TP - Injection SQL

1 Injection SQL :

L'injection SQL est une attaque qui consiste à insérer du code SQL malveillant dans une requête pour manipuler une base de données. Elle se produit lorsque les entrées utilisateurs ne sont pas sécurisées.

1.1 Exemple 1 : utilisation de OR

Si une requête SQL est construite directement avec une entrée utilisateur :

```
$nom = $_GET['nom'];
$sql = "SELECT * FROM utilisateurs WHERE nom = '$nom'";
$result = mysqli_query($conn, $sql);
```

Un attaquant peut entrer `' OR '1'='1`, ce qui transforme la requête en :

```
SELECT * FROM utilisateurs WHERE nom = '' OR '1'='1'
```

Cela renvoie tous les utilisateurs (faille d'accès aux données).

1.2 Exemple 2 : utilisation des caractères de commentaire

Si une requête SQL est construite directement avec une entrée utilisateur :

```
$login = $_POST["login"];
$password = $_POST["password"];
query = "SELECT spec_id, spec_mot_de_passe FROM spectateurs WHERE spec_login = '$login'
AND spec_mot_de_passe = '$password'";
$result = mysqli_query($conn, $query);
```

Un attaquant peut entrer `admin' # ` , ce qui transforme la requête en :

```
SELECT spec_id, spec_mot_de_passe FROM spectateurs WHERE spec_login = 'admin' # '
commantaire ignoré
```

Cela permet de se connecter en admin sans mot de passe.

1.3 Exercice :

Accéder au site vulnérable se situant sur le RPI :

http://172.xx.yy.zzz/VOTRENOM/WRONG/index.php

Essayer de se connecter aux comptes des différents utilisateurs en mettant en pratique les 2 exemples précédents.

Donner le login saisi :

Donner le mot de passe saisi :

2 Apport de cours :

2.1 Le mot clef : UNION

Le mot-clé **UNION** permet de **combiner** les résultats de **plusieurs requêtes** SELECT en une seule. Il est souvent utilisé pour fusionner des données provenant de différentes tables ayant des structures similaires.

```
SELECT colonne1, colonne2 FROM table1
UNION
SELECT colonne1, colonne2 FROM table2;
```

Exemple :

```
 SELECT fil_titre, fil_auteur FROM film
 UNION
 SELECT cin_nom, cin_ville FROM  cinema;
```

2.2 La table système : INFORMATION_SCHEMA

INFORMATION_SCHEMA est une base de données système intégrée à **MySQL**. Elle contient des métadonnées sur toutes les bases de données du serveur soit les tables, les colonnes, les index, les utilisateurs, les privilèges …

Cette requête permet de connaître le nom des tables de la base de données courante :

```
SELECT table_name FROM INFORMATION_SCHEMA.TABLES ;
```

Cette requête permet de connaître le nom des tables et leurs champs dans la base de données courante :

```
SELECT column_name, table_name FROM information_schema.columns;
```

2.3 Exemple 3 : visualisation du contenu des tables

Si une requête SQL est construite directement avec une entrée utilisateur, cette vulnérabilité permet de visualiser des champs dans la base de données.

```
$film = $_GET['film'];
$sql = "SELECT fil_nom, fil_auteur FROM film WHERE fil_nom LIKE ' %$film%'";
$result = mysqli_query($conn, $sql);
```

Il est donc assez facile de transformer cette requête en une requête :

```
 SELECT fil_titre, fil_auteur FROM film
 UNION
 SELECT column_name, table_name FROM information_schema.columns;
```

2.4 Exercice :

Accéder à la page : **http://172.xx.yy.zzz/VOTRENOM/WRONG/recherche_film.html**

Visualiser les films contenant ‘le’

Visualiser la liste des tables et leurs champs en mettant en pratique l’exemple 3.

Donner la recherche à saisir :

En utilisant les informations extraites de la précédente recherche.

