Sécurité des systèmes d'information

TD 1 : Base de données statistiques et anonymat

Ce TP est une introduction aux BD statistiques et à l'anonymat des données sur OracleXE, .

Récupérez l'archive tp_anonymization.zip

Pour ce TP, vous utiliserez exclusivement la ligne de commande "sqlplus".

1 PARTIE 1 - Base de données statistiques

Dans cette partie, nous considérons la table statistique et la vue suivante :

- Table STATS : name varchar(50), gender char(1), age NUMBER, insurance varchar(50), leucocyte NUMBER. Le champ 'name' contient le nom d'un patient, 'gender' son genre (H ou F), 'age' son age, 'insurance' sa compagnie d'assurance (ex : MGEN, MATMUT, MAIF, ...), et 'leucocyte' son taux de leucocyte qui est considérée comme une données privée ultra sensible.
- Vue STATS_VIEW : gender char(1), insurance varchar(50), leucocyte NUMBER. Il s'agit d'une projection de la table STATS sur les colonnes gender, insurance et leucocyte.

1. Utilisation de fonctions statistiques
   - Connectez-vous en tant qu'administrateur (system).
   - Lancer le script STAT.sql
   - Connectez vous en tant qu'utilisateur USER_STAT (mot de passe : oracle)
   - Cet utilisateur a-t-il accès à la table STATS et à la vue STATS_VIEW ?
   - Cet utilisateur a accès aux fonctions WHERE_CLAUSE( ), ROW_COUNT( ), SUM_LEUCOCYTE( ) et peut interroger la vue STATS_VIEW uniquement au travers de ces fonctions. La première fonction permet de visualiser une clause WHERE formée à partir de la clause donnée en paramètre, la seconde donne le nombre de patients correspondant satisfaisant une clause WHERE donnée, et la troisième donne leur taux de leucocyte. Vous pouvez lancer ces fonctions via la requête SQL suivante : SELECT <propriétaire_fonction>.<nom_fonction> (' <contenu_clause_where> ') from dual ;
   - Ces fonctions prennent en paramètre une chaine de caractères (clause <contenue_clause_where>) indiquant la clause WHERE d'une requête SQL posée sur une vue STATS_VIEW. La clause <contenue_clause_where> pourra être par exemple : insurance = "MATMUT" (attention : le symbole " est obtenu par deux guillemets simples successifs). Vous devrez consulter la table ALL_OBJETS pour trouver l'utilisateur propriétaire de ces fonctions permettant l'appel.
   - Quel est le nombre de patients assurés à la MATMUT, et la somme de leur taux de leucocyte ?
   - Quel est le nombre total de patients dans la base ?
2. Attaque de fonctions statistiques non protégées

— Vous savez que le patient Dubois est un homme qui est assuré à la MGEN.
— Utiliser les fonctions statistiques pour obtenir le taux de leucocyte de Dubois.

3. Attaque de fonctions statistiques protégées
   — Connectez-vous en tant qu'administrateur.
   — Lancer le script STAT_PROTECT_1.sql
   — Connectez-vous en tant qu'utilisateur USER_STAT.
   — Relancer l'attaque conduite à la section précédente. Fonctionne-t-elle toujours ?
   — Cette fois, les fonctions sont protégées, et seuls les résultats statistiques obtenus à partir d'au moins 2 tuples sont accessibles. Trouver une séquence d'appels aux fonctions statistiques protégées permettant d'obtenir quand même le taux de leucocyte de Dubois.
4. Attaques de fonctions statistiques protégées
   — Connectez-vous en tant qu'administrateur.
   — Lancer le script STAT_PROTECT_2.sql
   — Connectez-vous en tant qu'utilisateur USER_STAT.
   — Relancer l'attaque conduite à la section précédente. Fonctionne-t-elle toujours ?
   — Cette fois, les fonctions sont protégées, et seuls les résultats statistiques obtenus à partir d'au moins 2 tuples et au plus n-1 tuples, sont accessibles. Trouver une séquence d'appels aux fonctions statistiques protégées permettant d'obtenir quand même le taux de leucocyte de Dubois.

