#### Project report on

**Database access through SQL injection**

**A Dissertation submitted in partial fulfillment of the Academic requirements for the award of the degree of**

**Bachelor of Technology**

## In

**Computer Science & Engineering (Cyber Security)**

K Akshith (22H51A6228)

M Neeraj (22H51A6232)

P Ruchitha (22H51A6240)

**Under the esteemed Guidance of**

**Dr.R.Venkateswara Reddy (Associate Professor and HOD,CSC)**

**Department of Cyber Security**

**CMR COLLEGE OF ENGINEERING & TECHNOLOGY**

**(Autonomous)**

**(NAAC Accredited with ‘A+’ Grade & NBA Accredited) (Approved by AICTE, Permanently Affiliated to JNTU Hyderabad)**

**KANDLAKOYA, MEDCHAL ROAD, HYDERABAD-501401**

#### CMR COLLEGE OF ENGINEERING & TECHNOLOGY

**(Autonomous)**

**(NAAC Accredited with ‘A+’ Grade & NBA Accredited) (Approved by AICTE, Permanently Affiliated to JNTU Hyderabad)**

**KANDLAKOYA, MEDCHAL ROAD, HYDERABAD-501401**

**DEPARTMENT OF CYBER SECURITY**



#### CERTIFICATE

This is to certify that the Mini Project -1 report entitled “**Database access through SQL injection** ”being submitted by **K Akshith (22H51A6228), M Neeraj (22H51A6232), P Ruchitha (22H51A6240)** in partial fulfillment for the award of **Bachelor of Technology in Computer Science and Engineering (Cyber Security)** is a record of bonafide work carried out his/her under my guidance and supervision.

The results embodied in this project report have not been submitted to any other University or Institute for the award of any Degree.

1. LYDIA Dr. R. Venkateswara Reddy

Assistant Professor Associate Professor & HOD

Dept. of CSC Dept. of CSC

#### ACKNOWLEDGEMENT

With great pleasure I want to take this opportunity to express my heartfelt gratitude to all the people

`who helped in making this project a grand success.

I am grateful to **A. LYDIA**, Assistant Professor, Dept. of Computer Science and Engineering for her valuable suggestions and guidance during the execution of this project.

I would like to thank **Dr. R. Venkateswara Reddy**, Head of the Department of Computer Science and Engineering, for his moral support throughout the period of my study in CMRCET.

I am highly indebted to **Major Dr. V.A. NARAYANA**, Principal CMRCET, for giving permission to carry out this project in a successful and fruitful way.

I would like to thank the Teaching & Non- teaching staff of the Department of Computer Science and Engineering for their co-operation.

Finally, I express my sincere thanks to **Mr. CH. GOPAL REDDY**, Secretary, CMR Group of Institutions, for his continuous care. I sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project work.

K Akshith (22H51A6228)

M.NEERAJ (22H51A6232)

P. Ruchitha (22H51A6240)

#### ABSTRACT

* SQL injection is a prevalent cybersecurity threat where attackers exploit vulnerabilities in web applications to execute malicious SQL commands.
* Attackers manipulate input fields to inject SQL queries that alter database commands, potentially gaining unauthorized access to sensitive data.
* Successful SQL injection attacks can lead to data theft, data loss, unauthorized data manipulation, and in severe cases, full compromise of the database.
* Effective prevention strategies include input validation, use of parameterized queries, and regular security audits to detect and mitigate vulnerabilities.
* Understanding and implementing robust security measures are crucial in safeguarding databases against SQL injection attacks, ensuring data integrity and user trust.

**Table Of Content**

|  |  |  |
| --- | --- | --- |
| **CHAPTERS** | **DESCRIPTION** | **PAGE NUMBERS** |
| 1 | INTRODUCTION | 2 |
| 1.1 | AIM | 3 |
| 1.2 | SCOPE | 4 |
| 2 | LITERATURE REVIEW | 6 |
| 3 | EXISTING SOLUTIONS | 8 |
| 4 | PROPOSED SYSTEM | 11 |
| 4.1 | REQUIREMENT ANALYSIS | 12 |
| 4.1.1 | HARDWARE REQUIREMENTS | 12 |
| 4.1.2 | SOFTWARE REQUIREMENTS | 12 |
| 4.2 | MERITS AND DEMERITS | 13 |
| 5 | DESIGN DESCRIPTION | 15 |
| 5.1 | CONCEPTUAL DESIGN | 15 |
| 6 | IMPLEMENTATION AND DISCUSSION | 16 |
| 6.1 | IMPLEMENTATION | 17 |
| 7 | RESULT | 20 |
| 8 | CONCLUSION AND FUTURE ENHANCEMENT | 21 |
| 8.1 | CONCLUSION | 21 |
| 8.2 | ENHANCEMENT | 21 |
| 8.3 | REFERENCES | 24 |

# CHAPTER 1

1

#### INTRODUCTION

* SQL injection is a technique where malicious SQL statements are inserted into an entry field for execution. It exploits vulnerabilities in poorly sanitized user inputs in web applications.
* The goal of SQL injection is to manipulate the database queries executed by the application's backend, potentially gaining unauthorized access to data or performing unintended operations.
* Attackers typically inject SQL commands into input fields such as login forms, search queries, or URL parameters. If the application fails to properly sanitize inputs, these commands can be executed by the database.
* Successful SQL injection attacks can lead to data theft, modification, or deletion. Attackers can extract sensitive information like usernames, passwords, credit card details, or even gain administrative access to the application or server.
* To prevent SQL injection, developers should use parameterized queries (prepared statements) or ORM frameworks that automatically sanitize inputs. Input validation and limiting database permissions are also crucial measures to mitigate this type of attack.