Visualiser les login et les mots de passe des spectateurs.

Donner la recherche à saisir :

2.5 Point cours : Hash

Les mots de passe ne sont pas stockés en clair dans une base de données, car si un pirate accède à la base de données, il pourra voir tous les mots de passe des utilisateurs directement et les utiliser pour se connecter aux comptes des victimes, essayer ces mots de passe sur d'autres sites, les revendre sur le dark web.

Le **hashing** est une technique qui permet de convertir une donnée (ex. un mot de passe) en une empreinte unique et de longueur fixe. Cette transformation est réalisée par une fonction de hachage, qui est un algorithme mathématique irréversible. Il est impossible de retrouver la donnée originale à partir du hash. Une seule lettre différente génère un hash très différent.

Les algorithmes **MD5, SHA1 et SHA-256** sont des algorithmes de hachage couramment utilisés.

2.6 Point cours : Rainbow Table

Une **Rainbow Table** (table arc-en-ciel) est une technique utilisée pour retrouver un mot de passe à partir de son empreinte (hash). Elle repose sur un précalcul de millions de valeurs hachées pour accélérer les attaques contre les bases de données.

Sur le site https://crackstation.net/ qui utilise des « RainbowTable » retrouver les mots de passe des spectateurs.

Donner les noms et les mots de passe des utilisateurs :

3 Les bonnes méthodes de codage :

3.1 Le bon site :

Tester les pages nommées

http://172.xx.yy.zzz/VOTRENOM/GOOD/index.php
http://172.xx.yy.zzz/VOTRENOM/GOOD/recherche_film.html

Ces pages sont codées pour être durcies face aux attaques en injection.

3.2 Télécharger les pages PHP :

Lancer un **Windows PowerShell**, saisir PowerShell dans la barre de recherche.

1. Se déplacer dans votre répertoire Documents/
2. Créer un répertoire PHP
3. Se déplacer dans le répertoire PHP
4. Copier les fichiers contenu dans le RPI sur votre PC, via la commande :

```
PC C:\Users\Eleve\Documents\PHP>scp -r NOM@172.xx.yy.zz:/var/www/html/VOTRENOM/*  .
```

3.3 Comparaisons :

Dans le répertoire nommé PHP sur votre PC, comparer les fichiers contenus dans le répertoire GOOD et dans le répertoire WRONG.

Identifier les changements permettant au site web d'être endurci face aux attaques d'injection SQL.

Rendre ce compte-rendu complété sur Pronote

4 Résumé des bonnes méthodes de codage :

Pour se protéger d'une attaque par injection SQL :

- Utiliser des requêtes préparées (paramétrées)

Le mot `stmt` est une abréviation de "statement", qu'on peut traduire par instruction préparée. On le retrouve dans les scripts PHP qui interagissent avec une base de données de manière sécurisée.

`mysqli_prepare()` et `mysqli_stmt_bind_param()` permettent de réaliser des requêtes sécurisées.

- Ne jamais afficher d'erreurs SQL aux utilisateurs
- Limiter les permissions SQL uniquement les droits nécessaires
- Filtrer et valider les entrées des utilisateurs

Si un pirate accède à la base de données, il pourra voir tous les mots de passe des utilisateurs directement et les utiliser pour se connecter aux comptes des victimes et également essayer ces mots de passe sur d'autres sites (attaque credential stuffing).

De nombreuses lois et normes (RGPD en Europe, PCI-DSS pour les paiements en ligne) **interdisent** le stockage des mots de passe en clair et exigent leur protection par **hachage**.

Les algorithmes de hachage MD5 et SHA-1 sont obsolètes !

Pour bien stocker les mots de passe :

- Utiliser un algorithme de hachage sécurisé (bcrypt, Argon2, PBKDF2).
- Ajouter un sel unique à chaque mot de passe avant de le hacher.
- Ne jamais réutiliser un même sel pour plusieurs utilisateurs.