We Test Pens Incorporated

COMP90074 - Web Security Assignment

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

Report delivered: 04/06/2023

Executive Summary

This report will cover penetration tests performed by “We Test Pens Incorporated” (the attacker) on bank of UniMelb Pty. Ltd. which will be referred as “the organisation” throughout this report and will cover five vulnerabilities, exploitations, and a risk assessment based on the ISO31000 Risk Matrix (See appendix 1). It is the organisation’s risk appetite that can determine final decisions for any further action regarding any of these issues.
Two user credentials are provided for the purpose of testing as a user and the branch manager. Additionally, the scope of work is provided as “http://assignment-plutus.unimelb.life/”. Bank of UniMelb provides users a self-service banking experience and is limited in time and budget and all pen tests are performed with care on production. Tools utilised by the attacker include python codes and Burb’s Intruder (https://portswigger.net/burp).

Overall assessment
Four vulnerabilities are detected as ‘Extreme risk’ while one vulnerability is assessed as High risk. Potential harm can be catastrophic and for most of the cases the possibility of occurrence is likely, and this report will allow for further explanation in each section.
The vulnerability of the highest concern has been identified as the ‘privilege escalation’ since the attacker can perform any administrative functionality in the banking system which is extremely dangerous, while the least concerning vulnerability is considered to be the ‘authentication bypass’, since the developer dashboard has not been compromised by the attacker.

Out Of scope
1. Insufficient password policies
2. Sensitive information over HTTP
3. Directory listing enabled
4. Lack of rate limiting on the login page
5. Server-status exposed

Scope of work
Following is a list of all vulnerabilities found and will discuss further in this report.

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

Table of Contents

Summary of Findings

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

Risk	Reference	Vulnerability
Extreme	Privilege escalation	Privilege escalation to admin role in admin page
Extreme	Authentication weakness	Account takeover in Change password functionality
Extreme	IDOR	Found in accepting an unused parameter in profile page
Extreme	Sensitive files / directories	Information disclosure found in /.Git directory
High	Authentication Bypass	Found in developer login page.

Detailed Findings

This section provides detailed descriptions of all the vulnerabilities identified.

Privilege escalation - Privilege escalation to admin role in admin page

Description	For vertical privilege escalation (or privilege elevation) the attacker will attempt to gain access to accounts with higher privileges. In this case the attacker targeted the administrator privilege. Once a privileged account is accessed, it can be used to invade other accounts. **Risk statement:** Unauthorized privilege escalation may occur due to incorrect access control settings in the application, which leads to gaining access to functionalities available specifically for “administrator” role. As a user, the attacker could modify the value of a cookie stored in browser and changed its access to admin role to be able to perform admin functionalities.
Proof of Concept	Steps to produce the result are below and a proof of concept with a flag is attached in the appendix 2. **(Figure 1: FLAG** Privilege escalation**)** 1. Change admin cookie, a Boolean value to “true” from its default “false” value after logging in to dashboard. This will allow access to Admin page (/admin.php) and can be done via browser cookie jar or request headers. 2. In the admin page, a hidden input found in a POST request to “/assigns.php” actioned on a button click named “promote user”. 3. A POST parameter’ value found with: {"user":"admin", "roleGroup": 1} on action of the button. To promote current user to admin, attacker could use username and a variety of roleGroup numbers to test outcome. 4. Finally, passing {"user":"avatandoust", "roleGroup": 9}, as a json in the post parameters, responded by “promoted!” instead of “unauthorised” and resulted in privilege escalation for the current user (avatandousts) **(Figure 6: privilege escalation)**

Consequence	An attacker could access the admin page of the UniMelb bank web application and promote themselves to admin. The consequences may vary on other functionalities available to admin role. In this case Consequences considered "**major**". Functionalities such as, edit, add, delete any branch information including branch managers or create and delete of bank accounts.
Likelihood	The possibility for attack is considered "**Likely**" and the reason is that the bank is utilising cookies to control users' privilege in an unsafe way where it can be exploited by an attacker changing the cookies.
Risk Rating	**Extreme** The administrator role in a banking application is considered to have a high-level of access. Users' private information can be leaked, edited, and/or removed. Bank accounts and their entities have access to important personal/private information, in which if breached can be extremely dangerous.
Recommendation	- A sample code that can be used for this scenario is provided in Appendix 2 including further discussions. (Figure 7 Privilege escalation suggestion: Using Cookie Authentication, utilizing a secret word and hash algorithm like MD5 will improve chance of exposing admin:) - Use Single sign-on (SSO) services can be useful to manage access controls. - Presenting less data to end users – for instance admin page in dashboard to be inaccessible. - Prevent normal users to promote other users.
References	https://www.crowdstrike.com/cybersecurity-101/privilege-escalation/ https://portswigger.net/web-security/access-control https://docstore.mik.ua/orelly/webprog/pcook/ch08_11.htm

