

# Sécurité web

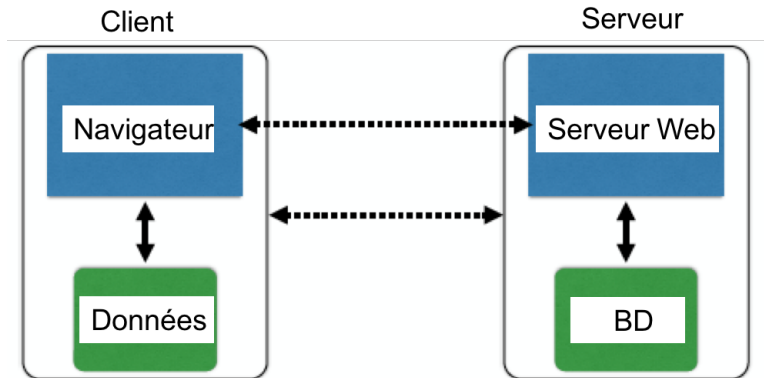
Mehdi Haddad  
`mehdi.haddad@u-pec.fr`

2016 - 2017

# Plan du cours

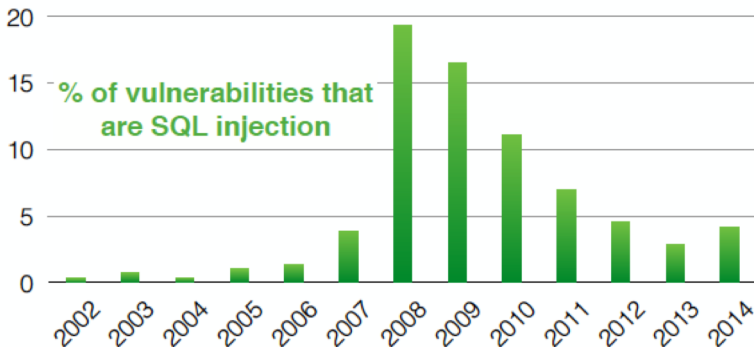
1. Rappel architecture web
2. Injection SQL
3. Contre-mesures injection SQL

# Architecture Web



La base de données (BD) est (souvent) une entité indépendante du serveur web (logiquement et physiquement).

# Injection SQL



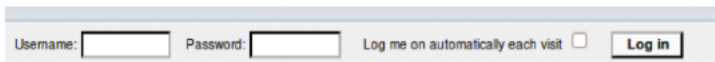
<http://web.nvd.nist.gov/view/vuln/statistics>

# Injection SQL

- ▶ Utilisation des vulnérabilités d'applications tierces accédant à la base de données
- ▶ Attaque classique consiste à introduire du code (malveillant) dans des champs d'un formulaire
- ▶ Si le formulaire est « mal » conçu le code est envoyé à la BD
- ▶ La BD exécute le code malveillant

# Injection SQL

## Site web



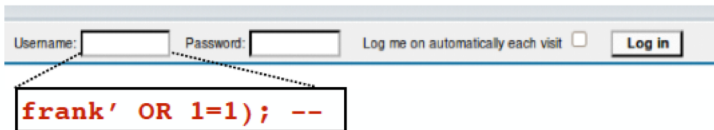
Username:  Password:  Log me on automatically each visit ☐

## Code (PHP)

```
$result = mysql_query("select * from Users  
                        where(name='$user' and password='$pass');");
```

L'objectif d'une injection SQL, sur cet exemple, est de se connecter sans connaître l'utilisateur ni le mot de passe.

# Injection SQL



A screenshot of a web application's login interface. It features a light blue header bar. Below the header, there are two input fields: 'Username:' and 'Password:'. To the right of these fields is a checkbox labeled 'Log me on automatically each visit' and a 'Log in' button. A red-bordered box is overlaid on the 'Password:' field, containing the text 'frank' OR 1=1); --'. Dotted lines connect the corners of this box to the corners of the 'Password:' input field.

Username:  Password:  Log me on automatically each visit ☐

**frank' OR 1=1); --**

```
$result = mysql_query("select * from Users  
    where(name='user' and password='$pass');");
```

```
$result = mysql_query("select * from Users  
    where(name='frank' OR 1=1); --  
    and password='whocares');");
```

# Injection SQL



A screenshot of a web login interface. It features a 'Username:' label followed by a text input field, a 'Password:' label followed by a text input field, a checkbox labeled 'Log me on automatically each visit', and a 'Log in' button. A dotted line connects the password input field to a box below it containing a red SQL injection payload.

frank' OR 1=1); DROP TABLE Users; --

```
$result = mysql_query("select * from Users  
where(name='$user' and password='$pass');");
```

```
$result = mysql_query("select * from Users  
where(name='frank' OR 1=1);  
DROP TABLE Users; --  
and password='whocares');");
```

(La fonction mysql\_query permet d'exécuter plusieurs requêtes)



