



COMPTE RENDU : TP GESTION DES DONNEES ET SECURITE

Auteur : TAMBOURA Dina

Numéro étudiant :22403684

Mail : Tambouradina@gmail.com

I. Introduction

Ce TP a pour objectif de :

1. Implémenter un modèle relationnel pour une base de données gérant les relations entre des clients et des agences immobilières.
2. Gérer les droits d'accès aux objets de la base pour garantir une sécurité organisationnelle.
3. Tester la sécurité des formulaires web contre les injections SQL et proposer des solutions pour éliminer les vulnérabilités.

Environnement de test :

- Conteneur Docker pour MySQL (version : latest).
- Conteneur Docker pour PHP avec Apache (version : PHP 7.4).
- Utilisation de SQLmap pour tester les injections SQL.
- Poste hôte : Ubuntu 22.04.

II. Partie I : Implémentation du Modèle Relationnel

1. Création des Tables

Nous avons créé quatre tables (« Client », « Agence », « Bien » et « Inscrit ») avec leurs contraintes associées.

```
mysql> SHOW Databases;
+-----+
| Database |
+-----+
| TP_IMMO  |
| information_schema |
| mysql    |
| performance_schema |
| sys      |
+-----+
5 rows in set (0.00 sec)

mysql> |
```

```
mysql> SHOW TABLES;
+-----+
| Tables_in_TP_IMMO |
+-----+
| Agence             |
| Bien               |
| Client             |
| Inscrit            |
+-----+
4 rows in set (0.01 sec)
```

Requêtes Utilisées :

1. TABLE CLIENT :

```
CREATE TABLE Client (
    IDC CHAR(3) PRIMARY KEY,
    NomPrenom VARCHAR(50) NOT NULL,
    Age INT NOT NULL,
    Salaire DECIMAL(10, 2) NOT NULL);
```

```
mysql> SELECT * FROM Client;
+----+-----+-----+-----+
| IDC | NomPrenom | Age | Salaire |
+----+-----+-----+-----+
| C01 | Alice Dupont | 30 | 2500.00 |
| C02 | Bob Martin | 45 | 3200.00 |
| C03 | Charlie Durand | 25 | 1800.00 |
+----+-----+-----+-----+
3 rows in set (0.00 sec)
```

2. TABLE AGENCE:

```
CREATE TABLE Agence (
    IDA CHAR(3) PRIMARY KEY,
    Nom VARCHAR(50) NOT NULL,
    Adresse VARCHAR(100) NOT NULL,
    Telephone VARCHAR(15) NOT NULL);
```

```
mysql> SELECT * FROM Agence
-> ;
```

IDA	Nom	Adresse	Telephone
A01	Agence Centre	123 Rue Principale	0123456789
A02	Agence Nord	456 Rue du Nord	0987654321

```
2 rows in set (0.00 sec)
```

3. TABLE BIEN:

CREATE TABLE Bien (

IDB CHAR(3) PRIMARY KEY,

Adresse VARCHAR(100) NOT NULL,

CodePostal CHAR(5) NOT NULL,

Type VARCHAR(10) NOT NULL,

IDC CHAR(3),

IDA CHAR(3),

LoyerMensuel DECIMAL(10, 2) NOT NULL,

Etat VARCHAR(10) CHECK (Etat IN ('Loue', 'Libre')),

FOREIGN KEY (IDC) REFERENCES Client(IDC),

FOREIGN KEY (IDA) REFERENCES Agence(IDA));

```
mysql> SELECT * FROM Bien;
```

IDB	Adresse	CodePostal	Type	IDC	IDA	LoyerMensuel	Etat
B01	12 Rue Verte	75001	T2	NULL	A01	1200.00	Libre
B02	34 Rue Bleue	75002	T3	C01	A01	1500.00	Louer
B03	56 Rue Jaune	75003	T1	NULL	A02	900.00	Libre

```
3 rows in set (0.00 sec)
```

