# TryHackMe

Julien ROYON CHALENDARD

CTF :
OWASP Top 10

Catégorie :
Web

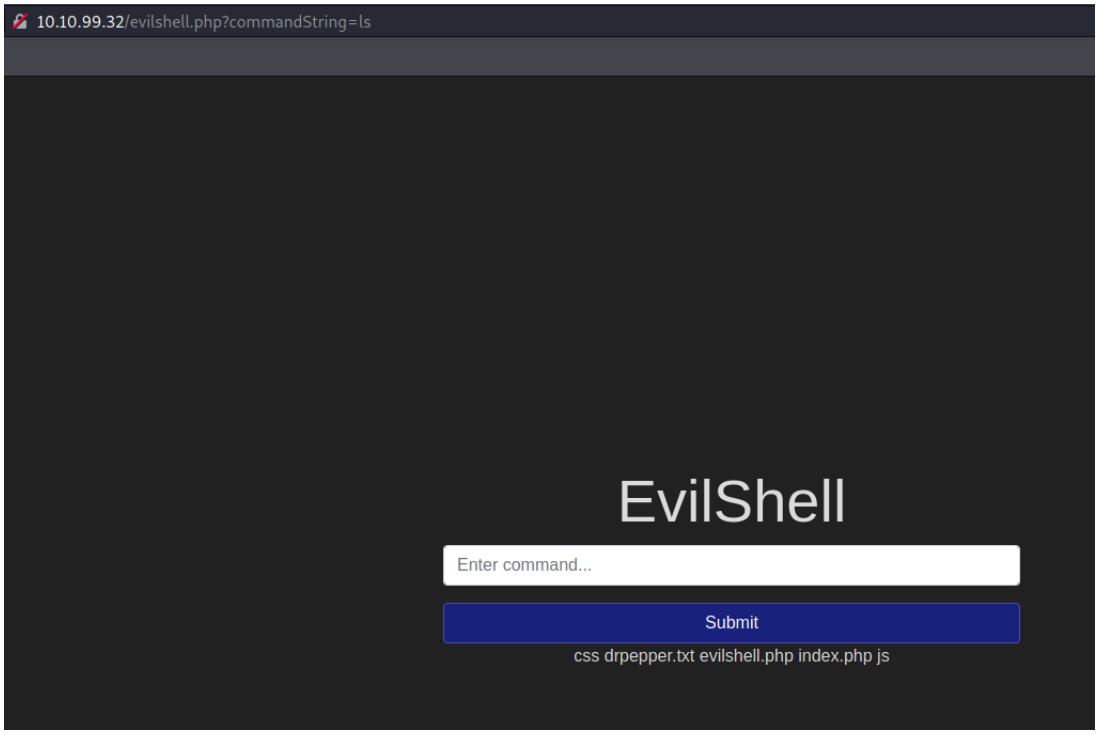
2020

# Sommaire

# 1 [Severity 1] Command Injection Practical

## 1.1 What strange text file is in the website root directory ?

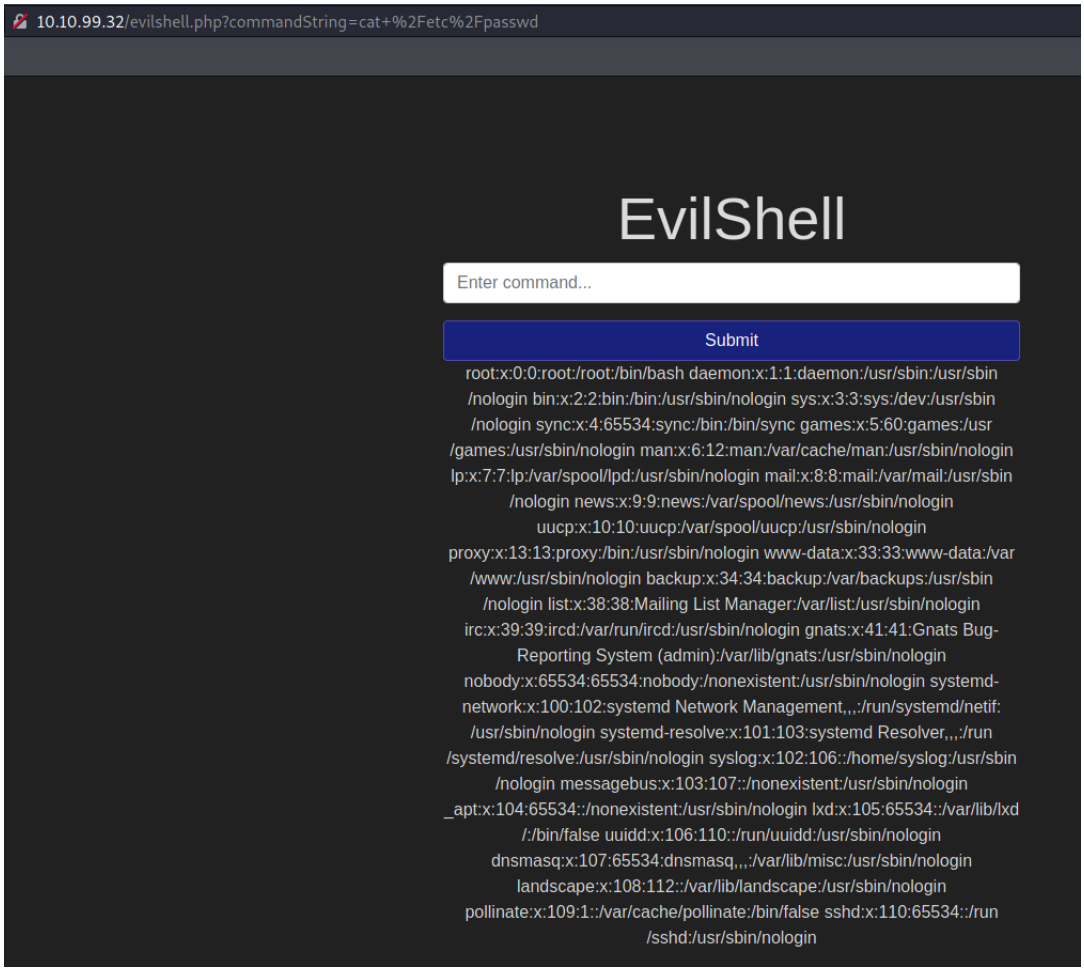
Pour cela, on utilisera la commande "ls" pour voir les fichiers à la racine de l'application web