#### AIM

* + The aim of our project is to obtain sensitive information stored in the database, such as user credentials, personal details, or financial records.
  + Modify existing data in the database, which can include altering records, deleting information, or inserting new data.
  + Gain control over the application's backend operations, potentially allowing for administrative privileges or executing arbitrary commands on the server.

#### SCOPE

* + Access sensitive information stored in the database, including usernames, passwords, credit card details, and personal records.
  + Modify or delete existing data within the database, altering records or transactions.
  + Gain unauthorized access to administrative functions or accounts with elevated privileges.
  + Execute commands on the database server, potentially compromising the entire server environment.
  + Exploit vulnerabilities to disrupt or disable the application, impacting availability and functionality.

##### Web Development

* + Web development, also known as website development, refers to the tasks associated with creating, building and maintain websites and web applications that run online on a browser, it may, however, also include web design, web programming, and database management.
  + Web development is closely related to the job of designing the features and functionality of apps (web design). The basic tools involved in web development are programming languages like HTML (Hyper Text Markup Language), CSS (Cascading Style Sheets), JavaScript, Flask.



# CHAPTER 2

#### LITERATURE REVIEW

##### Halfond, Viegas, and Orso (2006):

This method involves injecting malicious SQL code into an entry field for execution, exploiting

vulnerabilities in an application's software to manipulate or access the data base directly. According to

comprehensive review by Halfond, Viegas, and Orso (2006), SQL injection attacks can lead to

unauthorized access to sensitive data, data corruption, and even the complete compromise of the affected

system.

##### Ali and Ismail (2012):

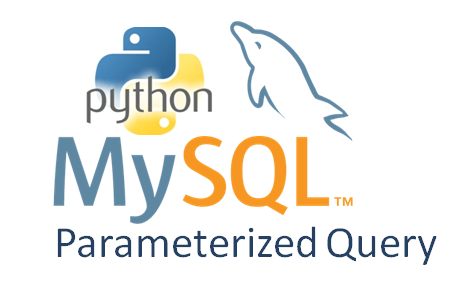
A more recent study by Ali and Ismail (2012) delves into the evolving nature of SQLi techniques and the sophistication of attacks. They categorize SQL injection attacks into several types, including union-based, error-based, and blind SQL injection, each with distinct characteristics and detection challenges.

# CHAPTER 3

#### EXISTING SOLUTION

##### Parameterized Queries:

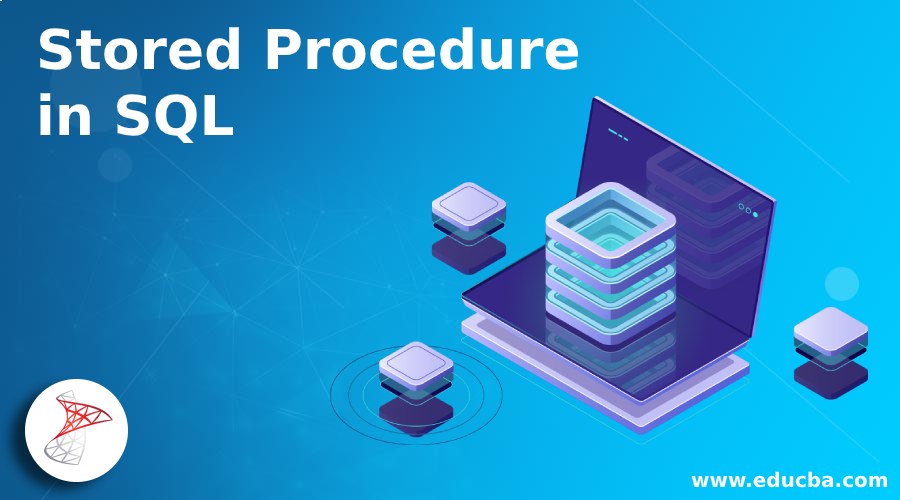
Parameterized queries ensure that SQL code is not directly concatenated with user input. Instead, placeholders are used, and user inputs are passed as parameters, which the database engine treats as data, not executable code.



**Fig 1:** Parameterized Queries

##### Stored Procedures:

Stored procedures are precompiled collections of SQL statements stored in the database. By using stored procedures, the SQL code is predefined and separated from user inputs, reducing the risk of SQL injection.



**Fig 2:** Stored Procedures

##### Input Validation and Sanitization:

Input validation involves checking user inputs to ensure they conform to expected formats (e.g., no special characters, length constraints).