4. TABLE INSCRIT

CREATE TABLE Inscrit (

IDC CHAR(3),

IDA CHAR(3),

PRIMARY KEY (IDC, IDA),

FOREIGN KEY (IDC) REFERENCES Client(IDC),

FOREIGN KEY (IDA) REFERENCES Agence(IDA));

```
mysql> SELECT * FROM Inscrit;  
+-----+-----+  
| IDC | IDA |  
+-----+-----+  
| C01 | A01 |  
| C02 | A01 |  
| C01 | A02 |  
| C02 | A02 |  
| C03 | A02 |  
+-----+-----+  
5 rows in set (0.00 sec)
```

2. Ajout des Contraintes et Données

Pour tester les requêtes, nous avons ajouté des données aux tables avec les commandes suivantes :

```
INSERT INTO Client (IDC, NomPrenom, Age, Salaire) VALUES ('C01', 'Alice Dupont', 30,  
2500.00);
```

```
INSERT INTO Client (IDC, NomPrenom, Age, Salaire) VALUES ('C02', 'Bob Martin', 45,  
3200.00);
```

```
INSERT INTO Agence (IDA, Nom, Adresse, Telephone) VALUES ('A01', 'Agence Centre',  
'123 Rue Principale', '0123456789');
```

```
INSERT INTO Agence (IDA, Nom, Adresse, Telephone) VALUES ('A02', 'Agence Nord', '456  
Rue du Nord', '0987654321');
```

```
INSERT INTO Bien (IDB, Adresse, CodePostal, Type, IDC, IDA, LoyerMensuel, Etat)  
VALUES ('B01', '12 Rue Verte', '75001', 'T2', NULL, 'A01', 1200.00, 'Libre');
```

```
INSERT INTO Bien (IDB, Adresse, CodePostal, Type, IDC, IDA, LoyerMensuel, Etat)  
VALUES ('B02', '34 Rue Bleue', '75002', 'T3', 'C01', 'A01', 1500.00, 'Loue');
```

```
INSERT INTO Inscrit (IDC, IDA) VALUES ('C01', 'A01');
```

```
INSERT INTO Inscrit (IDC, IDA) VALUES ('C02', 'A01');
```

3. Requêtes pour Répondre aux Questions

Voici l'ensemble des requêtes SQL réalisées :

- **Adresse des biens non encore loués :**

```
SELECT Adresse FROM Bien WHERE Etat = 'Libre';
```

```
mysql> SELECT Adresse FROM Bien WHERE Etat = 'Libre';
+-----+
| Adresse |
+-----+
| 12 Rue Verte |
| 56 Rue Jaune |
+-----+
2 rows in set (0.00 sec)
```

- **Identifiants des clients n'ayant rien loué :**

SELECT IDC FROM Client WHERE IDC NOT IN (SELECT DISTINCT IDC FROM Bien WHERE Etat = 'Loue');

```
mysql> SELECT IDC FROM Client WHERE IDC NOT IN (SELECT DISTINCT IDC FROM Bien WHERE Etat = 'Loue');
+----+
| IDC |
+----+
| C02 |
| C03 |
+----+
2 rows in set (0.01 sec)
```

- **Nombre de biens non encore loués :**

SELECT COUNT(*) AS NombreBiensNonLoues FROM Bien WHERE Etat = 'Libre';

```
mysql> SELECT COUNT(*) AS NombreBiensNonLoues FROM Bien WHERE Etat = 'Libre';
+-----+
| NombreBiensNonLoues |
+-----+
| 2 |
+-----+
1 row in set (0.01 sec)
```

- **Nombre de biens loués par chaque agence :**

SELECT IDA, COUNT(*) AS NombreBiensLoues FROM Bien WHERE Etat = 'Loue' GROUP BY IDA;

```
mysql> SELECT IDA, COUNT(*) AS NombreBiensLoues FROM Bien WHERE Etat = 'Loue' GROUP BY IDA;
+----+-----+
| IDA | NombreBiensLoues |
+----+-----+
| A01 | 1 |
+----+-----+
1 row in set (0.00 sec)
```