Le fichier étrange est : drpepper.txt

## 1.2 How many non-root/non-service/non-daemon users are there ?

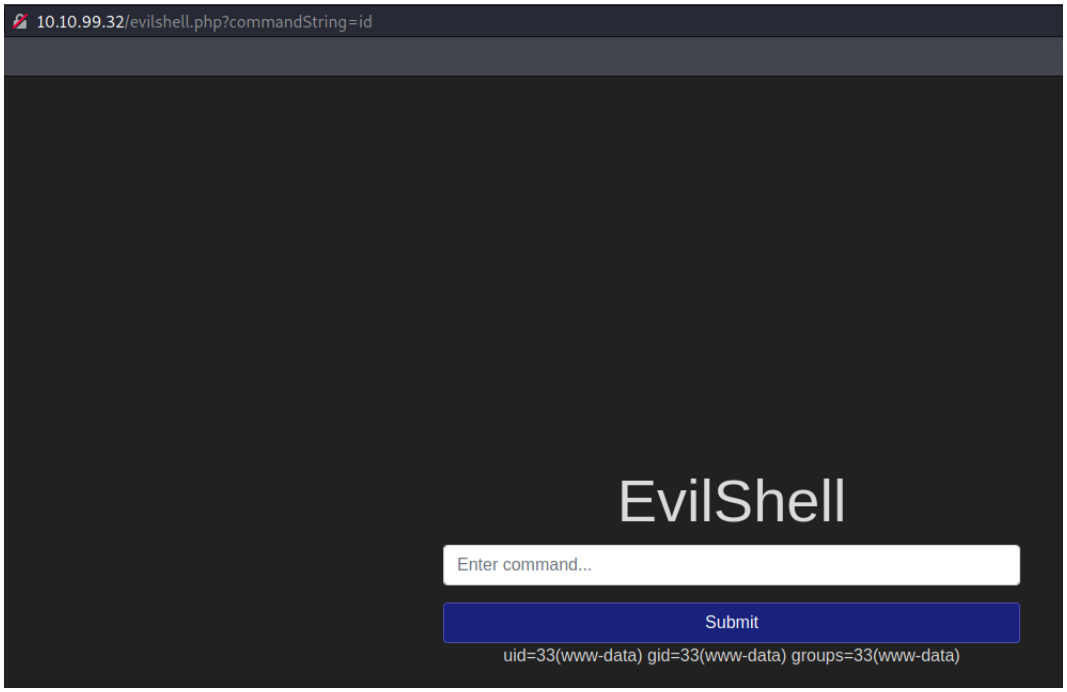
Pour voir les utilisateurs enregistrés dans le système, on peut aller voir le fichier passwd




Les utilisateurs non service ont un id qui commence par 1000. Dans notre cas, il n'y a aucun utilisateur avec 1000 en id. Donc il y en a 0

## 1.3 What user is this app running as ?

On utilisera la commande "id" ou "whoami" pour voir ça



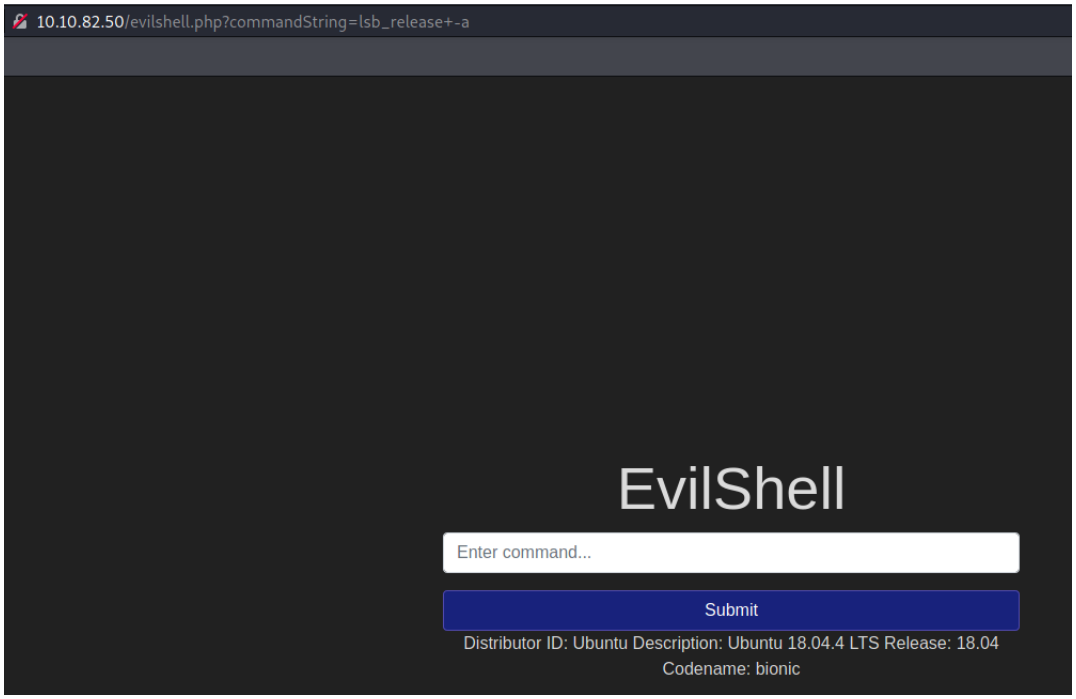

Nous interagissons avec l'utilisateur www-data

## 1.4 What is the user's shell set as ?

Pour voir cela, on peut aussi se baser sur la capture d'écran du fichier passwd. En regardant l'utilisateur www-data, on voit "/usr/sbin/nologin"