**Fig 3:** Input Validation and Sanitization

##### Web Application Firewalls:

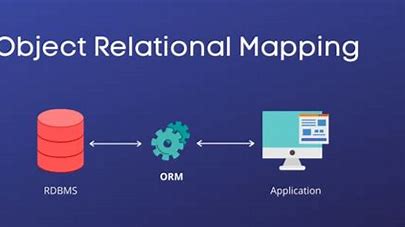
WAFs provide an additional layer of security by monitoring, filtering, and blocking potentially harmful traffic to web applications.



##### ORM:

**Fig 4:** Web Application Firewalls

Utilizing ORM (Object-Relational Mapping) tools like Hibernate or Entity Framework which automatically handle query building and parameterization, reducing SQL injection risks.



**Fig 5:** ORM (Object-Relational Mapping)

# CHAPTER 4

#### PROPOSED SYSTEM

To prevent SQL injection attacks, implementing parameterized queries (also known as prepared statements) is essential. This technique ensures that user inputs are treated as data rather than executable code, thus safeguarding against malicious inputs that attempt to alter SQL queries. By separating the SQL code from the data, parameterized queries make it impossible for attackers to manipulate the SQL commands through user inputs.

Additionally, employing robust input validation and sanitization practices further enhances security. This involves rigorously checking and cleaning user inputs to ensure they conform to expected formats and do not contain potentially harmful content. Using ORM (Object-Relational Mapping) frameworks can also help abstract database interactions, reducing the likelihood of direct SQL injection vulnerabilities.

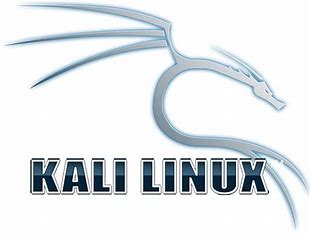
#### REQUIREMENT ANALYSIS

##### Software Requirements

* + - * Windows 7, Linux, fire fox
      * Web application framework
      * Scanning Tools

##### Hardware Requirements

* + - * System 32 or 64 bit with 4 GB or 8 GB RAM
      * Network Security Devices
      * CPU
      * RAM

##### MERITS AND DEMERITS Merits:

* + - * Exploitation for Testing and Education
      * Highlighting Security Flaws
      * User-Friendly Interface

