

SECURISER UNE APPLICATION

Table des matières

I.	Enoncé.....	2
II.	Base de données	3
III.	Fichier de configuration	3
A.	Cas 1 – fichier de configuration	3
B.	Cas 2 – fichier environnement	3
A.	Cas 3 – fichier environnement – avancé.....	3
IV.	Les formulaires et leurs failles	4
A.	Injection de commandes.....	4
B.	Cross Site Scripting (XSS).....	4
C.	CSRF.....	4
V.	Les utilisateurs	5
A.	Brut Force.....	5
B.	Injection SQL	5
VI.	Téléchargement de fichier	5
VII.	Pour aller plus loin	6

I. Enoncé

Vous venez d'intégrer une équipe de développement et on vous demande de sécuriser une application déjà existante.

L'application permet de corriger certaines failles connues et les bonnes pratiques à mettre en place pour palier à ces failles devraient être automatiquement mises en place.



Sécuriser ses applications en PHP

A propos

Ce site est un site d'exercices pour la sécurité des sites en PHP.
Il comprend des failles de sécurité, aussi il ne faut pas l'installer sur un serveur.

Les formulaires

[Injection de commandes](#)
[Cross Site Scripting \(XSS\)](#)
[CSRF](#)

Objectifs

Ce site est une application Web PHP/MySQL vulnérable.
Son objectif principal est d'expliquer lors de formations la sécurité et à tester les outils.
L'objectif de ce site est de pratiquer certaines des vulnérabilités Web les plus courantes

AVERTISSEMENT!

L'application est vulnérable ! Ne le téléchargez pas dans le dossier html public de votre hébergeur ou sur tout autre serveur Internet.

Il est recommandé de le tester en local ou d'utiliser une machine virtuelle.

Clause de non-responsabilité

Nous n'assumons aucune responsabilité quant à la manière dont quiconque utilise cette application.

Nous avons précisé les objectifs de l'application et elle ne doit pas être utilisée à des fins malveillantes.

Si votre serveur Web est compromis via une installation de cette application, ce n'est pas notre responsabilité, c'est la responsabilité de la ou des personnes qui l'ont téléchargé et installé.

Installation

1. Créer la base de données phpsec
2. Importer le script [phpsec.sql](#) dans la base de données phpsec

Schéma de la BDD

phpsec users	
id	int
login	varchar(255)
password	varchar(255)
name	varchar(255)

phpsec comments	
id	int
name	varchar(255)
comment	text
publish	tinyint(1)

Site créé par [CE SOFT](#) pour [CE FORMATION](#)

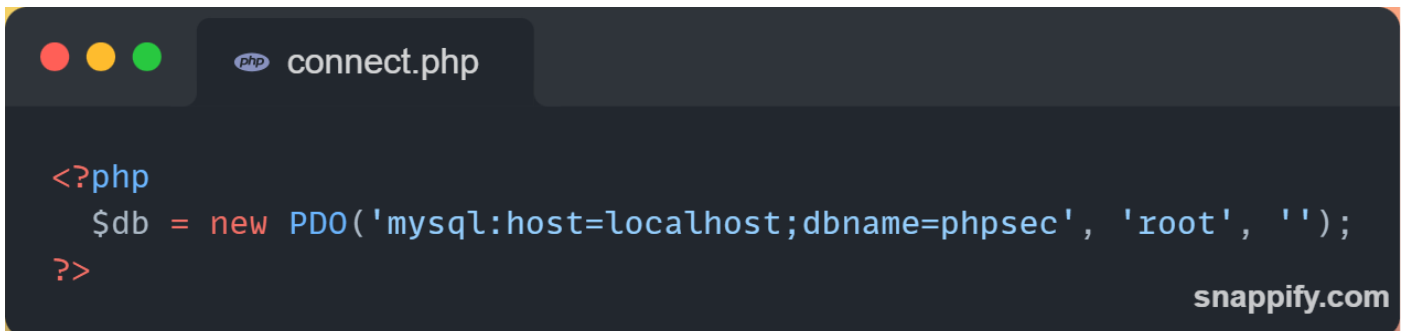
II. Base de données

Vous trouvez la base de données un peu sommaire et vous constatez des manquent, vous la corrigez au fur et à mesure.

Un export de celle-ci sera mise à jour au niveau du projet.

III. Fichier de configuration

Vous constatez que les éléments de connexion à la base de données se trouvent directement dans le fichier de connexion.



```
<?php
$db = new PDO('mysql:host=localhost;dbname=phpsec', 'root', '');
?>
```

A. Cas 1 – fichier de configuration

Vous décidez de mettre en place un fichier de configuration comprenant des constantes utilisables partout dans le code.

B. Cas 2 – fichier environnement

Vous décidez de mettre en place un fichier d'environnement (.env) comprenant des associations clé valeur (var=valeur), et vous mettez en place une fonction qui permet de parcourir le fichier et de convertir ces associations en constantes.

A. Cas 3 – fichier environnement – avancé

Vous décidez d'utiliser un fichier d'environnement mais vous mettez en place une classe pour y accéder.

IV. Les formulaires et leurs failles

A. Injection de commandes

Vous constatez qu'il est possible de faire une injection de commandes, alors que le formulaire ne doit permettre de tester qu'une adresse IP.

Vous décidez de mettre en place le correctif pour ne pas permettre de saisir autre chose qu'une adresse IP.

Saisir une adresse IP:

B. Cross Site Scripting (XSS)

Vous constatez qu'il est possible d'ajouter du code malveillant dans les commentaires.

Nom *

Commentaire *

Mon premier commentaire

Christel

Cous décidez de mettre en place le correctif pour ne plus interpréter le code.

C. CSRF

Vous constatez que le formulaire de changement de mot de passe peut être renvoyé plusieurs fois.

Nouveau mot de passe :

Confirmer le mot de passe :

Vous décidez de mettre en place les correctifs nécessaires pour empêcher cela et pour être sûr que c'est bien le site lui-même qui demande la modification du mot de passe.

V. Les utilisateurs

A. Brut Force

Vous constatez que les mots de passe ne sont pas assez forts et qu'un hacker peut donc faire du brut force

Username:

Password:

[Se connecter](#)

Vous décidez de mettre en place les contraintes suivantes :

- Mise en place des mots de passe forts (et vérifiez l'envoi du formulaire en post).
- Hacher le mot de passe
- Stocker en BDD les tentatives de connexion
- Définir la limite du nombre de tentatives
- Définir le devenir d'un nombre de tentatives trop élevé (bloqué 15 minutes, compte désactivé, Adresse IP en blacklist)

B. Injection SQL

Vous testez l'injection SQL au niveau du formulaire de connexion et vous constatez que cela est possible.

Username:

Password:

[Se connecter](#)

Vous décidez de :

- Utiliser des requêtes préparées
- Nettoyer les entrées de l'utilisateur

VI. Téléchargement de fichier

Vous testez le téléchargement de fichier et vous constatez qu'il n'y a aucune vérification.

Vous mettez en place les contraintes suivantes :

- Autoriser uniquement les types de fichier nécessaires (images jpg, png)
- Vérifier les informations du fichier
- Renommer le fichier

VII. Pour aller plus loin

Il vous reste un peu de temps par rapport au planning attendu.

Vous décidez d'empêcher l'affichage de la liste du contenu des répertoires.

Vous avez constaté que l'application a été développée en PHP Procédural, vous décidez de transformer l'application en POO (Programmation Orientée Objet), voire en MVC.