Sécurité des données et gestion des utilisateurs

C'est quoi la sécurité d'une base de données

- La sécurité de la base de données est le processus, les outils et les contrôles qui sécurisent et protègent les bases de données contre les menaces accidentelles et intentionnelles.
- La sécurité des bases de données doit traiter et protéger les éléments suivants :
  - données de la base de données ;
  - système de gestion de la base de données (SGBD) ;
  - toutes les applications associées ;
  - serveur de base de données
  - infrastructure informatique et/ou réseau utilisée pour accéder à la base de données.

Types de sécurité de base de données

Il existe plusieurs formes et type de sécurité de bases de données, entre autres:

- **Contrôles des accès au réseau : L**'accès au réseau où est installé le serveur de bases de données, doit être limité au niveau minimum d'autorisations nécessaires.
- **Contrôles d'accès administratifs: U**n nombre minimal d'utilisateurs doit avoir accès à la base de données, cela se fait en créant des **comptes utilisateurs**.
- **Authentification:** Comme les contrôles d'accès, l'authentification permet d'identifier avec précision les utilisateurs avant qu'ils aient accès aux données. Elle peut prendre la forme de mots de passe, codes PIN, cartes magnétiques, etc.
- **Masquage des données** : il existe des logiciels de masquage des données cachent les informations en remplaçant des lettres et chiffres par d'autres caractères. Ainsi, si une personne non autorisée parvient à accéder aux données, elle ne pourra pas consulter les informations clés.

Constat et problématique

- La plupart du temps, l'accès au serveur se fait par le compte d'administration système **sans aucun mot de passe.**
- Aucun utilisateur ni rôle n'ayant été créé pour l'exploitation des données, c'est donc le **propriétaire** des bases par défaut (**root**) qui exerce ses droits. Ceux-ci étant illimités il est possible pour les utilisateurs finaux de modifier supprimer ou insérer dans toutes les bases y compris les bases système, les schémas, les données comme le code (procédures stockées et triggers notamment).
- Bien évidemment, cet état de fait laisse la porte grande ouverte aux attaques de serveurs SGBD,

Solution

- La solution consiste à établir :

1. Un accès sécurisé au serveur (Authentification)
2. Créer des utilisateurs dotés de privilèges différents pour l'exploitation des données des bases de données.

L'authentification

- La page d'authentification permet de saisir le nom d'un utilisateur et son mot de passe. L'utilisateur par défaut est **root** et sans mot de passe.
- Donc comment créer un utilisateur et comment lui attribuer un mot de passe?

Création d'un utilisateur

- **Syntaxe:**

```
CREATE USER 'login'@'hote' [IDENTIFIED BY 'mot_de_passe'];
```

- **Login**

  Le login est un simple identifiant. Il n'est pas obligatoire de l'entourer de guillemets, sauf s'il contient des caractères spéciaux comme - ou @. C'est cependant conseillé.

- **Hôte**

  L'hôte est **l'adresse de l'ordinateur à partir de laquelle l'utilisateur va se connecter**. Si l'utilisateur se connecte à partir de la machine sur laquelle le serveur MySQL se trouve, on peut utiliser **'localhost'**. Sinon, on utilise en général une **adresse IP** ou un **nom de domaine.**

Création des utilisateurs

- **Exemples:**

```
CREATE USER 'Stagiaire1'@'localhost' IDENTIFIED BY 'stagiaireOfppt';

CREATE USER 'Stagiaire2'@'194.28.12.4';

CREATE USER 'Utilisateur'@'arb.brab.net' IDENTIFIED BY 'azerty';
```

- Il est également possible de permettre à un utilisateur de se connecter à partir de plusieurs hôtes différents, en utilisant le joker %, on peut préciser des noms d'hôtes partiels ou permettre à l'utilisateur de se connecter à partir de n'importe quel hôte.

```
CREATE USER 'UnUtilisateur'@'%' IDENTIFIED BY 'abc123';
```

Renommer l'utilisateur

- Pour modifier l'identifiant d'un utilisateur (login et/ou hôte), on peut utiliser*

```
RENAME USER ancien_utilisateur  TO  nouvel_utilisateur
```

- Exemple :

```
RENAME USER 'Stagiaire1'@'localhost'  TO 'FirstStag'@'localhost';
```

Mot de passe du compte utilisateur

- Le mot de passe de l'utilisateur est donné par la clause IDENTIFIED BY. Ce qui signifie que l'utilisateur peut se connecter sans mot de passe. **Évitez au maximum les utilisateurs sans mot de passe.**

- Lorsqu'un mot de passe est précisé, il n'est pas stocké tel quel dans la table *mysql.user*. Il est d'abord hashé, et c'est cette valeur hashée qui est stockée.