##### Demerits:

* + - * Security Risks
      * System Compromise

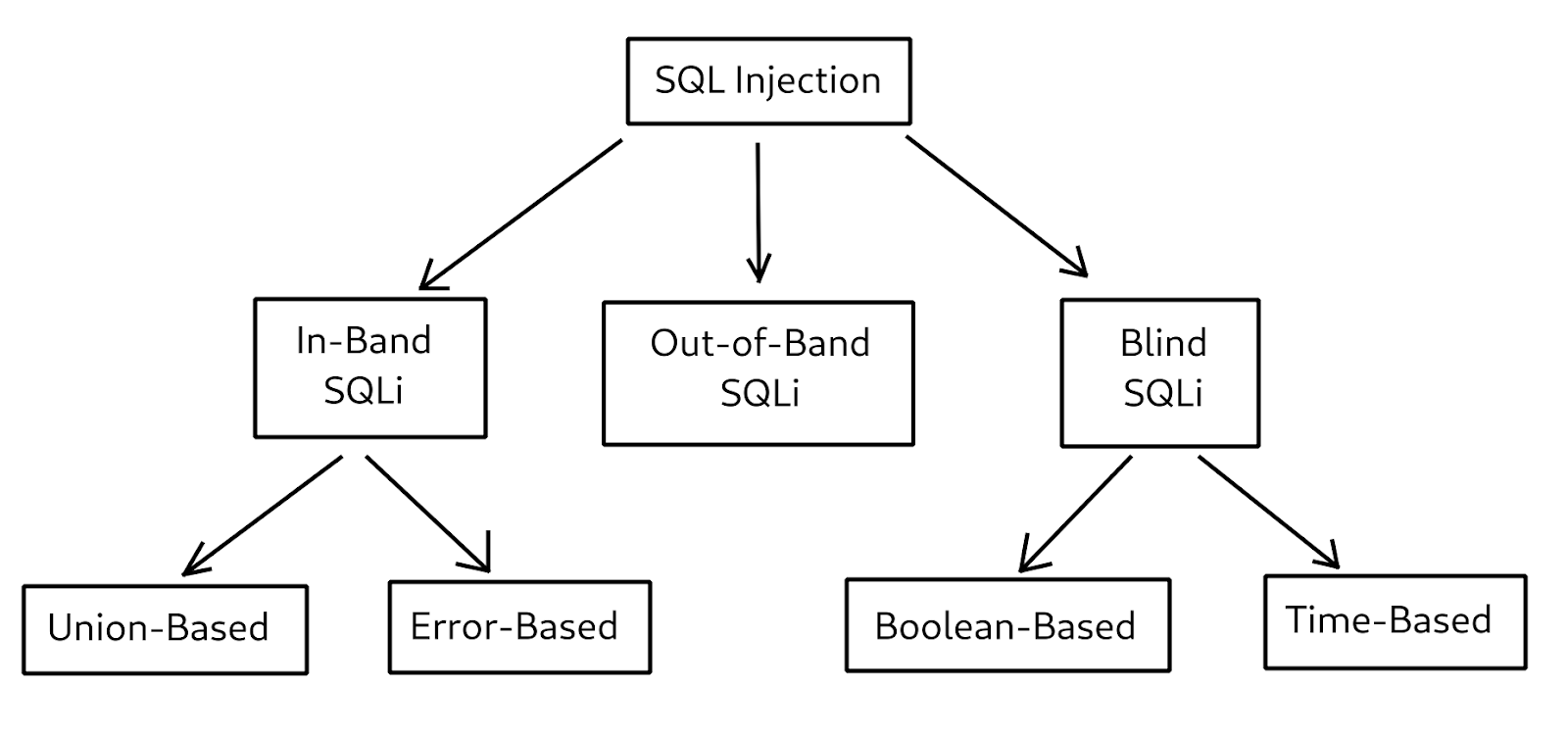
# CHAPTER 5

14

#### DESIGN DESCRIPTION

##### 5.1 CONCEPTUAL DESIGN

The diagram shows the steps involved in Database access through SQL injection



**Fig 6:** Data base access through SQL injection

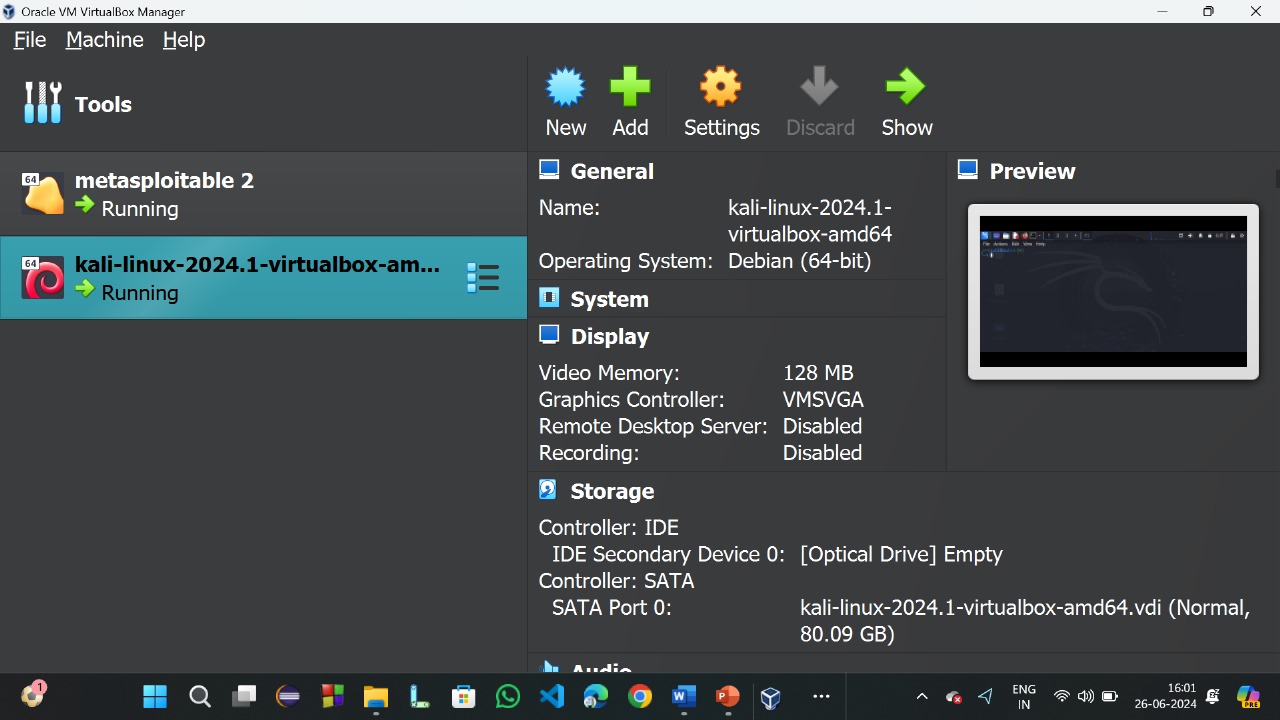
# CHAPTER 6

#### IMPLEMENTATION AND DISCUSSION

* 1. **IMPLEMENTATION**

**Reconnaissance:**

Reconnaissance through SQL injection involves attackers probing a database to gather information without initially causing damage. They craft specific SQL queries to extract metadata about the database structure, such as table names, column names, and data types. This preliminary exploration helps them understand the database's architecture and vulnerabilities, which can be exploited later for more targeted and damaging attacks.



**Fig 6:** Reconnaissance

## Fire fox :

To prevent such threats, it is crucial to use parameterized queries and input validation, ensuring user inputs are treated as data, not executable code.

