We Test Pens Incorporated

COMP90074 - Web Security Assignment 2

Xiande Wen

905003

**PENETRATION TEST REPORT FOR**

**Bank of UniMelb Pty. Ltd. - WEB APPLICATION**

**Report delivered: 04/06/2021**

# Executive Summary

We Test Pens Incorporated has carried out an exhaustive penetration test of web application <http://assignment-zeus.unimelb.life/> at the request of Bank of UniMelb Pty. Ltd.

At the conclusion of the test, five vulnerabilities (and their associated risks) have been uncovered:

1. Sensitive testing / development files / directories disclosure;
2. Bypassing client-side authentication;
3. Privilege escalation;
4. Weak authentication; and
5. Insecure direct object references.

Their risks range in severity from extreme to high, with the most concern falling on the sensitive testing / development files / directories disclosure vulnerability in the Apache directory listing (Finding 1). This extreme-risk vulnerability exposes the entire system to the attacker and might lead to many other uncovered vulnerabilities and reveal system and database credentials, which will result in users’ personal information leakage and potentially direct financial losses.

Another extreme vulnerability is bypassing the client-side authentication in the web application’s developer login functionality (Finding 2). This vulnerability allows the attacker authenticates into the system as the developer and acquire the developer privilege. The attacker could alter the code for his/her own benefit, which might lead to credentials and personal information leakage, financial losses and denial of services.

The first high-risk vulnerability is the privilege escalation in the admin panel (Finding 3). It allows the attacker to promote him/herself to an admin and perform higher privileged actions, which might lead to the system takeover and unauthorised closure of users’ accounts for denial of services. Another high-risk vulnerability is the weak authentication in the password change functionality (Finding 4). It allows the attacker to take over other users’ accounts to steal their personal information and perform unauthorised actions. The attacker could even take over the branch manger’s accounts to perform higher-privileged actions. The last high-risk vulnerability is the insecure object references vulnerability in the user profile page (Finding 5) and it could leak out every user’s personal information.

All these vulnerabilities are fatal to a bank company since users’ trusts are the key for the company to survive in the industrial and these trusts can be easily destroyed and hardily earned back if one of the vulnerabilities ever happens. The slightest possibility of personal data leakage and direct financial losses should never be allowed.

Based on these findings, the website is not secure enough for production release. The high-risk vulnerabilities can cause severe data breach, financial losses and serious damage to the business reputation and they have to be mitigated before release.

Table of Contents

[Executive Summary 2](#_Toc73828715)

[Summary of Findings 5](#_Toc73828716)

[Detailed Findings 6](#_Toc73828717)

[Finding 1 - Sensitive testing / development files / directories disclosure vulnerability presents in the Apache directory listing 6](#_Toc73828718)

[**Description** 6](#_Toc73828719)

[**Proof of Concept** 6](#_Toc73828720)

[**Impact** 6](#_Toc73828721)

[**Likelihood** 6](#_Toc73828722)

[**Risk Rating** 6](#_Toc73828723)

[**References** 6](#_Toc73828724)

[**Recommendation** 6](#_Toc73828725)

[Finding 2 - Client-side authentication bypassing vulnerability presents in the developer login functionality 7](#_Toc73828726)

[**Description** 7](#_Toc73828727)

[**Proof of Concept** 7](#_Toc73828728)

[**Impact** 7](#_Toc73828729)

[**Likelihood** 7](#_Toc73828730)

[**Risk Rating** 7](#_Toc73828731)

[**References** 7](#_Toc73828732)

[**Recommendation** 7](#_Toc73828733)

[Finding 3 - Privilege escalation vulnerability presents in the admin panel 8](#_Toc73828734)

[**Description** 8](#_Toc73828735)

[**Proof of Concept** 8](#_Toc73828736)

[**Impact** 8](#_Toc73828737)

[**Likelihood** 8](#_Toc73828738)

[**Risk Rating** 8](#_Toc73828739)

[**References** 8](#_Toc73828740)

[**Recommendation** 8](#_Toc73828741)

[Finding 4 - Weak authentication vulnerability presents in the password change functionality 9](#_Toc73828742)

[**Description** 9](#_Toc73828743)

[**Proof of Concept** 9](#_Toc73828744)

[**Impact** 9](#_Toc73828745)

[**Likelihood** 9](#_Toc73828746)

