |
|---|---|
| OWASP_2021_A01 | https://owasp.org/Top10/A01_2021-Broken_Access_Control/ |
| OWASP_2017_A05 | https://owasp.org/www-project-top-ten/2017/A5_2017-Broken_Access_Control.html |

## ***Missing Anti-clickjacking Header***

**URL:** http://localhost:5173/

**Risk:** High

**Confidence:** Medium

**Parameter:** x-frame-options

**Attack:**

**Evidence:**

**CWE ID:** 1021

**WASC ID:** 15

**Source:** Passive (10020 - Anti-clickjacking Header)

**Alert Reference:** 10020-1

**Input Vector:**

Description:

The response does not include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options to protect against Clickjacking' attacks.

**Solution:**

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 Policys "frame-ancestors" directive.

**Reference:**

https://developer.mozilla.org/en-lJS/docs/Web/HTTP/Headers/X-Frame-Options

**Alert Tags:**

| Key | Value |
|---|---|
| OWASP_2021_A05 | https://owasp.org/Top10/A05_2021-Security_Misconfiguration/ |
| WSTG-v42-CLNT-09 | https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/ |
| OWASP_2017_A06 | https://owasp.org/www-project-top-ten/2017/A6_2017-Security_Misconfiguration.html |

# MEDIUM-RISK VULNERABILITIES:

Medium-risk vulnerabilities are less severe compared to critical and high-risk vulnerabilities but still pose a notable risk to the security of the application. They may not lead to immediate exploitation or compromise, but they still require attention and remediation to prevent potential security incidents.

The medium-risk vulnerability found in this VAPT is as follows:

## ***X-Content-Type-Options Header Missing***

**URL:** http://localhost:5173/

**Risk:** Medium

**Confidence:** Medium

**Parameter:** x-content-type-options

**Attack:**

**Evidence:**

**CWE ID:** 693

**WASC ID:** 15

**Source:** Passive (10021 - X-Content-Type-Options Header Missing)

**Input Vector:**

Description:

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, Ourrent (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.

Other Info:

This issue still applies to error type pages (401, 403, 500, etc.) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type.

At "High" threshold this scan rule will not alert on client or server error responses.

**Solution:**

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.

**Reference:**

http://msdn.microsoft.com/enq-l s/library/ie/gg622941 aspx

https://owasp.org/www-community/Security Headers

**Alert Tags:**

| Key | Value |
|---|---|
| OWASP_2021_A05 | https://owasp.org/Top10/A05_2021-Security_Misconfiguration/ |
| OWASP_2017_A06 | https://owasp.org/www-project-top-ten/2017/A6_2017-Security_Misconfiguration.html |

# LOW-RISK VULNERABILITIES:

Low-risk vulnerabilities are the least severe and may not pose an immediate threat to the security of the application. However, they still represent areas where security improvements can be made to enhance the overall security posture.

The low-risk vulnerabilities found in this VAPT are as follows:

## ***Information Disclosure - Suspicious Comments***

**URL:** http://localhost:5173/

**Risk:** Low

**Confidence:** Low

**Parameter:**

**Attack:**

**Evidence:** from

**CWE ID:** 200

**WASC ID:** 13

**Source:** Passive (10027 - Information Disclosure - Suspicious Comments)

**Input Vector:**

Description:

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.

Other Info:

The following pattern was used: \bFROM\b and was detected in the element starting with: "<script type=”module”>

```
import RefreshRuntime from "/@react-refresh"

RefreshRuntime.injectIntoGlobalHook(window)
```

**Solution:**

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

**Alert Tags:**

| Key | Value |
|---|---|
| OWASP_2021_A01 | https://owasp.org/Top10/A01_2021-Broken_Access_Control/ |
| WSTG-v42-INFO-05 | https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/ |
| OWASP_2017_A03 | https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure.html |

# CONCLUSION

In conclusion, the Vulnerability Assessment and Penetration Testing (VAPT) conducted on the Library Management System (LMS) web application has unveiled a spectrum of security vulnerabilities across various risk categories. These vulnerabilities pose significant risks to the confidentiality, integrity, and availability of the system and its data. The identified vulnerabilities include critical, high-risk, medium-risk, and low-risk issues, each requiring prompt attention and remediation to mitigate potential security breaches and data leaks.

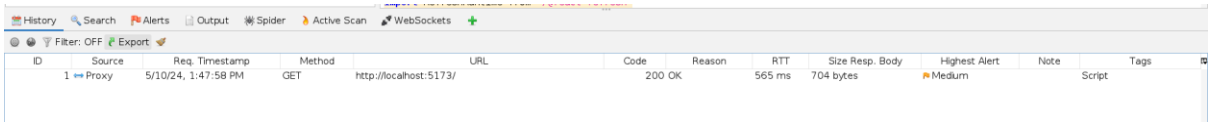
Addressing these vulnerabilities and implementing the recommended security measures is imperative to fortify the security posture of the LMS application. By prioritizing remediation efforts and adhering to industry best practices, the LMS application can bolster its resilience against cyber threats and ensure the continued protection of sensitive information and critical functionalities.

