Rapport du TP6 : SQL_INJECTION

Démonstration d'une Injection SQL via DevContainer

Objectif : Mettre en place un environnement de développement isolé pour démontrer comment une requête SQL mal protégée peut être exploitée par un attaquant.

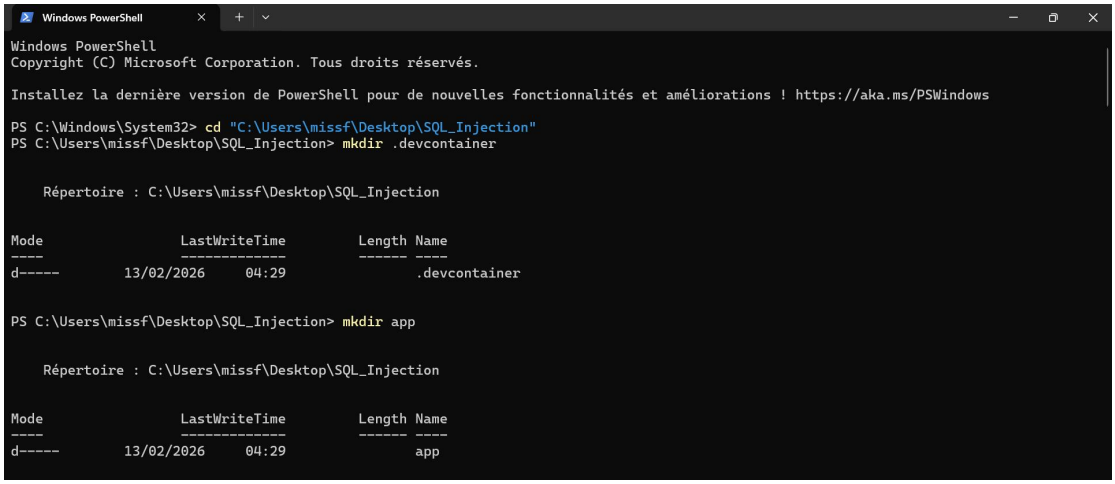
1. Préparation de l'environnement Local

Étape 1 : Structure du dossier

Je commence par créer l'espace de mon travail .

Étape 2 : Configuration du conteneur (DevContainer)

J'ai crée mes fichier .devcontainer/devcontainer.json et je colle le code JSON fourni dans l'énoncé.



Étape 3 : Créer le fichier docker compose.yml

Crée le fichier docker-compose.yml à la racine. Ce fichier va piloter deux services :

Dans le fichier **app** : mon serveur Node.js.

db : ma base de données MariaDB.

```
PS C:\Users\missf\Desktop\SQL_Injection> @'
>> {
>>   "name": "SQL Injection Demo",
>>   "dockerComposeFile": "docker-compose.yml",
>>   "service": "app",
>>   "workspaceFolder": "/usr/src/app",
>>   "customizations": {
>>     "vscode": {
>>       "settings": {
>>         "terminal.integrated.shell.linux": "/bin/bash"
>>       },
>>       "extensions": [
>>         "ms-azuretools.vscode-docker"
>>       ]
>>     }
>>   }
>> }
```

Étape 4 : Initialisation du code

package.json : j'ai créer le fichier pour définir les dépendances (express et mysql).

server.js : C'est le cœur de l'application.

Dans server.js, la ligne vulnérable est la suivante :