[**Risk Rating** 9](#_Toc73828747)

[**References** 9](#_Toc73828748)

[**Recommendation** 9](#_Toc73828749)

[Finding 5 - Insecure direct object references vulnerability presents in the user profile page 10](#_Toc73828750)

[**Description** 10](#_Toc73828751)

[**Proof of Concept** 10](#_Toc73828752)

[**Impact** 10](#_Toc73828753)

[**Likelihood** 10](#_Toc73828754)

[**Risk Rating** 10](#_Toc73828755)

[**References** 10](#_Toc73828756)

[**Recommendation** 10](#_Toc73828757)

[Appendix I - Risk Matrix 11](#_Toc73828758)

[Appendix 2 - Additional Information 12](#_Toc73828759)

# Summary of Findings

A brief summary of all findings appears in the table below, sorted by Risk rating.

|  |  |  |
| --- | --- | --- |
| **Risk** | **Reference** | **Vulnerability** |
| Extreme | Finding 1 | Sensitive testing / development files / directories disclosure vulnerability presents in the Apache directory listing |
| Extreme | Finding 2 | Client-side authentication bypassing vulnerability presents in the developer login functionality |
| High | Finding 3 | Privilege escalation vulnerability presents in the admin panel |
| High | Finding 4 | Weak authentication vulnerability presents in the password change functionality |
| High | Finding 5 | Insecure direct object references vulnerability presents in the user profile page |

# Detailed Findings

## Finding 1 - Sensitive testing / development files / directories disclosure vulnerability presents in the Apache directory listing

|  |  |
| --- | --- |
| **Description** | The **sensitive testing / development files / directories disclosure** vulnerabilitypresents in the Apache directory listing, which reveals the system code and expose everything to the attacker. The attacker could discover other vulnerabilities and exploit them further by inspecting the code. System and database credentials might be leaked as well, which would result in severe personal data breach and might lead to users’ direct financial losses.  The exploitation requires no authentication so the likelihood that it happens significantly increases. |
| **Proof of Concept** | This vulnerability can be exploited by directly inspecting the testing / development files / directories after the attacker manages to brute-force them out. For a detailed walkthrough, see [Appendix 2, Section 1](#_Section_1_–). |
| **Impact** | **Catastrophic:** An attacker could access different versions of all the code files of the system, which will leak out any system and database credentials that exist in the file. By inspecting the code, the attacker could find more uncovered vulnerabilities which could put the security of users’ personal and financial information at higher risk. |
| **Likelihood** | **Almost Certain:** The exploitation of this vulnerability requires no authentication and there is a clear hint pointing to the [test/](http://assignment-zeus.unimelb.life/test/) directory in the comment. It is almost certain that it will be exploited. |
| **Risk Rating** | **Extreme:** Referring to the risk matrix in [Appendix 1](#_Appendix_I_-), catastrophic impact and almost certain likelihood suggest a high risk. The consequence of such exploitation will cause great damage to the business reputation and incur an extreme user churn rate. |
| **References** | [1] <https://www.simplified.guide/apache/disable-directory-listing> |
| **Recommendation** | It is recommended that:   1. Disable the directory listing for testing / development files / directories completely. For example [1], add `.htaccess` to the target directory and add `Options -Indexes` in that file. 2. Remove unnecessary comments. For example, the comment at the bottom of login page is useless but revealing the sensitive information. |

## Finding 2 - Client-side authentication bypassing vulnerability presents in the developer login functionality

|  |  |
| --- | --- |
| **Description** | The **client-side authentication bypassing** vulnerability presents in the **developer login** functionality. Malicious users can acquire developer privilege and alter the system code from the frontend pages for their own benefit, which could lead to system takeover, users’ personal information leakage and financial losses.  The exploitation requires the attacker to have a set of valid credentials so if the admin team does a strict background checking for their customers, the likelihood will decrease to some extent. |
| **Proof of Concept** | This vulnerability can be exploited by first finding out the password for developer login from de-obfuscated JavaScript code. For a detailed walkthrough, see [Appendix 2, Section 2](#_Section_2_–). |
| **Impact** | **Catastrophic:** An attacker could alter the system code for their own benefit, which could lead to system takeover, users’ personal information leakage and financial losses. |
| **Likelihood** | **Likely:** Client-side authentication is a common place for attackers to look at. The exploitation requires the attacker to have a set of valid credentials so strict background checking might decrease the likelihood but it is usually difficult to identify a malicious user. |
| **Risk Rating** | **Extreme:** Referring to the risk matrix in [Appendix 1](#_Appendix_I_-), catastrophic impact and likely likelihood suggest an extreme risk. The consequence of such exploitation will cause great damage to the business reputation and incur an extreme user churn rate. |
| **References** | None |
| **Recommendation** | It is recommended that:   1. Remove developer login entry from normal users’ view; 2. Replace client-side authentication with server-side authentication as what is done for the login page. |