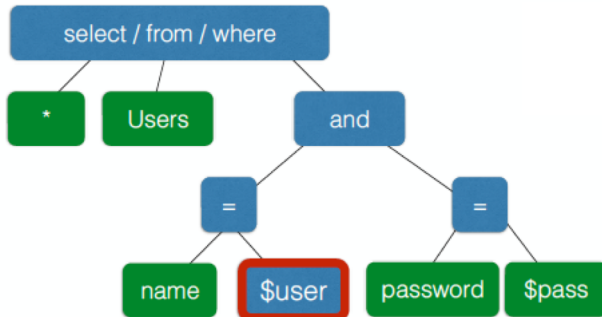
# Injection SQL : origine du problème

```
$result = mysql_query("select * from Users  
where(name='$user' and password='$pass');");
```

- ▶ Cette chaîne de caractère combine du **code** et des **données** fournies par l'utilisateur (\$user et \$pass).
- ▶ Lorsque la frontière entre le code et les données est "floue", des vulnérabilités peuvent apparaître (e.g., buffer overflow).

# Injection SQL : origine du problème

```
$result = mysql_query("select * from Users  
where(name='$user' and password='$pass');");
```



**\$user** devrait être des données et non du code

# Injection SQL : contre mesures

- ▶ On ne peut pas préfigurer que les données fournies par l'utilisateur.
- ▶ Il nécessaire de vérifier / valider les données fournies par l'utilisateur.
- ▶ Rendre fiables les données fournies
  - ▶ Vérifier qu'elles ont le format attendu.
  - ▶ Modifier les données pour les rendre conforme au format attendu.

# Injection SQL : contre mesures

- ▶ Supprimer certains caractères
- ▶ Echapper certains caractères
- ▶ Vérifier les données fournies
- ▶ Utiliser les requêtes préparées (prepared statement)

# Injection SQL : contre mesures

- ▶ Supprimer certains caractères

' ; -- #

- ▶ Inconvénients

- ▶ Certaines applications utilisent ces caractères
- ▶ Comment savoir quels sont les « mauvais » caractères

# Injection SQL : contre mesures

- Echapper certains caractères

Modifier ' en \'

Modifier ; en \ ;

Modifier - en \-

Modifier \ en \\

- Inconvénients

- Difficile à réaliser manuellement

```
magic_quotes_gpc = On
```

(obsolète depuis les dernières versions de php)

```
mysql_real_escape_string()
```

- Besoin d'utiliser ces caractères dans le code SQL.
- Echapper les caractères peuvent ne pas suffire.

## Injection SQL : contre mesures

- ▶ Vérifier les données fournies
- ▶ Exemple : vérifier qu'un entier appartient à un intervalle.
- ▶ Principe : il est plus simple de rejeter une requête que de « réparer » les données fournies. En effet les modifications peuvent introduire de nouvelles erreurs/failles.
- ▶ Inconvénients : Difficile pour des données complexes ou difficilement énumérable

## Prepared Statements

- ▶ Traiter les données fournies par l'utilisateur en fonction de leurs types.
- ▶ Sépare le code et les données.
- ▶ La compilation de la requête est réalisée sans avoir les données de l'utilisateur.

```
$result = mysql_query("select * from Users  
                        where(name='$user' and password='$pass');");
```

```
$db = new mysql("localhost", "user", "pass", "DB");
```

```
$statement = $db->prepare("select * from Users  
                        where(name=? and password=?);");
```

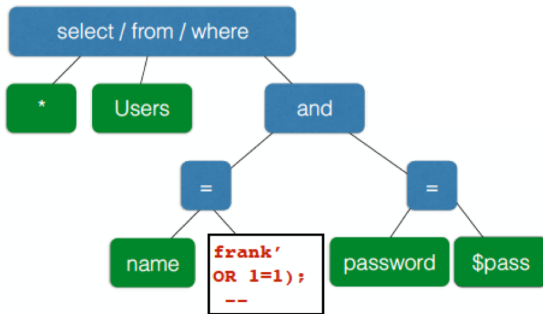
```
$statement->bind_param("ss", $user, $pass);
```

```
$statement->execute();
```



# Prepared Statements

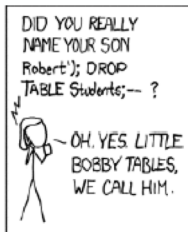
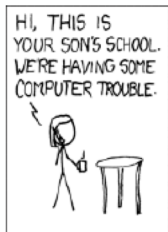
```
$statement = $db->prepare("select * from Users  
    where(name=? and password=?);");  
$stmt->bind_param("ss", $user, $pass);
```



- ▶ La structure de l'arbre de la requête est fixée.
- ▶ Les données fournies par l'utilisateur ne peuvent impacter que les feuilles de l'arbre.

# Prepared Statements

- ▶ Les requêtes SQL sont précompilées.
- ▶ Seulement les emplacements des données (symbole ?) sont inclus dans la requête.
- ▶ Lorsque les données sont liées (bind) leur type est fixé.
- ▶ Les données sont considérées d'un certain type, il n'y a plus de compilation.
- ▶ Elle ne peuvent pas être considérées comme étant du code.



<http://xkcd.com/327/>