

Rapport du TP 1

NISSIA Christian

AMANA Lydie

Partie I

Le travail s'est fait sous Kali linux 2021.

L'installation a été effectuée sans problème avec la version prebuilt.

- La commande **sudo xl list** permet d'afficher les domaines comme sur la capture suivante :

```
(root@kali)~# sudo xl list
Name      ID   Mem VCPUs   State   Time(s)
Domain-0  0   7870    4    r----- 180.6

(root@kali)~#
```

- Quantité de ressource allouée au dom0

Mémoire : 7870

VCPUs : 4

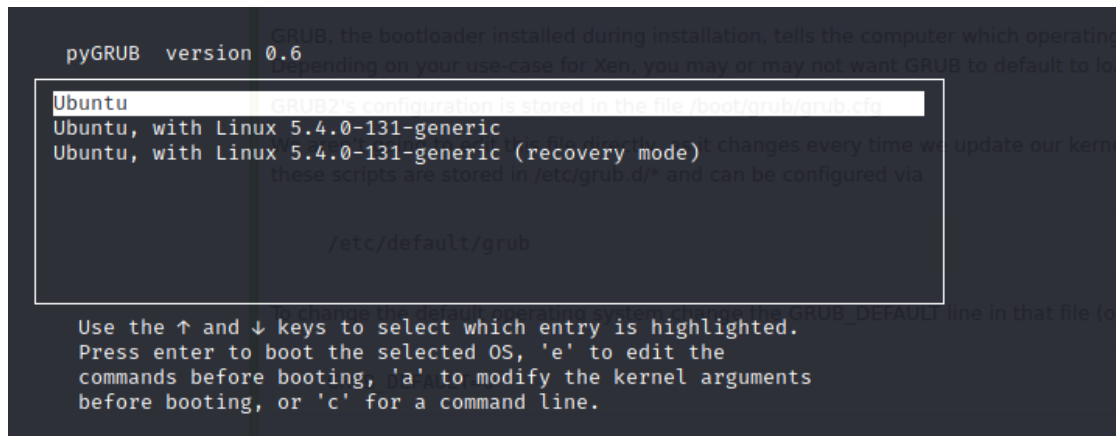
- Nous pouvons dire que le fichier de configuration que nous avons créé est incomplet car il manque le login.
- La commande **sudo xl list** affiche la liste des domaines, dans laquelle nous pouvons voir le domain 0

```
(root@kali)~# sudo xl list
Name      ID   Mem VCPUs   State   Time(s)
Domain-0  0   7870    4    r----- 1637.4

(root@kali)~#
```

Démarrage de la VM

Au démarrage, le Grub de la machine virtuelle Ubuntu démarre comme sur la figure précédente.



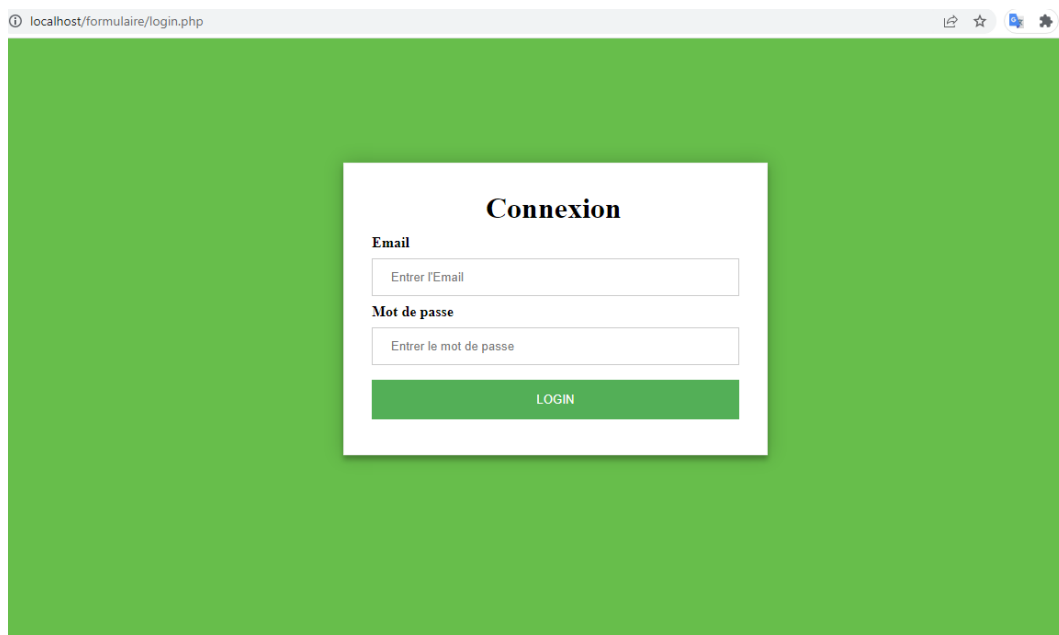
Nous pouvons donc nous connecter en tant que root avec le mot de pas que nous avons défini plus haut.

Partie 2

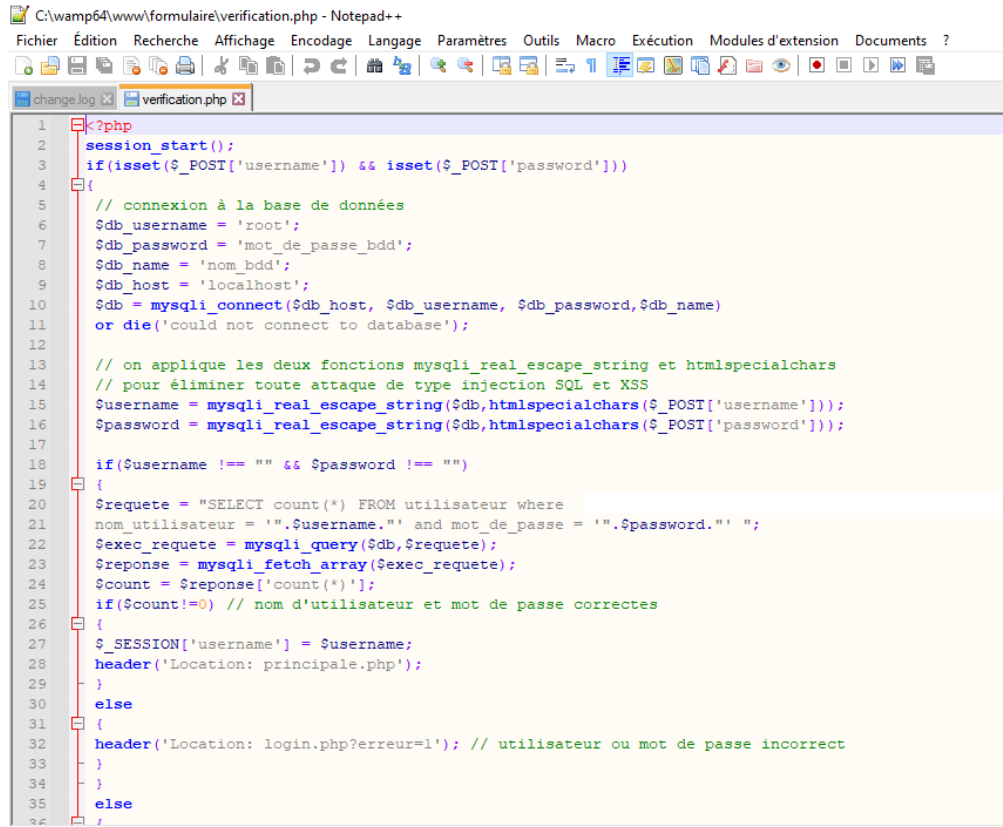
Execution d'une attaque SQLi et déni de service

La plateforme

Le formulaire de connexion



Le code serveur d'authentification



```
1 <?php
2 session_start();
3 if(isset($_POST['username']) && isset($_POST['password']))
4 {
5     // connexion à la base de données
6     $db_username = 'root';
7     $db_password = 'mot_de_passe_bdd';
8     $db_name = 'nom_bdd';
9     $db_host = 'localhost';
10    $db = mysqli_connect($db_host, $db_username, $db_password,$db_name)
11    or die('could not connect to database');
12
13    // on applique les deux fonctions mysqli_real_escape_string et htmlspecialchars
14    // pour éliminer toute attaque de type injection SQL et XSS
15    $username = mysqli_real_escape_string($db,htmlspecialchars($_POST['username']));
16    $password = mysqli_real_escape_string($db,htmlspecialchars($_POST['password']));
17
18    if($username != "" && $password != "")
19    {
20        $requete = "SELECT count(*) FROM utilisateur where
21        nom_utilisateur = '". $username.'" and mot_de_passe = '". $password.'" ";
22        $exec_requete = mysqli_query($db,$requete);
23        $reponse = mysqli_fetch_array($exec_requete);
24        $count = $reponse['count(*)'];
25        if($count!=0) // nom d'utilisateur et mot de passe correctes
26        {
27            $_SESSION['username'] = $username;
28            header('Location: principale.php');
29        }
30        else
31        {
32            header('Location: login.php?erreur=1'); // utilisateur ou mot de passe incorrect
33        }
34    }
35    else
36    {
37    }
```

SQLi Testing

- 1) Changement du mot de passe de l'administrateur

Il nous suffit de faire du SQL Injection et ajouter la ligne suivante dans le champ du mot de passe

```
sqlcmd -S .\PlateSpinDB -E -Q "ALTER LOGIN sa WITH PASSWORD = '{SangaNdoleOkok2038}'"
```

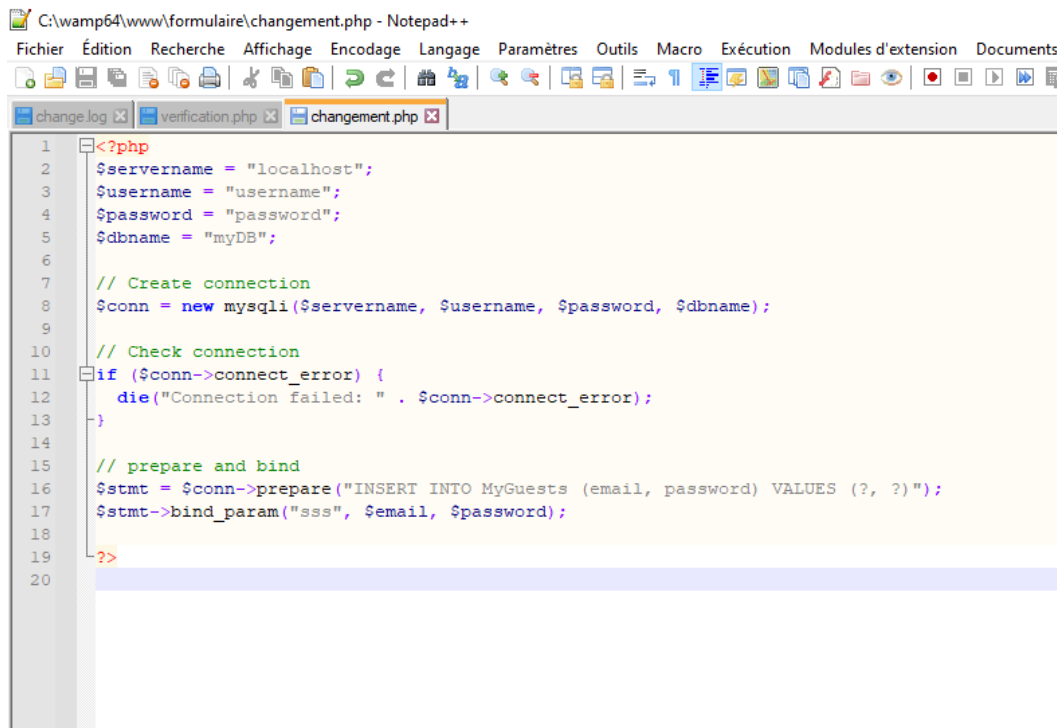
- 2) Une deuxième attaque qui aura pour but de nous authentifier quel que soit les informations

```
sqlcmd -S .\PlateSpinDB -E -Q "ALTER LOGIN sa WITH PASSWORD = '{ $db_password }'"
```

- 3) Pour empêcher ce genre d'attaques, il faut limiter vérifier les caractères récupérés dans les champs du formulaire avant de les envoyer à la base de données.

Correction et coût sur la performance

- 1) Utilisation des *prepared request/statements* pour gérer l'authentification.

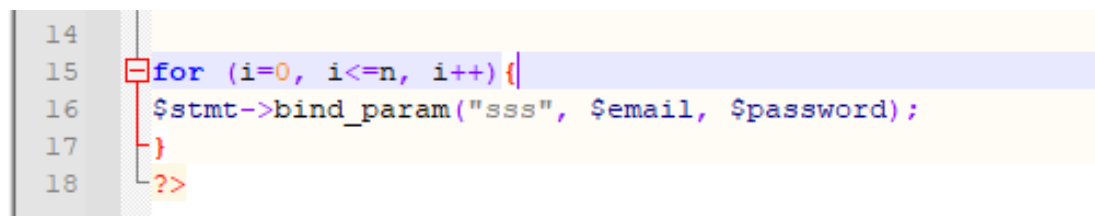


```
C:\wamp64\www\formulaire\changement.php - Notepad++
Fichier  Édition  Recherche  Affichage  Encodage  Langage  Paramètres  Outils  Macro  Exécution  Modules d'extension  Documents

change.log  verification.php  changement.php

1  <?php
2  $servername = "localhost";
3  $username = "username";
4  $password = "password";
5  $dbname = "myDB";
6
7  // Create connection
8  $conn = new mysqli($servername, $username, $password, $dbname);
9
10 // Check connection
11 if ($conn->connect_error) {
12     die("Connection failed: " . $conn->connect_error);
13 }
14
15 // prepare and bind
16 $stmt = $conn->prepare("INSERT INTO MyGuests (email, password) VALUES (?, ?)");
17 $stmt->bind_param("sss", $email, $password);
18
19 ?>
20
```

- 2) On constate que les attaques n'aboutissent plus
- 3) un script qui effectuera n requêtes d'authentification par seconde



```
14
15 for (i=0, i<=n, i++) {
16     $stmt->bind_param("sss", $email, $password);
17 }
18 ?>
```