**CIRM – Web Application Vulnerability Assessment and Penetration Testing Report**

**Created for :** CIRM – GMS Application

**Prepared for :** GMS - VAPT

**Date of Preparation :** October 25, 2023

**Status :** Done

|  |  |
| --- | --- |
| **Document Control** | |
| Reference | CM/WA/09-001 |
| Status | Approved |
| Version | 3.0 |
| Date | 25th October 2023 |
| Author | Joseph Martin |
| Approved By | Rajesh Sampath |

|  |  |
| --- | --- |
| **Scope Information** | |
| Company | CIRM |
| Application | CIRM – Web application |
| Test Type | Web Application VAPT |

**Contents**

[Introduction 4](#_Toc148707755)

[Objective and Methodology 4](#_Toc148707756)

[Terminology and Score 5](#_Toc148707757)

[Retest Summary 5](#_Toc148707758)

[Information Gathering – Web Application 6](#_Toc148707759)

[Port Scanning Results 6](#_Toc148707760)

[Scope 6](#_Toc148707761)

[Limitations / Out of Scope 6](#_Toc148707762)

[Vulnerabilities Summary 7](#_Toc148707763)

[High Severity Vulnerabilities 8](#_Toc148707764)

[Vulnerability #1: Broken Authentication 8](#_Toc148707765)

[Vulnerability #2: Session Fixation 8](#_Toc148707766)

[Medium Severity Vulnerabilities 8](#_Toc148707767)

[Vulnerability #3: Host Header Injection 8](#_Toc148707768)

[Vulnerability #4: Open Redirection 8](#_Toc148707769)

[Vulnerability #5: Missing rate limiting 8](#_Toc148707770)

[Vulnerability #6: No Session Termination After Password Change 9](#_Toc148707771)

[Low Severity Vulnerabilities 9](#_Toc148707772)

[Vulnerability #7: Information exposure through an error message 9](#_Toc148707773)

[Vulnerability #8: Account Lockout Policy not enforced. 9](#_Toc148707774)

[Vulnerability #9: Missing Security Headers 9](#_Toc148707775)

[Vulnerability #10: Web server fingerprinting 10](#_Toc148707776)

[Vulnerability #11: Outdated Components Used 11](#_Toc148707777)

[Vulnerability #12: Missing Security Mechanism in Cookie 12](#_Toc148707778)

[Vulnerability #13: Weak Session Management 13](#_Toc148707779)

[Vulnerability #14: Direct IP Access Enabled 14](#_Toc148707780)

[Vulnerability #15: Version disclosure 14](#_Toc148707781)

[Vulnerability #16: Non-Expiry and Reusable CSRF Token 15](#_Toc148707782)

[Note: 17](#_Toc148707783)

[Lack of Two-Factor Authentication: Not Fixed 17](#_Toc148707784)

[Lack of Contact Details Verification: Not Fixed 18](#_Toc148707785)

**Introduction**

Security assessment is a process that enables an understanding of threats for better defence. Penetration testing simulates methods that intruders adopt to gain unauthorized access to an organization’s network systems, proceeding to compromise them. Most attackers follow conventional approaches to attempt a penetration.

Our security testing components focus on critical and high-severity vulnerabilities and strive to unearth application-level security issues to help provide valuable insights to development teams.

**Objective and Methodology**

To find potential vulnerabilities latent in web application interfaces and implement a simulated exploit to assess the possibilities of compromise, cover all attack vectors and trace the attack surface.

The core intent of running a VAPT test on the target infrastructure / applications is to evaluate the ease of gaining unauthorized access to the system by using different types of real-world exploits and common attack patterns to access the network or data. The exercise offers visibility into the possible impact of the flaw on the underlying network, operating system, database etc. using many methods a malicious hacker would attempt.

**Our testing methodologies, investigative process and procedures are aligned with SANS, NIST and OWASP standards, testing guides and best practices for application / infrastructure security appraisal. These guidelines are followed in order to elevate the level of risk awareness to globally recognized standards.**

A diagram of a process

Description automatically generated with medium confidence

**Terminology and Score**

**CVE** is a dictionary of publicly known information security vulnerabilities and exposures.

CVE’s common identifiers enable data exchange between security products and provide a baseline index point for evaluating coverage of tools and services. An information security "vulnerability" is a mistake in software application, configuration or operating system that can be directly used by a hacker to gain access to a system or network.

