Fiche pratique : Sécurisation MySql

mercredi 7 mai 2025 17:30

Objectifs : Mettre en place la **sécurité logique MySQL**, la **structuration des accès** et la **configuration côté application** :

Introduction

Lors d'une installation par défaut, **MySQL n'a pas de mot de passe universel prédéfini** pour le compte administrateur (root). Bien souvent, l'installation se fait sans mot de passe.
Cette mise en œuvre de base quoique pratique est trop permissive en matière de sécurité. La sécurisation des données (MySQL) est différente en termes de processus selon les architectures. Dans une architecture N1, le serveur est unique "stand-alone", il devra être sécurisé. Dans une architecture N2 qui contribue à diminuer les surfaces d'attaques, il faudra aussi sécuriser les serveurs.
Cette fiche dédiée à la partie mysql aborde donc les bonnes pratiques selon les architectures.

I. Serveur unique (Architecture N1 : stand-alone)

Objectif :

Séparer les rôles applicatifs, base de données et administration pour mieux contrôler les accès, les privilèges et la sécurité du moteur MySQL.

Contexte :

Apache, PHP, MySQL sont sur **une seule machine**. On souhaite **sécuriser l'installation locale**.

Problématiques :

- L'accès au moteur et à l'ensemble des bases de données doit se faire avec un administrateur "root" qui doit se connecter en local ou à un serveur unique référencé
- L'accès à la base de données "cyber" doit se faire par un utilisateur "applicatif", celui-ci ne doit pas pouvoir se connecter au moteur, ni à d'autres bases de données
- Les programmes fonctionnent en local, on doit restreindre les connexions externes

Étapes de sécurisation de Mysql :

1. **Exécution de mysql_secure_installation** pour :
    - Prérequis : sudo mysql_secure_installation
    - définir un mot de passe root,
    - supprimer les utilisateurs anonymes,
    - supprimer la base test,
    - désactiver les connexions root distantes.

2. **Restriction des connexions externes (fichier mysqld.cnf)**
    - Par défaut, MySQL est configuré pour **n'écouter que sur l'interface locale** (127.0.0.1), ce qui **empêche toute connexion distante => à vérifier**
    - sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf
    - Ajout ou modification de la ligne : bind-address = 127.0.0.1
    - sudo systemctl restart mysql

3. **Création d'un utilisateur dédié à l'application** :
    - Prérequis : sudo mysql

- CREATE USER 'appuser'@'localhost' IDENTIFIED BY 'motdepasse';
  GRANT SELECT, INSERT, UPDATE, DELETE ON cyber.* TO 'appuser'@'localhost';
    - Et création d'un **administrateur spécifique (optionnel)**
- CREATE USER 'adminsql'@'localhost' IDENTIFIED BY 'motdepasse';
- GRANT ALL PRIVILEGES ON *.* TO 'adminsql'@'localhost' WITH GRANT OPTION;
- FLUSH PRIVILEGES;
    - Et supprimer/désactiver l'utilisateur root (optionnel)
- Supprimer : DROP USER 'root'@'localhost';
- Désactiver : RENAME USER 'root'@'localhost' TO 'root_old'@'localhost';

Étapes de configuration de PHP

1. **Connexion PHP (ex. mysqli)** :
   $conn = new mysqli("localhost", "appuser", "motdepasse", "cyber");
   a. BP : Utiliser Mysqli avec le fichier config.php
   b. BP ++ : Utiliser PDO avec le fichier config.php

2. **Sécurisation du fichier config.php** (Bonnes pratiques)
   - Accès réservé au serveur (non exposé via le web)
   - Inclusion par le script PHP, non accessible en direct
   - // config.php
     - define('DB_HOST', 'localhost');
     - define('DB_USER', 'utilisateurweb');
     - define('DB_PASS', 'MotDePasse');
     - define('DB_NAME', 'ma_base');
   - // programme en php
     - require 'config.php';
     - $conn = new mysqli(DB_HOST, DB_USER, DB_PASS, DB_NAME);
   - Le fichier config.php contenant des identifiants sensibles ne doit pas être dans un répertoire accessible via le navigateur.
   - Droit spécifique (Lecture seule, non exposé)
     - <!> Non exposé : **pas dans** /var/www/html
     - <!> Non exposé : /home/app/config.php
     - <!> Non exposé : utiliser .htaccess
       - <Files "config.php">
       - Require all denied
       - </Files>
     - <!> Non exposé : Restreindre les droits du fichier
       - chmod 640 config.php
       - chown www-data:www-data config.php

