

**A Mini Project REPORT ON**

**“Website Database Hacking using SQLMap Tool | Ethical Hacking - SQL Injection Attackect”**

**Submitted By**

STUDENT NAME ROLL NO.

Nikhil Malvi B-50

Under The Guidance Of

**Prof. Pooja Dehankar**

IN THE FULFILLMENT OF

**Mini -Project REPORT**



**DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE ENGINEERING**

1. **Y. 202-25**

**Ajeenkya D. Y. Patil School of Engineering,**

**Lohegaon, Pune – 412 105**

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE ENGINEERING**



**CERTIFICATE**

This is to certify that **Nikhil Malvi**, a student of **B-50**, from **Ajeenkya D. Y. Patil School of Engineering**, has successfully completed the **Mini Project** in **Website Database Hacking using SQLMap Tool | Ethical Hacking - SQL Injection Attack** during the academic year **2024-2025**. This project involved in-depth analysis and hands-on application of SQL Injection techniques, showcasing the practical skills and understanding of ethical hacking and cybersecurity measures.

Date: 24/04/2025

|  |  |  |
| --- | --- | --- |
| **Prof. Pooja Dehankar** | **Dr. Bhagyashree Dhakulkar** | **Dr. F. B. Sayyad** |
| **Mentor** | **HOD-AI&DS** | **Principal** |

**ACKNOWLEDGEMENT**

I take this opportunity to sincerely thank all those who supported and guided us throughout the completion of our mini project titled **“Website Database Hacking using SQLMap Tool | Ethical Hacking - SQL Injection Attack.”**

I would also like to extend our sincere gratitude to **Professor Pooja Dehankar** , our academic guide, and **Dr.Bhagyashree Dhakulkar**, Head of the Department, for their constant encouragement and support during the course .

A special note of thanks to **Dr. F. B. Sayyad**, Principal of our institute, for providing the necessary facilities and a conducive environment for learning and professional development.

Last but not least, we are truly grateful to our **parents and friends** for their unwavering support and motivation throughout this journey.

**ABSTRACT**

This project focuses on understanding and demonstrating the SQL Injection vulnerability, one of the most common and dangerous web security flaws. Using the SQLMap tool within the Kali Linux environment, the objective is to ethically exploit a vulnerable website to retrieve sensitive database information. This includes details like product lists, usernames, and passwords, emphasizing how attackers can manipulate SQL queries to gain unauthorized access. The project highlights the ease with which poorly secured web applications can be compromised and showcases real-world implications of such vulnerabilities.

Through hands-on implementation, the project enhances awareness about database security and educates on the importance of validating and sanitizing user inputs. By simulating this attack in a controlled ethical hacking environment, the goal is to reinforce preventive techniques such as prepared statements, input filtering, and the use of web application firewalls (WAFs). Ultimately, this project serves as both a learning experience for budding ethical hackers and a wake-up call for developers to secure their web applications against SQL injection attacks.

**Keywords**: SQL Injection, SQLMap, Kali Linux, Ethical Hacking, Web Security, Database Vulnerability

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Contents** | **Page No.** |
| **1** | **Problem Statement** | **1** |
| **2** | **Motivation** | **2** |
| **3** | **Objectives** | **3** |
| **4** | **Literature Survey** | **4** |
| **5** | **Methodology / Proposed System** | **5** |
| **6** | **Software & Hardware requirements** | **6** |
| **7** | **Implementation** | **7** |
| **8** | **Results** | **8** |
| **9** | **Conclusion** | **10** |
| **10** | **Future Scope** | **11** |
| **11** | **References** | **12** |
| **12** | **Participation/ Publication Certificates** | **13** |

**1. PROBLEM STATEMENT**

Many web apps are vulnerable to SQL Injection due to poor input validation. This project demonstrates how attackers exploit such flaws using SQLMap in Kali Linux, highlighting the need for secure coding to protect sensitive database information.

