# Sécuriser son site internet en PHP procédurale

## Sommaire

# La fondation OWASP

**Open Web Application Security Project (OWASP)** est une communauté en ligne travaillant sur la sécurité des applications Web. Sa philosophie est d'être à la fois libre et ouverte à tous. Elle a pour vocation de publier des recommandations de sécurisation Web et de proposer aux internautes, administrateurs et entreprises des méthodes et outils de référence permettant de contrôler le niveau de sécurisation de ses application web


Tous les ans cette fondation réactualise son top ten des risques de sécurité applicatifs Web les plus critiques.

Voici leur classement à la date du 1er février 2021 :

1. L'injection SQL
2. L'authentification et la gestion des systèmes interrompues (Broken Authentification and Session Management) : correspond au risque de casser la gestion de l'authentification et de la session. Comprend notamment le vol de session ou la récupération de mots de passe.
3. L'exposition des données sensibles (Sensitive Data Exposure) : correspond aux failles de sécurité liées aux données sensibles comme les mots de passe, les numéros de carte de paiement ou encore les données personnelles et la nécessité de chiffrer ces données.
4. XML External Entities (XXE)
5. Les contrôles d'accès cassés (Broken Access Control) : Les restrictions sur ce que les utilisateurs authentifiés sont autorisés à faire ne sont souvent pas correctement appliquées. Les attaquants peuvent exploiter ces failles pour accéder à des fonctionnalités et / ou des données non autorisées, comme accéder aux comptes d'autres utilisateurs, afficher des fichiers sensibles, modifier les données d'autres utilisateurs, modifier les droits d'accès, etc.
6. Mauvaise configuration de la sécurité : Une mauvaise configuration de la sécurité est le problème le plus courant. Cela est généralement le résultat de configurations par défaut non sécurisées, de configurations incomplètes ou ad hoc, d'un stockage dans le cloud ouvert, d'en-têtes HTTP mal configurés et de messages d'erreur détaillés contenant des informations sensibles. Non seulement tous les systèmes d'exploitation, Frameworks, bibliothèques et applications doivent être configurés en toute sécurité, mais ils doivent également être corrigés / mis à niveau en temps opportun.
7. Les failles XSS (Cross-Site Scripting (XSS) : Elles se produisent chaque fois qu'une application inclut des données non approuvées dans une nouvelle page Web sans validation ou échappement appropriée, ou met à jour une page Web existante avec des données fournies par l'utilisateur à l'aide d'une API de navigateur qui peut créer du HTML ou du JavaScript. XSS permet aux attaquants d'exécuter des scripts dans le navigateur de la victime qui peuvent détourner les sessions des utilisateurs, dégrader des sites Web ou rediriger l'utilisateur vers des sites malveillants.
8. Désérialisation non sécurisée (Insecure Deserialization).
9. Utilisation de composants avec des vulnérabilités connues (Using Components with Known Vulnerabilities) : Les composants, tels que les bibliothèques, les Frameworks et d'autres modules logiciels, s'exécutent avec les mêmes privilèges que l'application. Si un composant vulnérable est exploité, une telle attaque peut faciliter de graves pertes de données ou une prise de contrôle du serveur. Les applications et les API utilisant des composants avec des vulnérabilités connues peuvent saper les défenses des applications et permettre diverses attaques et impacts.
10. Surveillance insuffisante des tentatives de connexions (Insufficient Logging & Monitoring).

# Se protéger des injections SQL

Pour prévenir les injections SQL, il faut faire appel aux requêtes préparées. Ce sont des requêtes dans lesquels les paramètres sont interprétés indépendamment de la requête elle-même. De cette manière, il est impossible d'effectuer des injections. Dans tous les systèmes de gestion de bases de données, deux méthodes sont utilisées : prepare() qui prépare la requête et execute() qui exécute la requête avec les paramètres.