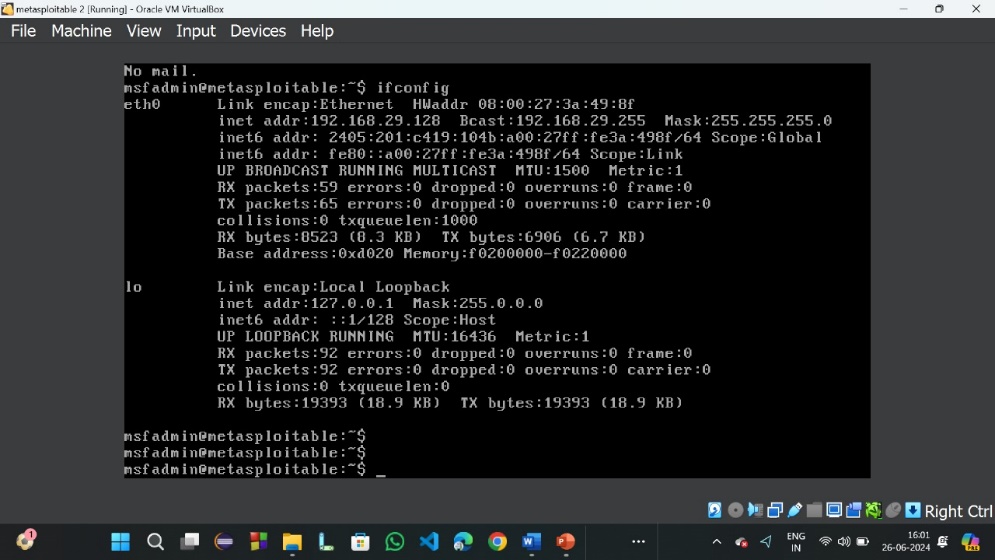
### 

### Fig 7:Fire fox

**Metasploitable for location:**

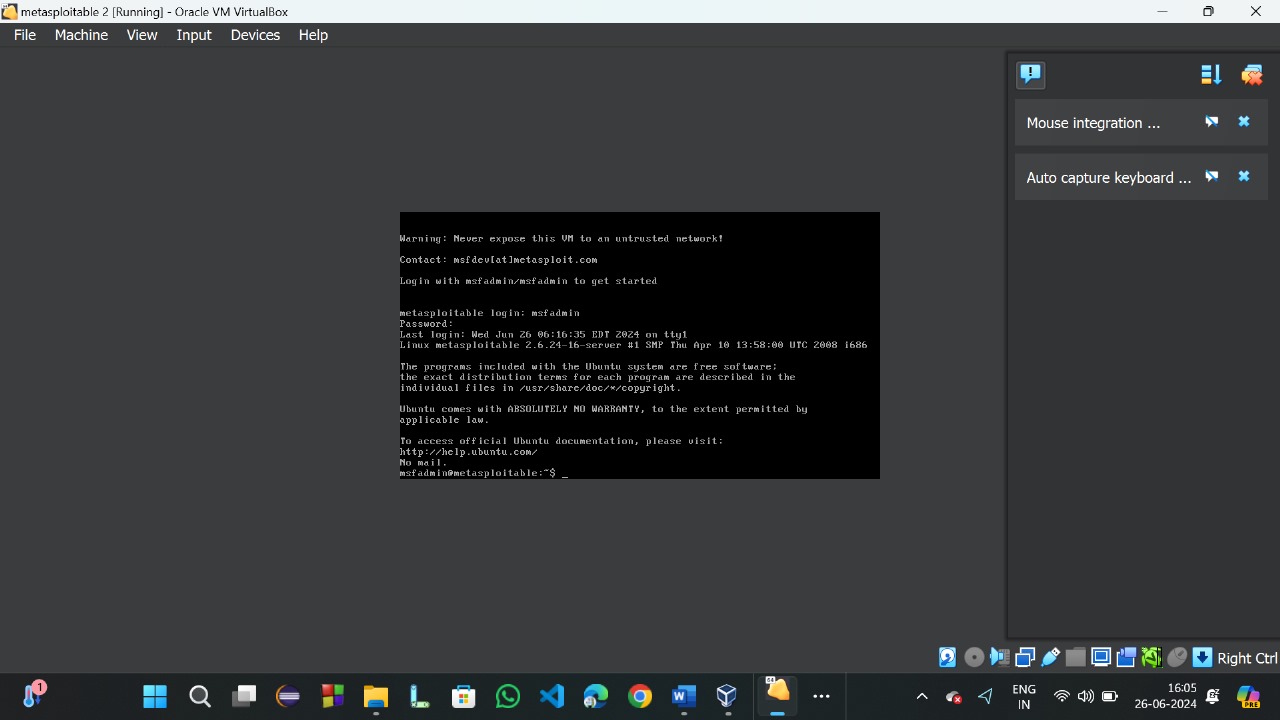
## Gaining access:

It is the phase where an attacker obtains control over the target. Be it a network or a web application, “Gaining Access” is only the beginning.