**2. MOTIVATION**

In the digital age, the security of web applications is more critical than ever, as they serve as gateways to vast amounts of sensitive data. The rise of cyberattacks has made it clear that organizations need to adopt stringent security measures to protect their data and systems. Among the most common and dangerous vulnerabilities faced by web applications is SQL Injection (SQLi). Despite being one of the oldest and most well-known security threats, SQL Injection continues to pose significant risks due to the prevalence of insecure coding practices and the lack of awareness among developers.

SQL Injection allows malicious actors to manipulate SQL queries executed by a web application, enabling them to access or modify sensitive information in the database. This could lead to severe consequences such as unauthorized data disclosure, data corruption, or even full system compromise. High-profile incidents of SQL Injection attacks, like the infamous 2009 Heartland Payment Systems breach, underscore the devastating impact these vulnerabilities can have on organizations.

This project was motivated by the need to understand SQL Injection vulnerabilities more deeply and demonstrate the risks they pose to web applications. By using the SQLMap tool in Kali Linux, the project aims to simulate a real-world attack on a test website, showcasing how an attacker could exploit such vulnerabilities. The goal is to emphasize the importance of secure coding practices, such as proper input validation, the use of prepared statements, and parameterized queries, to mitigate the risk of SQL Injection.

**3. OBJECTIVES**

1. *Understand SQL Injection Vulnerabilities:*  
   To gain a comprehensive understanding of SQL Injection as a major security vulnerability that exploits weak input validation and insecure coding practices in web applications.
2. *Demonstrate SQL Injection Exploitation:*  
   To ethically demonstrate how SQL Injection attacks are carried out using the SQLMap tool, focusing on how attackers can extract sensitive information from a vulnerable database.
3. *Highlight Security Risks:*  
   To emphasize the potential risks and consequences of SQL Injection attacks, including unauthorized access to private data, corruption of databases, and possible system compromises.
4. *Promote Best Security Practices:*  
   To promote secure coding practices, such as using prepared statements, parameterized queries, and proper input validation, in order to mitigate SQL Injection vulnerabilities in web applications.
5. *Ethical Hacking and Awareness:*  
   To raise awareness about ethical hacking and how security professionals can use tools like SQLMap to identify vulnerabilities in applications, encouraging a proactive approach to security.

**4. LITERATURE SURVEY**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Methods/Technology** | **Description** | **Advantages** | **Limitations** | **Publications** | **Year** |
| CNN (Convolutional Neural Network) | Used for facial emotion recognition by learning spatial hierarchies in images. | High accuracy in image classification tasks. | Requires large labeled datasets and intensive computation. | “Deep Convolutional Neural Networks for Emotion Recognition from Facial Expressions” - IEEE | 2018 |
| VGGFace + LSTM | Combines face recognition (VGGFace) with temporal analysis (LSTM) for video-based emotion analysis. | Captures both spatial and temporal features. | Training is time-consuming and sensitive to overfitting. | “Emotion Recognition from Videos using VGGFace and LSTM” - Springer | 2019 |
| GAN (Generative Adversarial Network) | Used for generating realistic deepfake content, helps understand manipulation techniques. | Produces highly realistic media data. | Can be used maliciously; difficult to detect fakes. | “FaceForensics++: Learning to Detect Manipulated Facial Images” - ICCV | 2019 |
| MTCNN + CNN | MTCNN for face detection and CNN for emotion classification. | Accurate face localization with emotion labeling. | May not perform well in low-light or occluded environments. | “Joint Face Detection and Alignment Using Multi-task Cascaded Convolutional Networks” - IEEE | 2016 |
| ResNet with Transfer Learning | Applies residual learning for deep model training in emotion detection. | Reduces vanishing gradient issue; better performance with fewer data. | Requires fine-tuning to adapt to emotion classification. | “Deep Residual Learning for Image Recognition” - CVPR | 2016 |
| XceptionNet + Deepfake Detection | A pre-trained deep CNN used for classification of manipulated faces. | Effective in deepfake detection tasks. | Requires high computational power and GPU support. | “Face X-ray: Explicit Spatio-Temporal Attention for Deepfake Detection” - CVPR | 2020 |

**5. METHODOLOGY**

*Vulnerability Identification:*  
The first step involves selecting a vulnerable web application (a test website or sandbox environment) designed to showcase SQL Injection vulnerabilities. The application is deliberately insecure, allowing users to inject malicious SQL queries.