- **Salaire moyen des inscrits d'une agence donnée (exemple : IDA = 'A01') :**

SELECT AVG(Salaire) AS SalaireMoyen FROM Client WHERE IDC IN (SELECT IDC FROM Inscrit WHERE IDA = 'A01');

```
mysql> SELECT AVG(Salaire) AS SalaireMoyen FROM Client WHERE IDC IN (SELECT IDC FROM Inscrit WHERE IDA = 'A01');
+-----+
| SalaireMoyen |
+-----+
| 2850.000000 |
+-----+
1 row in set (0.00 sec)
```

III. Partie II - Gestion des Droits d'Accès aux Objets de la Base

Objectif

Attribuer des permissions d'accès aux objets de la base de données MySQL en fonction des rôles et utilisateurs mentionnés dans l'énoncé.

1. Création des Utilisateurs

Pour chaque utilisateur, un compte MySQL a été créé avec un mot de passe pour garantir une authentification unique. Voici les commandes exécutées :

-- Créer les utilisateurs avec un mot de passe

```
CREATE USER 'Brisefer'@'%' IDENTIFIED BY 'password123';  
CREATE USER 'Genial'@'%' IDENTIFIED BY 'password123';  
CREATE USER 'Lebut'@'%' IDENTIFIED BY 'password123';  
CREATE USER 'Rectitude'@'%' IDENTIFIED BY 'password123';
```

2. Attribution des Droits

Les permissions ont été configurées en fonction des rôles assignés à chaque utilisateur. Voici les détails :

a. Permissions pour Mr. Brisefer (Administrateur Général)

Mr. Brisefer a tous les droits sur la base de données.

```
GRANT ALL PRIVILEGES ON TP_IMMO.* TO 'Brisefer'@'%';
```

b. Permissions pour Mme Rectitude (Responsable des Agences)

Mme Rectitude a les droits suivants :

- **Consultation, insertion et mise à jour** sur les tables Client, Agence et Bien.
- **Aucune permission de suppression** pour garantir l'intégrité des données.

-- Droits sur la table Agence

```
GRANT SELECT, INSERT, UPDATE ON TP_IMMO.Agence TO 'Rectitude'@'%';
```

-- Droits sur les tables Client et Bien

```
GRANT SELECT, INSERT, UPDATE ON TP_IMMO.Client TO 'Rectitude'@'%';
```

```
GRANT SELECT, INSERT, UPDATE ON TP_IMMO.Bien TO 'Rectitude'@'%';
```

c. Permissions pour Mr. Lebut (Chargé de Clientèle)

Mr. Lebut est responsable des relations clients et a les droits suivants :

- **Insertion, suppression et mise à jour** sur Client et Inscrit.
- **Consultation uniquement** sur la table Agence.

-- Droits sur Client et Inscrit

```
GRANT INSERT, DELETE, UPDATE ON TP_IMMO.Client TO 'Lebut'@'%';
```

```
GRANT INSERT, DELETE, UPDATE ON TP_IMMO.Inscrit TO 'Lebut'@'%';
```

-- Droit de consultation sur Agence

```
GRANT SELECT ON TP_IMMO.Agence TO 'Lebut'@'%';
```

d. Permissions pour Mr. Genial (Gestionnaire des Biens)

Mr. Genial gère les biens immobiliers et a les droits suivants :

- **Consultation, insertion, suppression et mise à jour** sur la table Bien.
- **Consultation uniquement** sur la table Agence.