Authentication weakness – Result in account takeover

Description	An account takeover via changing users' passwords with no permission by the attacker if given a username. In this case the attacker, as a normal user, could change the password of the branch manager. This vulnerability is in change password functionality in settings. **Risk statement:** Account takes over can occur due to a lack in filtering unauthorised requests and not isolating functionalities which leads to changing any user's password to a new requested phrase by attacker and eventually taking over an account username and password credentials.
Proof of Concept	Steps to produce the result are below and a proof of concept with a flag is attached in the appendix 2. **(Figure 2: FLAG** Authentication weakness leading to account takeover**)** 1. As a result of monitoring, change password functionality in "settings" revealed that a user can change their password without providing the old password. 2. Knowing the branch manager username and using it in the Post request to "/change-password.php" by removing "old" field parameter, which is the old password of the user, allowed the password for the branch manager username change successfully. For this, using Burp Suite repeat functionality the attacker removed old parameter and only passed new and username.
Consequence	An attacker could change the password of a given user without knowing their current password. This would incur **major** consequences such as information disclosure and leakage in a banking system.
Likelihood	The occurrence of this attack is considered **likely**, knowing this system will be available to any bank account owner.
Risk Rating	**Extreme** This vulnerability will result in account takeovers, given any username.
Recommendation	- Back-end check for a user to only be able to change own passwords. - SSO and VPN for inside organisation access controls (branch managers and bank employees) - Multi vector authentications by sending a user SMS or Email and ask for acknowledgment over changing the password.
References	https://perception-point.io/guides/cybersecurity/understanding-account-takeover-ato-9-defensive-measures/

Insecure direct object references (IDOR) – Found in accepting an unused parameter in profile page.

Description	This vulnerability is a type of access control to use user-supplied inputs to access objects directly. It was discovered in a hidden variable in profile URL. User ID is used in this URL to access users' information. **Risk statement:** IDOR on obtaining objects may occur dure to lack in controlling parameters and user access controls which leads to information leaks and data breaches of bank users' details and information.
Proof of Concept	Steps to produce the result are below and a proof of concept with a flag attached in the appendix 2**. (Figure 5: FLAG** IDOR via hidden parameter**)** 1. Manual testing on different pages with known parameters such as, user, id, etc. since Disburser dictionary could not find any interesting path for attacker. 2. After witnessing changes in user information populating in profile page, a python code written to automatically detect a specific information in response contents (in this case searched for FLAG word using "find" functionality) 3. To use the code, run idor.py provided with report. Provide username and password and allow it to request login and get credentials sessions for further requests. 4. The code will show an ID when the word is found inside content.
Consequence	An attacker could access other users' information and reveal their data in profile page. The consequence of this attack is considered **catastrophic** considering other attacks and how the attacker can overtake all user accounts.
Likelihood	Possibility of this to occur is **likely** and requires scanning URLs and a logged in user.
Risk Rating	**Extreme** The reason is that the attacker can access data for all users based on their ID and can brute force ID numbers to cause further problems.
Recommendation	- Disable or remove functionalities are not used in backend. - Make client-side inputs clear and strict. In this case make sure the id of the requested user is equal to the id of the attacker(requester).
References	https://portswigger.net/web-security/access-control/idor

