We Test Pens Incorporated

COMP90074 - Web Security Assignment 3
Jiaxuan Feng
965867

# PENETRATION TEST REPORT FOR Bank of UniMelb Pty. Ltd. - WEB APPLICATION

**Report delivered: 3/6/2021**

## Executive Summary

This report is for vulnerabilities found in website [http://assignment-zeus.unimelb.life/](http://assignment-zeus.unimelb.life/).

We Test Pens Incorporated has carried out an exhaustive penetration test of this web application, and as a result, five vulnerabilities (and their associated risks) have been uncovered. Their severity from high to low is as follows:

1. Authentication weakness leading to account takeover.
2. Privilege escalation.
3. Sensitive files / directories left behind during testing /development.
4. IDOR via a hidden parameter.
5. Bypassing client-side authentication.

Authentication weakness leading to account takeover: In Setting page, a user can change his own password by sending request to the server. However, an attacker can use this functionality to change other users' password by modifying a request sent to the server.

Privilege escalation: In this web application, there is an Admin page which should only be accessed by administrators. But an authenticated normal user can successfully visit this page by changing parameter values of the request sent to the server. After this, this user can also click "Promote" button and modify body parameters in the request to promote anyone as an administrator.

Sensitive files / directories left behind during testing /development: With an automated tool scanning this website, a directory with sensitive files can be found, and it can be accessed by anyone. And in this directory, an attacker can easily find the many sensitive information such as the development log.

IDOR via a hidden parameter: In User Profile page, a user should view his own profile, while when a user request this page from server, there is a hidden parameter in the request. This hidden parameter is to indicate which user's profile will be return, and an attacker can change the value of this parameter to view other users' profile page, and this can lead to information leakage.

Bypassing client-side authentication: In Developer Login page, there is a sensitive information leakage when a user views the source code of this page. An attacker can use the information he get to successfully login as a developer role.

# Table of Contents

Executive Summary 2

Summary of Findings 5

Detailed Findings 6

Finding 1 - Authentication weakness leading to account takeover in Settings page 6

**Description** 6

**Proof of Concept** 6

**Impact** 6

**Likelihood** 6

**Risk Rating** 6

**References** 6

**Recommendation** 6

Finding 2 - Privilege escalation in Admin page 7

**Description** 7

**Proof of Concept** 7

**Impact** 7

**Likelihood** 7

**Risk Rating** 7

**References** 7

**Recommendation** 7

Finding 3 - Sensitive files / directories left behind during testing /development. 8

**Description** 8

**Proof of Concept** 8

**Impact** 8

**Likelihood** 8

**Risk Rating** 8

**References** 8

**Recommendation** 8

Finding 4 - IDOR via a hidden parameter in User Profile page 9

**Description** 9

**Proof of Concept** 9

**Impact** 9

**Likelihood** 9

**Risk Rating** 9

**References** 9
**Recommendation** 9
Finding 5 - Bypassing client-side authentication in Developer Login page 10
**Description** 10
**Proof of Concept** 10
**Impact** 10
**Likelihood** 10
**Risk Rating** 10
**References** 10
**Recommendation** 10
Appendix I - Risk Matrix 11
Appendix 2 - Additional Information 12
Finding 5 Bypassing client-side authentication in Developer Login page 12

## Summary of Findings

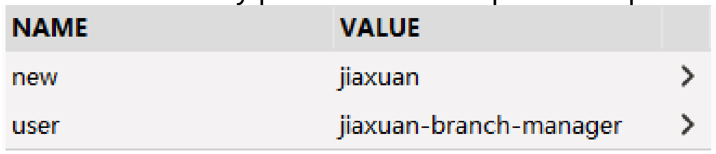
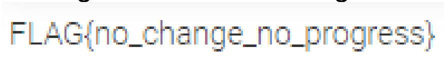
A brief summary of all findings appears in the table below, sorted by Risk rating.