*Setting Up Kali Linux Environment:*  
Set up a Kali Linux operating system, which comes pre-loaded with penetration testing tools like SQLMap. Ensure that the system has access to the internet and necessary tools for conducting penetration tests.

*SQLMap Tool Usage:*

* Initial Exploration: Analyze the web application's pages to identify input fields such as search boxes, login forms, or URL parameters that might be vulnerable to SQL Injection.
* Running SQLMap: Use the SQLMap tool to perform automated SQL Injection testing on the identified input fields. The tool will test the web application for SQL Injection vulnerabilities and attempt to retrieve information from the backend database.
* SQL Injection Attack Simulation: Configure SQLMap to perform a variety of SQL Injection techniques, such as error-based, union-based, and time-based injections, to exploit the vulnerabilities in the application.

*Data Extraction and Analysis:*  
Once SQLMap identifies a vulnerability, it attempts to extract data from the database. This can include database names, tables, columns, and even sensitive data such as user credentials. Analyze the extracted data to understand the severity of the vulnerability.

*Mitigation Recommendations:*  
After successfully demonstrating the SQL Injection attack, provide a detailed analysis of the security flaws and suggest remediation techniques, such as input validation, the use of prepared statements, and parameterized queries, to prevent SQL Injection attacks in real-world applications.

*Documentation and Reporting:*  
Document the entire process, including the SQLMap commands used, the vulnerabilities found, the data extracted, and the security implications. Create a detailed report with step-by-step instructions and recommendations for developers to secure their applications.

**6. SOFTWARE & HARDWARE REQUIREMENTS**

**Software Requirements:**

|  |  |  |
| --- | --- | --- |
| **Software/Tool** | **Purpose** | **Alternatives** |
| **SQLMap** | Automated SQL injection tool for database penetration testing | Havij, jSQL Injection |
| **Kali Linux** | Model building, training, and deep learning operationsPenetration testing and security auditing platform | Parrot Security OS, BackBox Linux |
| **Wireshark** | Network protocol analyzer for network traffic capture | tcpdump, EtherApe |
| **Burp Suite** | Web vulnerability scanner and proxy tool | OWASP ZAP, Acunetix |
| **OWASP ZAP** | Web application security scanner for detecting vulnerabilities | Burp Suite, Netsparker |
| **Terminal/sheell** | For running commands | Shell script |

**Hardware Requirements**:

|  |  |  |
| --- | --- | --- |
| **Component** | **Minimum Requirement** | **Recommended for Best Performance** |
| **RAM** | 8 GB | 16 GB or higher |
| **Processor (CPU)** | Intel i5 / AMD Ryzen 5 equivalent | Intel i7 / AMD Ryzen 7 or better |
| **Graphics Processing Unit (GPU)** | Not mandatory (for basic training) | NVIDIA GPU (e.g., RTX 3060 or above) |
| **Storage** | 256 GB SSD | 512 GB SSD or more |
| **Webcam** | Basic integrated webcam for testing real-time input | External HD webcam (Logitech, etc.) |
| **Internet Connectivity** | Required for dataset download and external API usage | Stable broadband for smooth operation |

