

Injections SQL, introduction

- Définitions:

- Injection SQL : Exploiter les failles de sécurité par de personnes non autorisées et malintentionnées.
- Injection SQL : on en parle plus pour les applications Web
- Les principes d'injections SQL sont simples, alors leurs préventions est simple.

- Exemple: En PDO

`SELECT * from utilisateurs where nom = ?` – En théorie cette requête ramène les informations (mot de passe) d'un utilisateur dont le nom est en paramètre.

En principe, seules les personnes connaissant la valeur du paramètre nom pourront chercher les informations correspondantes

Injections SQL

Injection SQL peuvent se produire en modifiant une requête de façon à ce qu'elle soit toujours exécutée (retourne toujours vrai) en changeant la clause WHERE ou avec un opérateur UNION.

Imaginez maintenant que quelqu'un soit malintentionné remplace la requête par:

```
SELECT * from utilisateurs where nom ='Patoche' OR 1=1;
```

Comme 1=1 est tout le temps vrai, alors la requête va renvoyer les informations de tous les utilisateurs.

Ou encore

```
SELECT Description FROM produits  
WHERE Description like '%Chaises'
```

- *UNION*

```
SELECT username FROM dba_users  
WHERE username like '%'
```

Injections SQL

Les attaques par injection de code tentent d'ajouter des instructions SQL supplémentaires ou des commandes à l'instruction SQL existante .

Ce type d'attaque est fréquemment utilisé contre les applications Microsoft SQL Server (à cause de sa commande EXECUTE), mais fonctionne rarement avec une base de données Oracle

- En PL / SQL et Java , Oracle ne prend pas en charge plusieurs instructions SQL par requête de base de données . Ainsi , l'attaque d'injection commune suivante ne fonctionnera pas contre une base de données Oracle via un PL / SQL ou une application Java. Cette déclaration se traduira par une erreur

```
SELECT * FROM users
```

```
WHERE username = 'bob' and PASSWORD = 'mypassword';
```

```
DELETE FROM users
```

```
WHERE username = 'admin';
```

Par contre si la requête est placée dans un bloc anonyme, ceci est va causer un problème

Injections SQL

Les injections par appels de fonctions:

En principe l'utilisation des fonctions permet de ne pas faire d'injections SQL par ajout de commandes SQL. De plus les SGBD sont en principe sécurisé ce qui rend l'accès à ces fonctions difficile.

Par contre, les SGBD dont ORACLE fournissent certaines fonctions qui permettent la gestion du système, l'exploitation du réseau etc. Ces fonctions sont contenues dans des packages sécurisés (Administration). Ces fonctions peuvent-être utilisées pour attaquer la base de données.

Injections SQL

Comment s'en prévenir: ?

1. Valider toutes les entrées.
2. Vérifier le format des données saisies et notamment la présence de caractères spéciaux ;
3. Ne pas afficher de messages d'erreur explicites affichant la requête ou une partie de la requête SQL. Personnaliser vos message erreur.
4. Supprimer les comptes utilisateurs non utilisées, notamment les comptes par défaut ;
5. Eviter les comptes sans mot de passe ;
6. Restreindre au minimum les privilèges des comptes utilisés ;
7. Les procédures stockées (systèmes ou non) non nécessaires à l'application doivent- être supprimées ou avec un accès restreint.
8. Utiliser des procédures stockées.
9. En PDO, utiliser le `prepare()` à la place du `query()` et du `exec()` (Utiliser le `PDOStatement`)
10. ET surtout utiliser des variables de liaison