| **Risk** | **Reference** | **Vulnerability** |
|---|---|---|
| Extreme | Finding 1 | Authentication weakness leading to account takeover in Settings page, change password functionality. |
| Extreme | Finding 2 | Privilege escalation in Admin page Promote user functionality. |
| Extreme | Finding 3 | Sensitive files / directories left behind during testing /development. |
| High | Finding 4 | IDOR via a hidden parameter in User Profile page. |
| Medium | Finding 5 | Bypassing client-side authentication in Developer Login page. |
| | | |
| | | |
| | | |
| | | |

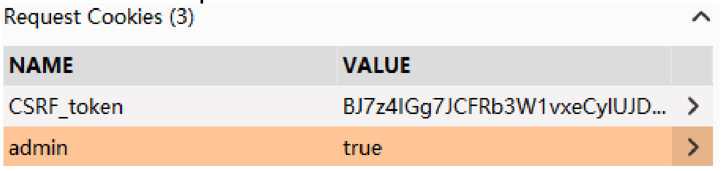
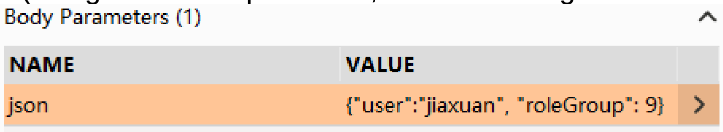
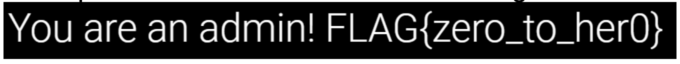
## Detailed Findings

This section provides detailed descriptions of all the vulnerabilities identified.

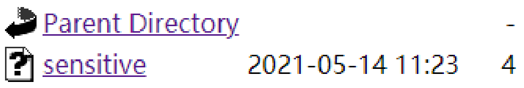
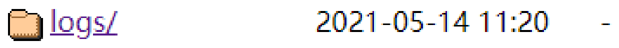
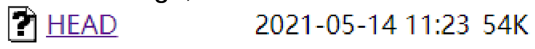
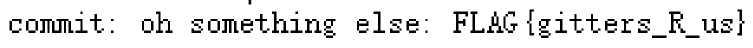
### Finding 1 - Authentication weakness leading to account takeover in Settings page

| | |
|---|---|
| **Description** | In this web application, users can change their own password in settings page. The client side will send three parameters to server to achieve this functionality. An attacker can use it to change other users' password by modifying these parameters. |
| **Proof of Concept** | Step1: Log into the web application, click the right-side settings button. Open Burp Suite and proxy.<br>Step2: Input anything in both current and new password input boxes. Click "SAVE CHANGES" to send a request.<br>Step3: In Burp Suite, modify this request. Change body parameters: Remove "old" parameter. Change the value of "new" as your new password. Change the value of "user" as your branch manager, then forward this request. The screenshot of body parameters in Burp Suite inspector:<br><br>NAME / VALUE<br>new / jiaxuan<br>user / jiaxuan-branch-manager<br>Step4: Use the new password of Step3 to login to your branch manager account, and flag will show in settings page:<br><br>FLAG{no_change_no_progress} |
| **Impact** | **Catastrophic:** With this vulnerability, an attacker can take over other users' account, even a branch manager. Due to this web application is for a bank, the impact should be Catastrophic. |
| **Likelihood** | **Possible:** If an attacker wants to exploit this vulnerability, he first needs a valid account to login this website, then he also needs to guess out the username of other users. Thus, I think it should be possible. |
| **Risk Rating** | **Extreme:** This vulnerability's impact is catastrophic, and likelihood is possible, according to ISO31000 Risk Matrix (Appendix 1), the risk rating should be extreme. |
| **References** | [1] Lecture 16 – In-Depth Authentication Testing slides |
| **Recommendation** | The server needs to force to match the old password. Assign a default value to the parameter "old". If this parameter is removed in a request, the server will handle it as the default value, and it will not match with correct password. |