## 1.5 What version of Ubuntu is running ?

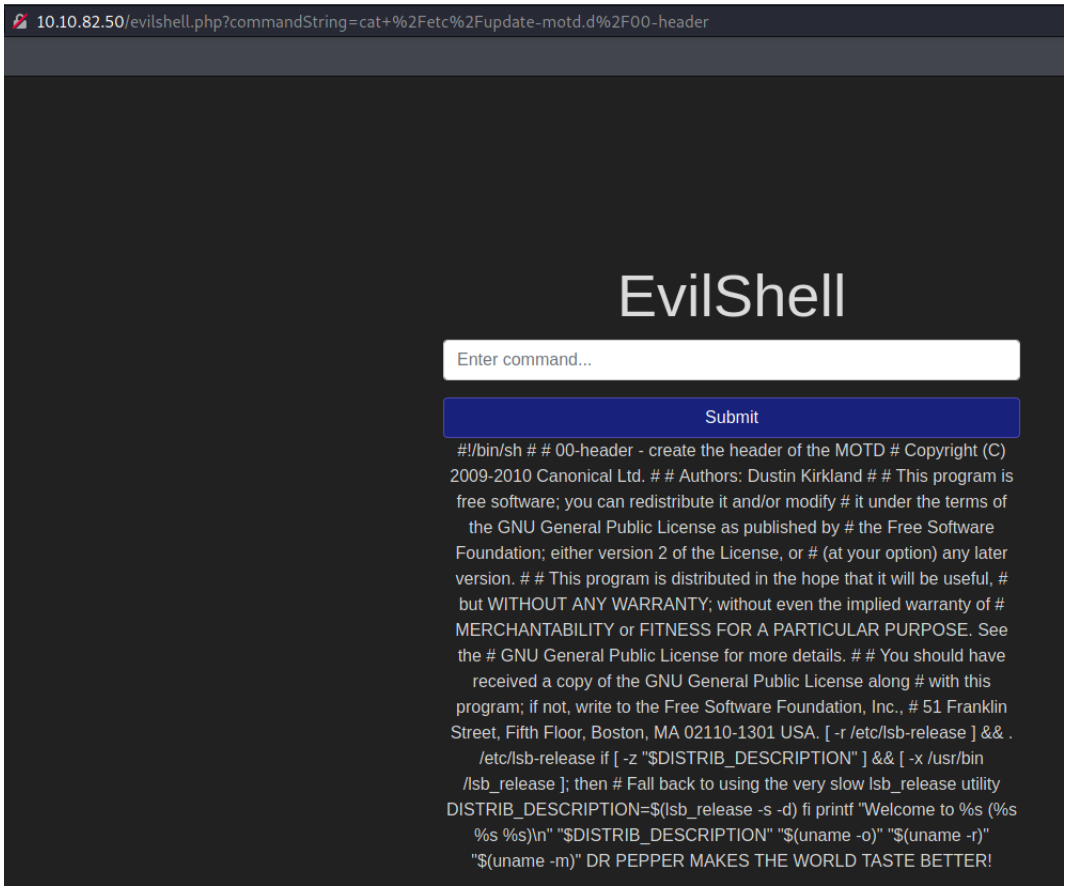
On utilisera la commande lsb_release -a pour avoir des informations sur l'OS




Ubuntu est a la version 18.04.4

## 1.6 Print out the MOTD. What favorite beverage is shown ?

On utilisera la commande "cat /etc/update-motd.d/00-header"

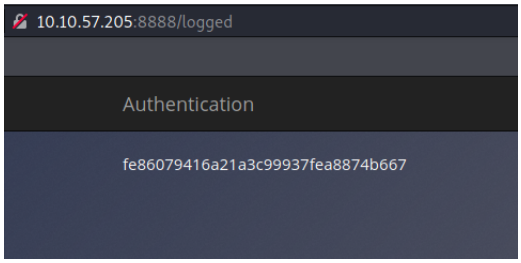



La boisson préférée est : Dr Pepper

# 2 [Severity 2] Broken Authentication Practical

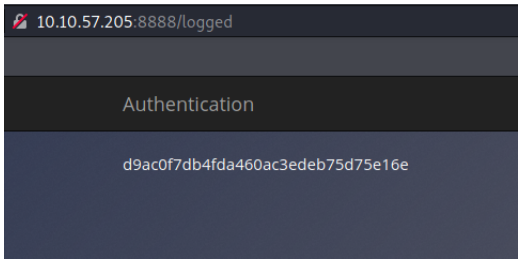
## 2.1 What is the flag that you found in darren's account ?

On suit le tutoriel, on crée un compte en mettant en espace avec darren. Lors de la connexion on mettant juste darren comme pseudo, on pourra utiliser notre mot de passe et se connecter au compte de darren.



