TP 1:   
Identification, démonstration et correction des failles majeures.

SQL Injections

# Le mécanisme de la faille

#### English

**In short:**

* SQL injection is a code injection technique that might destroy your database.
* SQL injection is one of the most common web hacking techniques.
* SQL injection is the placement of malicious code in SQL statements, via web page input.

**Detailed information:**

In computing, SQL injection is a code injection technique used to attack data-driven applications, in which malicious SQL statements are inserted into an entry field for execution (e.g. to dump the database contents to the attacker).   
SQL injection must exploit a security vulnerability in an application's software, for example, when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and unexpectedly executed.

**In conclusion:**

SQL injection attacks allow attackers to spoof identity, tamper with existing data, cause repudiation issues.

#### French

**La version courte:**

* L'injection SQL est une technique d'injection de code qui peut détruire votre base de données.
* L'injection SQL est l'une des techniques de piratage les plus courantes sur le web.
* L'injection SQL consiste à insérer du code malveillant dans des instructions SQL via les entrées d'une page web.

**Plus d’information détaillée :**

En informatique, l'injection SQL est une technique d'injection de code utilisée pour attaquer des applications basées sur des bases de données, dans laquelle des instructions SQL malveillantes sont insérées dans un champ de saisie afin d’être exécutées (par exemple, pour extraire le contenu de la base de données au profit de l’attaquant).

L’injection SQL exploite une vulnérabilité de sécurité dans le logiciel d’une application, notamment lorsque les entrées utilisateur ne sont pas correctement filtrées pour les caractères d’échappement des chaînes de caractères dans les instructions SQL, ou lorsque ces entrées ne sont pas strictement typées et sont exécutées de manière inattendue.

**En conclusion :**

Les attaques par injection SQL permettent aux attaquants d’usurper une identité, de manipuler des données existantes et de provoquer des problèmes de répudiation.

# Les différentes variantes

SQL injections typically fall under three categories:

* In-band SQLi (Classic)
  + Error-based SQL injection
  + Union-based SQL injection
* Inferential SQLi (Blind)
  + Boolean-based blind SQL injection
  + Time-based blind SQL injection
* Out-of-band SQLi

## In-band SQLi (Classic)

Injection SQL intrabande, où l'attaquant reçoit le résultat sous forme de réponse directe en utilisant le même canal de communication. Par exemple, le résultat de l'attaque sera affiché et l'attaquant sera en mesure de modifier la requête d'origine et de recevoir les résultats directs de la requête modifiée.

### Error-based SQL injection

L'injection SQL basée sur les erreurs est un sous-type d'injection SQL intrabande où le résultat renvoyé à l' attaquant est une chaîne d'erreur de base de données.

*Consequences of error-based SQL injection:*

1. Obtenez des informations sur le type et la version de la base de données afin d' utiliser différentes techniques d'attaque pour un type/une version de base de données spécifique.
2. Obtenez des informations sur la structure de la base de données pour essayer des injections SQL plus spécifiques une fois que la structure est connue.
3. Extraire des données de la base de données. Bien que le processus soit beaucoup plus long et plus complexe que l'affichage direct du résultat d'une requête, un attaquant peut être en mesure de manipuler les erreurs pour exfiltrer des données de la base de données.

### Union-based SQL injection

L'injection SQL basée sur l'union est un sous-type d'injection SQL intrabande dans lequel l'attaquant utilise la clause SQL UNION pour recevoir un résultat qui combine des informations légitimes avec des données sensibles.

*Consequences of union-based SQL injection*

Il s'agit du type d'injection SQL le plus dangereux, car il permet à l'attaquant d'obtenir directement presque toutes les informations de la base de données.

## Inferential SQLi (Blind)

L'injection SQL aveugle est un type d'injection SQL dans lequel l'attaquant ne reçoit pas de réponse évidente de la base de données attaquée et reconstruit la structure de la base de données étape par étape en observant le comportement du serveur de base de données et de l'application.

L'attaquant peut être en mesure de faire ce qui suit :

* Vérifiez si d'autres types d'injections SQL sont possibles
* Obtenir des informations sur la structure de la base de données
* Extraire les données de la base de données

### Boolean-based blind SQL injection

L'injection SQL aveugle booléenne est un sous-type d'injection SQL aveugle dans lequel l'attaquant observe le comportement du serveur de base de données et de l'application après avoir combiné des requêtes légitimes avec des données malveillantes à l'aide d'opérateurs booléens.

### Time-based blind SQL injection

Idem mais des requêtes avec des données malveillantes qui entraînent des retards.

## Out-of-band SQL injection

L'injection SQL hors bande est un type d'injection SQL dans lequel l'attaquant ne reçoit pas de réponse de l'application attaquée sur le même canal de communication, mais est en mesure de faire en sorte que l'application envoie des données à un point de terminaison distant qu'il contrôle.

L'injection SQL hors bande n'est possible que si le serveur que vous utilisez dispose de commandes qui déclenchent des requêtes DNS ou HTTP. Cependant, c'est le cas avec tous les serveurs SQL populaires.

## Prevention

Le seul moyen efficace de prévenir tous les types de vulnérabilités SQLi dans les applications Web est d'utiliser des requêtes paramétrées pour accéder aux bases de données SQL.   
Si votre langage de programmation ne prend pas en charge les requêtes paramétrées, mais que votre moteur de base de données prend en charge les procédures stockées, vous pouvez utiliser des procédures stockées avec des instructions préparées à la place.