Sensitive files/directories – information disclosure in /.Git directory

Description	This vulnerability revealed git directories and files including the version control history (left behind since testing in production). All version control data are stored in "/.git" folder. This vulnerability was found by scanning the directories and comments hinting to the exploitation. Dirburster tool and dictionaries were used for scanning the paths however the vulnerability was found using a manual test. Initial path was recognised in a commented section in login page. **Risk statement:** Sensitive files and directories exposure may occur due to not removing the test files and development/test environments after moving to production phase which leads to information disclosure about the server, organisation, and application.
Proof of Concept	Steps to produce the result are below and a proof of concept with a flag is attached in the appendix 2. (Figure 4: FLAG Sensitive files / directories left behind during testing/development.) 1. After recognising /test/ path and understanding a path to a sensitive file with a 250 written inside and as the attacker we suspected this path. 2. Upon more investigation using scanners and similar articles on this, test/.git path tested and result was recoded. Going through files and folders gave more information about branches, commits and versions.
Consequence	An attacker could access git version information through test/.git path and the consequence differs depends on the information saved during testing time on git and is considered **moderate**.
Likelihood	Possibility for this attack to occur is **almost certain**, since it is happening under public URL in login page with no login credential required.
Risk Rating	**Extreme** The reason for this decision is that Git contains important information and revealing its information can problematic, but the underlying data is not usually in stored in git.
Recommendation	- Preventing user to scrawl in websites path using web application firewalls or request controls. This can be done either in back end or using ".htaccess" to disable directories. - Removing unnecessary comments that may reveal any sensitive data (in this case, data about the structure of the site and directories – comment in login page)
References	https://portswigger.net/web-security/information-disclosure/exploiting

Authentication Bypass – Found in developer's login page.

Description	This vulnerability is bypassed by decoding JJencode for a JavaScript function in the developer login page. This might be left over after development and caused by transforming to production too quickly hence the actual functionality is not available anymore. **Risk assessment:** Developer's login authentication may be bypassed due to availability in a public page and not limiting its visuality to developers which may lead to logging in as a developer and accessing functionalities only available to them.
Proof of Concept	Steps to produce the result are below and a proof of concept with a flag is attached in the appendix 2. **(Figure 3: FLAG** Bypassing client-side authentication**)** 1- An encoded JavaScript function (named authenticate) found in developer-login page, followed by a comment that explained its encoded with JJ-encoder. 2- Using a decoder to understand the function, using "53lu" online tool, removing extra spacing. 3- Decoder provided a password check in client side based on user input. Using this password, an alert shown, and a flag discovered, with a congrats message for logging in successfully.
Consequence	An attacker could access an encoded JavaScript function in client side. The consequence can differ on developer accounts access and functionality, but assuming the functionality is limited and are including tests because of alert box appearing instead of seeing more developer applications, consequence is **minor**.
Likelihood	This function is available to all users and **likely** to be seen and decrypted via any online decryption tool.
Risk Rating	**High** The reason is that the possibility of revealing this function to users is high considering the consequences above.
Recommendation	- Similar check functionality for developers as for privilege escalation code in appendix 2. - Reduce visibility of developer login page in dashboard to developers only. Removing the page and making a separate dashboard. - Using SSO for sign in and provide developers with appropriate credentials.
References	https://www.53lu.com/tool/jjencode/

Appendix I - Risk Matrix

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

Likelihood	Consequence				
	Negligible	Minor	Moderate	Major	Catastrophic
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

