**Assessment duration:** 7 days

**Report Created On:** 12/03/2023

**Assessment Performed By:** Subhash Paudel

**Document Control**

**Statement of Confidential**

The contents of this document have been developed by 38 SECURITY. 38 SECURITY considers the contents of this document to be proprietary and business confidential information. This information is to be used only in the performance of its intended use. This document may not be released to another vendor, business partner or contractor without prior written consent from 38 SECURITY. Additionally, no portion of this document may be communicated, reproduced, copied, or distributed without the prior consent of 38 SECURITY.

The contents of this document do not constitute legal advice. 38 SECURITY’s offer of services that relate to compliance, litigation or other legal interests are not intended as legal counsel and should not be taken as such. The assessment detailed herein is against a fictional company for training and examination purposes, and the vulnerabilities in no way affect 38 SECURITY external or internal infrastructure.

**Engagement Contacts**

|  |  |  |
| --- | --- | --- |
| Inlanefreight Contacts | | |
| Primary Contact | **Title** | **Primary Contact Email** |
| Rachel Williams | Chief Executive Officer | rachel@inlanefreight.local |
| Secondary Contact | **Title** | **Secondary Contact Email** |
| William Ley | Chief Technical Officer | wley@inlanefreight.local |

|  |  |  |
| --- | --- | --- |
| Assessor Contact | | |
| Assessor Name | **Title** | **Assessor Contact Email** |
| 38 SECURITY | Security Consultant | subhash@38security.corp.com.au |

**Proprietary Information**

The content of this document should be considered proprietary information and should not be disclosed outside of organization.

Table of Contents

[1. Executive Summary 5](#_Toc129584414)

[2. Scope 6](#_Toc129584415)

[2.1 Target Scope 6](#_Toc129584416)

[3. Approach and Methodology 7](#_Toc129584417)

[3.1 Web Application Penetration Testing 7](#_Toc129584418)

[4. Risk Ratings 10](#_Toc129584419)

[5. Summary of Findings 11](#_Toc129584420)

[5.1 Vulnerabilities by Impact 11](#_Toc129584421)

[6. Detailed Findings 13](#_Toc129584422)

[6.1 R1 – Stored HTML Injection 13](#_Toc129584423)

[6.2 R2 – Cross Site Request Forgery (CSRF) 15](#_Toc129584424)

[6.3 R3 – Directory Traversal 17](#_Toc129584425)

[6.4 R4- Missing Error Handling 19](#_Toc129584426)

[6.5 R5- Using Components with Known Vulnerabilities 21](#_Toc129584427)

[6.6 R6- Application Accessible using IP 22](#_Toc129584428)

[6.7 R7- Cookie Not Marked as Secure 24](#_Toc129584429)

[6.8 R8- Server Banner Disclosure 26](#_Toc129584430)

[7. Appendix 28](#_Toc129584431)

[8. Conclusion 28](#_Toc129584432)

[9. Additional resources 29](#_Toc129584433)

# Executive Summary

IT Sec Vulnerability Assessment team was engaged by **38 Security** team to conduct security review of web application during the period **8th March 2023 to 14th March 2023.**

The prime objective of this security exercise is to assess & identify potential cyber risks associated with webapplication and remediate those identified risks by following effective Risk Management methodology.

Globally accepted Web Application Security Framework – OWASP (Open Web Application Security Project) provides minimum standard for Web Application Security Assessment. The Web Application Security Testing methodology has been aligned with the OWASP standards and industry best practices such as SANS, WASC, CWE, etc. The IT Security Vulnerability assessment team identified **08 vulnerabilities (02 High, 02 Medium** and **04 Low)** in the target web application.

Based on the vulnerabilities discovered as the part of this exercise, IT Security Vulnerability Assessment team strongly recommends taking immediate action and calculated measures to protect this application.

Risk rating of the vulnerabilities detected have been based on our knowledge of business requirement, priorities, and infrastructure.

Risk details have been documented under section 6, which includes corresponding recommendations. It is advised to prioritize and address the identified vulnerabilities at the earliest opportunity to reduce the overall security exposure.

# Scope

IT Sec Vulnerability Assessment team performed a security review of Web application to discover security issues that could lead to the compromise or abuse of systems which may impact negatively on business or reputation.