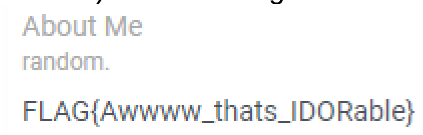
## Finding 2 - Privilege escalation in Admin page

| | |
|---|---|
| **Description** | In this website, the admin page should only be accessed by administrators. In the request sent to the server, a Boolean parameter "admin" is used to distinguish role of users. But a general user can change its value and thus visit the admin page successfully, and can use admin's functionality "promote" to make him an admin. |
| **Proof of Concept** | Step1: Login to the website, then open Burp Suite and proxy.<br>Step2: Click the Admin button on the left side of the web to send a request, then use Burp Suite to modify this request. In Request Cookies, change the value of admin as true, and forward this request:<br><br>Request Cookies (3)<br>NAME / VALUE<br>CSRF_token / BJ7z4IGg7JCFRb3W1vxeCyIUJD...<br>admin / true<br>Step3: Click "PROMOTE USER" button and use Burp Suite to change the request: In Request Cookies, change the value of admin as true same as step2. In body parameters, modify the value of json: replace admin with your username, replace 1 with 9 (9 is get from Burp intruder, the return length is different):<br><br>Body Parameters (1)<br>NAME / VALUE<br>json / {"user":"jiaxuan", "roleGroup": 9}<br>Then forward this request, the webpage shows my account has been promoted, and we can find the flag in dashboard page:<br><br>You are an admin! FLAG{zero_to_her0} |
| **Impact** | **Catastrophic:** With this vulnerability, an attacker can promote his own account as an admin. An admin has higher privileges, may be able to assign role to users and view users' sensitive information, such as password. It is very serious for a bank application, so the impact should be catastrophic. |
| **Likelihood** | **Possible:** This vulnerability is not very easy to find. An attacker needs brute force to get the role group of his account. Besides, an attacker must have a valid account of this website before, so the likelihood should be possible. |
| **Risk Rating** | **Extreme:** This vulnerability's impact is catastrophic, and likelihood is possible, according to ISO31000 Risk Matrix (Appendix 1), the risk rating is extreme. |
| **References** | [1] Lecture 16 – In-Depth Authentication Testing slides |
| **Recommendation** | Remove parameter "admin" in the request. Let the server to determine if a user is an administrator by his username, then return different response. |

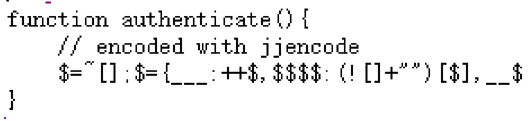
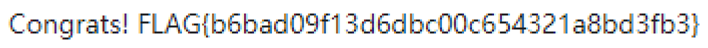
## Finding 3 - Sensitive files / directories left behind during testing /development.

| | |
|---|---|
| **Description** | During testing/development, some files record the log. And these files may contain sensitive information and can be gained by an attacker. Using dirbuster, we can explore the structure of a website and find such files or directories. |
| **Proof of Concept** | Step1: Use dirbuster and its own word lists to automatically scan this website: http://assignment-zeus.unimelb.life/ . From the result, there are a directory and a file with a 200 response: /test/sensitive<br>Step2: Visit this directory using URL: http://assignment-zeus.unimelb.life/test/<br><br>Parent Directory -<br>sensitive 2021-05-14 11:23 4<br>Step3: Click the sensitive file, and the webpage shows 250.<br>Step4: Because the sensitive file has been changed, we need to find the change log. According to GitHub, visits URL: http://assignment-zeus.unimelb.life/test/.git ,(.git directory is usually used by git directory to record logs). Then we can find logs directory there:<br><br>logs/ 2021-05-14 11:20 -<br>Click into logs, there is a HEAD file:<br><br>HEAD 2021-05-14 11:23 54K<br>Step5: Click into HEAD file, and the flag can be found in commit comments part:<br><br>`commit: oh something else: FLAG{gitters_R_us}` |
| **Impact** | **Major:** With this vulnerability, and attacker can explore the history of the web development. If the developer made mistakes and leak sensitive information in previous version, the attacker can also find it even if the developer has modified it. Thus, I think it should be major. |
| **Likelihood** | **Almost Certain:** An attacker does not need a valid account to exploit this vulnerability, and most automated tools can find this sensitive directory. Besides, the directory .git is also very common. Thus, I think it is almost certain. |
| **Risk Rating** | **Extreme:** This vulnerability's impact is major, and likelihood is possible, according to ISO31000 Risk Matrix (Appendix 1), the risk rating should be extreme. |
| **References** | [1] https://gitlab.com/kalilinux/packages/dirbuster |
| **Recommendation** | The server should change the access permissions of this sensitive directory. Only developers can access it. If other users try to do that, the server should return unauthorised. |