## 2.2 What is the flag that you found in arthur's account ?

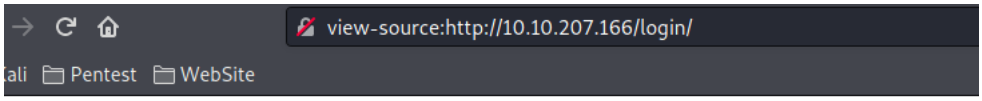
On procède de la même façon pour arthur




# 3 [Severity 3] Sensitive Data Exposure (Challenge)

## 3.1 What is the name of the mentioned directory ?

On parcourt l'application web et on regarde le code source de chaque page. C'est dans la page /login que l'on trouve quelque chose d'intéressant.

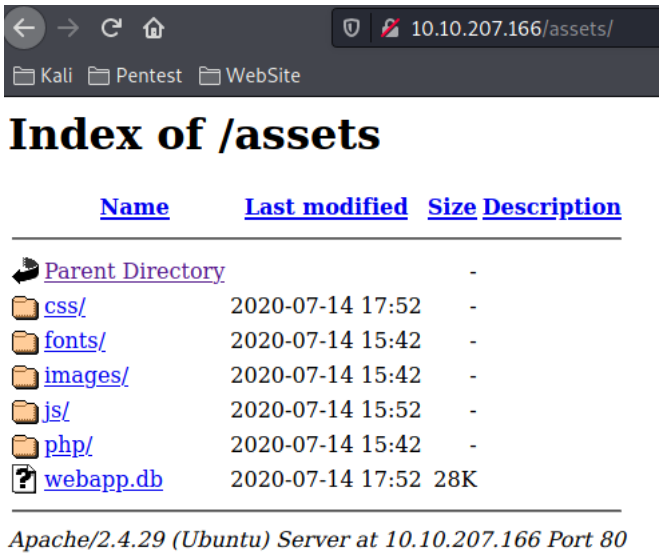



```
<!DOCTYPE html>
<html>
    <head>
        <title>Login</title>
        <meta name="viewport" content="width=device-width, user-scalable=no">
        <meta charset="utf-8">
        <link rel="shortcut icon" type="image/x-icon" href="../favicon.ico">
        <link type="text/css" rel="stylesheet" href="../assets/css/style.css">
        <link type="text/css" rel="stylesheet" href="../assets/css/loginStyle.css">
        <link type="text/css" rel="stylesheet" href="../assets/css/orkney.css">
        <link type="text/css" rel="stylesheet" href="../assets/css/icons.css">
        <script src="../assets/js/jquery-3.5.1.min.js"></script>
        <script src="../assets/js/loginScript.js"></script>
    </head>
    <body>
        <header>
            <a id="home" href="/">Sense and Sensitivity</a>
            <a id="login" href="/login">Login</a>
        </header>
        <div class=background></div>
        <!-- Must remember to do something better with the database than store it in /assets... -->
        <main>
            <div class="content">
                <form method="POST" action="/api/login">
                    <input type="text" name="username" placeholder="Username"><br>
                    <input type="password" name="password" placeholder="Password"><br>
                    <input id="loginBtnFunc" type="submit" value="Login!">
                </form>
                <i id="loginBtnStyle" class="material-icons">arrow_forward</i>
                        </div>
        </main>
        <footer><span>&copy; Sense and Sensitivity, 2020</span></footer>
    </body>
</html>
```

Dans un commentaire, on y voit que la base de données est stockée dans : /assets

## 3.2 Navigate to the directory you found in question one. What file stands out as being likely to contain sensitive data ?

Il y a un fichier différent des autres qui est : webapp.db




## 3.3 Use the supporting material to access the sensitive data. What is the password hash of the admin user ?

On suit ce qu'il y a dans l'énoncé et on voit l'empreinte du mot de passe de l'admin ainsi que d'autres utilisateurs.

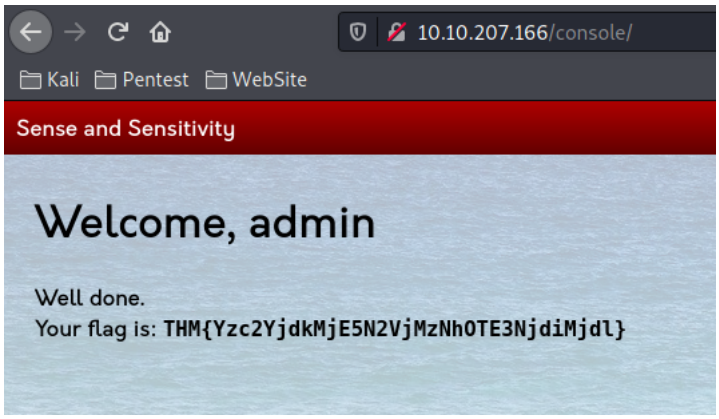
```
kali@kali:~/Pentest/TryHackMe/OWASP Top 10$ sqlite3 webapp.db
SQLite version 3.34.0 2020-12-01 16:14:00
Enter ".help" for usage hints.
sqlite> .tables
sessions  users
sqlite> PRAGMA table_info(users);
0|userID|TEXT|1||1
1|username|TEXT|1||0
2|password|TEXT|1||0
3|admin|INT|1||0
sqlite> SELECT * FROM users;
4413096d9c933359b898b6202288a650|admin|6eea9b7ef19179a06954edd0f6c05ceb|1
23023b67a32488588db1e28579ced7ec|Bob|ad0234829205b9033196ba818f7a872b|1
4e8423b514eef575394ff78caed3254d|Alice|268b38ca7b84f44fa0a6cdc86e6301e0|0
sqlite>
```

## 3.4 What is the admin's plaintext password ?

On utilise CrackStation comme indiqué dans l'énoncé pour retrouver le mot de passe de l'admin : qwertyuiop


## 3.5 Login as the admin. What is the flag ?

On se connecte en tant qu'admin maintenant que nous avons son mot de passe. On obtient alors le flag du challenge




