IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, April 2026
Impact Factor: 8.2

Real-Time Security System for Detecting and Preventing Brute Force and SQL Injection Attacks

J Jeba Stanly[1], V.Dharshini[2], S. Jansivijila[3], M. Kaviya[4], M. Prakashini[5]
Assistant Professor, Dept. of CSE[1]
Students, Dept. of CSE[2-5]
N.S.N. College of Engineering and Technology, Karur, India

Abstract: *In today's digital world, web applications are increasingly vulnerable to cyber attacks such as brute force and SQL injection. These attacks pose serious threats by enabling unauthorized access, data breaches, and loss of sensitive information. To address these challenges, this project presents a real-time security system designed to detect and prevent brute force and SQL injection attacks effectively. The system is developed using Flask and integrates multiple security mechanisms to continuously monitor and protect the authentication process. It analyzes user inputs to identify suspicious patterns and prevents malicious SQL queries from being executed, thereby ensuring secure interaction with the database.*

In addition, the system monitors login activities by tracking failed login attempts from each IP address to detect brute force attacks. When the number of failed attempts exceeds a predefined threshold, the system automatically blocks the attacker's IP address to prevent further access. It also maintains detailed log files that record attack information such as IP address, attack type, and timestamp, enabling effective monitoring and analysis of security incidents. By combining real-time detection, automated prevention, and logging mechanisms, the proposed system enhances overall application security and safeguards user data against common cyber threats..

Keywords: *Brute Force Attack, SQL Injection, Web Application Security, Cyber Security, Intrusion Detection, Flask Framework, MySQL Database, Authentication Security, Input Validation, IP Blocking, Real-Time Monitoring*

I. INTRODUCTION

In today's digital era, web applications are widely used across various domains such as education, banking, healthcare, and e-commerce. With the rapid growth of online platforms, the risk of cyber threats has also increased significantly. Among these threats, brute force attacks and SQL injection attacks are considered some of the most common and dangerous security vulnerabilities. These attacks can lead to unauthorized access, data leakage, and severe damage to system integrity.

Traditional authentication systems primarily rely on username and password mechanisms, which are often vulnerable to repeated login attempts and malicious input injections. Brute force attacks involve systematically trying multiple password combinations until the correct one is found, while SQL injection attacks exploit input fields to manipulate database queries and gain unauthorized access.

To overcome these challenges, this project proposes a **Real-Time Security System for Detecting and Preventing Brute Force and SQL Injection Attacks**. The system is designed using Flask and integrates multiple security layers such as input validation, attack detection, IP blocking, logging, and email alert mechanisms.

The proposed system continuously monitors user activity and analyzes input data to detect suspicious behavior. It identifies SQL injection attempts using pattern matching techniques and prevents malicious queries from being executed. Additionally, it tracks failed login attempts to detect brute force attacks and blocks the attacker's IP address once a threshold is exceeded.

DOI: 10.48175/IJARSCT-33446

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 10, April 2026 Impact Factor: 8.2

This system provides a lightweight, efficient, and scalable solution for securing web applications without requiring complex hardware or high computational resources. By combining real-time monitoring with automated response mechanisms, the system enhances application security and ensures a safe user authentication environment.

II. LITERATURE SURVEY / RELATED WORK

Web application security has been an active area of research due to the increasing number of cyber attacks targeting online systems. Various techniques have been proposed to detect and prevent brute force and SQL injection attacks.

In [1], researchers analyzed different types of SQL injection attacks and proposed prevention techniques using input validation and query parameterization. While effective, the system required strict implementation and developer awareness. Similarly, [2] discussed authentication vulnerabilities and suggested secure login mechanisms, but it lacked real-time attack detection capabilities.

A brute force detection model was introduced in [3], where login attempts were monitored and restricted after a certain threshold. Although this method reduces attack attempts, it does not provide alert mechanisms or logging features. In [4], an intrusion detection system (IDS) was proposed to monitor network traffic and detect anomalies, but it required complex configurations and high computational resources.

The use of OWASP security guidelines was highlighted in [5], which provided best practices for preventing web vulnerabilities such as SQL injection and cross-site scripting. However, these guidelines are theoretical and require practical implementation.

Recent studies such as [6] focused on combining multiple security techniques, including logging and alert systems, to enhance application security. These systems showed improved performance but lacked simplicity and ease of deployment.

The proposed system improves upon existing approaches by integrating:

- Real-time SQL injection detection using pattern matching
- Brute force detection using login attempt tracking
- Automatic IP blocking mechanism
- Email alert system for administrator notification
- Lightweight and easy deployment using Flask

This makes the system more practical, efficient, and suitable for real-world applications.