**Vulnerability** - A weakness which allows a hacker to break into / compromise a systems security.

**Exploit** - Code which allows an attacker to take advantage of a vulnerable system.

**Payload** - Actual code which runs on the system after exploitation.

**CWE -** Common Weakness Enumeration is a tangible set of software weaknesses that

enable better selection of security tools that can find these weaknesses.

**Retest Summary**

We assessed the attack environment of **CIRM – Web Application** with a view to validate the fixes introduced for the vulnerabilities identified in the previous section

The penetration testing performed on the target website discovered that most of the vulnerabilities are fixed and the fixes are working as desired. Some vulnerabilities have not been fixed. Our observations are detailed in the latter sections of this report.

The specific goals of the penetration test were as follows:

-Determine whether a remote attacker could penetrate the web application.

-Ascertain the impact of a security breach on data confidentiality and systems availability.

The aforementioned targets have been successfully met, the results of which are elucidated in the report.

It is important to note that seemingly minor design and functionality issues could be leveraged in attempts to compromise the application and the web server. We suggest deployment of the recommended mitigation techniques and controls as well as security protocols to secure the website and databases against all highlighted threats

**Information Gathering – Web Application**

|  |  |
| --- | --- |
| **IP Address** | 147.135.51.166 |
| **URL in Scope** | https://test-grants.cirm.ca.gov |
| **Server** | Nginx 1.18.0 (Ubuntu) / Apache httpd 2.4.18 |

**Port Scanning Results**

|  |  |  |
| --- | --- | --- |
| **Port** | **State** | **Service** |
| 80/tcp | Open | Apache httpd 2.4.18 |
| 113/tcp | Closed | ident |
| 443/tcp | Open | nginx 1.18.0 (Ubuntu) |
| 2000/tcp | Open | cisco-sccp |
| 5060/tcp | Open | sip |
| 8008/tcp | Open | http |
| 8010/tcp | Closed | xmpp |

**Scope**

The scope includes validation of the above-mentioned endpoints in line with OWASP Top 10 guidelines, NIST and SANS recommendations for potential vulnerabilities.

**Limitations / Out of Scope**

* We had no knowledge of the server-side code that makes up the application, therefore we had to make assumptions based on the information we were able to locate.
* We were limited by the amount of time we had allocated to perform our assessment, whereas a malicious individual may potentially have unlimited time.

**Vulnerabilities Summary**

|  |  |  |  |
| --- | --- | --- | --- |
| **Severity** | **Vulnerability** | **Impact** | **Status** |
| High | Broken authentication | Loss of Confidentiality | Fixed |
| High | Session fixation | Account takeover | Fixed |
| Medium | Host Header Injection | Phishing Attacks. | Fixed |
| Medium | Open Redirection | Phishing Attacks. | Fixed |
| Medium | Missing Rate Limit | Denial of Service Attack | Fixed |
| Medium | No Session Termination After Password Change | Account takeover | Fixed |
| Low | Information disclosure through error message | Loss of Confidentiality | Fixed |
| Low | Account Lockout Policy Not Enforced | Brute force attacks | Fixed |
| Low | Missing security headers | Possible XSS attack | Not Fixed |
| Low | Web server fingerprinting | Loss of Confidentiality | Not Fixed |
| Low | Outdated Component Used | Security Best Practice Violation. | Not Fixed |
| Low | Missing Security Mechanism in Cookie Attribute | Possible XSS and MITM attacks | Not Fixed |
| Low | Weak Session Management | MITM Attacks, Account Takeover | Not Fixed |
| Low | Direct IP Access Enabled | Information disclosure. | Fixed |
| Low | Version disclosure | Information disclosure. | Not Fixed |
| Low | Non-Expiry and Reusable CSRF Token | Data manipulation | Not Fixed |

**High Severity Vulnerabilities**

**Vulnerability #1: Broken Authentication**

|  |  |
| --- | --- |
| Vulnerability Details | Session is not being validated before authorizing a user’s request for a resource. |
| Status | Fixed |
| Remarks | The issue is marked as fixed since the file is intentionally designed for public accessibility, and as such, it does not pose a vulnerability related to broken authentication mechanisms. |