# Target Scope

As a part of the security review process, IT Sec Vulnerability Assessment team had discussed with **BNEW SAN SAS SDU India VAS 1 s**team and identified the key applications to undergo the security review process. Because of this the below application has been identified to perform the security review.

Web Application Penetration testing activities were executed as follows:

1. **Information Gathering:** Examining Web application's content, functionality, and security mechanisms.
2. **Threat Profiling:** Identify security objectives, survey & decompose the application, and design the attack vectors based on credible threats.
3. **Automated Scanning**: Discover vulnerabilities by covering OWASP, WASC and SANS top security issues.
4. **Manual Penetration Testing:** Manual assessment to re-validate the findings from automated scan and eliminate false positives. Assess business logic flaws and server side & client-side vulnerabilities.
5. **Analysis & Reporting:** Analyse the identified security vulnerability and document the findings with credible risk ratings. Report documentation with appropriate recommendation or mitigation measures.

# Approach and Methodology

Hack The Box Academy performed testing under a “black box” approach May 12, 2022, to May 31, 2022 without credentials or any 38d knowledge of Inlanefreight’s internally facing environment with the goal of identifying unknown weaknesses. Testing was performed from a non-evasive standpoint with the goal of uncovering as many misconfigurations and vulnerabilities as possible. Testing was performed remotely via a host that was provisioned specifically for this assessment. Each weakness identified was documented and manually investigated to determine exploitation possibilities and escalation potential. Hack The Box Academy sought to demonstrate the full impact of every vulnerability, up to and including internal domain compromise. If Hack The Box Academy were able to gain a foothold in the internal network, Inlanefreight allowed for further testing including lateral movement and horizontal/vertical privilege escalation to demonstrate the impact of an internal network compromise.

# Web Application Penetration Testing

The common vulnerabilities in Open Web Application Security Project (OWASP) provide a minimum standard for application penetration testing. The IT Sec Vulnerability Assessment team follows web application security assessment methodology has been aligned with the OWASP standards and industry’s leading security consortiums such as SANS, CWE, WASC, etc. IT Sec Vulnerability Assessment team has followed the one-phase approach to assess the application.

**Phase 1: Grey Box Assessment -** Technique to test the software product or application with partial knowledge of the internal workings of an application. It can be considered as non-intrusive and unbiased because it does not require tester to have access to the source code. By combining the input of developers & testers the overall product quality can be improved. In this process, context specific errors that are related to systems are commonly identified. It will increase the testing coverage by concentrating on all the layers of any complex system. Tests are designed based on the knowledge of algorithm, architectures, internal states, or other high -level descriptions of the program behavior.

The following figure depicts the steps taken to find vulnerabilities, expose weaknesses, and identify deviations from accepted best practices in the assessed application.

**Diagram

Description automatically generated**

***Highlights of activities that were performed in each of the stages are described below:***

|  |  |
| --- | --- |
| **Stage 1** | **Profiling**  In this stage, profiling of the target application is performed by identifying user entry points, interfaces to external or internal applications, identifying roles with varying trust levels and determine the data flow path with indication on privilege boundaries. |
| **Stage 2** | **Automated Scanning**  Automated application vulnerability scanners (i.e., commercial and open source) are used to scan for application specific vulnerabilities covering all OWASP, WASC and SANS references. |
| **Stage 3** | **Vulnerability Detection**  This phase involved a complete hybrid approach of identifying the application security vulnerabilities with automated tools and scripts along with manual assessment to eliminate false positives and negatives.  Manual assessment uses various vulnerability databases to identify vulnerabilities that were missed during automated scans in addition to security verification of business logic flaws, broken access controls and more. |
| **Stage 4** | **Vulnerability Exploitation**  The primary focus in this phase is on using manual security testing techniques to exploit the systems, which included several exploits to assess the application hardening measures, cryptography issues, authentication & authorization controls, session management module, business logic flaws and various validation measures.  Attack scenarios for production environment will use a combination of exploit payloads in strict accordance with agreed rules of engagement. |
| **Stage 5** | **Reporting**  All exploitable security vulnerabilities in the target application are recorded with associated CVSS v3 based scores are reported to the client. The identified security vulnerability is thoroughly assessed and reported along with appropriate recommendation or mitigation measures. |