Résumé d'une configuration mono-serveur

- **Principes à retenir :**
  - 1 utilisateur MySQL = 1 application
  - 1 base = 1 application
  - Pas d'utilisation de root dans les scripts
  - Utilisation de noms d'hôtes pour découpler les accès des adresses IP
  - Accès administrateur réservé à une machine spécifique

II. Architecture distribuée N2 (multi-serveurs)

Objectif :

Séparer les rôles applicatifs, base de données et administration pour mieux contrôler les accès, les privilèges et la sécurité du moteur MySQL.

Contexte :

- Plusieurs **serveurs applicatifs** (PHP/Apache) distincts
- Un **serveur MySQL centralisé**
- Un **serveur d'administration** dédié aux opérations MySQL

Problématiques :

- L'accès au moteur et à l'ensemble des bases de données doit se faire avec un administrateur "root", connecté en local ou depuis un hôte autorisé.
- L'accès à la base de données "cyber" doit se faire par un utilisateur "applicatif", celui-ci ne doit pas pouvoir se connecter au moteur, ni à d'autres bases de données
- Cet utilisateur "applicatif" ne peut se connecter qu'à partir du serveur applicatif
- Les programmes fonctionnent sur le serveur applicatif, ils ne peuvent exécuter des requêtes qu'à partir du serveur applicatif, on doit donc interdire l'exécution de programmes à partir d'autres serveurs.

Règles de configuration :

- **Utilisation de noms d'hôtes (ou DNS) au lieu d'adresses IP**
  - Cela permet d'éviter de modifier les permissions MySQL en cas de changement d'IP.
  - <!> Fichiers hosts à jour sur tous les serveurs
- **Création d'utilisateurs par serveur applicatif**
- **Chaque utilisateur "applicatif"** ne peut se connecter **que depuis son serveur applicatif** et n'a accès **qu'à sa base dédiée**.
- **L'administrateur** a un accès complet, mais **limité à un hôte spécifique** (ex. srv_adminwin).
- Les serveurs applicatifs (PHP) doivent être configurés pour accéder au serveur de base de données (srv_db), à la base de données (appx_db) avec des utilisateurs "applicatifs".
- Les connexions peuvent être faites en mysqli ou PDO, selon les bonnes pratiques du projet.

Normalisation des serveurs et bases

Désignation	Rôle
srv_app1	Serveur 1 applicatif (Apache + PHP) – Peut être remplacé par son @IP
srv_app2	Serveur 2 applicatif (Apache + PHP) – Peut être remplacé par son @IP
srv_db	Serveur de base de données MySQL centralisée
srv_adminwin	Serveur d'administration distant pour gérer le moteur MySQL
app1_db	Base de données dédiée à srv_app1, hébergée sur srv_db
app2_db	Base de données dédiée à srv_app2, hébergée sur srv_db

Étapes de configuration MySQL sur srv_db

1. Fichier mysqld.cnf

a. Par défaut, MySQL est configuré pour **n'écouter que sur l'interface locale** (127.0.0.1), ce qui **empêche toute connexion distante**, même si les comptes MySQL sont correctement créés.
b. Fichier : # Par défaut :
   i. bind-address = 127.0.0.1
c. Fichier : À modifier pour accepter les connexions des serveurs applicatifs :
   i. bind-address = 0.0.0.0

d. Pour **restreindre plus finement les interfaces** :
   i. bind-address = 192.168.10.50 # @IP du srv_db sur le réseau local privé

