# Injections SQL

## Définition

L'injection SQL est une **attaque qui vise à exploiter une vulnérabilité dans une application web** pour manipuler directement les requêtes SQL envoyés à une Base de Données. L'attaquant peut injecter du code SQL malveillant dans des champs de saisie ou des URL, ce qui peut entraîner des conséquences graves pour la sécurité de la base de données..

## Exemples concrets

Ces exemples ne montrent pas d'injection



*Figure 1: Page d'accueil*

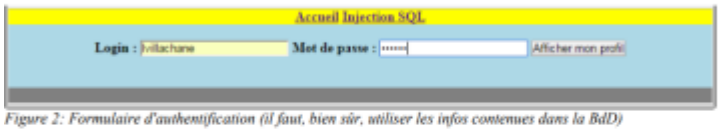


*Figure 2: Formulaire d'authentification (il faut, bien sûr, utiliser les infos contenues dans la BdD)*



*Figure 3: Retour du formulaire après saisie et soumission*



*Figure 4: Erreur sur page inexistante (URL modifiée à la main)*

## Se protéger : validation des données entrantes

Voir Validation des données entrantes

## Se protéger : principes des requêtes préparées

### Généralités

La plupart des SGBD modernes utilisent un mécanisme dit “Requêtes sécurisées” pour sécuriser les injections SQL. Ce mécanisme a pour but d'améliorer la sécurité et l'efficacité des requêtes SQL.

* La requête est analysée puis optimisée avant d'être exécutée, sans inclure les données réelles (les

paramètres) au moment de sa préparation.

* Une requête préparée peut être réutilisable plusieurs fois sans être recompilée à chaque fois.

* Cela garantit que le comportement de la requête reste constant.

* Les requêtes préparées s'exécutent plus rapidement lorsqu'elles sont utilisées plusieurs fois.

Les requêtes préparées offrent plusieurs avantages par rapport aux requêtes classiques. Elles permettent de Sécuriser les requêtes en évitant l'insertion de code malveillant et en améliorant les performances des requêtes répétées, car la requête n'a besoin d'être analysée qu'une seule fois, même si elle est exécutée plusieurs fois avec des données différentes. Cela renforce la sécurité en empêchant l'exécution de code SQL injecté et améliore les performances des requêtes répétées. De plus, elles “figent” la structure de la requête, ne laissant aucune place à la modification malveillante de cette dernière, réduisant considérablement la surface d'attaque pour les injections SQL.

## Paramètres anonymes

**Définition:**: Ce sont des marqueurs utilisés dans les requêtes préparées pour indiquer les emplacements des valeurs à injecter, sans leur donner de noms explicites. Ils ont représentés par des points d'interrogation **(?)** dans la requête SQL.

Lorsque la requête est exécutée, un tableau contenant les valeurs des paramètres est fourni, et ces valeurs sont liées aux marqueurs anonymes dans l'ordre de leur apparition dans la requête. Cela permet de sécuriser les requêtes sans risquer d'injecter du code malveillant dans la Base de Données.

## Paramètres nommés

**Définition**: Une alternative aux paramètres anonymes dans les requêtes préparées. Au lieu d'utiliser des points d'interrogation pour marquer l'emplacement des valeurs, chaque paramètre est identifié par un nom précédé d'un deux points (:). Cela permet de spécifier les paramètres indépendamment de leur position dans la requête et d'améliorer la lisibilité, surtout pour les requêtes complexes. Un tableau associatif est ensuite utilisé pour lier chaque paramètre nommé à une valeur.

## Mise en œuvre PHP

### Paramètres anonyme

```
$sql = "INSERT INTO uneTable (uneColonne, uneAutreColonne)
        VALUES (?,?)";
```

**Description**: Les valeurs des paramètres sont représentées par des points d'interrogation ***?*** dans la requête.

```
 $pdo_stmt->execute(array(150, 'rouge'))
```

**Exécution**: Lors de l'exécution de la requête,vous passes un tableau de valeurs dans l'ordre des

points d'interrogation. Simplicité, mais il faut toujours respecter l'ordre des paramètres dans le tableau lors de l'exécution.

#### Paramètres Nommés

```
$sql = "INSERT INTO uneTable (uneColonne, uneAutreColonne)
        VALUES (:uneValeur, :uneAutreValeur)";
```

**Description**: Les paramètres dans la requête sont identifiés par des noms spécifiques, précédés de deux-points **:** comme **:uneValeur**

```
$pdo_stmt->execute(array(':uneValeur' => 150, ':uneAutreValeur' =>
'rouge'));
```

**Exécution**: Lors de l'exécution, vous passez un tableau associatif, où chaque clé est le nom du paramètre et chaque valeur est celle à utiliser. Nous n'avons pas besoin de suivre l'ordre des paramètres dans la requête. C'est plus lisible et flexible, surtout quand la requête est complexe.

### Mise en œuvre Java

#### Paramètres anonyme

#### Paramètres Nommés

#### Binding

From:
https://wiki.siochaptalqper.fr/ - **Wiki SIO Chaptal**

Permanent link:
**https://wiki.siochaptalqper.fr/doku.php?id=bloc3:sqlinjection&rev=1742235334**

Last update: **2025/03/17 19:15**