Il n'est pas recommandé de s'appuyer uniquement sur d'autres méthodes de prévention, telles que les listes blanches, les listes noires ou le filtrage/échappement des entrées. Les pirates malveillants peuvent trouver un moyen de contourner cet assainissement.

# Demo

[GitHub link](https://github.com/StijnVM6/secure-programming-tp1)

### Database (SQLite)

A screenshot of a computer

AI-generated content may be incorrect.

### Vulnerable login path

A computer screen shot of text

AI-generated content may be incorrect.





### CRUD:

<http://localhost:3000/login>

curl -X POST <http://localhost:3000/login>

-H "Content-Type: application/json"

-d '{"username":"admin","password":"securepassword123"}'

**VS**

curl -X POST http://localhost:3000/login

-H "Content-Type: application/json"

-d '{"username":"admin";--","password":"anything"}'

A black background with white text

AI-generated content may be incorrect. VS A black background with white text

AI-generated content may be incorrect.

### Problem

Prisma ORM docs : [Link](https://www.prisma.io/docs/orm/prisma-client/using-raw-sql/raw-queries#raw-queries-with-relational-databases)

The double hyphen (--) comments out the rest of the SQL statement, thus turning the password check into a comment, bypassing authentication!

This allows SQL injection because user input is directly added to the query string.

### Solution

We must use parameterized queries instead of directly injecting user input into SQL statements.

Since we're using Prisma as our ORM, it already helps prevent injection by default.

#### Solution 1:

A computer screen shot of colorful text

AI-generated content may be incorrect.

Parameterized queries ensure that user input is treated as data, not code !

No more SQL injection, because ${} injects values safely.

#### Solution 2:

A computer screen with text

AI-generated content may be incorrect.

Here we use the Prism ORM which automatically parameterizes user input to prevent SQL injection.

# ENGLISH

## In-band SQLi (Classic)

In-band SQL injection is a type of SQL injection where the attacker receives the result as a direct response using the same communication channel. For example, if the attacker performs the attack manually using a web browser, the result of the attack will be displayed in the same web browser. The simplest type of in-band SQL injection is when the attacker is able to modify the original query and receive the direct results of the modified query.

### Error-based SQL injection

Error-based SQL injection is a subtype of in-band SQL injection where the result returned to the attacker is a database error string.

##### Consequences of error-based SQL injection:

Returning an error string to an attacker may seem harmless. However, depending on the structure of the application and the type of database, the attacker may use the received error string to:

* Get information about the type and version of the database to use different attack techniques for a specific database type/version.
* Get information about the structure of the database to try more specific SQL injections once the structure is known.
* Get data out of the database. While the process is much longer and more complex than directly displaying the result of a query, an attacker may be able to manipulate the errors to exfiltrate data from the database.

### Union-based SQL injection

Union-based SQL injection is a subtype of in-band SQL injection where the attacker uses the UNION SQL clause to receive a result that combines legitimate information with sensitive data.

##### Consequences of union-based SQL injection

Union-based SQL injection is the most dangerous type of SQL injection because it lets the attacker directly obtain almost any information from the database.

### Prevention

The only fully effective way to prevent all types of SQLi vulnerabilities in web applications, including in-band SQLi, is to use parameterized queries (also known as prepared statements) to access SQL databases. If your programming language does not support parameterized queries but your database engine supports stored procedures, you may use stored procedures with prepared statements instead.

Relying purely on other prevention methods, such as whitelists, blacklists, or input filtering/escaping, is not recommended. Malicious hackers may find a way around such sanitization.

Additionally, you should configure your environment not to display database errors – this helps mitigate even undiscovered error-based SQL injection vulnerabilities. However, it has no effect on other types of SQL injections and therefore you should not treat it as a prevention method.

## Inferential SQLi (Blind)

Blind SQL injection is a type of SQL injection where the attacker does not receive an obvious response from the attacked database and instead reconstructs the database structure step-by-step by observing the behaviour of the database server and the application.

Performing an attack using blind SQL injections takes much longer than in the case of in-band SQL injections but can yield the same results. Based on the behaviour of the database server and the application, the attacker may be able to do the following:

* Check if other types of SQL injections are possible
* Get information about the structure of the database
* Get data out of the database

### Boolean-based blind SQL injection

Boolean-based blind SQL injection is a subtype of blind SQL injection where the attacker observes the behaviour of the database server and the application after combining legitimate queries with malicious data using boolean operators.

### Time-based blind SQL injection

Time-based blind SQL injection is a subtype of blind SQL injection where the attacker observes the behaviour of the database server and the application after combining legitimate queries with SQL commands that cause time delays.

### Prevention

The only fully effective way to prevent all types of SQLi vulnerabilities in web applications, including blind SQLi, is to use parameterized queries.

## Out-of-band SQL injection

Out-of-band SQL injection (OOB SQLi) is a type of SQL injection where the attacker does not receive a response from the attacked application on the same communication channel but instead is able to cause the application to send data to a remote endpoint that they control.

Out-of-band SQL injection is only possible if the server that you are using has commands that trigger DNS or HTTP requests. However, that is the case with all popular SQL servers.

### Prevention

Parameterized queries !