# _ Les différents types d'attaques

## SOMMAIRE

## Présentation

Aujourd'hui, PHP est le langage de programmation le plus apprécié pour rendre les pages d'un site internet dynamiques. Cela étant, il présente certaines brèches de sécurité en raison du fait qu'avec ce langage, les serveurs web sont accessibles au public. Découvrez les failles que l'on rencontre le plus fréquemment chez les sites internet développés avec PHP ainsi que les méthodes d'attaque les plus utilisées par les hackers

## Injection SQL

Les injections SQL consiste à utiliser des données qui vont modifier une requête SQL de bases pour un autre usage que celui prévu initialement. Ces failles sont possibles lorsque les données d'entrées ne sont pas protégées. Imaginons par exemple une requête SQL qui permet de sélectionner l'identifiant d'un utilisateur à partir de son login et de son mot de passe.

```
SELECT identifiant
FROM utilisateur
WHERE login = 'nom_utilisateur' AND password = 'XC5AF32';
```

Cette requête permet de récupérer l'identifiant de l'utilisateur si le login et le mot de passe sont corrects. S'ils ne sont pas corrects, la requête ne renvoie rien et l'application peut donc indiquer à l'utilisateur qu'il a dû se tromper de mot de passe.

Dans cette requête, le login est entré par l'utilisateur. Or, si celui-ci indique que son login est "admin' —" alors il peut se connecter même sans avoir le bon mot de passe. Voici à quoi ressemblera la requête avec un tel nom d'utilisateur:

```
SELECT identifiant
FROM utilisateur
WHERE login = 'admin' --' AND password = 'faux_password';
```

Les 2 tirets signifient que le reste de la requête est en commentaire. Dès lors, il est possible de se connecter sur le compte de l'administrateur et d'avoir accès à toutes les données sensibles d'un site ou d'une application

## Solution pour se protéger des injections SQL

Pour éviter de telles failles avec le langage PHP, il convient de nettoyer les données d'entrées avant de les utiliser dans des requêtes SQL. Il est possible d'utiliser PDO pour éviter ces erreurs, mais quand ce n'est pas possible il convient juste d'utiliser les fonctions **mysqli_real_escape_string()** ou **mysqli_real_escape_string()** (pour les anciennes versions de PHP). Ces fonctions protègent les caractères spéciaux d'une chaîne de caractères pour que cette chaîne puisse être utilisée dans une requête.

Ci-dessous on retrouve une fonction PHP qu'il convient d'appeler pour toutes les données à insérer dans une requête.

En PHP 5

```php
function sanitize_string($str) {
      if (get_magic_quotes_gpc()) {
            $sanitize = mysqli_real_escape_string(stripslashes($str));
      } else {
            $sanitize = mysqli_real_escape_string($str);
      }
      return $sanitize;
}
```

Avantages de la fonction

Cette fonction est à utiliser sur chaque **$_GET** ou **$_POST** qui iront dans une requête SQL. Elle transforme notamment le guillemet simple en son équivalent en entité HTML. Il est pratique d'utiliser une telle fonction car si un jour le système de base de données est modifié, il suffit juste de changer la fonction **mysqli_real_escapte_string()** à l'intérieur au lieu de faire des modifications dans tout le reste du code.

## Faille XSS

XSS vient de Cross-Site Scripting et comme l'acronyme CSS était déjà pris pour Cascading Style Sheets, on a utilisé un X pour « cross » (croix en anglais).

La faille consiste à injecter un script arbitraire dans une page pour provoquer une action bien définie. Les autres utilisateurs exécutent le script sans s'en rendre compte dès l'ouverture de la page.

Cross veut également dire traverser, car l'un des buts de la faille est d'exécuter un script permettant de transmettre des données depuis un site vers un autre.

Ce problème se situe principalement au niveau des cookies, car on peut par exemple récupérer les cookies d'un site A depuis un site B. On peut ainsi récupérer les cookies de n'importe qui, même de l'administrateur d'un site.

Notez par ailleurs qu'on peut exploiter la faille XSS en JavaScript mais aussi avec d'autres langages.

