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 jdbc

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 chnageant 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 certaine fonctions qui permettent la gestion du système, l'exploitation du réseau etc. Ces fonctions sont contenues dans des packages sécurisé (Administration). Ces fonctions peuvent-être utilisées pour attaquer la base de données.

Injections SQL

Prévenir les injections SQL en JDBC,

- Utilisez le PreparedStatement au lieu du Statement
- Faire une liaison de variable (bind)→ requête avec ?

```
String name = request.getParameter("name");
PreparedStatement pstmt =
conn.prepareStatement("insert into EMP (ENAME) values ('" + name + "')");
pstmt.execute();
pstmt.close();
```

IL faut plutôt faire ceci:

```
PreparedStatement pstmt =
conn.prepareStatement ("insert into EMP (ENAME) values (?)");
String name = request.getParameter("name");
pstmt.setString (1, name);
pstmt.execute();
pstmt.close();
```

Injections SQL

En JDBC, utilisez les CallableStatement avec les procédures stockées et les fonctions.:

prepareCall("{call proc (?,?)}"); →Liaison de variables

Éviter d'utiliser le CallableStatement avec les bloc anonymes.

La protection la plus efficace contre les attaques par injection SQL est l'utilisation de variables de liaison. En plus cette pratique permettra également d'améliorer les performances des applications.

les variables bind (liaison) doivent être utilisées pour chaque instruction SQL, peu importe quand et où l'instruction SQL est exécutée. Ceci est la norme de codage interne d'Oracle et devrait également être la norme de votre organisation

Injections SQL

Les valeurs de vos variables Bind ne doivent pas servir de nom de tables ou de nom de colonnes.

Utiliser les variables de liaison dans la clause LIKE

```
String name = request.getParameter("name");

conn.prepareStatement("SELECT id FROM users WHERE name LIKE '%" + name + "%'");
```

pstmt = conn.prepareStatement("SELECT id FROM users WHERE name LIKE ?");

Injections SQL

Comment s’en prévenir: ?

- Valider toutes les entrées.
- Vérifier le format des données saisies et notamment la présence de caractères spéciaux ;
- Ne pas afficher de messages d'erreur explicites affichant la requête ou une partie de la requête SQL. Personnaliser vos message erreur.
- Supprimer les comptes utilisateurs non utilisées, notamment les comptes par défaut ;
- Eviter les comptes sans mot de passe ;
- Restreindre au minimum les privilèges des comptes utilisés ;
- Les procédures stockées (systèmes ou non) non nécessaires à l’application doivent-être supprimées ou avec un accès restreint.
- ET surtout utiliser des variables de liaison