## Finding 3 - Privilege escalation vulnerability presents in the admin panel

|  |  |
| --- | --- |
| **Description** | The **privilege escalation** vulnerability presents in the **admin panel**. Malicious users can promote themselves to admin users hence acquire higher privilege, which could lead to users’ personal information leakage and potentially financial losses.  The exploitation requires the attacker to have a set of valid credentials so if the admin team does a strict background checking for their customers, the likelihood will decrease to some extent. |
| **Proof of Concept** | This vulnerability can be exploited by using the admin panel to promote malicious users to admin users. For a detailed walkthrough, see [Appendix 2, Section 3](#_Section_3_–). |
| **Impact** | **Major:** An attacker could view any sensitive information shown in the admin panel and perform any actions that an admin is allowed to, which could lead to users’ personal information leakage and potentially financial losses. |
| **Likelihood** | **Possible:** Cookie storage is a common place for an attacker to look at. The exploitation requires the attacker to have a set of valid credentials so strict background checking might decrease the likelihood, although it is usually difficult to identify a malicious user. Since the “Promote User” button is not working until some modifications are made to the hidden value, the likelihood that it happens might decrease further. |
| **Risk Rating** | **High:** Referring to the risk matrix in [Appendix 1](#_Appendix_I_-), major impact and possible likelihood suggest a high risk. Privilege escalation should not happen at all in a banking system which usually contains users’ highly sensitive information. |
| **References** | None |
| **Recommendation** | It is recommended that:   1. Remove admin panel entry from normal users’ view; 2. Remove the cookie value which checks if the current user is an admin user and do that on the server side using the user’s session id. |

## Finding 4 - Weak authentication vulnerability presents in the password change functionality

|  |  |
| --- | --- |
| **Description** | The **weak authentication** vulnerability presents in the **password change** functionality. Malicious users can take over other users’ accounts and authenticate into the system as branch managers, which could lead to unauthorised closure of user accounts and personal information leakage.  The exploitation requires the attacker to have a set of valid credentials so if the admin team does a strict background checking for their customers, the likelihood will decrease to some extent. |
| **Proof of Concept** | This vulnerability can be exploited by altering the request body posted to the password change endpoint. For a detailed walkthrough, see [Appendix 2, Section 4](#_Section_4_–). |
| **Impact** | **Major:** An attacker can authenticate into the system as branch managers, which could lead to unauthorised closure of user accounts and personal information leakage. |
| **Likelihood** | **Possible:** Client-side authentication is a common place for attackers to look at. The exploitation requires the attacker to have a set of valid credentials so strict background checking could decrease the likelihood. |
| **Risk Rating** | **High:** Referring to the risk matrix in [Appendix 1](#_Appendix_I_-), major impact and possible likelihood suggest a high risk. Changing password should have strong authentication mechanism in place as the it is often targeted for account takeover. |
| **References** | None |
| **Recommendation** | It is recommended that:   1. Enable multi-factor authentication for changing password and logging in; 2. Make sure the password authentication implemented correctly and ensure the strong password complexity. |

## Finding 5 - Insecure direct object references vulnerability presents in the user profile page

|  |  |
| --- | --- |
| **Description** | The **insecure direct object references** vulnerability presents in the **user profile** page. |
| **Proof of Concept** | <https://github.com/danielmiessler/SecLists/blob/25d4ac4/Discovery/Web-Content/burp-parameter-names.txt> |
| **Impact** | **Negligible/Minor/Moderate/Major/Catastrophic: An attacker could do xyz with this vulnerability** |
| **Likelihood** | **Rare/Unlikely/Possible/Likely/Almost Certain: The possibility of the vulnerability being exploited** |
| **Risk Rating** | **Informational / Low / Medium / High / Extreme**  This is the reasoning behind my rating |
| **References** | **http://example.com** |
| **Recommendation** | The fix goes here |

