

**Sri Shankara's degree**

**College Kurnool**

Rayalaseema University

**Assignment-SQLMAP**

Shaik Reena Tasleem

21360008019

SQLMap is a powerful open-source penetration testing tool that automates the process of detecting and exploiting SQL injection vulnerabilities in web applications. Its primary purpose is to assess the security of web applications by identifying potential vulnerabilities that could allow attackers to execute arbitrary SQL queries on the application's backend database.

Here's a breakdown of its purpose and usage:

- **Detecting SQL Injection Vulnerabilities:** SQLMap can be used to scan web applications for SQL injection vulnerabilities by sending various types of SQL injection payloads to different parameters of the web application (e.g., URL parameters, form fields, cookies). It analyzes the responses from the server to determine if the application is vulnerable to SQL injection.
- **Exploiting SQL Injection Vulnerabilities:** Once SQL injection vulnerabilities are identified, SQLMap can be used to exploit them. It can automatically extract information from the database, such as database schema, tables, columns, and data records. It can also execute arbitrary SQL queries on the database, depending on the level of access gained through the vulnerability.
- **Enumerating Database Information:** SQLMap can perform a comprehensive enumeration of database information, including database management system (DBMS) type, database version, database users, privileges, and more. This information is crucial for understanding the structure and configuration of the database, which aids in further exploitation or security analysis.
- **Dumping Data:** SQLMap can extract data from the database by dumping database tables and their contents. This feature is particularly useful for attackers seeking sensitive information stored in the database, such as user credentials, personally identifiable information (PII), or proprietary data.
- **Advanced Techniques and Customization:** SQLMap provides various advanced techniques and customization options to optimize the detection and exploitation of SQL injection vulnerabilities. Users can specify custom injection payloads, tamper with HTTP requests/responses, specify injection point identification methods, and more.
- **Reporting:** SQLMap generates detailed reports summarizing the results of the vulnerability assessment and exploitation process. These reports typically include information such as identified vulnerabilities, extracted data, and recommendations for remediation.

However, it's important to note that SQLMap should be used responsibly and ethically. Unauthorized or malicious use of SQLMap against web applications without proper authorization is illegal and can lead to severe legal consequences. It should only be used by security professionals or ethical hackers in controlled environments with proper

authorization from the application owner.

Sqlmap is an open-source penetration testing tool. It comes with a powerful detection engine. It automates the process of detecting & taking over the database server. There is the total of six SQL injection tool techniques are present. This is the highest amount of tools present than others. When we are going to extract the password from a vulnerable database, often the passwords are in hash form. It can detect the hash & can mention which type of hash was that.

## Features:

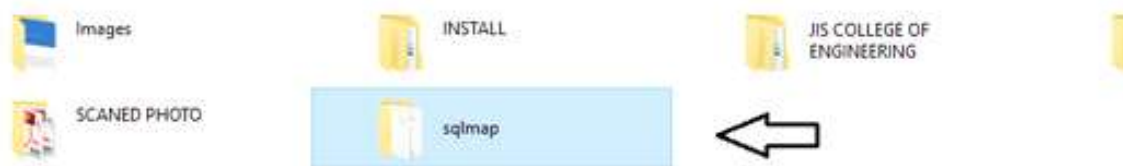
- It supports extracting user, password hashes, tables etc.
- We can download & update any file from the database server underlying file system.

## Downloading & Installation:

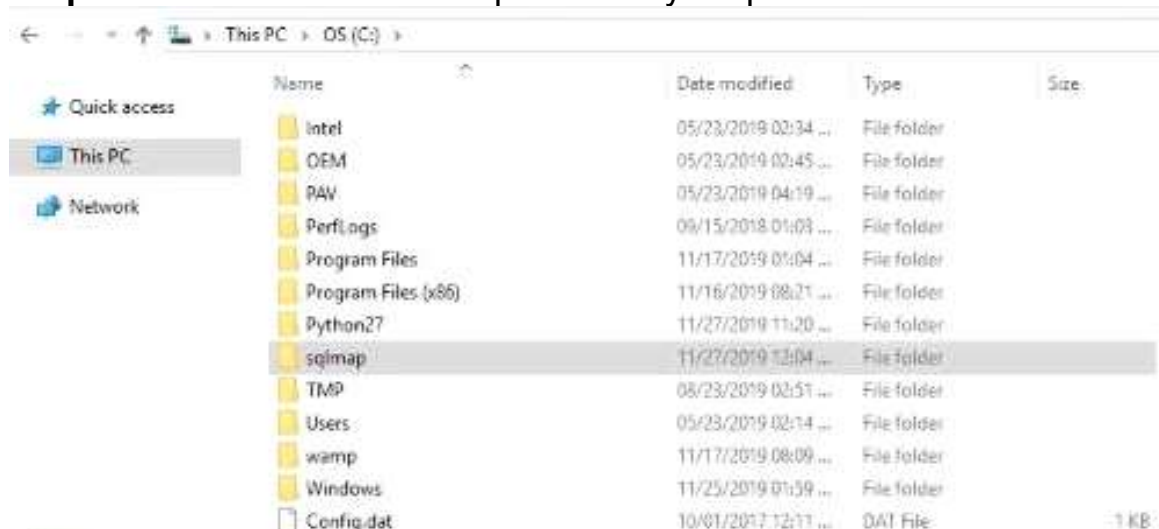
**Step 1:** Browse to this [link](#).