**Vulnerability #2:** **Session Fixation**

|  |  |
| --- | --- |
| Vulnerability Details | Session Fixation is an attack that permits an attacker to hijack a valid user session. |
| Status | Fixed |
| Remarks | Pre- and post-login cookies are different; hence the issue has been marked fixed. |

**Medium Severity Vulnerabilities**

**Vulnerability #3: Host Header Injection**

|  |  |
| --- | --- |
| Vulnerability Details | Host header Injection. |
| Status | Not Fixed |
| Remarks | The issue has been resolved by implementing server-side validations to prevent redirections. |

**Vulnerability #4: Open Redirection**

|  |  |
| --- | --- |
| Vulnerability Details | Open Redirection due to invalidated input. |
| Status | Fixed |
| Remarks | This issue is marked as fixed since it’s no longer redirecting to malicious third-party websites. |

**Vulnerability #5: Missing rate limiting**

|  |  |
| --- | --- |
| Vulnerability Details | Missing rate limiting leads to replay attacks. This vulnerability also allows spamming of an email message to a particular user/provider |
| Status | Not Fixed |
| Remarks | The issues have been resolved, and the rate limiting is now set at 3 requests per second. If a user attempts more than 3 requests per second, the application will throw a "503 Service Temporarily Unavailable" error. |

**Vulnerability #6: No Session Termination After Password Change**

|  |  |
| --- | --- |
| Vulnerability Details | Active Sessions available after a password change |
| Status | Not fixed |
| Remarks | The issues have been fixed; the session gets terminated once the user changes the password. |

**Low Severity Vulnerabilities**

**Vulnerability #7: Information exposure through an error message**

|  |  |
| --- | --- |
| Vulnerability Details | Server information getting exposed through an error message |
| Status | Fixed |
| Remarks | Default error pages are disabled, and custom error pages are implemented. |

**Vulnerability #8: Account Lockout Policy not enforced.**

|  |  |
| --- | --- |
| Vulnerability Details | Account is not being locked after a specific number of wrong attempts. |
| Status | Fixed |
| Remarks | The issues have been resolved, and the rate limiting is now set at 3 requests per second. If a user attempts more than 3 requests per second, the application will throw a "503 Service Temporarily Unavailable" error. |

**Vulnerability #9: Missing Security Headers**

|  |  |
| --- | --- |
| Vulnerability Details | HTTP Response Headers for Security have not been enforced. |
| Description | We have ascertained that certain security headers are missing in the  Response which weakens the security posture of the application. |
| Status | Not Fixed |
| Affected Instances | Entire Application |
| Recommendation | We recommend adding the necessary missing security headers in entire application:  Add Content-Security-Policy header.  Add Permission-policy header |

**Evidence of the defect:**

Application server lacks security headers which allow attackers to perform phishing.

A screenshot of a computer

Description automatically generated

*Figure 1: Missing Security Headers*

**Vulnerability #10: Web server fingerprinting**

|  |  |
| --- | --- |
| Vulnerability Details | Web-server version details are exposed in the response headers. |
| Description | It is observed that the application is disclosing the web server information in the response banner. |
| Status | Not Fixed |
| Affected Instances | Response Headers |
| Recommendation | Remove Web server information from the banner. |

**Evidence of the defect:**

The response header of the application is revealing the web server being used.

A screenshot of a computer

Description automatically generated

*Figure 2: web server fingerprinting*

**Vulnerability #11: Outdated Components Used**

|  |  |
| --- | --- |
| Vulnerability Details | Outdated components used. |
| Description | components that are used in the application are outdated |
| Status | Not Fixed |
| Affected instances | <https://test-grants.cirm.ca.gov> |
| Recommendation | Update of software components to prevent the exploitation of known vulnerabilities. Apply security patches promptly to mitigate risks associated with outdated components. |

**Evidence of the defect:**

The application server uses outdated components.

A close-up of a computer code

Description automatically generated

*Figure 3: Vulnerable Component Used*

**Vulnerability #12: Missing Security Mechanism in Cookie**

|  |  |
| --- | --- |
| Vulnerability Details | No Security mechanisms implemented in the cookie attributes. |
| Description | We have ascertained that certain security attributes are missing in the cookies which weakens the security posture of the application. |
| Status | Not Fixed |
| Affected Instances | Entire Application |
| Recommendation | Therefore, it is recommended to set secure attribute value to true and add a secure "SameSite" attribute. |

**Evidence of the defect:**

The application does not set secure attribute value to true and “samesite” to strict or lax.