2 PARTIE 2 - Anonymat et Chiffrement (package Oracle de chiffrement)

Dans cette partie, nous utiliserons le package DBMS_OBFUSCATION_TOOLKIT.

1. Pseudonymat par hachage
   — Connectez-vous en tant qu'administrateur.
   — Interrogez Oracle XE pour connaitre la liste des fonctions cryptographique disponibles dans le package de chiffrement.
   — Créez une nouvelle table STAT_ANONYM sur le même format que la table STATS.
   — Insérez le contenu de la table STATS dans la table STAT_ANONYM en remplaçant les noms des patients par une clé anonyme, obtenue par hachage cryptographique.
   — Pour hacher une chaine de caractères, vous pouvez utiliser :

```
SELECT rawtohex(
UTL_RAW.cast_to_raw (
DBMS_OBFUSCATION_TOOLKIT.md5(input_string => 'texte' ) ) )
FROM dual;
```

   Ou si t1 est est la table crée par : create table t1 (nom varchar2(32)) ;

```
SELECT rawtohex(
UTL_RAW.cast_to_raw (
DBMS_OBFUSCATION_TOOLKIT.md5(input_string => nom ) ) )
FROM t1;
```

— Donnez les droits nécessaires à l'utilisateur USER_STAT pour qu'il puisse accéder à la table STAT_ANONYM.

2. Attaque au pseudonymat par hachage
   — Connectez-vous en tant qu'utilisateur USER_STAT.
   — En supposant que cet utilisateur connaisse le nom d'un patient (par exemple le patient 'franck'), retrouvez toutes les données (age, genre, leucocyte) correspondant à ce patient dans la table STAT_ANONYM.
   — Même sans avoir cette information, essayez via des outils existant sur le web de déchiffrer les données hachées.
   — Lancer le script NOMS.sql
   — Une table NOMS contenant une série de noms potentiels vous est fournie dans le script NOMS.sql.
   — Retrouvez dans la table STAT_ANONYM chaque ligne correspondant à un nom de la table NOMS.
   — Combien de lignes de la table STAT_ANONYM parvenez-vous à dé-anonymiser ?
3. Pseudonymat par chiffrement
   — Connectez-vous en tant qu'administrateur.
   — Créez une table CLEF avec un champs VAL de type CHAR(8) dans laquelle vous stockerez une clé (secrète) que vous choisirez.
   — Le script CRYPT.sql crée les fonctions ENCRYPT et DECRYPT que vous pourrez utiliser. Compléter le script (chaque série de points d'interrogation est à remplacer par un nom de variable).
   — Lancer le script CRYPT.sql
   — Si le script a été correctement modifié, vous obtenez en sortie :
   Function created. PADING('TEST') ———————————— ———————— ———————— test—- Function created. ENCRYPT('TEST','12345678') ———————————— ———————————— 2FD0D87C2F466C21
   Function created. DECRYPT('2FD0D87C2F466C21','12345678') ———————— ———————— ———————————— test
   — Expliquer les modification apportées au script CRUPT.sql
   — Insérez le contenu de la table STATS dans la table STAT_ANONYM2 en remplaçant les noms des patients par une valeur anonyme obtenue par chiffrement du nom avec votre clé secrète.
4. Attaque au pseudonymat par chiffrement
   — Connectez-vous en tant qu'utilisateur USER_STAT.
   — En supposant que cet utilisateur connaisse le nom d'un patient comme précédemment, peut-il retrouver les données correspondant à ce patient dans la table STAT_ANONYM2 ?

— Lancer le script LISTING.sql
— Une table LISTING contenant une série de noms potentiels avec leur genre, age, et compagnie d'assurance, vous est fournie dans le script LISTING.sql.
— Dé-anonymiser la table STAT_ANONYM en utilisant LISTING.sql.
— Combien de lignes de la table STAT_ANONYM parvenez-vous à dé-anonymiser ?

5. 2-Anonymat
— Connectez-vous en tant qu'administrateur.
— Anonymiser (à la main) la table STAT_ANONYM2 de manière à la rendre 2 anonyme.
— Connectez-vous en tant qu'utilisateur USER_STAT.
— Montrer que l'attaque précédente n'est plus opérante.