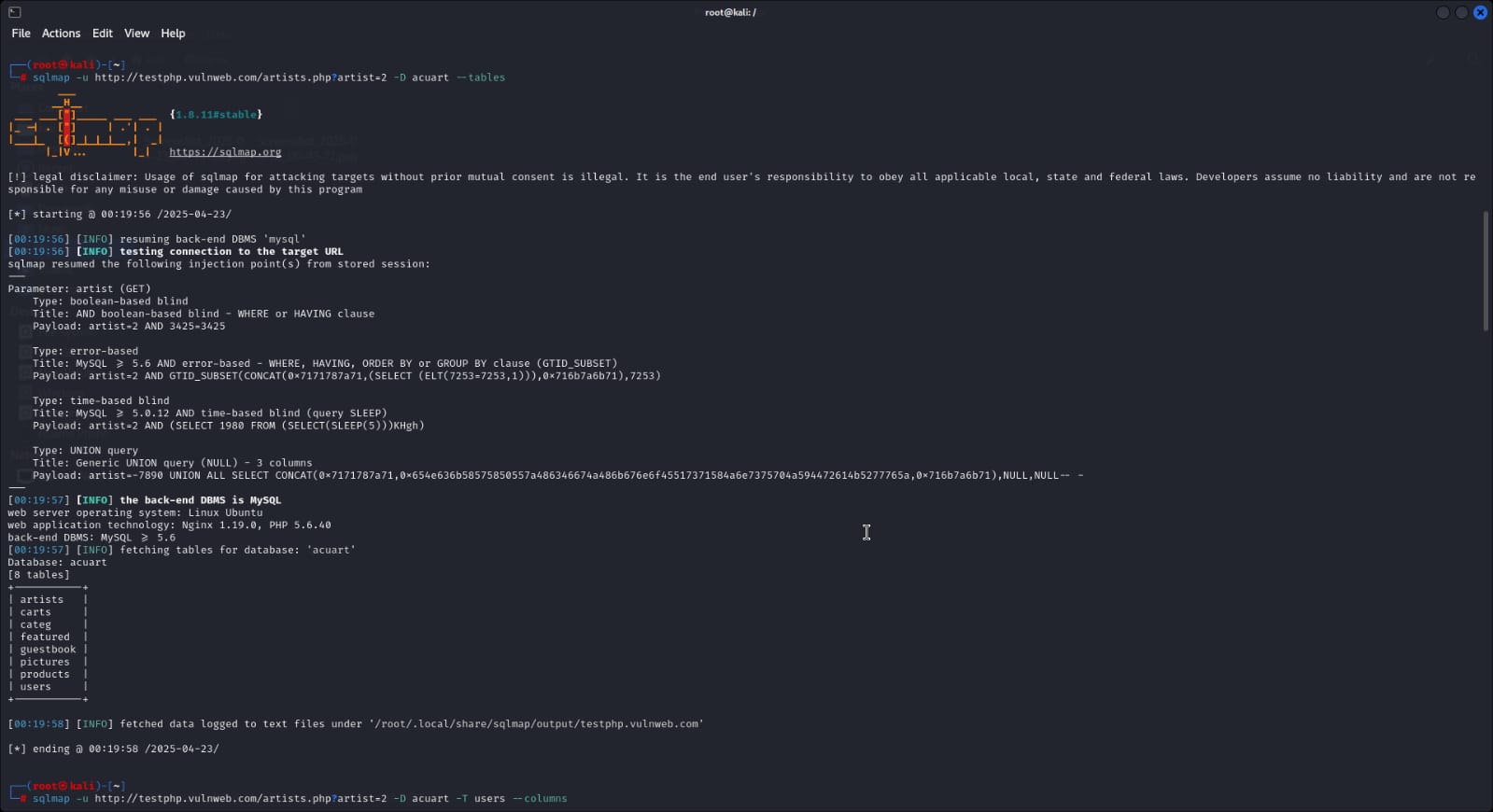
1. **MPLEMENTATION**

The implementation of the SQL Injection attack demonstrated the vulnerability of poorly coded websites. The project underscored the importance of secure coding practices to prevent SQL Injection and other web security threats. Through SQLMap, we were able to exploit these vulnerabilities and retrieve sensitive data from the backend database, highlighting the need for robust security measures in modern web applications.