**Step 2:** Click on the zip file on the right side & download the file.

**Step 3:** Then you have to extract the zip file. And then rename it to 'sqlmap'



**Step 4:** Then cut the folder & paste it to your pc C drive

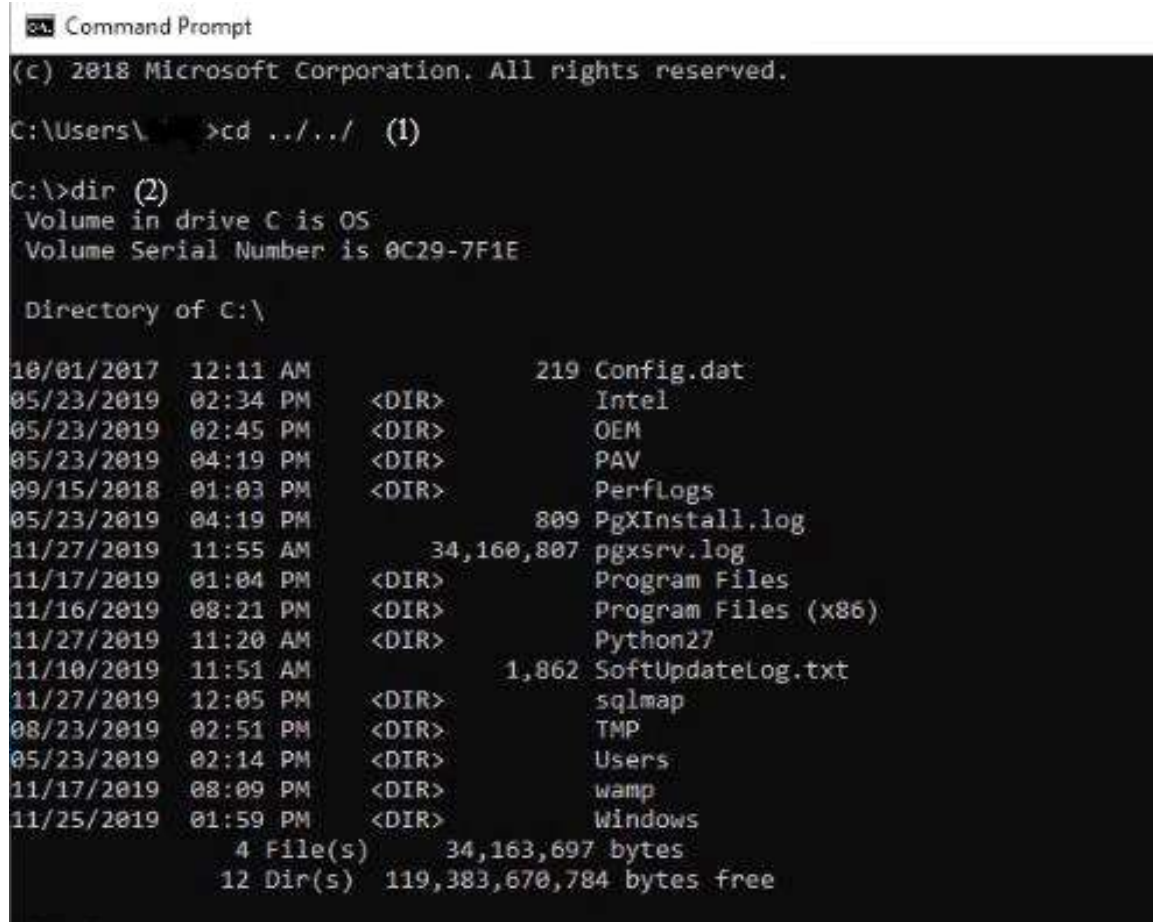


**Step 5:** Open Command Prompt from the start menu.

**Step 6:** Write down the following command one by one

```
cd ../ ../
```

```
dir
```



```
Command Prompt
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\ >cd ../ ../ (1)

C:\>dir (2)
Volume in drive C is OS
Volume Serial Number is 0C29-7F1E

Directory of C:\

10/01/2017  12:11 AM                219 Config.dat
05/23/2019  02:34 PM             <DIR>      Intel
05/23/2019  02:45 PM             <DIR>      OEM
05/23/2019  04:19 PM             <DIR>      PAV
09/15/2018  01:03 PM             <DIR>      PerfLogs
05/23/2019  04:19 PM                809 PgXInstall.log
11/27/2019  11:55 AM          34,160,807 pgxsrv.log
11/17/2019  01:04 PM             <DIR>      Program Files
11/16/2019  08:21 PM             <DIR>      Program Files (x86)
11/27/2019  11:20 AM             <DIR>      Python27
11/10/2019  11:51 AM            1,862 SoftUpdateLog.txt
11/27/2019  12:05 PM             <DIR>      sqlmap
08/23/2019  02:51 PM             <DIR>      TMP
05/23/2019  02:14 PM             <DIR>      Users
11/17/2019  08:09 PM             <DIR>      wamp
11/25/2019  01:59 PM             <DIR>      Windows
               4 File(s)      34,163,697 bytes
              12 Dir(s)  119,383,670,784 bytes free
```

**Step 7:** Then write another some commands

```
cd sqlmap
```

```
sqlmap.py
```

```
Command Prompt - sqlmap.py
11/27/2019 11:55 AM 34,160,807 pgxsrv.log
11/17/2019 01:04 PM <DIR> Program Files
11/16/2019 08:21 PM <DIR> Program Files (x86)
11/27/2019 11:20 AM <DIR> Python27
11/10/2019 11:51 AM 1,862 SoftUpdateLog.txt
11/27/2019 12:05 PM <DIR> sqlmap
08/23/2019 02:51 PM <DIR> TMP
05/23/2019 02:14 PM <DIR> Users
11/17/2019 08:09 PM <DIR> wamp
11/25/2019 01:59 PM <DIR> Windows
4 File(s) 34,163,697 bytes
12 Dir(s) 119,383,670,784 bytes free

C:\>cd sqlmap (1)
C:\sqlmap>sqlmap.py (2)

  H
  |
  | [1.3.11.106#dev]
  | [C]
  | [V...]
  | http://sqlmap.org

Usage: sqlmap.py [options]

sqlmap.py: error: missing a mandatory option (-d, -u, -l, -m, -r, -g, -c, --list-tampers, --wizard, --dependencies). Use -h for basic and -hh for advanced help

Press Enter to continue...
```

It will give the proper output & hence your installation is successful.