**Fig 8:**Gaining access

**COMMAND:**



**Fig 9**

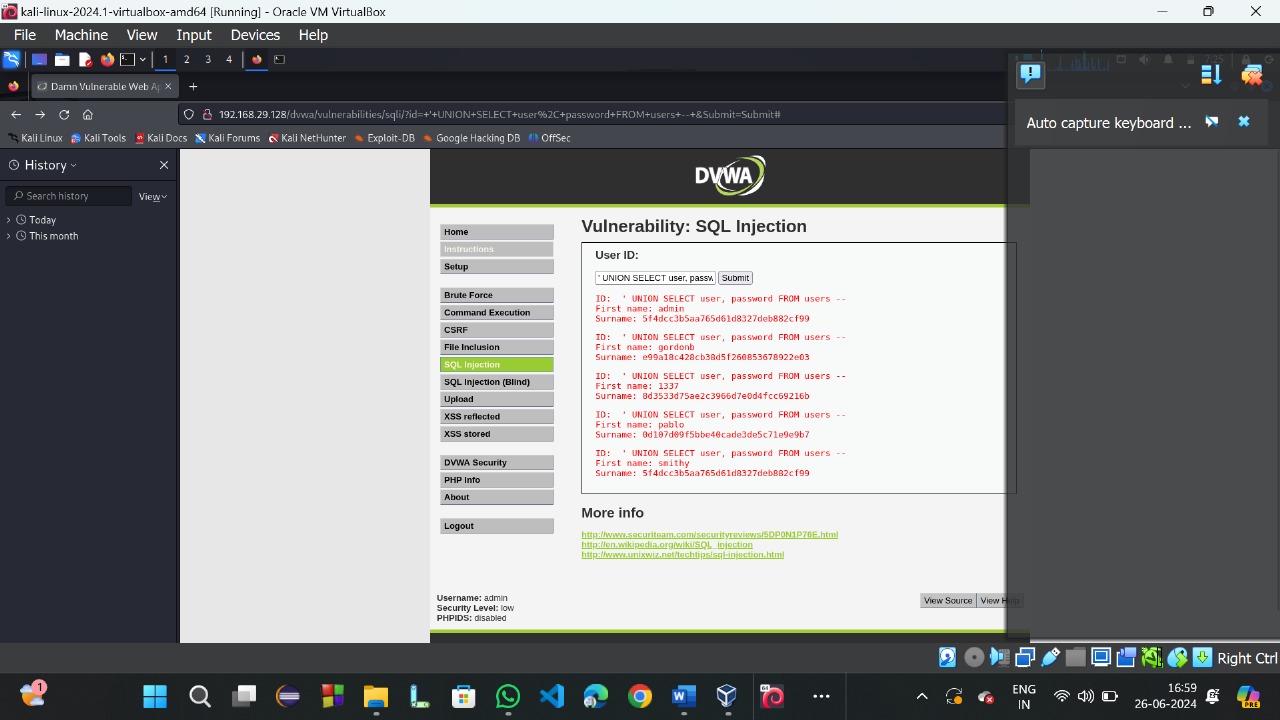


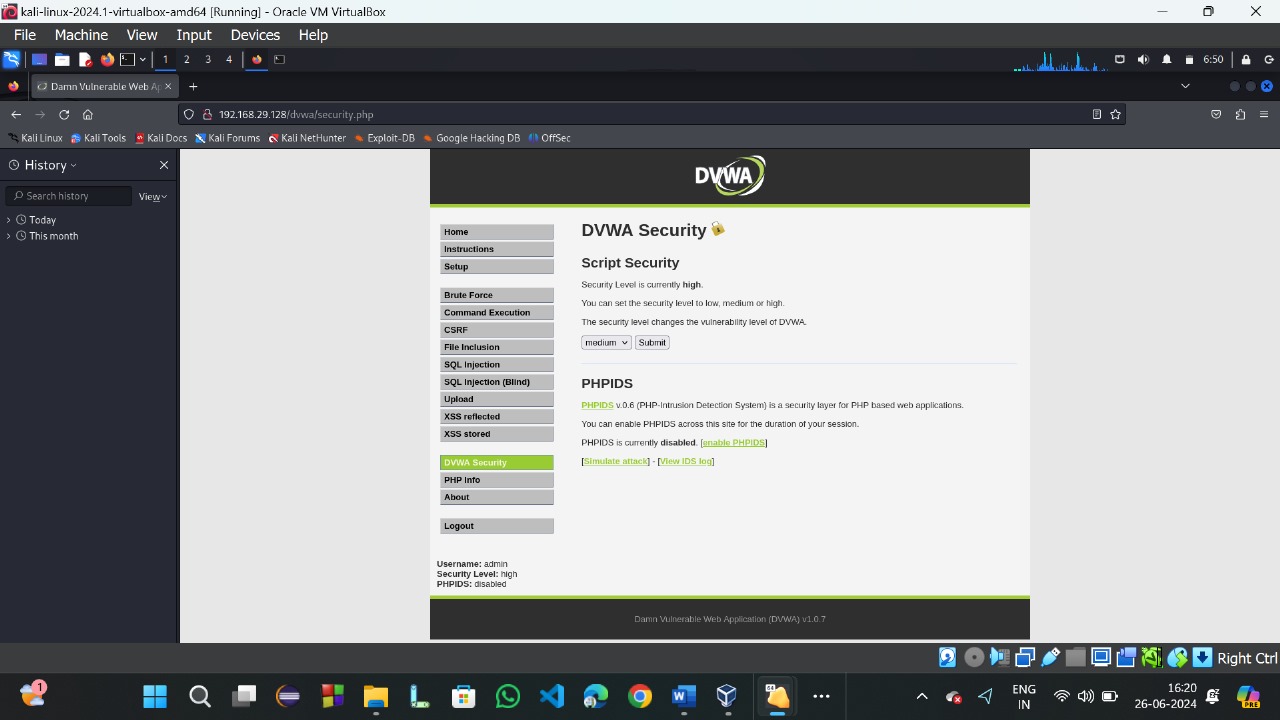
Fig 10