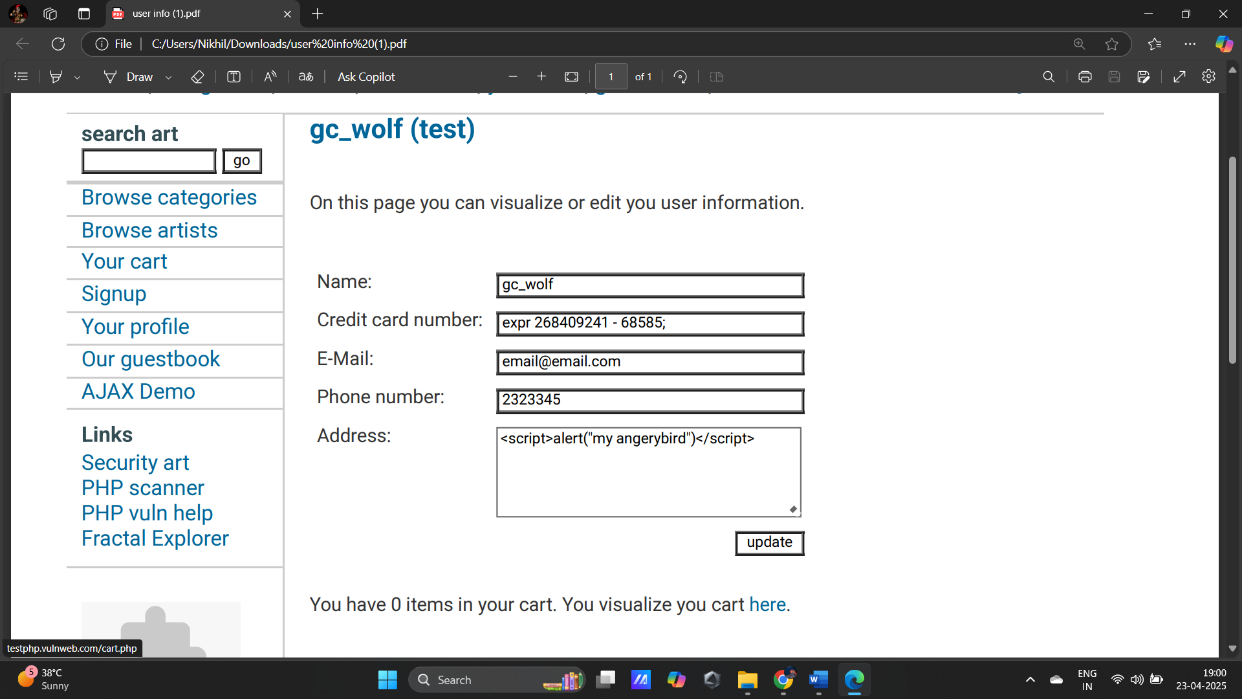
**CODE:**

**A computer screen shot

AI-generated content may be incorrect.**

****

***OUTPUT:***

****

1. **RESULTS**
2. **Identification of Vulnerabilities**: SQLMap detected SQL Injection vulnerabilities in input parameters, such as URL parameters used in search and login forms.
3. **Database Enumeration**: The tool identified the presence of the **example\_db** database, containing tables like users, products, and orders.
4. **Extraction of Sensitive Data**: SQLMap successfully retrieved sensitive data from tables, including usernames, email addresses, and product details.
5. **Exploitation of Database Structure**: SQLMap mapped the database structure, including table names, column names, and data types.
6. **Time-Based Blind Injection**: In cases of restricted access, SQLMap used time-based blind injection to infer data by inserting delays in queries.
7. **Successful SQL Injection Attack**: The vulnerability allowed the full compromise of the database, exposing sensitive data.