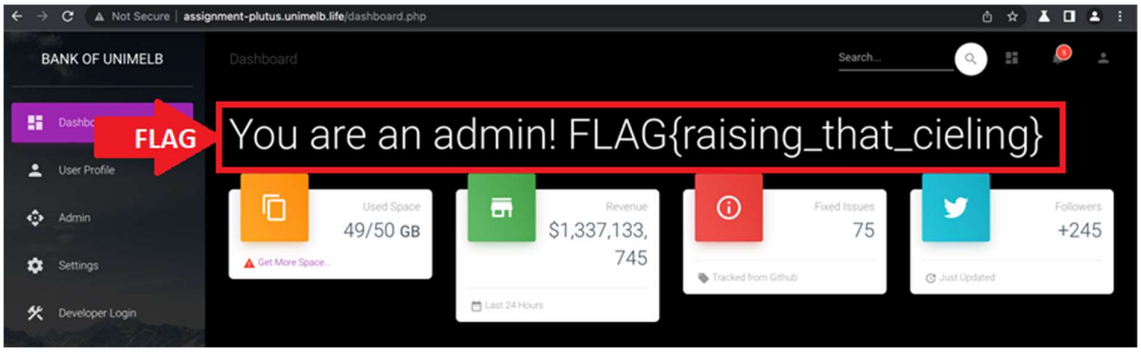

Figure 1: FLAG Privilege escalation

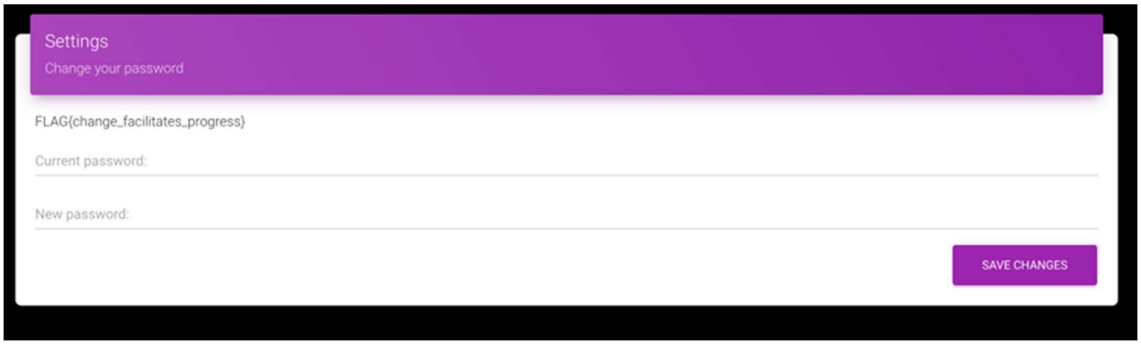

Figure 2: FLAG Authentication weakness leading to account takeover

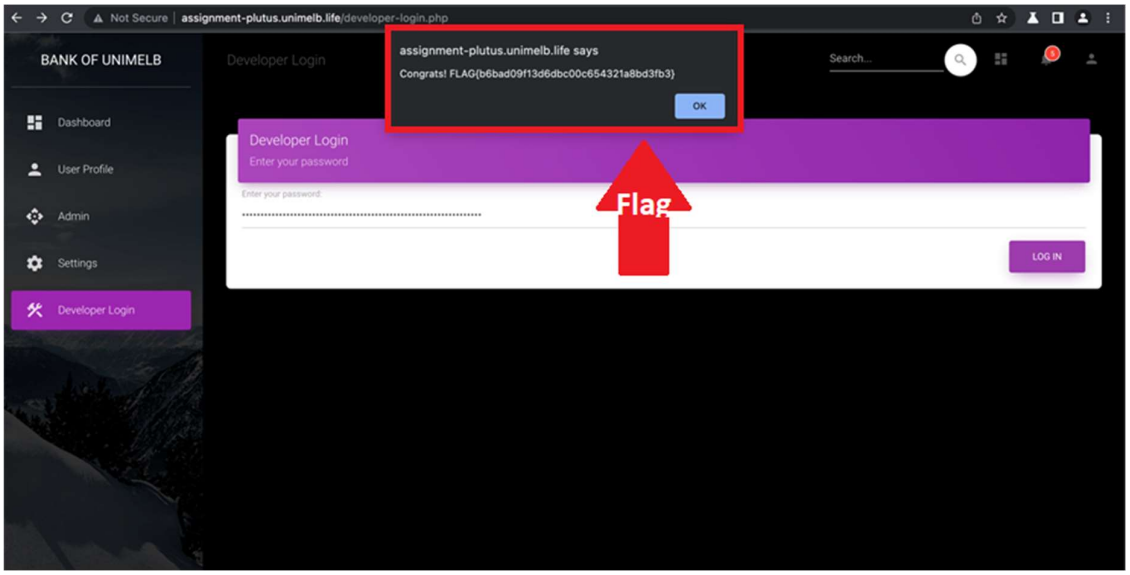

Figure 3: FLAG Bypassing client-side authentication

Flag: gitters_R_us

Figure 4: FLAG Sensitive files / directories left behind during testing/development.