## Finding 4 - IDOR via a hidden parameter in User Profile page

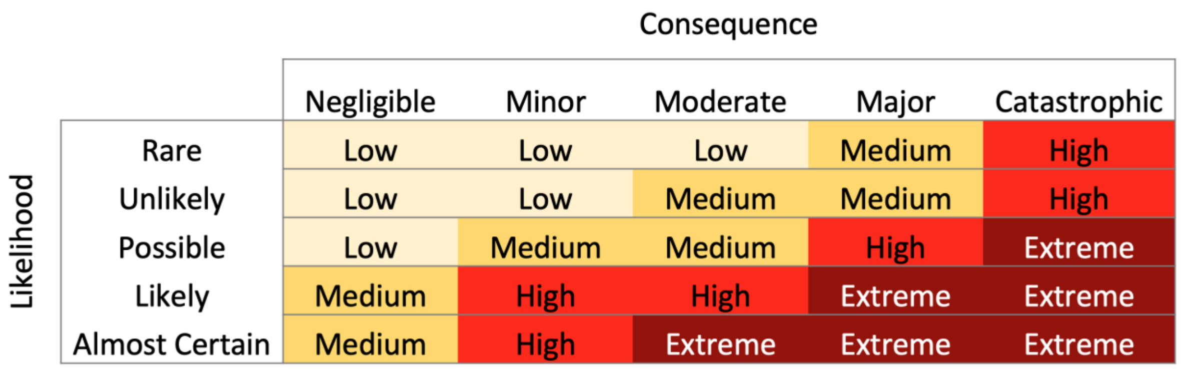
| | |
|---|---|
| **Description** | When an authenticated user visits User Profile page, the server should return the page about this certain user. While it is achieved by a hidden parameter "id". If it is not assigned any value, the page will return normally, but if a user adds a "id" after the URL and assign different values, the server will return profile of other users whose id matches successfully. |
| **Proof of Concept** | Step1: Log into the website http://assignment-zeus.unimelb.life/, and click the User Profile button on the left of the webpage.<br>Step2: Add "?id=333" after the URL, and enter, the final URL is http://assignment-zeus.unimelb.life/profile.php?id=333. ("id" is the most common parameter for users, and the value 333 is get from Burp intruder, the return length is different from other values.) Then the flag is found in the returned webpage:<br><br>About Me<br>random.<br>FLAG{Awwww_thats_IDORable} |
| **Impact** | **Moderate:** With this vulnerability, an attacker can view other users' profile page. However, there are not much information shown in this page, and this information is also not very sensitive. Thus, I think it should be moderate. |
| **Likelihood** | **Likely:** Only authenticated users can use User Profile functionality, but for a bank application, most people can apply for a valid account. Thus, I think the likelihood should be likely. |
| **Risk Rating** | **High:** This vulnerability's impact is moderate, and likelihood is likely, according to ISO31000 Risk Matrix (Appendix 1), the risk rating should be medium. |
| **References** | [1] Lecture 20 – IDOR and Method of Testing slides |
| **Recommendation** | When a user sends a request to the server, use cookies to verify this user's credential, to figure out who sends out this request, and then return a match user profile. |