-- Droits sur Bien

```
GRANT SELECT, INSERT, DELETE, UPDATE ON TP_IMMO.Bien TO 'Genial'@'%';
```

-- Droit de consultation sur Agence

```
GRANT SELECT ON TP_IMMO.Agence TO 'Genial'@'%';
```

3. Regroupement des Droits par Rôles

Pour simplifier la gestion, nous avons regroupé les permissions en rôles. Voici les commandes utilisées :

a. Création des Rôles

```
CREATE ROLE AdministrateurGeneral;
```

```
CREATE ROLE ResponsableAgences;
```

```
CREATE ROLE ChargeClientele;
```

```
CREATE ROLE GestionnaireBiens;
```

b. Attribution des Permissions aux Rôles

-- Droits pour AdministrateurGeneral

GRANT ALL PRIVILEGES ON TP_IMMO.* TO AdministrateurGeneral;

-- Droits pour ResponsableAgences

GRANT SELECT, INSERT, UPDATE ON TP_IMMO.Agence TO ResponsableAgences;

GRANT SELECT, INSERT, UPDATE ON TP_IMMO.Client TO ResponsableAgences;

GRANT SELECT, INSERT, UPDATE ON TP_IMMO.Bien TO ResponsableAgences;

-- Droits pour ChargeClientele

GRANT INSERT, DELETE, UPDATE ON TP_IMMO.Client TO ChargeClientele;

GRANT INSERT, DELETE, UPDATE ON TP_IMMO.Inscrit TO ChargeClientele;

GRANT SELECT ON TP_IMMO.Agence TO ChargeClientele;

-- Droits pour GestionnaireBiens

GRANT SELECT, INSERT, DELETE, UPDATE ON TP_IMMO.Bien TO GestionnaireBiens;

GRANT SELECT ON TP_IMMO.Agence TO GestionnaireBiens;

c. Attribution des Rôles aux Utilisateurs

GRANT AdministrateurGeneral TO 'Brisefer'@'%';

GRANT ResponsableAgences TO 'Rectitude'@'%';

GRANT ChargeClientele TO 'Lebut'@'%';

GRANT GestionnaireBiens TO 'Genial'@'%';

d. Activation des Rôles (Par Utilisateur)

SET DEFAULT ROLE AdministrateurGeneral FOR 'Brisefer'@'%';

SET DEFAULT ROLE ResponsableAgences FOR 'Rectitude'@'%';

SET DEFAULT ROLE ChargeClientele FOR 'Lebut'@'%';

SET DEFAULT ROLE GestionnaireBiens FOR 'Genial'@'%';

4. Vérification des Droits

Pour valider les permissions, nous avons utilisé la commande suivante pour chaque utilisateur :

SHOW GRANTS FOR 'Rectitude'@'%';

```
mysql> SHOW GRANTS FOR 'Rectitude'@'%';
+-----+
| Grants for Rectitude@% |
+-----+
| GRANT USAGE ON *.* TO 'Rectitude'@'%' |
| GRANT SELECT, INSERT, UPDATE ON 'TP_IMMO'. 'Agence' TO 'Rectitude'@'%' |
| GRANT SELECT, INSERT, UPDATE ON 'TP_IMMO'. 'Bien' TO 'Rectitude'@'%' |
| GRANT SELECT, INSERT, UPDATE ON 'TP_IMMO'. 'Client' TO 'Rectitude'@'%' |
| GRANT 'ResponsableAgences'@'%' TO 'Rectitude'@'%' |
+-----+
5 rows in set (0.00 sec)
```

SHOW GRANTS FOR 'Lebut'@'%';

```
mysql> SHOW GRANTS FOR 'Lebut'@'%';
+-----+
| Grants for Lebut@% |
+-----+
| GRANT USAGE ON *.* TO 'Lebut'@'%' |
| GRANT SELECT ON 'TP_IMMO'. 'Agence' TO 'Lebut'@'%' |
| GRANT INSERT, UPDATE, DELETE ON 'TP_IMMO'. 'Client' TO 'Lebut'@'%' |
| GRANT INSERT, UPDATE, DELETE ON 'TP_IMMO'. 'Inscrit' TO 'Lebut'@'%' |
| GRANT 'ChargeClientele'@'%' TO 'Lebut'@'%' |
+-----+
5 rows in set (0.00 sec)
```