# 4 [Severity 4 XML External Entity - eXtensible Markup Language

## 4.1 Full form of XML

Extensible Markup Language

## 4.2 Is it compulsory to have XML prolog in XML documents ?

No

## 4.3 Can we validate XML documents against a schema ?

Yes

### 4.4 How can we specify XML version and encoding in XML document ?

XML prolog

## 5 [Severity 4] XML External Entity - DTD

### 5.1 How do you define a new ELEMENT ?

!ELEMENT

### 5.2 How do you define a ROOT element ?

!DOCTYPE

### 5.3 How do you define a new ENTITY ?

!ENTITY

## 6 [Severity 4] XML External Entity - Exploiting

### 6.1 What is the name of the user in /etc/passwd

On se sert de ce qu'il y avait dans l'énoncé précédent pour construire notre payload. Et on obtient l'utilisateur : falcon

---


### 6.2 Where is falcon's SSH key located ?

Il faut une connaissance de Linux pour savoir que les clés SSH des utilisateurs sont stockés dans le dossier .ssh situé à la racine de leur home. La clé privée est id_rsa. Donc on obtient : /home/falcon/.ssh/id_rsa

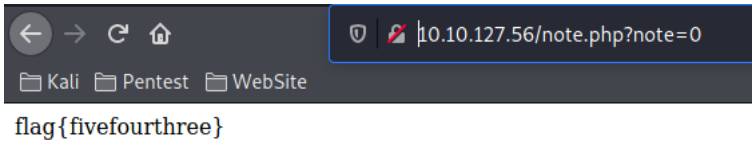
### 6.3 What are the first 18 characters for falcon's private key

On reprend exactement la même payload que pour etc/passwd mais on change la destination pour mettre la clé ssh


# 7 [Severity 5] Broken Access Control (IDOR Challenge)

## 7.1 Look at other users notes. What is the flag ?

Lorsqu'on se connecte, on voit que l'adresse URL est : http ://10.10.127.56/note.php ?note=1. On essaye alors de modifier la valeur de note pour voir ceux des autres utilisateurs. On essaie la valeur 0 et par chance, cela nous donne le flag.

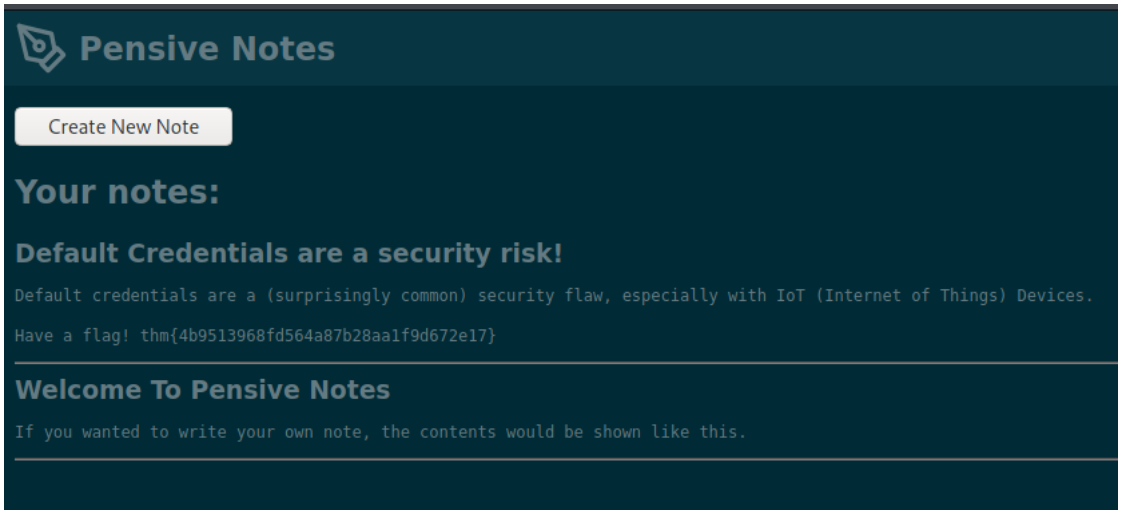



# 8 [Severity 6] Security Misconfiguration

## 8.1 Hack into the webapp, and find the flag !

En voyant le design de l'application web, on peut penser qu'il s'agit d'un CMS. Une recherche sur Google nous donne le GitHub du CMS : https ://github.com/NinjaJc01/PensiveNotes

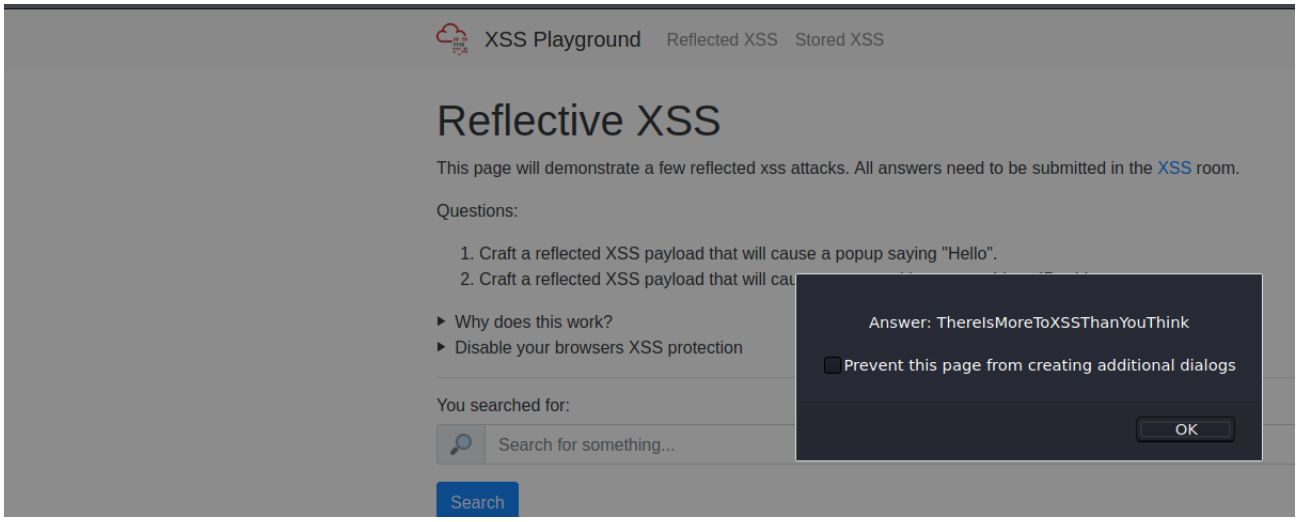
En bas de page, on y voit les identifiants pour s'y connecter. L'énoncé nous indique que le mot de passe n'a pas changé alors on peut se connecter sans problème dessus et alors le flag.