# Risk Ratings

The risk rating for each finding in this report is based on the Business Impact and Exploit vector of the vulnerability. Here’s a guide interpreting the risk rating:

|  |  |  |
| --- | --- | --- |
| **Risk Rating** | **CVSSv3 Score \*** | **Explanation** |
| **CRITICAL** | **9.0 – 10** | Vulnerability was discovered that has been rated as critical. It is recommended that corrective actions are implemented urgently. This category of risk should be monitored closely by management. |
| **HIGH** | **7.0 - 8.9** | Vulnerability was discovered that has been rated as important. It is recommended that corrective actions must be implemented within a short term. |
| **MEDIUM** | **4.0 - 6.9** | Vulnerability was discovered that has been rated as of medium criticality. It is recommended that corrective actions should be part of on-going security maintenance of the system. |
| **LOW** | **1.0 - 3.9** | Vulnerability was discovered that has been rated as of low criticality. Owner should consider whether to apply corrective measures as part of routine maintenance tasks or to accept the risk. |
| **INFO** | **0 - 0.9** | A finding was discovered that has been rated as of informational value which should be addressed to meet industry best practice. |

*(\*) IT Sec Vulnerability Assessment team has adopted the Common Vulnerability Scoring System (version 3). CVSS is a vendor independent, industry open standard. It is designed to convey vulnerability severity; help determine urgency and priority of response. The table below gives a key to the icons and symbols used through this report to provide a clear and concise risk scoring system.*

***For additional information, please refer:*** [*https://www.first.org/cvss/calculator/3.0*](https://www.first.org/cvss/calculator/3.0)

# Summary of Findings

The security review of web application has determined **08 vulnerabilities (02 High, 02 Medium** and **04 Low Risks)** during the assessment. If exploited, these vulnerabilities may lead to loss of confidentiality, might also increase system disruption and loss of intellectual property. The following graph depicts the total number of findings discovered during this assessment.

|  |  |  |  |
| --- | --- | --- | --- |
| Finding Severity | | | |
| High | **Medium** | **Low** | **Total** |
| 2 | **2** | **4** | **8** |

# Vulnerabilities by Impact

**Below table summarizes the list of security vulnerabilities findings.**

|  |  |  |
| --- | --- | --- |
| Sl. No | Vulnerability | Severity |
| 1 | Stored HTML Injection | **High** |
| 2 | CSRF (Cross Site Request forgery) | **High** |
| 3 | Directory Traversal | **Medium** |
| 4 | Missing Error Handling | **Medium** |
| 5 | Using Components with known Vulnerabilities | **Low** |
| 6 | Application Accessible using IP | **Low** |
| 7 | Cookie Not Marked as Secure | **Low** |
| 8 | Server Banner Disclosure | **Low** |