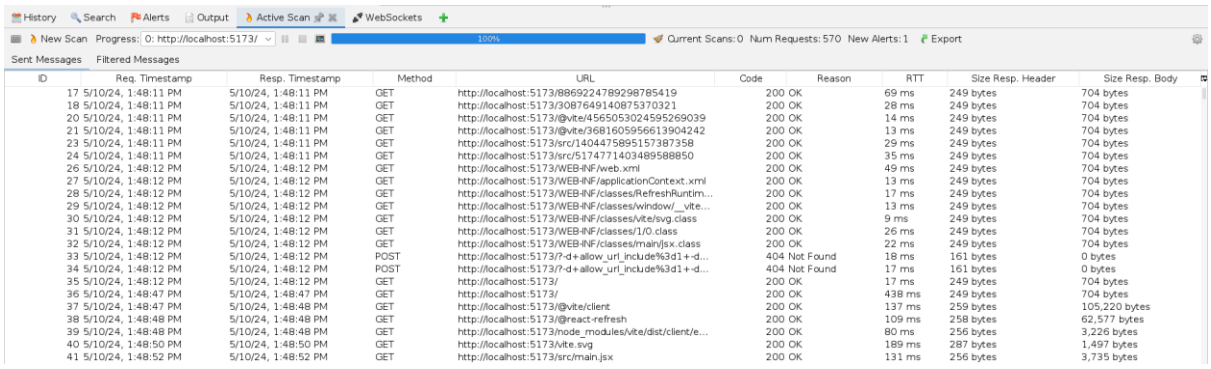
# APPENDICES

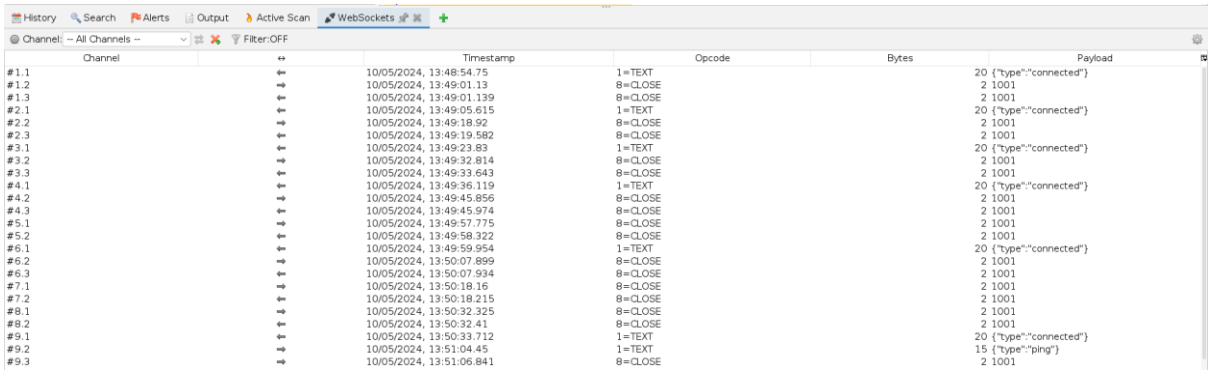
The appendices serve as a repository of detailed documentation pertinent to the Vulnerability Assessment and Penetration Testing (VAPT) performed on the LMS web application. They include:

**OWASP ZAP Logs:**

Logs generated during the OWASP ZAP scanning process provide valuable insights into the methodology employed during the scanning, the vulnerabilities identified, and the configurations utilized during the scanning process. They offer a deeper understanding of the scanning process and its outcomes.