Modifier le mot de passe

- Pour modifier le mot de passe d'un utilisateur, on peut utiliser la commande **SET PASSWORD** . Cependant, cette commande ne hashe pas le mot de passe automatiquement. Il faut donc utiliser la fonction **PASSWORD()**.

```
SET PASSWORD FOR 'Stagiaire2'@'194.28.12.4' = PASSWORD('ABC123');
```

- Dans les nouvelles versions:

```
ALTER USER 'user'@'hostname' IDENTIFIED BY 'newPass';
```

Supprimer un utilisateur

- Pour supprimer un utilisateur on se sert de la commande **DROP**.

```
DROP USER 'login utilisateur'@'hote'
```

Afficher la liste des utilisateurs

- Pour pouvoir voir les utilisateurs crées au niveau du serveur:

```
Select * from mysql.user
```

Les privilèges (Introduction)

- Lorsque l'on crée un utilisateur avec CREATE USER, celui-ci n'a au départ **aucun privilège**, **aucun droit**. En SQL, avoir un privilège, c'est avoir l'autorisation d'effectuer une action sur un objet(table, procédure,..).

- Un utilisateur sans aucun privilège ne peut rien faire d'autre que se connecter. Il n'aura pas accès aux données, ne pourra créer aucun objet (base/table/procédure/autre) ni en utiliser.

Ajout de privilèges

- Pour pouvoir ajouter un privilège à un utilisateur, il faut posséder le privilège **GRANT OPTION**. Pour l'instant, seul l'utilisateur "root" le possède. Étant donné qu'il s'agit d'un privilège un peu particulier, nous n'en parlerons pas tout de suite.

Ajout de privilèges

- Syntaxe:

```
GRANT privilege [(liste_colonnes)] [, privilege [(liste_colonnes)], ...]
ON [type_objet] niveau_privilege
TO 'utilisateur'@'hote' ;
```

- privilege : le privilège à accorder à l'utilisateur (SELECT, EXECUTE…)
- (liste_colonnes) : facultatif - liste des colonnes auxquelles le privilège s'applique
- niveau_privilege : niveau auquel le privilège s'applique
- type_objet : en cas de noms ambigus, il est possible de préciser à quoi se rapporte le niveau, TABLE ou PROCEDURE.

Privilèges relatifs à l'utilisation des données

- Les privilèges SELECT, INSERT, UPDATE et DELETE permettent aux utilisateurs d'utiliser ces mêmes commandes.

Les privilèges

Privilège	Action autorisé
SELECT	Permet à l'utilisateur d'utiliser la commande SELECT
UPDATE	Permet à l'utilisateur d'utiliser la commande UPDATE
DELETE	Permet à l'utilisateur d'utiliser la commande DELETE
INSERT	Permet à l'utilisateur d'utiliser la commande INSERT
CREATE TABLE	Permet la création des tables et bases de données
ALTER	Permet la modification de tables (avec ALTER TABLE)
DROP	Permet de supprimer de tables,
CREATE ROUTINE	Création de procédures stockées (ou fonctions stockées)
EXECUTE	Exécution de procédures stockées (ou de fonctions stockées)
TRIGGER	Création et suppression de triggers
CREATE USER	Gestion d'utilisateur (commandes CREATE USER, DROP USER, RENAME USER et SET PASSWORD)

Niveau de privilège

- Signifie les bases de données ou les objets de bases de données auxquels les privilèges vont être appliqués.

Niveau	Application du privilège
.	Privilège global : s'applique à toutes les bases de données, à tous les objets. Un privilège de ce niveau sera stocké dans la table *mysql.user*.
*	Si aucune base de données n'a été préalablement sélectionnée (avec `USE nom_bdd`), c'est l'équivalent de . (privilège stocké dans *mysql.user*). Sinon, le privilège s'appliquera à tous les objets de la base de données qu'on utilise (et sera stocké dans la table *mysql.db*).
*nom_bdd.**	Privilège de base de données : s'applique à tous les objets de la base *nom_bdd* (stocké dans *mysql.db*).
nom_bdd.nom_table	Privilège de table (stocké dans *mysql.tables_priv*).
nom_table	Privilège de table : s'applique à la table *nom_table* de la base de données dans laquelle on se trouve, sélectionnée au préalable avec `USE nom_bdd` (stocké dans *mysql.tables_priv*).
nom_bdd.nom_routine	S'applique à la procédure (ou fonction) stockée *nom_bdd.nom_routine* (privilège stocké dans *mysql.procs_priv*).