[Now get an additional 30% off on all GfG courses of your choice. Also get 90% Course fee refund in just 90 days. Dual savings offer ending soon, avail today!](#)

Here's a complete roadmap for you to become a developer: **Learn DSA -> Master Frontend/Backend/Full Stack -> Build Projects -> Keep Applying to Jobs**

And why go anywhere else when our [DSA to Development: Coding Guide](#) helps you do this in a single program! Apply now to our [DSA to Development Program](#) and our counsellors will connect with you for further guidance & support.

Identifying a vulnerable web application for testing purposes should be done ethically and legally, with proper authorization. Here are a few methods to find vulnerable web applications:

- **Vulnerable Web Application Databases:** Some organizations maintain databases of vulnerable web applications for educational and testing purposes. Examples include the OWASP (Open Web Application Security Project) Broken Web Applications Project and VulnHub. These platforms offer a variety of

intentionally vulnerable web applications that you can download and deploy in your testing environment.

- **Capture The Flag (CTF) Events:** Participating in Capture The Flag events, especially those focused on web security, can provide access to vulnerable web applications. CTF challenges often involve finding and exploiting vulnerabilities in web applications to gain points or solve puzzles. Platforms like Hack The Box, TryHackMe, and OverTheWire host CTF challenges regularly.
- **Vulnerability Disclosure Programs:** Some companies and organizations have vulnerability disclosure programs that allow security researchers to responsibly report security vulnerabilities they find in their web applications. By participating in such programs, you may gain access to vulnerable applications legally and with the organization's consent.
- **Online Vulnerable Applications:** Some websites and online platforms host intentionally vulnerable web applications for educational purposes. These applications are often designed specifically for practicing web security testing techniques. However, ensure that you have permission to test these applications, as unauthorized testing could be illegal.
- **Create Your Own Vulnerable Web Application:** If you have web development skills, you can create your own vulnerable web application for testing purposes. By intentionally introducing common vulnerabilities such as SQL injection, cross-site scripting (XSS), and insecure authentication mechanisms, you can practice identifying and exploiting these vulnerabilities in a controlled environment.