SHOW GRANTS FOR 'Genial'@'%';

```
mysql> SHOW GRANTS FOR 'Genial'@'%';
+-----+
| Grants for Genial@% |
+-----+
| GRANT USAGE ON *.* TO 'Genial'@'%' |
| GRANT SELECT ON 'TP_IMMO'. 'Agence' TO 'Genial'@'%' |
| GRANT SELECT, INSERT, UPDATE, DELETE ON 'TP_IMMO'. 'Bien' TO 'Genial'@'%' |
| GRANT 'GestionnaireBiens'@'%' TO 'Genial'@'%' |
+-----+
4 rows in set (0.00 sec)
```

SHOW GRANTS FOR 'Brisefer'@'%';

```
mysql> SHOW GRANTS FOR 'Brisefer'@'%';
+-----+
| Grants for Brisefer@% |
+-----+
| GRANT USAGE ON *.* TO 'Brisefer'@'%' |
| GRANT ALL PRIVILEGES ON 'TP_IMMO'.* TO 'Brisefer'@'%' |
| GRANT 'AdministrateurGeneral'@'%' TO 'Brisefer'@'%' |
+-----+
3 rows in set (0.00 sec)
```

Difficultés Rencontrées

- **Gestion des privilèges granulaire** : Identifier les permissions minimales nécessaires pour chaque utilisateur tout en évitant des privilèges excessifs.
- **Compatibilité MySQL** : Certaines commandes relatives aux rôles nécessitent MySQL 8.0 ou une version plus récente.
- **Configuration réseau Docker** : Les conteneurs doivent être correctement connectés au même réseau pour garantir l'accessibilité de MySQL.

IV. Partie 3 : Test de Sécurité avec SQLmap

Commande Utilisée

```
sqlmap -u "http://localhost:8080/inscription.php" --data="idc=C01&ida=A01" --batch
```

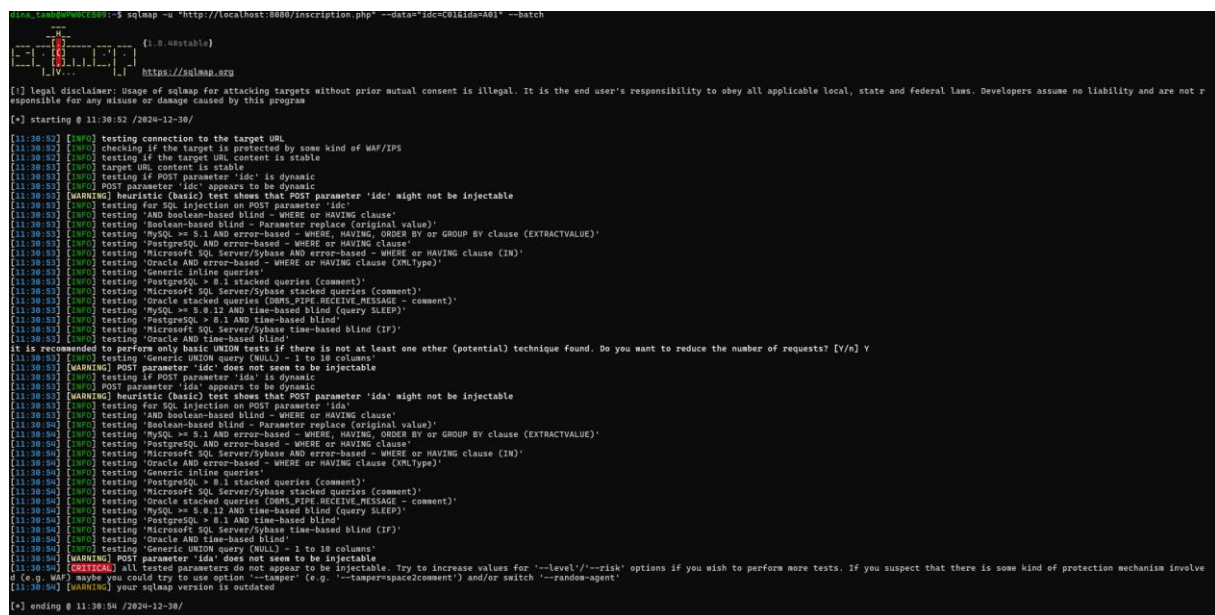
Résultats

- **Pas de vulnérabilité d'injection SQL détectée** sur les paramètres idc et ida.
- Utilisation de requêtes préparées (PDO) dans le fichier inscription.php a été efficace pour éliminer les risques d'injections.

