# TP1
# Exploitation des applications Web

### *Configuration de la machine*

Pour cette séance, vous avez à votre disposition une image Docker configurée avec un serveur HTTP qui publie une application Web vulnérable (PHP + MySQL).

Pour mettre en place le TP lancer la commande ci-dessous :

```
$docker run -d -p 80:80 registry.gitlab.com/piloo/tps:tpweb
```

Pour accéder à l'application, rendez vous à l'URL suivante et cliquer sur « Create /Reset database » :

```
http://localhost
```

### *Quelques exemples d'outils*

Pour faciliter l'exploitation des failles, il est recommandé d'installer les **plugins Firefox** suivants :

- HackBar
- Web developer ou Firebug
- Live HTTP Headers

### *Structure du TP*

Nous allons découvrir les failles applicatives Web faisant parties du TOP 10 de l'OWASP. Cette séance vous permettra d'apprendre à les exploiter est ainsi connaître les erreurs à éviter lors de vos développements.

## Prérequis

Il faut se connecter au site Web avec les identifiants ci-dessous :

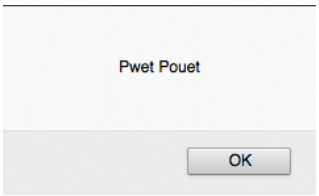
Login : admin
Password : password

La plateforme Web dispose de plusieurs niveaux de difficultés. Nous allons commencer par le niveau le plus facile « **LOW** » qui sera appliqué par défaut.

## Exercice 1 - (XSS)

**Objectif :**

Se rendre dans la page « XSS reflected » et exploiter la faille de XSS afin d'afficher un message d'alerte comme ci-dessous :




**Les questions à se poser :**

- Où puis-je injecter du code dans un site Web ??
- Où ce code sera-t-il interprété (Côté client, serveur) ? Quelles sont mes possibilités d'attaques ?

## Exercice 2 - (XSS)

**Objectif :**

Se rendre dans la page « XSS reflected » et exploiter la faille de XSS pour être redirigé vers le site de GOOGLE.

**Question à se poser :**

- Comment faire une redirection en JavaScript ?

## Exercice 3 - (XSS)

**Objectif :**

Nous avons vu que la faille de XSS permettait de rediriger vers un site Web, exécuter du code JavaScript. Maintenant, il est temps de construire votre première attaque

Forger une URL à envoyer par mail à l'administrateur du site pour lui voler son cookie de session. Une fois cette étape réussie, utiliser les informations obtenues pour s'authentifier sur le site. Pour simuler votre attaque, vous pouvez déjà voir ce qui se passe quand vous cliquez sur votre lien.

**Les questions à se poser :**

- Comment récupérer vos cookies en Javascript ?
- Comment envoyer les cookies récupérés vers un site distant contrôlé ?
- Comment faire pour monter un site distant (contrôlé) en attente des cookies ? sur Apache ? sur netcat ?
- Comment utiliser les informations obtenues pour usurper l'identité de l'administrateur ?
- Cette attaque fonctionne t'elle systématiquement?

## Exercice 4 - (Injection de commandes)

**Objectif :**

Se rendre dans la page « Command Execution » et détourner son comportement initial pour obtenir le contenu du fichier « /etc/passwd »

**Question à se poser :**

- Comment chainer plusieurs commandes dans un Shell ?

## Exercice 5 - (Injection de commandes)

**Objectif :**

Toujours en exploitant la faille d'injection de commandes, obtener un shell interactif (Avec Putty Telnet ou netcat) sur le serveur.

**Les questions à se poser :**

- Que puis-je faire avec **netcat** ?
- Il y a deux versions de netcat, quelle est la différence?
- Quelles sont les alternatives pour obtenir un shell ?
- Quel langage peut être utilisé ?

## Exercice 6 – (Local File Injection)

Se rendre dans la page « **File Inclusion** » et exploiter la faille pour afficher le contenu du fichier « /etc/passwd ».

## Exercice 7 – (Local File Injection)

**Objectif :**

Exploiter la faille de LFI pour afficher le contenu du fichier de configuration de la connexion à MySQL (« /config/config.inc.php »).

**Les questions à se poser :**

- Pourquoi ai-je une page blanche ?

## Exercice 8 – (Injection SQL)

**Objectif :**

Rendez vous sur la page « **SQL injection** » et trouver quel caractère provoque des messages d'erreurs SQL :

```
You have an error in your SQL syntax; check the manual that corresponds to your MySQL
server version for the right syntax to use near
```

**Les questions à se poser :**

- Pourquoi ce caractère provoque t-il une erreur ?
- En quoi ce type de message peut-il nous aider ?

## Exercice 9 – (Injection SQL)

**Objectif :**

Trouver le nombre de colonnes visibles

Rappel :
Dans le requête ci-dessous, nous avons quatre colonnes (nom , prenom, login, password) :

```
SELECT nom,prenom,login,password FROM personne
```

**Les questions à se poser :**

- Quels opérateurs SQL nous permettent de deviner le nombre de colonnes ?
- A votre avis, pourquoi est-il nécessaire de connaître le nombre de colonnes ?
- A votre avis, à quoi ces colonnes visibles pourront-elles servir ?

## Exercice 10 – (Injection SQL)

**Objectif :**

Trouver les noms des bases de données et tables existantes.

**La question à se poser :**

- Quelle table contient le nom de toutes les tables connues de MySQL ?

## Exercice 11 – (Injection SQL)

**Objectif :**

Trouver les colonnes de la table « users » .

**La question à se poser :**

- Quelle table contient le nom de toutes les colonnes connues de MySQL ?

## Exercice 12 – (Injection SQL)

**Objectif :**

Vous disposez de toutes les informations sur la structure de la table « users ». Il ne reste plus qu'à extraire les couples « login:password ».

Quel est le mot de passe de l'utilisateur « pablo» ?

## Conclusion

Il ne faut pas faire confiance aux l'utilisateurs !!
Filtrer toutes les entrées, champs de saisie, paramètres d'URL, …