2. **Création des bases (rappel)**
   a. CREATE DATABASE app1_db;
   b. CREATE DATABASE app2_db;
3. **Création des utilisateurs applicatifs avec restriction par hôte**
   a. CREATE USER 'app1_user'@'**srv_app1**' IDENTIFIED BY 'motdepasse1';
   b. GRANT SELECT, INSERT, UPDATE, DELETE ON app1_db.* TO 'app1_user'@'**srv_app1**';
   c. CREATE USER 'app2_user'@'**srv_app2**' IDENTIFIED BY 'motdepasse2';
   d. GRANT SELECT, INSERT, UPDATE, DELETE ON app2_db.* TO 'app2_user'@'**srv_app2**';
4. **Création de l'administrateur distant limité à srv_adminwin**
   a. CREATE USER 'adminsql'@'srv_adminwin' IDENTIFIED BY 'adminpass';
   b. GRANT ALL PRIVILEGES ON *.* TO 'adminsql'@'srv_adminwin' WITH GRANT OPTION;
5. **Sécurisation de l'utilisateur root**
   a. Accès local uniquement
      i. CREATE USER 'root'@'localhost' IDENTIFIED BY 'rootpass';
      ii. GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT OPTION;
   b. Accès possible depuis l'hôte admin uniquement (optionnel)
      i. CREATE USER 'root'@'srv_adminwin' IDENTIFIED BY 'rootpass';
      ii. GRANT ALL PRIVILEGES ON *.* TO 'root'@'srv_adminwin' WITH GRANT OPTION;

Étapes de configuration de PHP depuis les serveurs applicatifs

1. **Sur srv_app1 :** Fichier config.php
   a. define('DB_HOST', 'srv_db');
   b. define('DB_NAME', 'app1_db');
   c. define('DB_USER', 'app1_user');
   d. define('DB_PASS', 'motdepasse1');
2. **Sur srv_app2 :** Fichier config.php
   a. define('DB_HOST', 'srv_db');
   b. define('DB_NAME', 'app2_db');
   c. define('DB_USER', 'app2_user');
   d. define('DB_PASS', 'motdepasse2');

Résumé d'une configuration multi-serveurs

- Chaque **serveur applicatif dispose d'un utilisateur MySQL dédié** à sa base.
- La **résolution par noms d'hôtes (srv_app1, srv_db)** permet de découpler la configuration des IP.
- Les **droits sont strictement limités par utilisateur et base**.
- L'**administration distante** passe par un compte dédié, autorisé uniquement depuis une machine d'admin.
- Le compte root n'est **jamais utilisé dans les connexions applicatives**.

III. Connexion côté application (PHP)

Exemple de fichier config.php pour serveur applicatif :

- define('DB_HOST', 'srv_db'); // nom DNS du serveur MySQL
- define('DB_NAME', 'app1_db');
- define('DB_USER', 'app1_user');
- define('DB_PASS', 'motdepasse1');

Exemple de connexion mysqli :

- $conn = new mysqli(DB_HOST, DB_USER, DB_PASS, DB_NAME);

Exemple en PDO :

- $dsn = "mysql:host=" . DB_HOST . ";dbname=" . DB_NAME . ";charset=utf8mb4";
- $pdo = new PDO($dsn, DB_USER, DB_PASS);

IV. Compléments (Quelques trucs MySql)

- Fichier principal : /etc/mysql/my.cnf ou /etc/my.cnf
- Activation des logs :
  - [mysqld]
  - log_error = /var/log/mysql/error.log
  - slow_query_log = 1
  - slow_query_log_file = /var/log/mysql/slow.log
- Emplacements typiques :
  - /etc/my.cnf (CentOS, RedHat, anciennes versions)
  - /etc/mysql/my.cnf (Debian, Ubuntu)
- mysqld.cnf : fichier spécifique au service MySQL (mysqld)
  - Emplacement typique :
    - /etc/mysql/mysql.conf.d/mysqld.cnf (Ubuntu/Debian modernes)
- À retenir :
  - On évite de modifier my.cnf, on modifie mysqld.cnf à la place.
- **my.ini est l'équivalent de my.cnf sous Windows**
  - **Ordre de recherche :**
    - **%PROGRAMDIR%\MySQL\MySQL Server X.Y\my.ini (répertoire d'installation, par exemple C:\Program Files\MySQL\MySQL Server 8.0\)**
    - **%WINDIR%\my.ini**
    - **%WINDIR%\my.cnf**
    - **C:\my.ini**
    - **C:\my.cnf**