# Detailed Findings

# R1 – Stored HTML Injection

|  |  |  |  |
| --- | --- | --- | --- |
| R1 – Stored HTML Injection | | | |
| Risk Rating: High | CVSS v3 score: **7.1** | Status: **Open** | OWASP Category: **A07** |
| CVSS v3 Vector String:  <https://www.first.org/cvss/calculator/3.0#CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:H/A:N>  Graphical user interface, text, application, chat or text message  Description automatically generated | | | |
| Finding:  During the penetration testing assessment, it was observed that in the application stored HTML injection, the payload is stored by the web server and delivered later, potentially to multiple users which is openly redirecting to external phishing pages or some attackers-controlled domain.  Affected URL: | | | |
| Risk Description:  An HTML injection vulnerability exists when user-supplied data is used by server-side scripts to generate the content in a fashion which allows injected HTML content to be rendered. HTML injection is very similar to Cross-Site-Scripting (XSS) in that it allows the inclusion of content by an attacker but differs in that an HTML injection vulnerability does not require the execution of a script. As a result, while an XSS vulnerability might allow an attacker to do anything that can be done in JavaScript, an HTML injection vulnerability typically only allows an attacker to insert non-active content such as a link or iframe to another website.  This injected content may be used by an attacker to help facilitate a social engineering campaign to steal a user's data, to take advantage of a cross-site-request-forgery vulnerability in the application, or to direct the user to another site where a malicious payload can be delivered. Attackers often inject malicious JavaScript, VBScript, ActiveX, and/or HTML into vulnerable applications to deceive the user to gather data from them.   |  | | --- | | **Recommendation**  The first step toward mitigating HTML Injection attacks is to validate every application input to prevent acceptance of any characters that may have special meaning within the application or in the destination of the data (in this case, the browser). All input should be sanitized; not just parameters that the user is supposed to provide, but all data in the request, including hidden fields, cookies, headers, the URL itself, and so forth. The recommended method of input validation is to employ a white-list filter of known-good characters, causing the application to accept only inputs that have expected values. Input validation routines should always check the data for length, range, type, and format.  In addition to input validation, output encoding provides a second layer of defense against these types of attacks. Output encoding can be used even when input validation isn't possible because the format and semantic meaning of the data is unknown, or because the input syntax may legitimately include what would usually be considered malicious. In the case of data being returned to the user's web browser, output encoding requires converting characters that may be treated as code (e.g., <, >, ", ') to their appropriate HTML entities ("&lt;", "&gt;", "&quot;", "&apos;"). This way, these characters will be treated as data by the web browser instead of being executed as code. |   Reference Link:  <https://www.imperva.com/learn/application-security/html-injection/>  <https://corneacristian.medium.com/methods-to-exploit-html-injection-17b4254035e>  <https://www.owasp.org/index.php/Testing_for_HTML_Injection_(OTG-CLIENT-003)>  Supporting Evidence: | | | |

# R2 – Cross Site Request Forgery (CSRF)

|  |  |  |  |
| --- | --- | --- | --- |
| R2– Cross Site Request Forgery (CSRF) | | | |
| Risk Rating: High | CVSS v3 score: **7.1** | Status: **Open** | OWASP Category: **A01** |
| CVSS v3 Vector String:  <https://www.first.org/cvss/calculator/3.0#CVSS:3.0/AV:N/AC:H/PR:L/UI:R/S:U/C:H/I:H/A:H>  Graphical user interface, text, application, chat or text message  Description automatically generated | | | |
| Finding:  During the penetration testing assessment, it is observed that the application is vulnerable to Cross Site Request Forgery vulnerability.  Affected URL: | | | |
| Risk Description:  Depending on the application, an attacker can mount any of the actions that can be done by the user such as adding a user, modifying content, deleting data. All the functionality that’s available to the victim can be used by the attacker. The only exception to this rule is a page that requires extra information that only the legitimate user can know (such as user’s password).  Recommendation:  The most common method to prevent Cross-Site Request Forgery (CSRF) attacks is to append CSRF tokens to each request and associate them with the user’s session. Such tokens should at a minimum be unique per user session but can also be unique per request. By including a challenge token with each request, the developer can ensure that the request is valid and not coming from a source other than the user.  Reference Link:  <https://owasp.org/www-community/attacks/csrf>  <https://owasp.deteact.com/cheat/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html>  Supporting Evidence:   1. Login to the application with valid credentials and open the Admin/ Roles 2. Capture the request from the proxy tool such as Burp suite 3. Generate the Anticsrf form as shown in the below screenshot 4. Log out to the application and send the crafted page the attacker. 5. As shown in the below screenshot, the application is vulnerable to CSRF attack. | | | |