# 9 [Severity 7] Cross-site Scripting

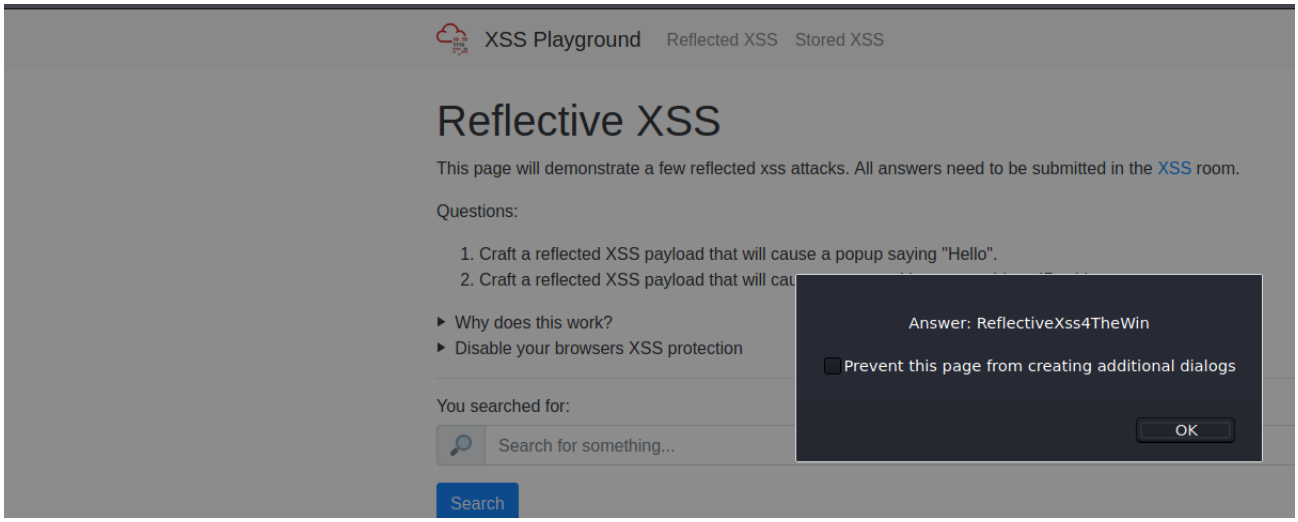
## 9.1 Navigate to http ://MACHINE_IP/ in your browser and click on the "Reflected XSS" tab on the navbar ; craft a reflected XSS payload that will cause a popup saying "Hello".

On enverra cette payload : <script>alert("Hello") ;</script> et on obtiendra le flag.



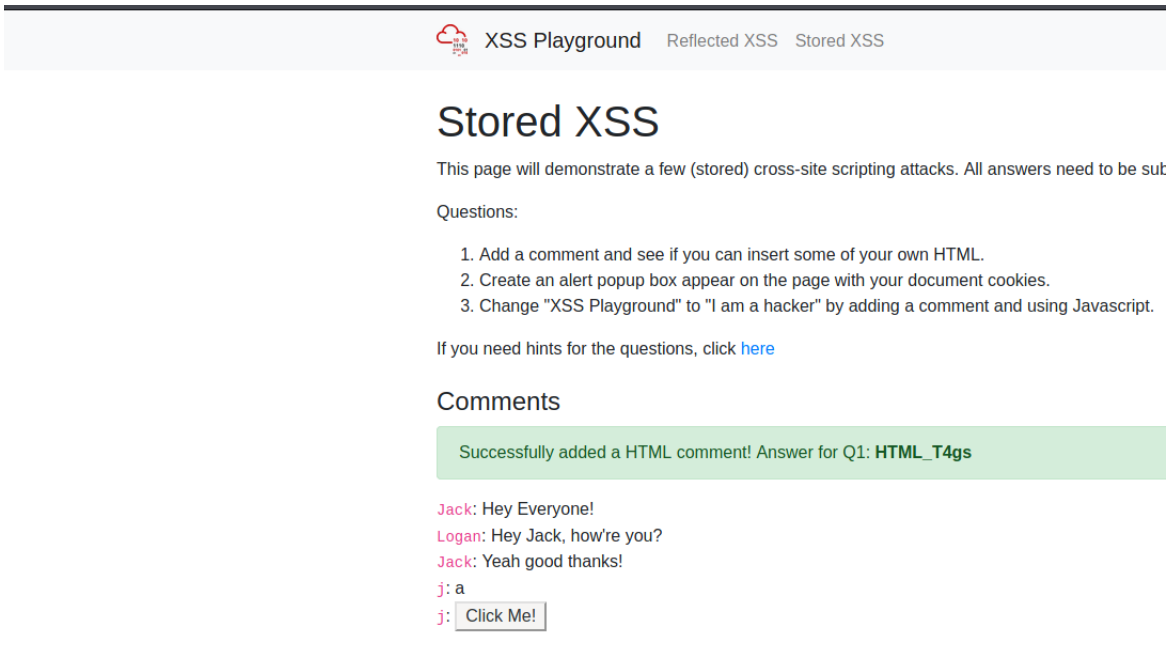

## 9.2 On the same reflective page, craft a reflected XSS payload that will cause a popup with your machines IP address.

On suit l'indice et on utilisera la payload : <script>alert(window.location.hostname) ;</script>