On distingue deux types de failles XSS:

- XSS permanent

C'est lorsque le script est stocké sur le serveur externe (base de données). Il est donc récupéré et exécuté à tous moments sur le site par n'importe quel utilisateur.

- XSS non permanent

Le script est souvent intégré à une URL et est exécuté sans être stocké sur un serveur.

On peut distinguer plusieurs possibilités non exhaustives d'exploitation de cette faille :

- Une redirection de la page afin de nuire aux utilisateurs ou pour tenter une attaque de phishing.
- Voler des sessions ou des cookies. (Donc se faire passer pour un autre utilisateur pour exécuter des actions)
- Rendre le site inaccessible en utilisant des alertes en boucle ou tout autre moyen nuisible.

## Solution pour se protéger des injections SQL

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 ceux que nous avons vus plus haut s'exécutent avec tout le mal qui s'ensuit.

Voici un exemple d’utilisation de cette fonction :

<?php echo htmlspecialchars($_POST['nom']); // echo affiche les données sur une page, alors on protège l'affichage avec la fonction htmlspecialchars?>

Au possible, il faut placer des cookies avec le paramètre **HttpOnly,** empêchant leur récupération avec JavaScript

### Est ce-que mon site est faillible ?

On peut chercher les problèmes à partir de nos codes sources directement mais aussi, et plus simplement, en ajoutant un

```
'
```

à la fin d'une url faillible. Si une erreur apparaît sur votre site du type, c'est qu'il y a potentiellement un souci :

Erreur dans l'exécution de la requête 'SELECT * FROM galerie WHERE id = 2''. Message de MySQL : 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 ''' at line 1

L'erreur apparaît sur le site car " id=2' " est directement utilisé pour former la requête alors que " ' " est un caractère spécial de SQL. Ce qui provoque donc une erreur de syntaxe.

Cette erreur nous indique donc que les données entrées par les utilisateurs ne sont pas vérifiées du côté serveur, et qu'il y a donc de fortes chances qu'on puisse aller plus loin.

## Faille CSRF

L'attaque XSRF (ou CSRF pour Cross Site Request Forgery) est une attaque silencieuse mais redoutable. Quand elle survient, la victime ne s'en rend souvent pas compte, et ses dégâts ne se font sentir que plus tard.

Le principe de l'attaque consiste à faire exécuter une opération qui demande des privilèges spéciaux par une personne authentifiée et autorisée sans qu'elle ne s'en rende compte.

Sur le back-office d'un site web par exemple, l'administrateur et ses modérateurs ont des privilèges qui leur permettent de gérer, à leur guise, le contenu présenté en front-office. Ils peuvent alors créer, modifier, supprimer... depuis un tableau de bord facile à utiliser.

Si par exemple, l'administrateur souhaite supprimer une entrée d'une base de données, il va tout simplement cliquer sur le bouton (ou lien) approprié. Ce lien enverra les paramètres qui permettent d'identifier l'enregistrement à supprimer, et puisque l'administrateur est authentifié et dispose des privilèges nécessaires pour supprimer l'entrée, alors l'opération se passe sans encombre.

L'attaque XSRF consiste donc à forger le lien de la suppression (dans ce cas) et l'envoyer à l'administrateur, et c'est lui qui l'exécutera à son insu.

Outre la suppression, le XSRF peut causer d'autres dommages au contenu du site Web comme:

L'ajout ou la modification de contenu d'une manière non autorisée.

Exécution des commandes systèmes via l'interface Web qui peuvent avoir des conséquences désastreuses.

## Comment se protéger de la faille CSRF ?

Il est important que le serveur vérifie l'action. Celle-ci doit bien être réalisée par la personne souhaitant le faire. Lorsqu'un utilisateur doit réaliser une action sensible, il est d'usage de générer un jeton/token unique et périssable. Lorsque l'utilisateur va finaliser l'action sensible, le serveur doit comparer ce jeton. Si c'est ok, l'action est réalisée. Sinon l'action ne doit pas aboutir. Pour un cyber-criminel, il est impossible de deviner à l'avance ce jeton. Ceci constitue donc une contre mesure efficace.

Les principaux frameworks (RubyOnRails, Symphony, etc.) utilisent par défaut ce type de protection. Les développeurs doivent faire attention à ne pas désactiver ces protections.