Si vous utilisez PDO sur une base de données MySQL, il faut savoir que par défaut les requêtes préparées ne sont pas réelles, elles sont émulées par PDO. Pour forcer PDO à utiliser des requêtes préparées réelles, il faut le préciser après la connexion à la base de données.

```php
$sth = $pdo->prepare("
     INSERT INTO voitures(modele, marque, annee, immatriculation, image)
     VALUES (:modele, :marque, :annee, :immatriculation, :image)
  ");

  $sth->execute(array(
                    ':modele' => $modele,
                    ':marque' => $marque,
                    ':annee' => $annee,
                    ':immatriculation' => $immatriculation,
                    ':image' => $image));
```

# Se protéger des failles XSS

La faille XSS consiste en l'injection de script HTML ou javascript dans la base de données via les formulaires.

Pour se prémunir de ses failles, il faut absolument utiliser les fonctions php **htmlspecialchars()** qui filtre les ‘<‘ et ‘>’ ou **htmlentities()** qui filtre toutes les entités html.

Ces fonctions doivent être utilisées sur des entrées utilisateurs qui s’afficheront plus tard sur votre site. Si elles ne sont pas filtrées, les scripts comme celui que vous pouvez voir à droite s’exécuteront avec tout le mal qui s’en suit.

Voici un exemple d’utilisation de ces fonctions lors de la récupération de données via un formulaire :

- htmlspecialchars():

```php
$nameCategory = htmlspecialchars($_POST['nameProduct']);
```

- htmlentities():

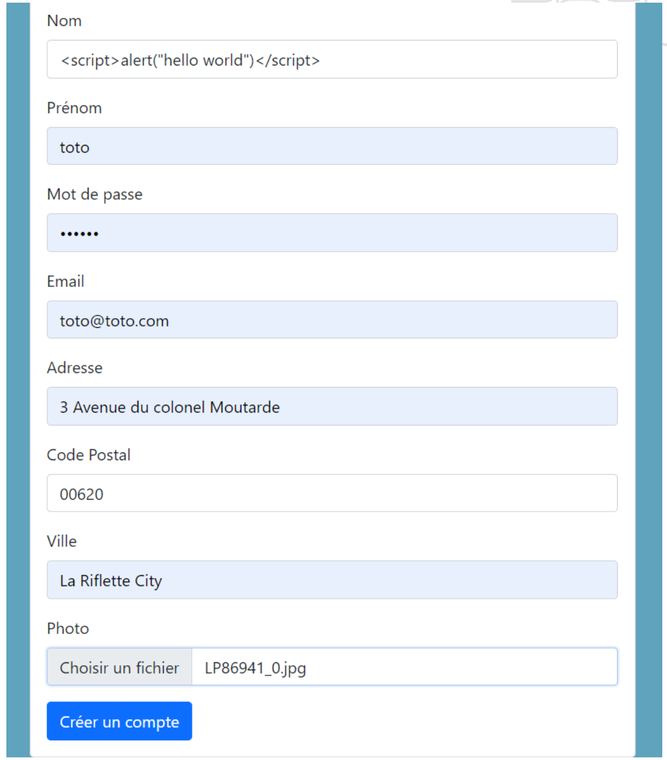
```php
$nameCategory = htmlentities($_POST['nameProduct']);
```

Ces deux fonctions appliquées aux données que vous récupérez en get ou en post va filtrer et remplacer tous les caractères spécifiques aux langages HTML et Javascript les remplaçant pas leur code HTML spécifique (*tableau des caractère HTML*).

Exemple :

<script>allert("hello world")</script> va être transformé en <script>allert("hello world")</script>

Ainsi le script malveillant ne sera plus interprété comme du HTML ou du Javascript.



# Les contrôles d'accès cassés (Broken Access Control)

Les contrôles d'accès cassés consistent à pouvoir accéder à des pages sans en avoir normalement le droit. Prenons pour exemple un site vitrine simple avec une page d'accueil et une page d'administration. La page d'administration n'est accessible que via une connexion sécurisée.

On peut croire que cela suffit mais désolé de vous décevoir, ce n'est pas le cas.

En effet, on peut accéder à l'espace d'administration via l'url. Il va donc falloir restreindre l'accès à cette page grâce aux variables $_SESSION crées lors de la connexion sécurisée.

Pour cela il suffit d'ajouter ce code en haut de toutes les pages où l'on veut restreindre l'accès.

```php
session_start();
if (!$_SESSION) {
    header('location:login.php');
}
```