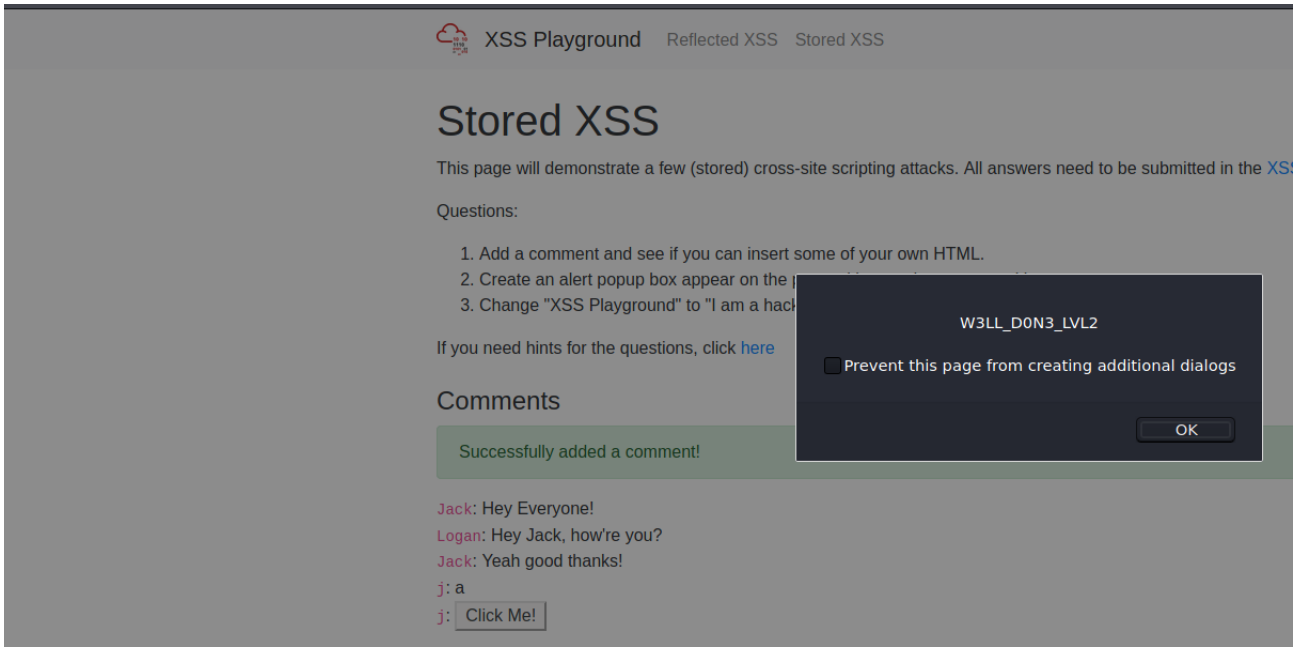

## 9.3 Then add a comment and see if you can insert some of your own HTML.

On décide d'ajouter un bouton dans les commentaires avec la payload : <button type="button">Click Me!</button>. Le flag apparaîtra alors après avoir mis le commentaire.



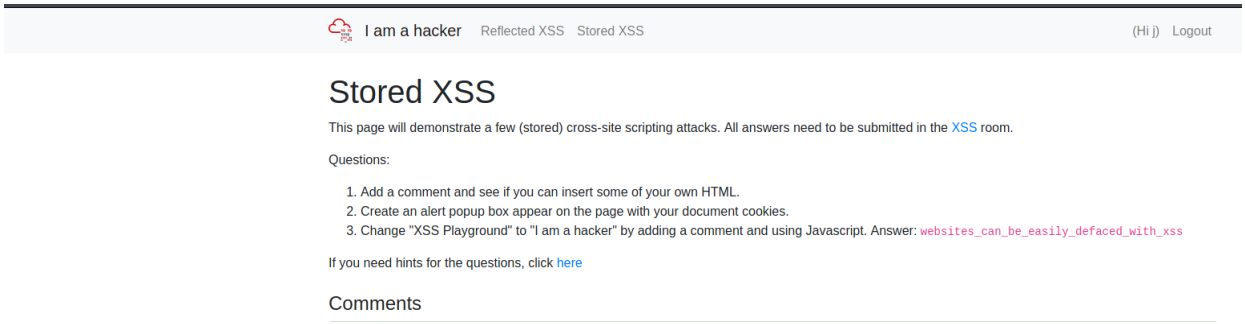

## 9.4 On the same page, create an alert popup box appear on the page with your document cookies.

On utilisera la payload : <script>alert(document.cookie) ;</script> pour obtenir le flag.