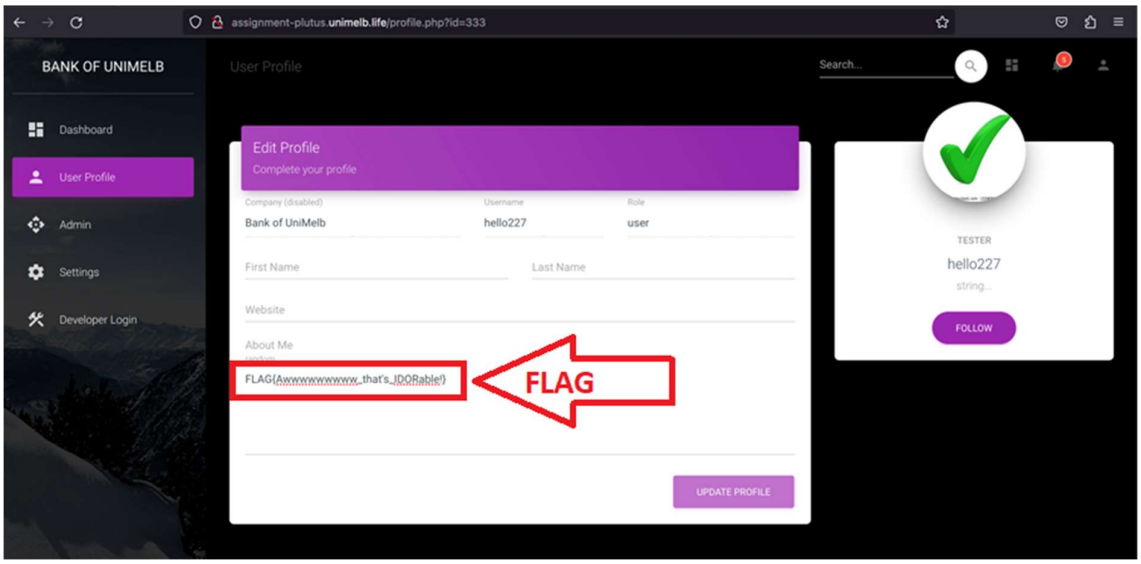

Figure 5: FLAG IDOR via hidden parameter

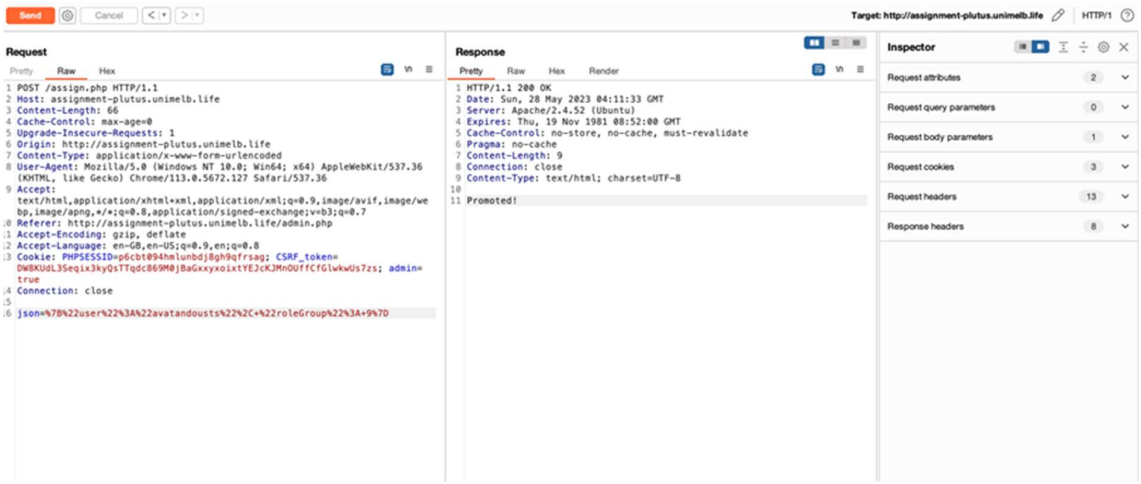

Figure 6: privilege escalation

PHP sample code:

```php
<?php
$secret_word_forAdmin = 'for the admin!;
$secret_word_forUser = 'for the users!;

if (pc_validate($_REQUEST['username'], $_REQUEST['password'])) {
    //seccuessfully login
if(user_role==admin) {
setcookie(isadmin, $_REQUEST['username'].','.md5($_REQUEST['username'].
    $secret_word_forAdmin));
}else {
setcookie(user, $_REQUEST['username'].','.md5($_REQUEST['username'].
    $secret_word_forUser));
}
}

?>
```

Figure 7 Privilege escalation suggestion: Using Cookie Authentication, utilizing a secret word and hash algorithm like MD5 will improve chance of exposing admin: