

RAPPORT DE COLLECTE D'INFORMATIONS

PROJET TEST D'INTRUSION PYTHON

Membres du projets

Noms et prénoms des membres	Etablissement	Filières et Niveau
Kignon Gninnaha Abel	UPB	Master 1 / SAS
Obin Yapi Hermann	UPB	Master 1 / SAS
Bamba N'giantchan Allassane	UPB	Master 1 / SAS
Nassa Grace Marie Paule	UPB	Master 1 / SAS
Kouame Akissi Esther	UPB	Master 1 / SAS

Objectif

Collecte d'informations : Détails sur les points d'entrée et les informations recueillies.

L'objectif de cette phase de reconnaissance est de collecter des informations détaillées sur les points d'entrée de l'application web WordPress et bWAPP, hébergées sur une machine virtuelle vulnérable (VM OWASP Broken Web Applications). Ces informations serviront à identifier des faiblesses exploitables lors des phases ultérieures de l'attaque. Il est crucial de bien cerner la structure de l'application et de relever toutes les données disponibles, qu'elles soient explicites ou cachées, car cela permet de formuler des attaques précises et efficaces (comme l'injection SQL, les attaques de brute force, ou l'exploitation de vulnérabilités de formulaires).

Résolution

La collecte d'informations (ou reconnaissance) est une phase essentielle où le testeur de pénétration rassemble des données sur les cibles potentielles. Pour notre scénario, nous nous focalisons sur les pages de connexion des applications WordPress et bWAPP. L'idée est d'extraire les éléments visibles et cachés qui pourraient nous indiquer des points d'entrée ou des faiblesses.

Les pages de connexion (ex. /wp-login.php pour WordPress et /login.php pour bWAPP) représentent des points d'entrée cruciaux, car elles sont souvent mal protégées et représentent des cibles de choix pour les attaques par brute force ou les injections SQL. C'est donc sur celles-ci, qu'on se concentrera.

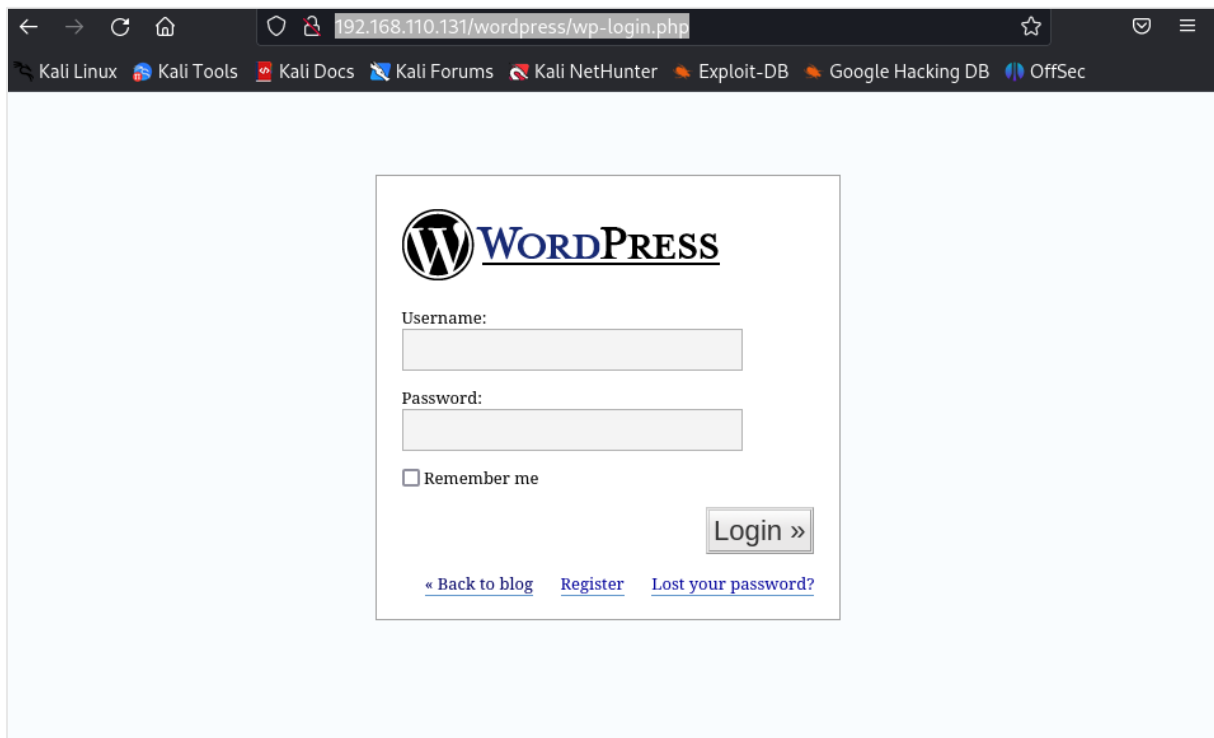
CAS de l'URL : <http://192.168.110.131/wordpress/wp-login.php>

Dans cette phase, nous avons implémenté un script Python simple pour automatiser la collecte d'informations à partir de différentes URL. Pour cela, nous avons utilisé les bibliothèques **requests** et **BeautifulSoup** (BS4), qui permettent respectivement de charger les pages web et d'analyser leur structure HTML afin d'extraire des données spécifiques comme les formulaires, les images, les liens, etc.

L'application en question se nomme WordPress et elle se trouve sur la VM OWASP Broken Web Applications.

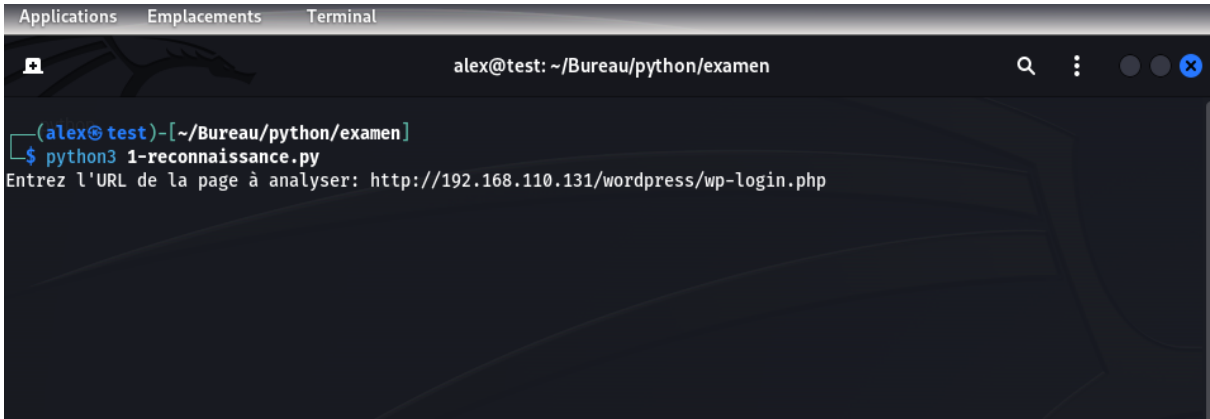
Script Python

```
python3 "/chemin/1-reconnaissance.py"
```



1)- Chargement de l'URL du site WordPress

On charge l'URL de notre page.



```

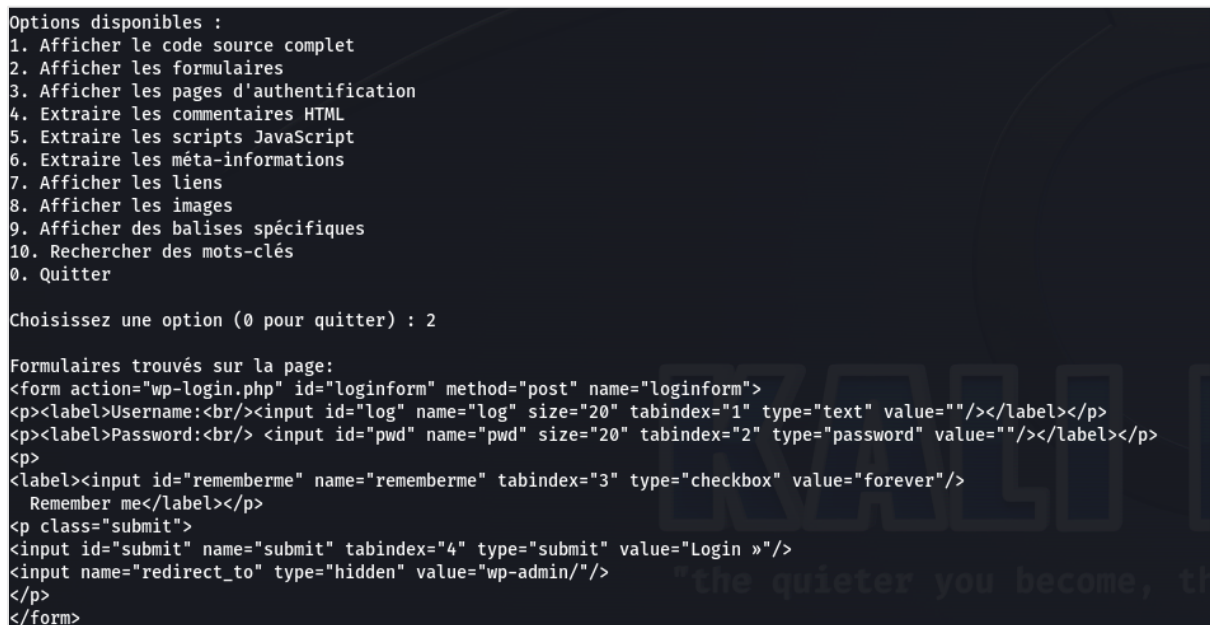
Applications  Emplacements  Terminal
alex@test: ~/Bureau/python/examen

(alex@test)-[~/Bureau/python/examen]
$ python3 1-reconnaissance.py
Entrez l'URL de la page à analyser: http://192.168.110.131/wordpress/wp-login.php

```

2)- Affichage du champs formulaire

Pour l’affichage du formulaire s’il en existe bien sûr, nous allons utiliser l’option 2 du script. Vous pouvez voir sur l’image ci-dessous le résultat de l’exécution.



```

Options disponibles :
1. Afficher le code source complet
2. Afficher les formulaires
3. Afficher les pages d'authentification
4. Extraire les commentaires HTML
5. Extraire les scripts JavaScript
6. Extraire les méta-informations
7. Afficher les liens
8. Afficher les images
9. Afficher des balises spécifiques
10. Rechercher des mots-clés
0. Quitter

Choisissez une option (0 pour quitter) : 2

Formulaires trouvés sur la page:
<form action="wp-login.php" id="loginform" method="post" name="loginform">
<p><label>Username:<br/><input id="log" name="log" size="20" tabindex="1" type="text" value=""/></label></p>
<p><label>Password:<br/> <input id="pwd" name="pwd" size="20" tabindex="2" type="password" value=""/></label></p>
<p>
<label><input id="rememberme" name="rememberme" tabindex="3" type="checkbox" value="forever"/>
Remember me</label></p>
<p class="submit">
<input id="submit" name="submit" tabindex="4" type="submit" value="Login »"/>
<input name="redirect_to" type="hidden" value="wp-admin"/>
</p>
</form>

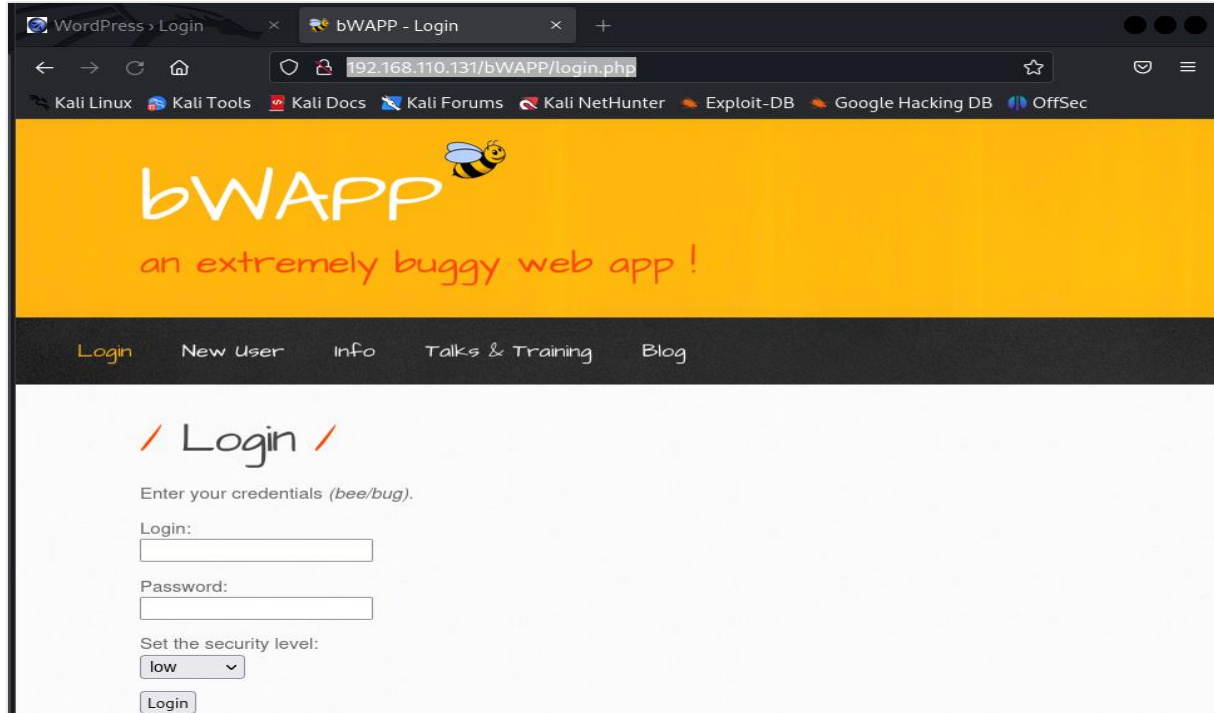
```

On pourra plus-tard faire du fuzzing sur ses champs de formulaire, observer les résultats et ensuite tenter de se connecter après avoir déceler des vulnérabilités exploitables. Notons comme le script le montre on a une palette de données qu’on peut extrait (Les images, les liens, les scripts sous javascript, les méta données etc...).

Après avoir identifié les champs du formulaire, nous pouvons envisager de passer à la phase de fuzzing ou d'injection, en testant des valeurs malicieuses dans les champs pour voir comment le serveur réagit. Par exemple, des attaques de brute force peuvent être automatisées pour tenter de casser le mot de passe de l'administrateur.

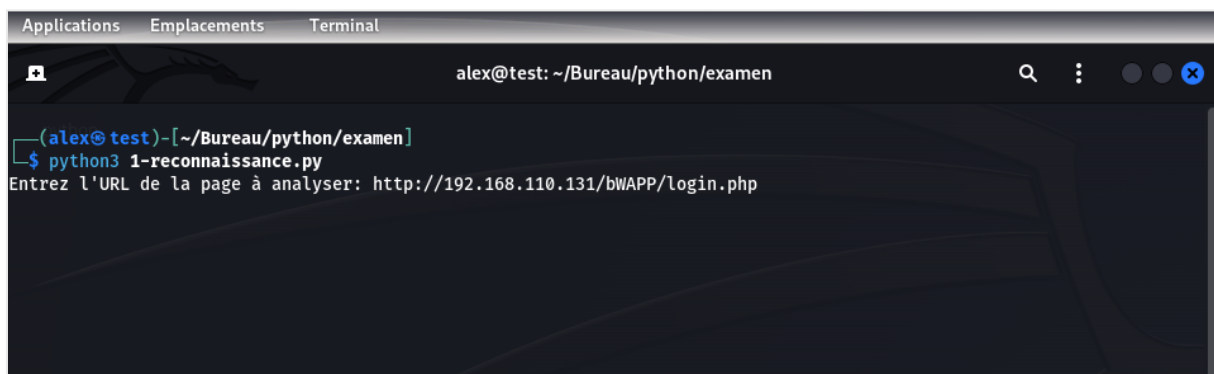
CAS de l'URL : <http://192.168.110.131/bWAPP/login.php>

Comme dans le cas précédemment vu, la même approche sera utilisée. Nous chargeons cette fois-ci l'URL de la page de connexion bWAPP (/login.php).



1)- Chargement de l'URL du site bWAPP

On va ici charger notre URL : <http://192.168.110.131/bWAPP/login.php>



En suivant les mêmes principes, on va extraire des informations de la page.

2)- Extraction de liens d'images de la page

On va donc utiliser ici, l'option 8 du script. Comme on l'a signifié auparavant, plusieurs extractions peuvent être fait dans le but de trouver des indices de vulnérabilité.

```
alex@test: ~/Bureau/python/examen

<a href="http://twitter.com/MME_IT" target="_blank">
  @MME_IT
</a>
on Twitter and ask for our cheat sheet, containing all solutions! / Need a
<a href="http://www.mmeit.be/bWAPP/training.htm" target="_blank">
  training
</a>
? / © 2014 MME BVBA
</p>
</div>
<div id="bee">
  
</div>
</body>
</html>

Options disponibles :
1. Afficher le code source complet
2. Afficher les formulaires
3. Afficher les pages d'authentification
4. Extraire les commentaires HTML
5. Extraire les scripts JavaScript
6. Extraire les méta-informations
7. Afficher les liens
8. Afficher les images
9. Afficher des balises spécifiques
10. Rechercher des mots-clés
0. Quitter

Choisissez une option (0 pour quitter) : 8

Images trouvées sur la page:
./images/blogger.png
./images/linkedin.png
./images/twitter.png
./images/facebook.png
./images/bee_1.png
```

Ces informations extraites, peuvent donner des informations qui permettront des accès non-autorisés sur l'application.