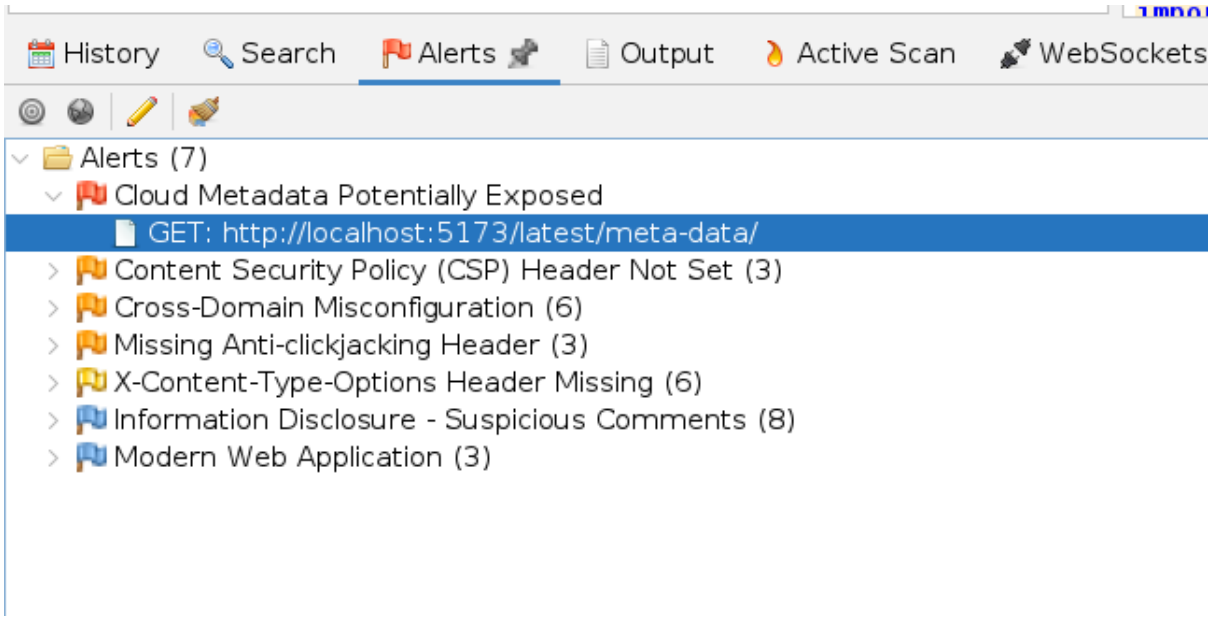


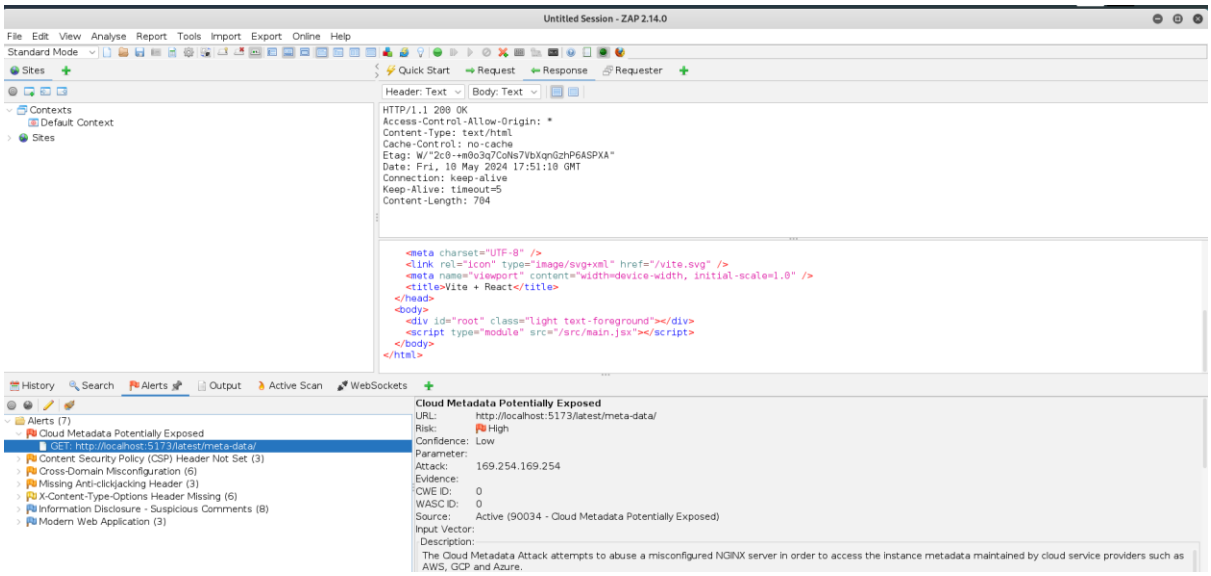



**Screenshots:**

Visual documentation, in the form of screenshots, captures instances of identified vulnerabilities, error messages, and any anomalous behaviors observed during the testing phase. These screenshots provide concrete evidence of vulnerabilities and aid in their comprehension.



**Exploit Code:**

Sample exploit code snippets or proof-of-concept (PoC) scripts are provided to demonstrate the potential exploitation of identified vulnerabilities. These exploit codes illustrate the severity and potential impact of each vulnerability, enhancing understanding and facilitating remediation efforts.

```
<!doctype html>
<html lang="en">
  <head>
    <script type="module">
import RefreshRuntime from "/@react-refresh"
RefreshRuntime.injectIntoGlobalHook(window)
window.$RefreshReg$ = () => {}
window.$RefreshSig$ = () => (type) => type
window.__vite_plugin_react_preamble_installed__ = true
</script>

    <script type="module" src="/@vite/client"></script>

    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
```