# Appendix 1 - Risk Matrix

All risks assessed in this report are in line with the ISO31000 Risk Matrix detailed below:



# Appendix 2 - Additional Information

## Section 1 – Sensitive testing / development files / directories disclosure exploitation walkthrough

The following steps provide a proof of concept for an exploitation of this vulnerability to retrieve sensitive information in the form of the flag.

|  |  |
| --- | --- |
| **Step** | **Explanation** |
| 1 | Inspecting [login.php](http://assignment-zeus.unimelb.life/login.php) gives the following comment pointing to [test/](http://assignment-zeus.unimelb.life/test/). |
| 2 | By visiting [test/](http://assignment-zeus.unimelb.life/test/), we know that the directory listing is enabled and it reveals the testing / development files / directories. |
| 3 | In order to acquire more information, we setup [dirbuster](https://gitlab.com/kalilinux/packages/dirbuster) to brute-force files and directories under [test/](http://assignment-zeus.unimelb.life/test/). We used [common.txt](https://github.com/danielmiessler/SecLists/blob/4a2ab64/Discovery/Web-Content/common.txt) as the wordlist. |
| 4 | From the results list of dirbuster, we can find a hidden directory [test/.git/](http://assignment-zeus.unimelb.life/test/.git/). |
| 5 | By walking through every files / directories under [test/.git/](http://assignment-zeus.unimelb.life/test/.git/), we can find the flag is in [test/.git/logs/HEAD](http://assignment-zeus.unimelb.life/test/.git/logs/HEAD).  FLAG{gitters\_R\_us} |

[Return to the main report](#_Proof_of_Concept)

## Section 2 – Client-side authentication bypassing exploitation walkthrough

The following steps provide a proof of concept for an exploitation of this vulnerability to retrieve sensitive information in the form of the flag.

|  |  |
| --- | --- |
| **Step** | **Explanation** |
| 1 | Inspecting [developer-login.php](http://assignment-zeus.unimelb.life/developer-login.php), we find that the authentication of developer login is reside in the client-side and the authentication function is obfuscated with jjencode. |
| 2 | De-obfuscating the code gives us the human-readable password. |
| 3 | Authenticating with that password gives us the flag.  FLAG{b6bad09f13d6dbc00c654321a8bd3fb3} |

[Return to the main report](#_Proof_of_Concept_1)

## Section 3 – Privilege escalation exploitation walkthrough

The following steps provide a proof of concept for an exploitation of this vulnerability to retrieve sensitive information in the form of the flag.

|  |  |
| --- | --- |
| **Step** | **Explanation** |
| 1 | Visiting the admin panel directly prompts us that it is “Unauthorised!”. By inspecting the cookies, we find that there is a cookie which is used by the client-side system to identify if the current user is an admin user. After changing that value to “true”, we are able to view the admin panel. |
| 2 | In the admin panel, two buttons are found. Clicking on the “Reset User” button prompts “Reset successful!” and clicking on the “Promote User” button prompts “Unauthorised!”. By inspecting the admin panel page, we find that there are a hidden field containing a JSON string which controls the user promotion action. |
| 3 | By brute-forcing different combinations of “user” and “roleGroup” values, we find that the response changes to “Prompted!” with “user” being the current user’s username and “roleGroup” being “9”. And back to the dashboard page, we find that the message changes and gives us the flag.  FLAG{zero\_to\_her0}    The brute-forcing steps are detailed in privilege\_escalation.py. |

[Return to the main report](#_Proof_of_Concept_2)

## Section 4 – Weak authentication exploitation walkthrough

|  |  |
| --- | --- |
| **Step** | **Explanation** |
| 1 | Inspecting the request body posted when changing the password, we find three parameters: “old”, “new” and “user”. |
| 2 | If an incorrect “old” password is provided, the response will be “Unauthorised!”. However, if we remove the “old” parameter and only post “new” and “user” parameters, the password will still be changed to the “new” password. Thus, we identify the vulnerability here. |
| 3 | Posting the request with “old” parameter removed and “user” parameter changed to some branch manager’s username also gives us the positive response. |
| 4 | Log into the system with the branch manager’s username and the new password and go to the password change page. We find the flag there.  FLAG{no\_change\_no\_progress} |

[Return to the main report](#_Proof_of_Concept_3)