Sécurité des Bases de Données (M2 INIS)

INJECTION SQL ET CONTRES MESURES

Benjamin Nguyen

Objectif : Ce TP étudie l'attaque par injection SQL, les contre-mesures applicatives, les droits d'accès et le principe d'exposition minimale, et la protection des données de l'application par du chiffrement coté client. Le TP se déroule en 2 parties:

- **Partie 1.** Construire une application JAVA/JDBC sur la base de scripts SQL et du programme Java/JDBC fournis. Vous réaliserez ensuite un ensemble d'attaques par injection SQL, permettant 1) de passer l'étape d'identification, 2) d'identifier le nom et la version du SGBD sous-jacent, 3) de récupérer le contenu de l'ensemble des tables autorisées à l'application.
- **Partie 2.** Améliorer la sécurité de l'application à l'aide 1) de contre mesures spécifiques implantées dans le code de l'application, 2) de droits d'accès en appliquant le principe du moindre privilège, et 3) à base de primitives de chiffrement implantées coté client.

Préparation. (documents et scripts fournis: **http://www.benjamin-nguyen.fr/SBD/)**
Pour commencer, connecter vous à Oracle XE avec la ligne de commande SQL (lancer la ligne de commande SQL avec `sqlplus`) puis lancer le script VENDOR.SQL fourni (commande: `START <chemin-du-fichier-script.sql>`).
La base suivante est crée et peuplée (les clé primaires sont soulignées, les clés étrangères sont en italique):

Table **vendeur** : VEN (NumVen, NomVen, Pays, Pwd)
Table **commande** : COM (NumCom, *NumCli*, FraisPort, AnCom);
Table **détail** : DET (*NumCom*, *NumPro*, Qte, Remise)
Table **produit :** PRO (NumPro, *NumFou*, NomPro, TypePro)
Table **fournisseur** : FOU (NumFou, NomFou, Pays, Tel);

Les champs sont décrits comme suit:

AnCom : Année de commande (ex 1992)
FraisPort : Frais de port
NomVen : Nom du vendeur
NomFou : Nom du fournisseur
NomPro : Nom du produit
NumVen : Numéro de vendeur
NumCom : Numéro de la commande
NumFou : Numéro de fournisseur
NumPro : Numéro de produit
Pays : Pays du vendeur ou du fournisseur
Qte : Quantité commandée
Remise : Remise effectuée
Tel : Téléphone du fournisseur
TypePro : Type de produit
Pwd: mot de passe du vendeur

Ensuite, ouvrir le programme exemple dans Eclipse. Pour cela, télécharger le package du TP (archive fournie sur le site du cours) et décompresser le dans c:\TPJAVA. Suivre les étapes suivantes:

1. Télécharger le package du TP et le décompresser dans c:\TPJAVA:
2. Ouvrir une perspective **Java (default)**
3. File/New/Project puis choisir Java Project
4. mettre un nom (e.g., Essai) puis cocher "create project from existing source" et "browser" pour arriver sur C:\TPJAVA\TestDB puis **FINISH**
5. Testez la compilation et l'exécution du programme. Il vous faudra mettre le bon couple login/password

Vérification: Modifiez les paramètres de connexion dans le code fourni, compilez le et testez le. Le contenu de la table VEN doit apparaître dans la console.

PARTIE 1. Conception de l'application base de données et injection SQL.

Q1 : **Connexion à la base depuis l'application Java/JDBC.**
Modifier le code de TestDB.java (fourni), afin de construire un programme qui :

- Se connecte à la base de données comme l'utilisateur «system»
- Demande un login (saisi au clavier) et un mot de passe (saisi au clavier) et va vérifier dans une table VEN l'existence du couple login/password (**Attention: c'est le numéro du vendeur NUMVEN qui lui sert de login, et le contenu de la colonne PWD qui lui sert de mot de passe**)
- Rejette l'accès si le couple n'apparaît pas dans la table.

Q2 : **Affichage des commandes**
Compléter le programme précédent (i.e. n'utilisez pas d'autre méthode d'interrogation de la base) pour, si l'accès n'est pas rejeté, permettre au vendeur de visualiser la liste de ses ventes (commandes), en affichant pour chaque vente le nom du produit, la quantité, et le numéro de la commande correspondante, triés par numéro de commande et date (ANCOM).

Q3 : **Contournement du mécanisme d'authentification**
Par injection de code SQL, contournez le mécanisme d'authentification. L'attaque devra permettre de passer l'étape d'authentification sans modifier le programme Java, ni apparaît un couple login/mot de passe existant dans la base.

Q4 : **Identification du système SGBD cible**
Par injection de code SQL, identifier le produit et la version du SGBD cible. (les informations sont dans la vue `v$version`)

Q5 : **Injection SQL - accès à la liste des objets autorisées à l'application**
Par injection de code SQL, interroger le dictionnaire de données pour récupérer la liste des objets dans la base de données (notamment les tables et vues) auxquelles l'application que vous attaquez a droit (tables **all_tables** et **all_views**).

Q6 : **Injection SQL - accès au contenu des objets autorisés**
Par injection de code SQL, récupérer le contenu de certains objets de votre choix accessibles à l'application.

Partie 2. Contre-mesures.
Q7 : **Contre-mesures applicatives : tests spécifiques**
Corrigez votre programme pour qu'il soit résistant à ces attaques, en ajoutant des mesures de contrôles/tests spécifiques pour empêcher l'injection SQL, et en utilisant des API plus restrictives de JDBC (requêtes paramétrées, interface JDBC "`PreparedStatment`").

Q8 : **Principe du moindre privilège - droits d'accès**
Limiter les droits de l'application de manière à ne laisser accessible que le minimum d'information.
Notamment, vous pourrez supprimer l'accès au dictionnaire de données, et affecter les privilèges minimum aux vendeurs (l'accès à ses commandes), de manière à ce que chaque vendeur ne voit que les informations qui le concerne.

Q9 : **Hachage/chiffrement des mots de passe**
Modifier votre programme pour qu'il demande une clé de chiffrement à l’utilisateur, et utilise cette clé pour chiffrer (resp. déchiffrer) certaines données de la base modifiées/insérées (resp. accédées) par l'utilisateur.