# 

# CHAPTER 7

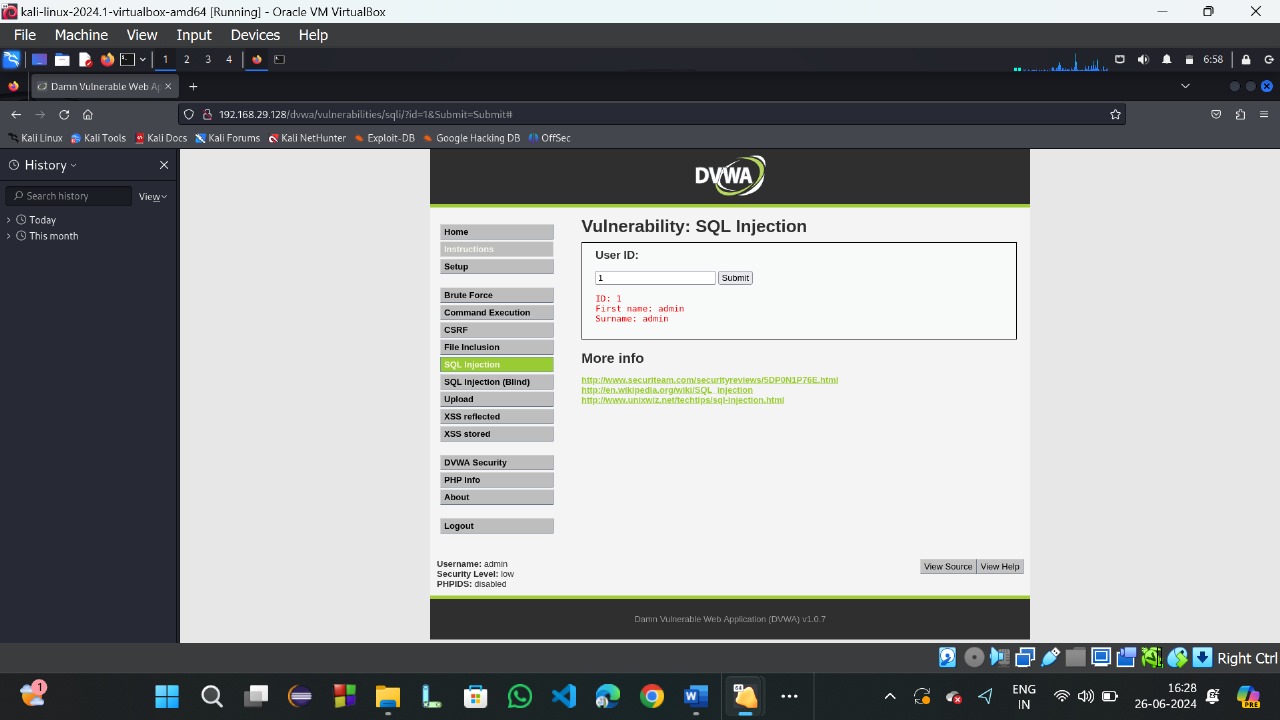
#### RESULT

we have successfully gained the access to the vulnerable system .Now we can take control of system