## 9.5 Change "XSS Playground" to "I am a hacker" by adding a comment and using Javascript.

En allant voir le code source, on voit que "XSS Playground" est dans une balise span qui a pour id "thm-title". En s'aidant de l'indice, on obtient :
<script>document.querySelector('#thm-title').textContent = 'I am a hacker'</script>




# 10 [Severity 8] Insecure Deserialization

## 10.1 Who developed the Tomcat application ?

The Apache Software Foundation

## 10.2 What type of attack that crashes services can be performed with insecure deserialization ?

Denial of Service

# 11 [Severity 8] Insecure Deserialization - Objects

## 11.1 Select the correct term of the following statement :

A Behaviour

# 12 [Severity 8] Insecure Deserialization - Deserialization

## 12.1 What is the name of the base-2 formatting that data is sent across a network as ?

Binary

# 13 [Severity 8] Insecure Deserialization - Cookies

## 13.1 If a cookie had the path of webapp.com/login , what would the URL that the user has to visit be ?

webapp.com/login

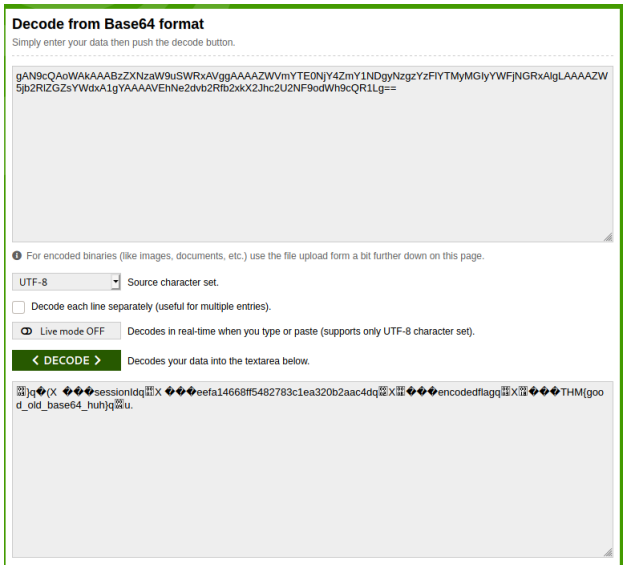
## 13.2 What is the acronym for the web technology that Secure cookies work over ?

HTTPS

# 14 [Severity 8] Insecure Deserialization - Cookies Practical

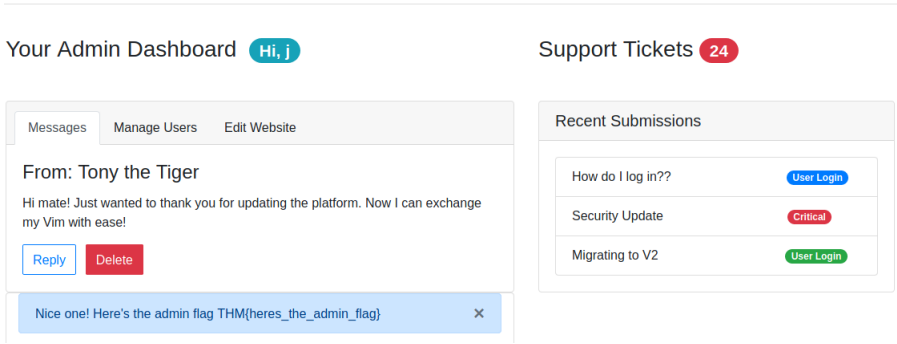
## 14.1 1st flag (cookie value)

Une fois connecté, on va dans Inspect l'Element puis dans Storage pour voir les cookies. La valeur du cookie "sessionId" ressemble a de l'héxadécimal. On le décode et parmi le résultat, on voit un flag.




## 14.2 2nd flag (admin dashboard)

On voit qu'il y a un cookie ayant pour nom "userType", il suffit de mettre comme valeur "admin" et de recharger la page pour être connecté avec les droits admin. On y voit le deuxième flag.




# 15 [Severity 8] Insecure Deserialization - Code Execution

## 15.1 flag.txt

On suit ce qu'il y a dans l'énoncé, on remplace le cookie "encodedPayload" par un reverse shell en bash, on recharge la page et on clique sur feedback.

Une fois sur le serveur, on recherche le fichier avec pour nom "flag.txt" et ensuite on peut le lire.

```
$ find / -name "flag.txt" 2>/dev/null
/home/cmnatic/flag.txt
$ ls
app.py
Dockerfile
index.html
launch.sh
__pycache__
requirements.txt
static
templates
user.html
venv
vimexchange.sock
wsgi.py
$ pwd
/home/cmnatic/app
$ cd ..
$ cat flag.txt
4a69a7ff9fd68
$
```

# 16 [Severity 9] Components With Known Vulnerabilities - Lab

## 16.1 How many characters are in /etc/passwd (use wc -c /etc/passwd to get the answer)

Une recherche sur Google avec les mots "bookstore exploit" permet de tomber sur un exploit https ://www.exploit-db.com/exploits/47887

On télécharge l'exploit et on l'exécute. On obtient un shell sur le serveur.

```
kali@kali:~/Pentest/TryHackMe/OWASP Top 10$ python3
47887.py  rce.py
kali@kali:~/Pentest/TryHackMe/OWASP Top 10$ python3 47887.py http://10.10.207.179/
> Attempting to upload PHP web shell ...
> Verifying shell upload ...
> Web shell uploaded to http://10.10.207.179/bootstrap/img/x8Iho8vJOo.php
> Example command usage: http://10.10.207.179/bootstrap/img/x8Iho8vJOo.php?cmd=whoami
> Do you wish to launch a shell here? (y/n): y
RCE $ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)

RCE $ wc -c /etc/passwd
1611 /etc/passwd

RCE $
```

Il y a 1611 caractères dans /etc/passwd

# 17 [Severity 10] Insufficient Logging and Monitoring

## 17.1 What IP address is the attacker using ?

Il y a plusieurs essais pour se connecter qui viennent de 49.99.13.16

## 17.2 What kind of attack is being carried out ?

Brute Force