Exemple 1

- On crée un utilisateur 'stagiaire'@'localhost', en lui donnant les privilèges SELECT, INSERT et DELETE sur la table *TABLE1*, et UPDATE sur les colonnes *nom*, *genre* et *commentaires* de la table *TABLE2*.

```
CREATE USER 'stagiaire'@'localhost' IDENTIFIED BY 'exemple2023';
GRANT SELECT,
      UPDATE (nom, prenom, `date de naissance`),
      DELETE,
      INSERT
ON gestion_commandes.employes
TO 'stagiaire'@'localhost';
```

Exemple 2

- On accorde le privilège SELECT à l'utilisateur 'stagiaire'@'localhost' sur la table *Gestion_commandes.clients*.

```
GRANT SELECT
ON gestion_commandes.clients
TO 'stagiaire'@'localhost';
```

Exemple 3

- On accorde à 'stagiaire'@'localhost' le privilège de créer et d'exécuter des procédures stockées dans la base de données *gestion_commandes*.

```
GRANT CREATE ROUTINE, EXECUTE
ON gestion_commandes.*
TO 'stagiaire'@'localhost';
```

Afficher la liste des privilèges d'un utilisateur

- En utilisant la syntaxe suivante, on aura la possibilité de vérifier les privilèges d'un utilisateur:

```
Show grants  for 'utilisateur'@'hote' ;
```

Révocation de privilèges

- Pour retirer un ou plusieurs privilèges à un utilisateur, on utilise la commande **REVOKE**.

```
REVOKE privilege [, privilege, ...]
ON niveau_privilege
FROM utilisateur;
```

- Exemple :

```
REVOKE DELETE ON Gestion_commandes.employes
FROM 'stagiaire'@'localhost';
```

Le privilège ALL

- Le privilège **ALL** (ou **ALL PRIVILEGES**), comme son nom l'indique, représente tous les privilèges. Accorder le privilège ALL revient donc à accorder tous les droits à l'utilisateur sauf **GRANT OPTION et aussi create user.**

```
GRANT ALL
ON gestion_commandes.employes
TO 'stagiaire'@'localhost';
```

```
REVOKE ALL privileges
ON Gestion_commandes.employes
FROM 'stagiaire'@'localhost';
```

Le privilège **GRANT OPTION**

- Un utilisateur ayant le privilège **GRANT OPTION** est autorisé à utiliser la commande **GRANT**, pour accorder des privilèges à d'autres utilisateurs.
  Ce privilège n'est pas compris dans le privilège ALL. Par ailleurs, un utilisateur ne peut accorder que les privilèges qu'il possède lui-même.
- Exemple:

```
GRANT SELECT, UPDATE, INSERT, DELETE, GRANT OPTION
ON gestion_commandes.*
TO 'Ofppt'@'localhost' IDENTIFIED BY 'ofppt2020';
```

Les rôles

- En plus d'accorder des autorisations à des utilisateurs individuels, dans MySQL, il est également possible de créer des rôles et d'accorder des autorisations aux rôles, puis d'attribuer des rôles aux utilisateurs.

- Cela facilite grandement la gestion des groupes d'utilisateurs avec des autorisations similaires. Pour créer un rôle de développeur Web, nous pouvons fournir la requête suivante :

```
CREATE ROLE 'nom_role';
```

Les rôles

- Exemple:

```
CREATE ROLE ‘webdeveloper’;

GRANT SELECT ON mysql.user TO ‘webdeveloper’;

GRANT ‘webdeveloper’ TO ‘rh_salmi’@’entrbc.com.net’;
```

- Un seul rôle peut être affecté à la fois à un utilisateur.

Activer le rôle pour un utilisateur

- Il faut que l'utilisateur se connecte, et indiquer à MySQL quels rôles vous souhaitez utiliser à l'aide de la requête suivante:

```
SET ROLE 'webdeveloper'
```

Les rôles

- Parfois c'est mieux d'affecter un rôle à un utilisateur lors de sa création.

```
CREATE USER ‘u2’@’%’ INDENTIFIED BY ‘foobar’ DEFAULT ROLE ‘webdeveloppper’;
```

Exclure un rôle d'un utilisateur

- Pour exclure un utilisateur d’un rôle, on peut utiliser la commande :

```
REVOKE ‘webdeveloppeur’ FROM ‘med34’@’localhost’
```