A screenshot of a computer

Description automatically generated

*Figure 4: Missing Attributes*

**Vulnerability #13: Weak Session Management**

|  |  |
| --- | --- |
| Vulnerability Details | Application is using weak session mechanism. |
| Description | A weak session mechanism vulnerability refers to a security flaw in a web application's session management system, where inadequate safeguards or poor implementation allow attackers to compromise user sessions. |
| Status | Not Fixed |
| Affected Instances | Session Management |
| Recommendation | It is recommended to use JWT based session management. |

**Evidence of the defect:**

The application uses a weak session id.

A screenshot of a computer

Description automatically generated

*Figure 5: Weak Session Management*

The session id is generated using MD5 hash algorithm.

A screenshot of a computer

Description automatically generated

*Figure 6: Weak Session Management*

**Vulnerability #14: Direct IP Access Enabled**

|  |  |
| --- | --- |
| Vulnerability Details | Direct IP access Enabled in the web server. |
| Status | Fixed |
| Recommendation | Direct IP access has been disabled in the server configuration. |

**Vulnerability #15: Version disclosure**

|  |  |
| --- | --- |
| Vulnerability Details | The web application is disclosing information on the rendering page. |
| Description | The web application exposes sensitive information and component versions on its rendering page, potentially exposing confidential data to unauthorized users |
| Status | Not Fixed |
| Affected Instances | <https://test-grants.cirm.ca.gov/> |
| Recommendation | Remove sensitive information disclosing in the application and keep all components, libraries, and frameworks, up to date. Regularly check for updates and security advisories for both Rails and Ruby. |

**Evidence of the defect:**

Avoid revealing sensitive information and component versions on the rendering page to prevent attackers from exploiting known vulnerabilities.

A screenshot of a computer

Description automatically generated

*Figure 8: Version Disclosure*

**Vulnerability #16: Non-Expiry and Reusable CSRF Token**

|  |  |
| --- | --- |
| Vulnerability Details | Application does not terminate old CSRF tokens. |
| Description | Cross-Site Request Forgery (CSRF) is a type of attack where an attacker tricks a user into unknowingly performing actions on a web application without their consent.  In this case, the application generates new tokens for every request but doesn't terminate old tokens, there is a significant vulnerability present. CSRF tokens are designed to be a one-time use mechanism. By generating new tokens for every request but not expiring or invalidating the old tokens, an attacker could exploit this behaviour. |
| Status | Not Fixed |
| Affected Instances | Entire Application |
| Recommendation | To mitigate the risks associated with this vulnerability, consider implementing the following recommendations:   * Single-Use Tokens: CSRF tokens should be designed to be used only once. After a token has been used for a specific action, it should be invalidated, making it useless for any future requests. * Token Expiry: Set an expiration time for CSRF tokens. Tokens should be generated with a limited lifespan, making them useless once they expire. A short expiration time is recommended to minimize the window of opportunity for attackers. * Token Binding: Bind the CSRF token to specific user sessions or actions. This ensures that the token is only valid for the intended user and the intended action, even if an attacker manages to acquire the token. |

**Evidence of the defect:**

The application generates new tokens for every request but doesn't terminate old tokens, there is a significant vulnerability present.

A screenshot of a computer

Description automatically generated

*Figure 9: Misconfigured CSRF Token*

By generating new tokens for every request but not expiring or invalidating the old tokens, an attacker can still perform CSRF attacks if the user's CSRF token gets exposed.

A screenshot of a computer

Description automatically generated

*Figure 10: Misconfigured CSRF Token*

**Note:**

**Lack of Two-Factor Authentication: Not Fixed**

* Missing Two-Factor Authentication Implementation in Login and Password Change Pages.
* Implement two-factor authentication (2FA) on both the login and user password pages.

A screenshot of a computer

Description automatically generated

*Figure 11: Missing 2FA Verification*

**Lack of Contact Details Verification: Not Fixed**

* No verification is in place when users change their email addresses in their contact details.
* Implement email verification mechanisms for user contact detail changes to ensure the security and integrity of account information.

<https://test-grants.cirm.ca.gov/contacts/edit_my_profile_contact/38860>

A screenshot of a computer

Description automatically generated

*Figure 12: No Email Verification*