# R3 – Directory Traversal

|  |  |  |  |
| --- | --- | --- | --- |
| R3 – Directory Traversal | | | |
| Risk Rating: Medium | CVSS v3 score: **5.8** | Status: **Open** | OWASP Category: **A5** |
| CVSS v3 Vector String:  <https://www.first.org/cvss/calculator/3.0#CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:C/C:L/I:N/A:N>  Graphical user interface, text, application, chat or text message  Description automatically generated | | | |
| Finding: During the penetration testing assessment, it is observed the application is vulnerable to directory traversal or path traversal vulnerability. A path traversal attack allows attackers to access directories that they should not be accessing, like config files or any other files/directories that may contains server’s data not intended for public.  Affected URL: | | | |
| Risk Description:  1. Giving appropriate permissions to directories and files. A PHP file typically runs as www-data user on Linux. We should not allow this user to access system files. But this doesn’t prevent this user from accessing web-application specific config files.  2. Process URI requests that do not result in a file request, e.g., executing a hook into user code, before continuing below.  3. When a URI request for a file/directory is to be made, build a full path to the file/directory if it exists, and normalize all characters (e.g., %20 converted to spaces).  4. It is assumed that a ‘Document Root’ fully qualified, normalized, path is known, and this string has a length N. Assume that no files outside this directory can be served.  5. Ensure that the first N characters of the fully qualified path to the requested file is the same as the ‘Document Root’.  6. Using a hard-coded predefined file extension to suffix the path does not limit the scope of the attack to files of that file extension.  Recommendation:  To protect the application from this weakness it is advised to follow these instructions:   * Never use attacker-controlled data as a filename or part of the filename when performing operations on files or folders. If filename should be based on the user's choice use predefined conditions instead of direct input. * Perform whitelist checks when working with files or directories using user-controlled input. * Use sandbox environments (e.g., jail, chroot) that enforce strict boundaries between the process and the operating system.   Reference Link:  <https://www.immuniweb.com/vulnerability/path-traversal.html#severity>  <https://owasp.org/www-community/attacks/Path_Traversal>  Supporting Evidence: | | | |

# R4- Missing Error Handling

|  |  |  |  |
| --- | --- | --- | --- |
| **R4 – Missing Error Handling** | | | |
| Risk Rating: **Medium** | CVSS v3 score: **2.4** | Status: **Open** | OWASP Category: **A7** |
| **CVSS v3 Vector String:**  <https://www.first.org/cvss/calculator/3.0#CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N> | | | |
| **Finding:**  During the penetration testing assessment, its observed the web application is showing information of internal search functionalities in the errror page.  **Affected URL**: | | | |
| **Risk Description:**  When an attacker explores a web site looking for vulnerabilities, the amount of information that the site provides is crucial to the eventual success or failure of any attempted attacks. If the application shows the attacker a stack trace, it relinquishes information that makes the attacker’s job significantly easier. For example, a stack trace might show the attacker a malformed SQL query string, the type of database being used, and the version of the application container. This information enables the attacker to target known vulnerabilities in these components.  **Recommendation:**   1. A web application must define a default error page for 404 errors, 500 errors, and to catch java.lang. Throwable exceptions prevent attackers from mining information from the application container’s built-in error response. 2. Handling standard HTTP error codes is useful and user-friendly in addition to being a good security practice, and a good configuration will also define a last-chance error handler that catches any exception that could possibly be thrown by the application.   **Reference:**  <https://www.matthewedgar.net/find-and-fix-404-errors/>  https://owasp.org/www-community/vulnerabilities/Missing\_Error\_Handling#:~:text=When%20an%20attacker%20explores%20a,the%20attacker%27s%20job%20significantly%20easier. | | | |
| **Supporting Evidence:** | | | |

# R5- Using Components with Known Vulnerabilities

|  |  |  |  |
| --- | --- | --- | --- |
| **R5 –** **Using Components with Known Vulnerabilities** | | | |
| Risk Rating: **Low** | CVSS v3 score: **2.4** | Status: **Open** | OWASP Category: **A7** |
| **CVSS v3 Vector String:**  <https://www.first.org/cvss/calculator/3.0#CVSS:3.0/AV:P/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N>  Graphical user interface, text, application, chat or text message  Description automatically generated | | | |
| **Finding:**  The application is using various vulnerable components which are vulnerable to different attacks if exploited successfully.  **Affected URL**: | | | |
| **Risk Description:**  The application is using various vulnerable components which are vulnerable to different attacks like Cross site scripting etc. if exploited successfully.  **Recommendation**  Upgrade to the latest version on below reported component.  **Reference Link:**  jQuery v1.7.2- <https://security.snyk.io/package/npm/jquery> | | | |
| **Supporting Evidence:** | | | |

# R6- Application Accessible using IP

|  |  |  |  |
| --- | --- | --- | --- |
| R6– Application Accessible using IP | | | |
| Risk Rating: Low | CVSS v3 score: **2.4** | Status: **Open** | OWASP Category: **A7** |
| CVSS v3 Vector String: <https://www.first.org/cvss/calculator/3.0#CVSS:3.0/AV:P/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N>  Graphical user interface, text, application, chat or text message  Description automatically generated | | | |
| Finding:  During the security assessment we found web application is accessible with IP address, instead of server name.  Affected URL: | | | |
| Risk Description:  Using an IP address instead of a host name with a DNS entry means attacker can get a layer of routing flexibility that can be used for further exploitation and may lead to Denial of service.  Recommendation  It is recommended to access the application through URL instead of IP address. | | | |
| Supporting Evidence: | | | |