Regardless of the method you choose, it's crucial to always obtain proper authorization before testing or exploiting any web application. Unauthorized testing can be illegal and unethical, and it may lead to legal consequences. Always ensure that you have explicit permission from the application owner or host before conducting any security testing.

Performing a basic SQL injection attack involves exploiting vulnerabilities in a web application that allows an attacker to manipulate SQL queries executed by the application's backend database. Here's a step-by-step guide to performing a basic SQL injection attack:

- **Identify Input Fields:** Identify input fields or parameters in the web application that may be vulnerable to SQL injection. Common vulnerable input fields include login forms, search forms, and URL parameters.
- **Understand the SQL Injection Point:** Determine the point where user input is incorporated into SQL queries without proper sanitization or validation. This could be in a URL parameter, a form field, or any other input mechanism.
- **Test for Vulnerability:** Start by entering simple characters like single quotes (') or double quotes (") into the input field. If the application responds with an error message or behaves unexpectedly, it might indicate a potential SQL injection vulnerability.

- **Inject SQL Code:** Once you've identified a potential vulnerability, try injecting SQL code into the input field to manipulate the SQL query executed by the application's backend database. Common SQL injection payloads include:
  - **' OR 1=1 --:** This payload typically results in a true condition, causing the application to retrieve all records from the database.
  - **'; DROP TABLE users; --:** This payload attempts to drop the "users" table from the database. Be cautious when using destructive payloads as they can cause permanent data loss.
- **Observe Application Response:** Pay attention to the application's response after injecting the SQL payload. Look for changes in behavior, error messages, or any unusual output that indicates successful injection.
- **Extract Information:** If the injection is successful, you can extract sensitive information from the database using additional SQL commands. For example:
  - Retrieve data from specific columns: **' UNION SELECT username, password FROM users; --**
  - Retrieve database version: **' UNION SELECT @@version; --**
  - Retrieve database name: **' UNION SELECT database(); --**
- **Exploit the Vulnerability:** Once you've identified the vulnerability and extracted relevant information, you can further exploit it by performing actions such as authentication bypass, data manipulation, or privilege escalation, depending on the application's functionality and your goals.
- **Report the Vulnerability:** If you discover a SQL injection vulnerability in a web application, responsibly report it to the application owner or administrator following their disclosure policy. Provide detailed information about the vulnerability and any steps needed to reproduce it.

Remember, always conduct security testing and ethical hacking activities within the bounds of the law and with proper authorization. Unauthorized access to or manipulation of web applications is illegal and unethical.

To document the steps for a basic SQL injection attack, you can create a structured document outlining each stage of the attack process. Here's a template you can follow:

## SQL Injection Attack Documentation

## **1. Introduction:**

- Brief overview of the SQL injection attack and its significance in web security testing.

## **2. Target Application:**

- Description of the web application under test, including its purpose, functionality, and URL.

## **3. Identification of Vulnerable Input Fields:**

- List of input fields or parameters identified as potentially vulnerable to SQL injection.

## **4. Understanding the SQL Injection Point:**

- Explanation of how user input is incorporated into SQL queries without proper validation.

## **5. Testing for Vulnerability:**

- Results of initial tests to identify signs of vulnerability, such as error messages or unexpected behavior.

## **6. Injection of SQL Code:**

- Description of SQL injection payloads used, including the payload itself and variations attempted.

## **7. Observation of Application Response:**

- Documentation of the application's response to injected SQL payloads, noting changes in behavior or error messages.

## **8. Extraction of Information:**

- SQL commands used to extract information from the database, along with the type of information retrieved.



## **9. Exploitation of the Vulnerability:**

- Actions taken to further exploit the SQL injection vulnerability, such as authentication bypass or data manipulation.

## **10. Reporting the Vulnerability:**

- Details of the responsible disclosure process, including communication with the application owner/administrator and recommendations for mitigation.

## **11. Conclusion:**

- Summary of key findings, lessons learned, and recommendations for improving web application security.

## **12. References:**

- Citations or links to resources consulted during the SQL injection attack testing.

You can fill in each section with relevant information and details specific to your testing scenario. This structured approach helps ensure that all aspects of the attack process are documented thoroughly and can be easily understood by others reviewing the report.

..... The End .....