Difficultés Rencontrées

- Problème initial de connexion refusée entre le conteneur PHP et le conteneur MySQL, résolu par une configuration correcte du réseau Docker.

Capture d'écran des Tests SQLmap



```
dina_tamboura@kali:~$ sqlmap -u "http://localhost:8080/inscription.php" --data="idc=C01&ida=A01" --batch
[1.0.0-stable]
https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 11:30:52 / 2024-12-30/

[11:30:52] [INFO] testing connection to the target URL
[11:30:52] [INFO] checking if the target is protected by some kind of WAF/IPS
[11:30:52] [INFO] testing if the target URL content is stable
[11:30:52] [INFO] target URL content is stable
[11:30:52] [INFO] testing if POST parameter 'idc' is dynamic
[11:30:52] [INFO] POST parameter 'idc' appears to be dynamic
[11:30:52] [WARNING] heuristic (basic) test shows that POST parameter 'idc' might not be injectable
[11:30:52] [INFO] testing for SQL injection on POST parameter 'idc'
[11:30:52] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[11:30:52] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[11:30:52] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[11:30:52] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[11:30:52] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[11:30:52] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[11:30:52] [INFO] testing 'Generic inline queries'
[11:30:52] [INFO] testing 'PostgreSQL >= 8.1 stacked queries (comment)'
[11:30:52] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[11:30:52] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[11:30:52] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[11:30:52] [INFO] testing 'PostgreSQL >= 8.1 AND time-based blind'
[11:30:52] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[11:30:52] [INFO] testing 'Oracle AND time-based blind'
[11:30:52] [INFO] it is recommended to perform only basic UNION tests if there is not at least one other (potential) technique found. Do you want to reduce the number of requests? [Y/n] Y
[11:30:52] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[11:30:52] [WARNING] POST parameter 'idc' does not seem to be injectable
[11:30:52] [INFO] testing if POST parameter 'ida' is dynamic
[11:30:52] [INFO] POST parameter 'ida' appears to be dynamic
[11:30:52] [WARNING] heuristic (basic) test shows that POST parameter 'ida' might not be injectable
[11:30:52] [INFO] testing for SQL injection on POST parameter 'ida'
[11:30:52] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[11:30:52] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[11:30:52] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[11:30:52] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[11:30:52] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[11:30:52] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[11:30:52] [INFO] testing 'Generic inline queries'
[11:30:52] [INFO] testing 'PostgreSQL >= 8.1 stacked queries (comment)'
[11:30:52] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[11:30:52] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[11:30:52] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[11:30:52] [INFO] testing 'PostgreSQL >= 8.1 AND time-based blind'
[11:30:52] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[11:30:52] [INFO] testing 'Oracle AND time-based blind'
[11:30:52] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[11:30:52] [WARNING] POST parameter 'ida' does not seem to be injectable
[11:30:52] [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for '--level'/'--risk' options if you wish to perform more tests. If you suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=space2comment') and/or switch '--random-agent'
[11:30:52] [WARNING] your sqlmap version is outdated
[*] ending @ 11:30:54 / 2024-12-30/
```

Conclusion

Ce TP a permis de mettre en place une base de données fonctionnelle et sécurisée. Les tests de sécurité avec SQLmap ont confirmé l'efficacité des protections contre les injections SQL. Cependant, il reste essentiel de continuer à tester et améliorer l'application pour prévenir d'autres types de vulnérabilités.