III. METHODOLOGY / SYSTEM DESIGN

The proposed system is designed to detect and prevent brute force and injection attacks in real time by monitoring user login activities and analyzing input data.

A. System Architecture

The system follows a modular architecture consisting of the following components:

1. **User Interface (Login Page)**
   Users enter their credentials through a web-based login form.
2. **Request Handling (Flask Server)**
   The Flask server processes incoming requests and manages application logic.
3. **IP Address Identification**
   The system retrieves the user's IP address to track activity.
4. **Security Layer**
   This is the core module responsible for:
   - Injection detection using pattern matching
   - Brute force detection using attempt counting

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, April 2026
Impact Factor: 8.2

5. **Database Layer (MongoDB)**
   Stores user credentials and verifies login details.
6. **Logging System**
   Records attack details such as IP address, attack type, and timestamp.
7. **Alert System**
   Sends email notifications to the administrator when an attack is detected.

B. Working Methodology

The system works as follows:

- The user submits login credentials through the interface
- The system checks if the IP is already blocked
- Input fields are analyzed for injection patterns
- If malicious input is detected, the IP is blocked
- If input is valid, credentials are verified in the database
- If login fails repeatedly, brute force detection is triggered
- The system blocks the IP after exceeding attempt limits
- Attack details are logged and email alerts are sent

B. Tools and Technologies Used

Technology	Purpose
Python	Core programming language
Flask	Web framework
MongoDB	Database
PyMongo	Database connectivity
HTML/CSS	Frontend design
JavaScript	Client-side interaction

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, April 2026
Impact Factor: 8.2

C. Workflow Diagram

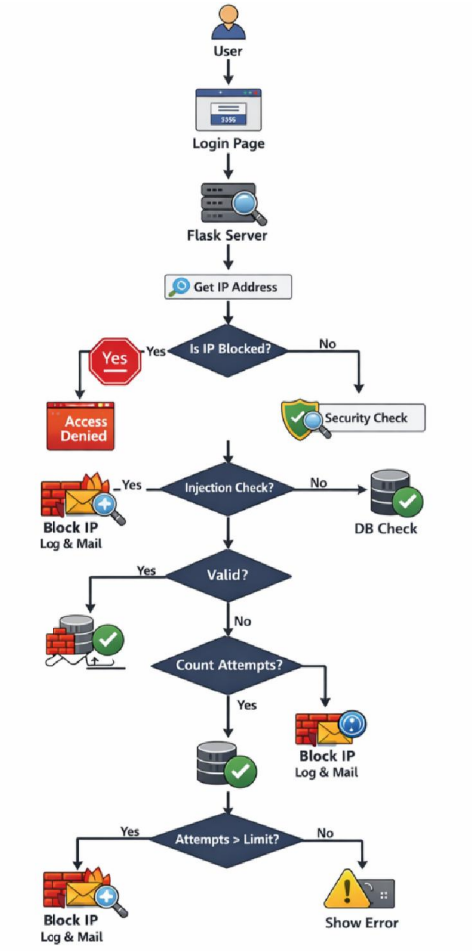

Fig 1 Flow Diagram

IV. Implementation

The implementation of the proposed system includes frontend, backend, security modules, and database integration.

A. User Interface

A simple login interface is developed using HTML and CSS. It allows users to enter their username and password securely. The interface is designed to be user-friendly and responsive.

B. Backend Processing

The backend is implemented using Flask, which handles user requests and processes login authentication. It integrates security checks before allowing access to the system.

C. Security Modules

The system includes two main security modules:

- **Injection Detection Module**

Analyzes user inputs using predefined patterns to identify malicious content such as SQL keywords or special characters.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, April 2026
Impact Factor: 8.2

• **Brute Force Detection Module**

Tracks failed login attempts based on IP address. If attempts exceed a threshold, the system blocks the IP.

D. Database Integration

MongoDB is used to store user credentials. The system queries the database to verify login details. Secure querying methods are used to prevent injection attacks.

E. Logging and Alert System

All detected attacks are recorded in log files with details such as IP address, attack type, and timestamp. Additionally, an email alert system is implemented using SMTP to notify the administrator in real time.

F. System Execution

The entire system runs in real time, ensuring immediate detection and response to threats. The integration of multiple modules ensures smooth operation and enhanced security.