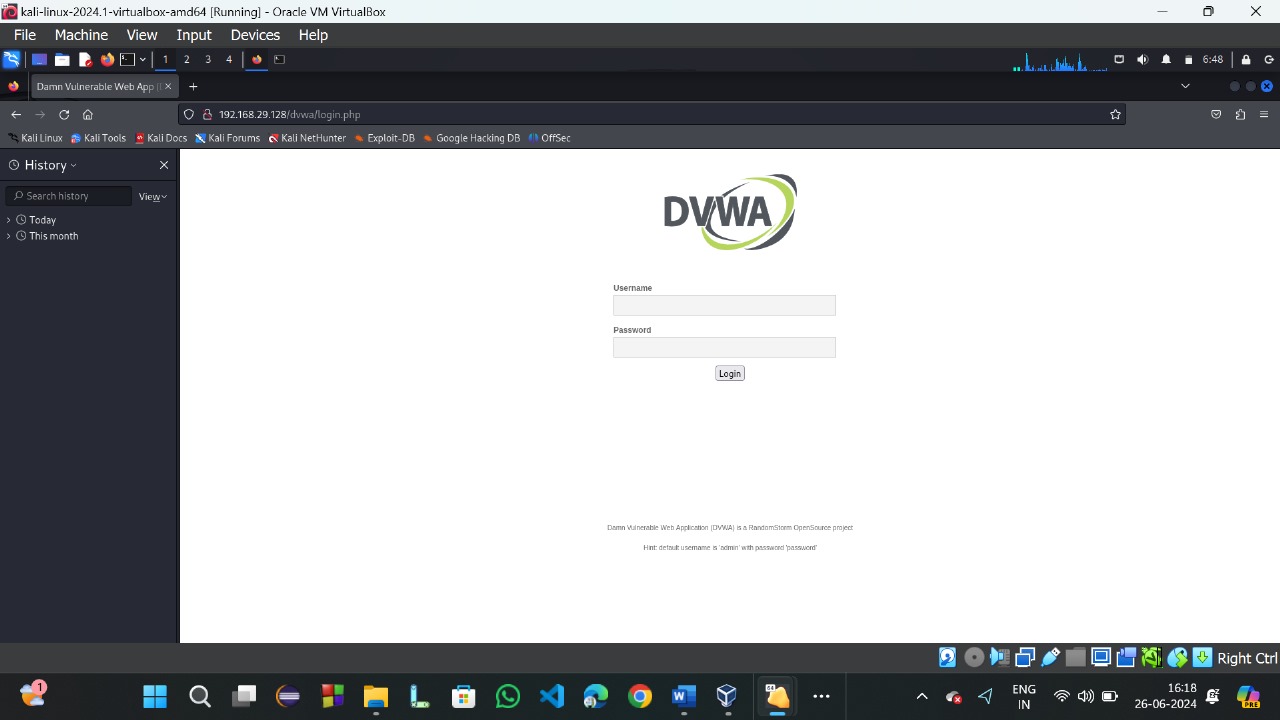
**Fig 11**

**Interaface:**



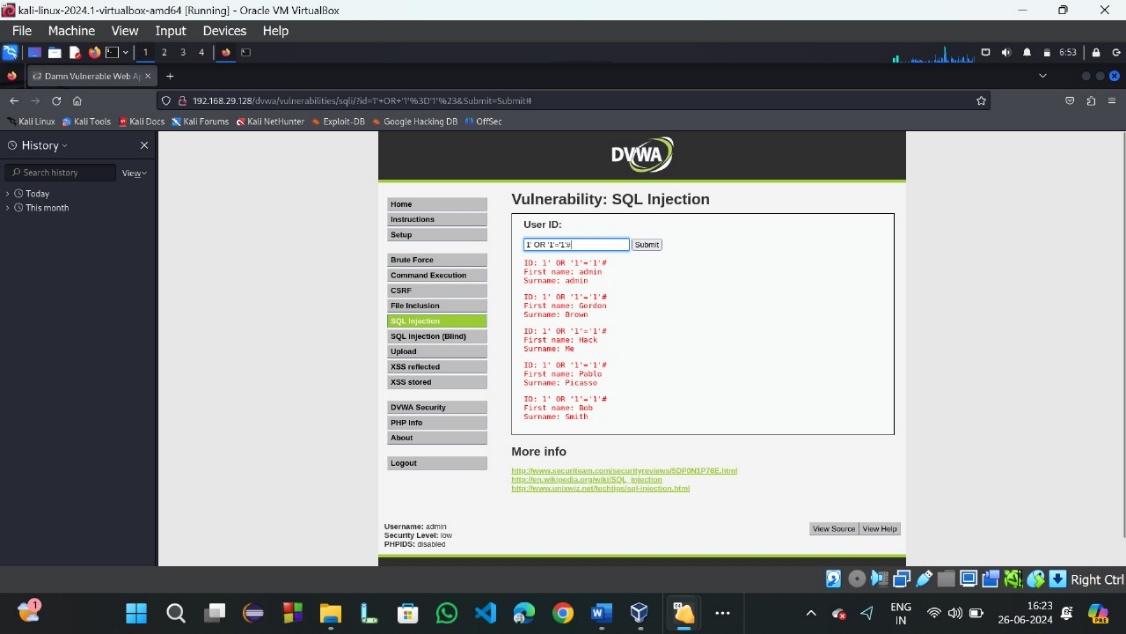
**Fig 12**

## Vulnerability -SQL injection:



**Fig 13**

***Kali Linux vulnerability DVWA report had been prepared and uploaded successfully in the website.***



## Website link:

https://cloud.google.com/sql/

# 

# CHAPTER 8

#### CONCLUSION AND FUTURE ENHANCEMEMT

##### 8.1. CONCLUSION

* In conclusion, SQL injection poses a significant threat to database security, with the potential to expose sensitive data, disrupt operations, and compromise entire systems.
* Addressing this risk requires implementing robust countermeasures such as parameterized queries, input validation, and the use of ORM frameworks.
* This project serves as a foundation for continuous improvement and commitment to security.
* By raising awareness and incorporating these security practices into the development lifecycle, organizations can safeguard their databases against SQL injection attacks, ensuring the integrity and confidentiality of their data.
* Thanks to the team members and stakeholders for their dedicated efforts.

##### FUTURE ENHANCEMENTS

* Future enhancements in mitigating SQL injection risks could involve the integration of advanced automated detection and prevention techniques within database management systems and application frameworks. This includes the development of AI-driven algorithms capable of dynamically analyzing query patterns and detecting anomalous behaviors indicative of SQL injection attempts.

#### REFERENCES

1. OWASP. (n.d.). SQL Injection."A Comprehensive Study on Data base access through SQL injection." International Journal of Computer Applications..
2. Sucuri. (2021). SQL Injection Attacks: Prevention and Protection ."A Survey on Data base access through SQL injection .
3. SANS Institute. (n.d.). SQL Injection .”An Approach on Data base access through SQL injection”.