## Finding 5 - Bypassing client-side authentication in Developer Login page

| | |
|---|---|
| **Description** | In Developer Login page, general users can view the source code and easily find a sensitive part which is encoded with jjencode, and it can be decoded by some online decoder very conveniently, then an attacker can use the information he gets form the decoded string to bypass the developer login authentication, and be regarded as a developer. |
| **Proof of Concept** | Step1: Log into the website http://assignment-zeus.unimelb.life/ and click the developer login button on the left side of the webpage.<br>Step2: View the page source code of Developer Login page, and can find the part of jjencode from line 58, the code is too long, so the screenshot is only a small part for indicating the position:<br><br>`function authenticate(){`<br>`    // encoded with jjencode`<br>`    $=~[];$={___:++$,$$$$:(![]+"")[$],__$`<br>`}`<br>Step3: Use an online decoder: https://lelinhtinh.github.io/de4js/. Copy all the part of jjencode code into the website, and click auto decode button, the full screenshot of the decode can be found in Appendix 2, Finding 5.<br>Step4: Copy the value get from last step, paste into the input box of developer page and click log in button, the value is: ashdfh2i3uh8f9erhf98h234f8ghw79ghr8egyh98hern98gh89j2w48fj403wofj<br>And the flag is popped out:<br><br>Congrats! FLAG{b6bad09f13d6dbc00c654321a8bd3fb3} |
| **Impact** | **Moderate:** Due to this vulnerability, an authenticated general user can login as developer. However, in this web application, there are not many rights for a developer role, so I think it should be moderate. |
| **Likelihood** | **Possible:** This vulnerability is obvious and very easy to find, but can only be exploited by an authenticated user, so the likelihood should be possible. |
| **Risk Rating** | **Medium:** This vulnerability's impact is moderate, and likelihood is possible, according to ISO31000 Risk Matrix (Appendix 1), the risk rating should be medium. |
| **References** | [1] https://lelinhtinh.github.io/de4js/ |
| **Recommendation** | The password of developer has been very complex, so it is enough to hide the function which include jjencode in users' browser. Only use submit form function in user end, and let the server to verify the password. |

## Appendix I - Risk Matrix

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

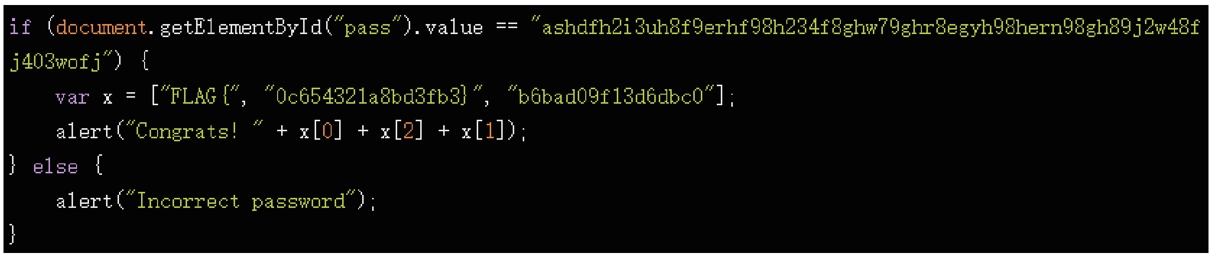


|  |  | Consequence |  |  |  |  |
|---|---|---|---|---|---|---|
|  |  | Negligible | Minor | Moderate | Major | Catastrophic |
| Likelihood | Rare | Low | Low | Low | Medium | High |
|  | Unlikely | Low | Low | Medium | Medium | High |
|  | Possible | Low | Medium | Medium | High | Extreme |
|  | Likely | Medium | High | High | Extreme | Extreme |
|  | Almost Certain | Medium | High | Extreme | Extreme | Extreme |

# Appendix 2 - Additional Information

## Finding 5 Bypassing client-side authentication in Developer Login page

Step3 screenshot:



```
if (document.getElementById("pass").value == "ashdfh2i3uh8f9erhf98h234f8ghw79ghr8egyh98hern98gh89j2w48f
j403wofj") {
    var x = ["FLAG{", "0c654321a8bd3fb3}", "b6bad09f13d6dbc0"];
    alert("Congrats! " + x[0] + x[2] + x[1]);
} else {
    alert("Incorrect password");
}
```