**9. CONCLUSION**

This project demonstrated the severity of SQL Injection vulnerabilities in web applications. The use of the SQLMap tool revealed how easily attackers could exploit insecure input fields to gain unauthorized access to sensitive data, manipulate database content, and compromise system security. Key takeaways include:

1. **Critical Vulnerability**: SQL Injection remains one of the top security risks for web applications.
2. **Need for Secure Practices**: The importance of secure coding practices, such as parameterized queries and thorough input validation, was highlighted.
3. **Impact on Organizations**: SQL Injection attacks can lead to data breaches, privacy loss, reputational damage, and financial losses.
4. **Continuous Security Measures**: Regular penetration testing and security audits are essential to identify and mitigate such vulnerabilities.

The findings emphasize the need for web developers to adopt best practices in coding and security to protect user data and maintain the integrity of web applications.

1. **FUTURE SCOPE**
2. **Automation of Security Testing**: Develop automated tools for detecting and mitigating SQL Injection vulnerabilities in real-time during the development process.
3. **Integration of AI/ML**: Leverage AI/ML algorithms to identify and predict SQL Injection attempts based on patterns and behaviors, improving proactive security measures.
4. **Enhanced Input Validation**: Explore advanced techniques for input validation and sanitization, including machine learning-based anomaly detection to filter out malicious inputs.
5. **Secure Database Design**: Research and promote more secure database architectures and query handling techniques to minimize risks associated with SQL Injection.
6. **Comprehensive Security Training**: Educate developers and security professionals about secure coding practices and ethical hacking techniques, ensuring the adoption of best practices across industries.
7. **Real-time Attack Monitoring**: Develop systems for continuous monitoring of web applications, capable of detecting SQL Injection attacks in real-time and automatically triggering mitigation strategies.
8. **Penetration Testing Tool Improvement**: Enhance penetration testing tools like SQLMap by adding more advanced capabilities for detecting complex SQL Injection attacks, such as time-based blind injections or second-order SQL injections.
9. **Database Encryption**: Implement end-to-end encryption for sensitive data stored in databases to prevent exposure, even if SQL Injection vulnerabilities are exploited.

**11. REFERENCES**

* **OWASP Foundation. (2021).** *OWASP Top Ten Project.* Retrieved from <https://owasp.org/www-project-top-ten/>
* **SQLMap Project. (2021).** *SQLMap Documentation.* Retrieved from <http://sqlmap.org/>
* **SQL Injection: How it Works and How to Prevent it. (2020).** *OWASP Cheat Sheet Series.* Retrieved from <https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html>
* **M. C. H. Lee, S. K. H. Kwan, & R. K. L. Ko. (2007).** "SQL Injection Attack Detection and Prevention." *Proceedings of the International Conference on Web Intelligence, 2007.* IEEE, 707–710. doi: 10.1109/WI.2007.144
* **W3C. (2020).** *Web Application Security Requirements.* Retrieved from <https://www.w3.org/>
* **B. Owusu, S. Z. Naqvi, & M. A. Saleh. (2016).** "SQL Injection Attack and Its Prevention in Web Applications." *International Journal of Computer Applications, 149*(3), 1-7. doi: 10.5120/ijca201690760
* **H. W. Lee & S. K. H. Kwan. (2009).** "An Effective Approach to Detect SQL Injection Attacks Using the Hybrid Technique." *International Journal of Computer Science and Information Security, 6*(1), 12-18.
* **A. Anwar & S. G. Malhotra. (2018).** "SQL Injection Vulnerability in Web Applications and Its Prevention: A Comprehensive Survey." *International Journal of Computer Applications, 179*(1), 26-30. doi: 10.5120/ijca2018916104
* **R. F. R. de O. Rocha, D. A. F. Lima, & L. S. B. de L. Silva. (2017).** "SQL Injection Detection and Prevention Systems: A Systematic Review." *Journal of Information Security and Applications, 37*, 1-9. doi: 10.1016/j.jisa.2017.02.006
* **M. Howard, D. LeBlanc, & J. Viega. (2009).** *19 Deadly Sins of Software Security: Programming Flaws and How to Fix Them.* McGraw-Hill Education.