Ici, les variables sont concaténées directement. C'est la porte ouverte à l'injection.

```
>> '@ | Out-File -FilePath .devcontainer\devcontainer.json -Encoding utf8
PS C:\Users\missf\Desktop\SQL_Injection> @'
>> services:
>>   app:
>>     image: node:14
>>     ports:
>>       - "3000:3000"
>>     environment:
>>       - DB_HOST=db
>>       - DB_USER=root
>>       - DB_PASSWORD=root
>>       - DB_NAME=login_db
>>     volumes:
>>       - ../app:/usr/src/app
>>     working_dir: /usr/src/app
>>     command: ["sh", "-c", "npm install -g nodemon && npm install && nodemon server.js"]
>>
>>   db:
>>     image: mariadb:latest
>>     environment:
>>       MARIADB_ROOT_PASSWORD: root
>>       MARIADB_DATABASE: login_db
>>     volumes:
>>       - db_data:/var/lib/mysql
>>
>> volumes:
>>   db_data:
```

```
>> '@ | Out-File -FilePath .devcontainer\docker-compose.yml -Encoding utf8
PS C:\Users\missf\Desktop\SQL_Injection> cd app
PS C:\Users\missf\Desktop\SQL_Injection\app>
PS C:\Users\missf\Desktop\SQL_Injection\app> # Créer le .gitignore
PS C:\Users\missf\Desktop\SQL_Injection\app> Invoke-WebRequest https://www.toptal.com/developers/gitignore/api/node -OutFile .gitigno
re
PS C:\Users\missf\Desktop\SQL_Injection\app>
PS C:\Users\missf\Desktop\SQL_Injection\app> # Créer le package.json
PS C:\Users\missf\Desktop\SQL_Injection\app> @'
>> {
>>   "name": "sql-injection-demo",
>>   "version": "1.0.0",
>>   "main": "server.js",
>>   "dependencies": {
>>     "express": "^4.17.1",
>>     "mysql": "^2.18.1"
>>   }
>> }
```

```
>> '@ | Out-File -FilePath package.json -Encoding utf8
PS C:\Users\missf\Desktop\SQL_Injection\app> @'
>> const express = require('express');
>> const mysql = require('mysql');
>> const bodyParser = require('body-parser');
>> const app = express();
>> const port = 3000;
>>
>> app.use(bodyParser.urlencoded({ extended: true }));
>>
>> const db = mysql.createConnection({
>>   host: process.env.DB_HOST,
>>   user: process.env.DB_USER,
>>   password: process.env.DB_PASSWORD,
>>   database: process.env.DB_NAME,
>> });
>>
>> function connectWithRetry() {
>>   db.connect((err) => {
>>     if (err) {
>>       console.error('Échec de la connexion à la base de données:', err.message);
>>       setTimeout(connectWithRetry, 2000);
>>     } else {
>>       console.log('Connecté à la base de données');
>>     }
>>   });
>> }
>> connectWithRetry();
>>
```

```
>> app.get('/', (req, res) => {
>>   res.send(`
>>     <h2>Connexion</h2>
>>     <form method="POST" action="/login">
>>       <input type="text" name="username" placeholder="Nom d'utilisateur" required><br><br>
>>       <input type="password" name="password" placeholder="Mot de passe" required><br><br>
>>       <button type="submit">Se connecter</button>
>>     </form>
>>   `);
>> });
>>
>> app.post('/login', (req, res) => {
>>   const { username, password } = req.body;
>>   // ATTENTION : Cette ligne est volontairement vulnérable
>>   const query = `SELECT * FROM users WHERE username = '${username}' AND password = '${password}'`;
>>
>>   db.query(query, (err, results) => {
>>     if (err) {
>>       res.status(500).send('Erreur SQL : ' + err.message);
>>       return;
>>     }
>>     if (results.length > 0) {
>>       res.send(`<h1>Connexion réussie !</h1> Bienvenue, ${results[0].username}.`);
>>     } else {
>>       res.send('Identifiants incorrects.');
>>     }
>>   });
>> });
>>
>> app.listen(port, () => {
>>   console.log(`Serveur lancé sur http://localhost:${port}`);
>> });
```

Déploiement et Configuration

Étape 5 : Lancement de l'environnement

Une notification en bas à droite de VS Code apparaîtra : **"Reopen in Container"**. Cliquez dessus. VS Code va alors installer Node.js et MariaDB automatiquement à l'intérieur de Docker.

<u>Étape 6 :</u> Initialisation de la Base de Données

Une fois que VS Code est "dans le conteneur", ouvre le terminal intégré de VS Code et tape la commande suivante pour entrer dans MariaDB :

Le mot de passe est : root

Une fois dans le prompt MariaDB (MariaDB [(none)]>), tape ces commandes SQL :

```
PS C:\Users\missf\Desktop\SQL_Injection> docker ps
CONTAINER ID   IMAGE           COMMAND                  CREATED          STATUS          PORTS
   NAMES
8b7f01d65cd4   node:14         "/bin/sh -c 'echo Co…"   27 minutes ago   Up 27 minutes   0.0.0.0:3000->3000/tcp, [::]:3000->3000/tcp
   sql_injection_devcontainer-app-1
7be6d808a9eb   mariadb:latest  "docker-entrypoint.s…"   27 minutes ago   Up 27 minutes   3306/tcp
   sql_injection_devcontainer-db-1
PS C:\Users\missf\Desktop\SQL_Injection> docker exec -it sql_injection_devcontainer-db-1 mariadb -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 4
Server version: 12.0.2-MariaDB-ubu2404 mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> USE login_db;
Database changed
MariaDB [login_db]> CREATE TABLE users (
    ->   id INT AUTO_INCREMENT PRIMARY KEY,
    ->   username VARCHAR(255) NOT NULL,
    ->   password VARCHAR(255) NOT NULL
    -> );
Query OK, 0 rows affected (0.032 sec)

MariaDB [login_db]> INSERT INTO users (username, password) VALUES ('testuser', 'password123');
Query OK, 1 row affected (0.005 sec)

MariaDB [login_db]> EXIT;
Bye
PS C:\Users\missf\Desktop\SQL_Injection>
```