V. Compléments (Quelques trucs/cmd sous Linux)

- sudo mysql
- sudo grep 'temporary password' /var/log/mysqld.log
- Message : Can't connect to local MySQL server through socket... signifie que **le serveur MySQL ne tourne pas**, ou que **le client essaie de se connecter via un socket Unix qui n'existe pas**
  - **Statut :** sudo systemctl status mysql
  - Statut : sudo service mysql status
  - Start : sudo systemctl start mysql
  - Connexion :
    - mysql -u root
    - mysql -u root -h 127.0.0.1 -p
- Message : ERROR 1698 (28000): Access denied for user 'root'@'localhost'
  - sudo mysql
  - SELECT user, host, plugin FROM mysql.user WHERE user = 'root';
    - Auth_socket => NOK
  - ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY

'TonMotDePasseFort'; FLUSH PRIVILEGES; EXIT;
    - mysql -u root -p

VI. Compléments (Quelques trucs/cmd sous Windows)

- Mysql -u root -p
- Suppression utilisateurs anonymes : DELETE FROM mysql.user WHERE user = '';
- Supprimer la base test :
    - DROP DATABASE IF EXISTS test;
    - DELETE FROM mysql.db WHERE Db='test' OR Db='test_%';
- Recharger les privilèges :
    - FLUSH PRIVILEGES;
- Créer un nouvel administrateur et désactiver root
    - Même commandes SQL que sous Linux :
    - CREATE USER 'adminsql'@'localhost' IDENTIFIED BY 'MotDePasseFort';
    - GRANT ALL PRIVILEGES ON *.* TO 'adminsql'@'localhost' WITH GRANT OPTION;
    - FLUSH PRIVILEGES;

    - -- Optionnel : désactiver root
    - RENAME USER 'root'@'localhost' TO 'root_old'@'localhost';
- Fichier de configuration : My.ini
    - Bloque les connexions distantes :
        - Ajout ou modification de la ligne : bind-address = 127.0.0.1

VII. Compléments (Différences Linux/Windows)

Élément	**Linux**	**Windows**
Script mysql_secure_installation	Oui	Non
Fichier de config	/etc/mysql/my.cnf	C:\ProgramData\MySQL\...\my.ini
Redémarrage	systemctl restart mysql	via services.msc
Gestion des permissions	Identique (SQL)	Identique (SQL)
Log MySQL	/var/log/mysql/	C:\ProgramData\MySQL\...\Data

VIII. Adaptation à un environnement déjà installé

Objectif :

Adapter les procédures de sécurisation MySQL à un environnement existant sans perturber les services en place.

Contexte :

Contrairement à une installation neuve (stand-alone ou multi-serveurs), un environnement existant peut déjà contenir des utilisateurs, des privilèges en production, des applications en fonctionnement.

Principes à respecter :

- Ne jamais supprimer un utilisateur sans avoir audité ses usages.
- Ne pas modifier brutalement les fichiers de configuration sans prévoir un redémarrage maîtrisé.
- Procéder par étapes vérifiées et réversibles.
- Sauvegarder avant de commencer !

Étapes à suivre :

1. Audit des comptes existants

SELECT user, host, plugin FROM mysql.user;
SHOW GRANTS FOR 'utilisateur'@'hôte';

- Identifier les utilisateurs techniques, administratifs, applicatifs.
- Vérifier les hôtes autorisés (localhost, %, IP, noms).

2. Création des comptes sécurisés

- Ajouter un compte applicatif restreint (host spécifique, base dédiée).
- Ajouter un compte admin (limité à un hôte d'administration, si pertinent).

CREATE USER 'appuser'@'srv_app1' IDENTIFIED BY 'motdepasse';
GRANT SELECT, INSERT, UPDATE, DELETE ON cyber.* TO 'appuser'@'srv_app1';

3. Tests de fonctionnement

- Modifier temporairement les scripts de test pour utiliser le nouveau compte.
- Valider la connexion PHP (mysqli ou PDO), depuis les bons serveurs.

4. Réduction progressive des privilèges

- Supprimer les droits trop larges (ex : GRANT ALL ON *.*).
- Renommer l'utilisateur root (ou le bloquer) si plus utilisé dans les scripts.

RENAME USER 'root'@'localhost' TO 'root_old'@'localhost';

5. Sécurisation des fichiers de configuration

- Vérifier bind-address, logs, droits sur config.php.
- Appliquer les bonnes pratiques de sécurisation même si le système est déjà actif.

6. Planification des redémarrages (si nécessaire)

- Sauvegarder la configuration avant toute modification du fichier mysqld.cnf ou my.ini.
- Planifier les interruptions sur créneaux dédiés.