# R7- Cookie Not Marked as Secure

|  |  |  |  |
| --- | --- | --- | --- |
| R7 – Cookie Not Marked as Secure & no HTTP Only flag | | | |
| Risk Rating: Low | CVSS v3 score: **2.4** | Status: **Open** | OWASP Category: **A7** |
| CVSS v3 Vector String: <https://www.first.org/cvss/calculator/3.0#CVSS:3.0/AV:P/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N> | | | |
| Finding:  During the penetration testing assessment, cookie not marked as secure & no HTTP Only flag set, and transmitted over HTTPS. This means the cookie could potentially be stolen by an attacker who can successfully intercept and decrypt the traffic, or following a successful man-in-the-middle attack. If the HttpOnly attribute is set on a cookie, then the cookie's value cannot be read or set by client-side JavaScript. This measure makes certain client-side attacks, such as cross-site scripting, slightly harder to exploit by preventing them from trivially capturing the cookie's value via an injected script.  Affected URL: | | | |
| Risk Description:  This cookie will be transmitted over a HTTP connection, therefore if this cookie is important (such as a session cookie), an attacker might intercept it and hijack a victim's session. If the attacker can carry out a man-in-the-middle attack, he/she can force the victim to make an HTTP request to steal the cookie.  Recommendation   * Mark all cookies used within the application as secure.   Reference:  <https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/session-cookie-not-marked-as-secure/>    <https://cwe.mitre.org/data/definitions/1004.html#:~:text=If%20the%20HttpOnly%20flag%20is,be%20exposed%20to%20unintended%20parties.&text=If%20the%20cookie%20in%20question,the%20identity%20of%20the%20user>. | | | |
| Supporting Evidence: | | | |

# R8- Server Banner Disclosure

|  |  |  |  |
| --- | --- | --- | --- |
| R8 – Server Banner Disclosure | | | |
| Risk Rating: Low | CVSS v3 score: **2.4** | Status: **Open** | OWASP Category: **A7** |
| CVSS v3 Vector String: <https://www.first.org/cvss/calculator/3.0#CVSS:3.0/AV:P/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N>  Graphical user interface, text, application, chat or text message  Description automatically generated | | | |
| Finding:  During the security assessment, it was confirmed that the web application discloses server banner along with the version.  Affected URL: | | | |
| Risk Description:  Verbose server information is sent in the HTTP responses from the server. The information included in the response contains the server’s name, type, and version number.  Below is an example of a HTTP response that contains verbose server banners:  HTTP/1.1 200 OK  Server: Apache 2.0  Cache-control: private  X-Powered-By: JSP/2.2  Content-Type: text/html;charset=utf-8  Content-Language: en-US  Content-Length: 3347  Verbose server banners provide additional information that allows an attacker to perform targeted attacks to the specific technology stack in use by the application and underlying infrastructure.  Recommendation  Verbose server information should be removed from all HTTP responses. This can be performed by modifying the server's configuration files or through the use and configuration of a web application firewall.  Reference:  <https://httpd.apache.org/download.cgi>  <https://www.owasp.org/index.php/Fingerprint_Web_Server_(OTG-INFO-002)> | | | |
| Supporting Evidence | | | |

# Appendix

All appendix of all findings here

# Conclusion

The security review identified **08 vulnerabilities (02 High, 02 Medium** and **04 Low)** in the target web application.

The overall control environment of is ‘**Poor’**.

On analysing the identified vulnerabilities during this security review, it appears that most of them might have crept in at different phases of the deployment and software development cycle. These findings underscore the need for vigorously applying a culture of security upon the entire length and breadth of the SDLC model that is being applied for developing the application. This would mean a continuous process of strengthening the threat model, risk identification and mitigation processes at each stage of the application development lifecycle.

# Additional resources

Nmap scan result, burp scan result, sqlmap result, zap results