G. Screenshot Example

Fig 2 Server Running

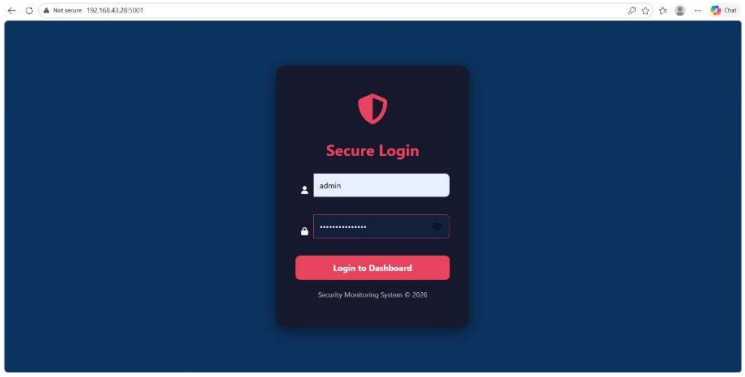

Fig 3 Login Page

Fig 4 Home Page

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 10, April 2026
Impact Factor: 8.2

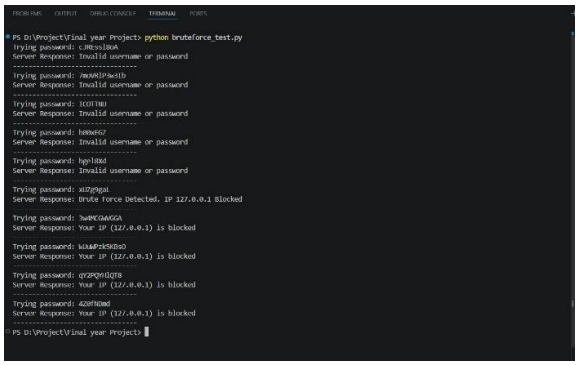

Fig 5 Brute Force Attack

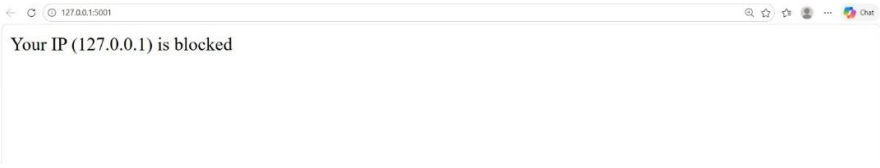

Fig 6 IP Blocked

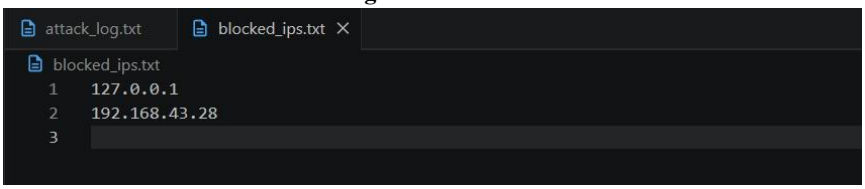

Fig 7 IP Blocked List

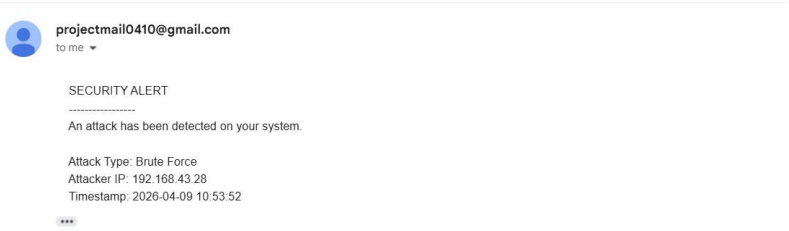

Fig 8 Received Mail After Attack

V. RESULTS AND DISCUSSION

The proposed real-time security system was successfully implemented and tested in a web-based environment using Flask and MongoDB. The system was evaluated based on its ability to detect and prevent brute force and injection attacks during the login process. Various test cases were executed by simulating malicious inputs and repeated login attempts.

The system effectively detected SQL injection attempts by analyzing user inputs for suspicious patterns such as special characters, logical operators, and query manipulation keywords. Upon detection, the system immediately blocked the attacker's IP address, logged the attack details, and generated an email alert to notify the administrator. Similarly, brute force attacks were simulated by repeatedly submitting incorrect login credentials. The system tracked these failed attempts and successfully identified abnormal behavior once the threshold limit was exceeded. The attacker's IP address was then blocked automatically, preventing further access.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 10, April 2026 Impact Factor: 8.2

The response time of the system was observed to be minimal, ensuring real-time detection and prevention without affecting the normal user experience. The logging mechanism accurately recorded all attack events, including IP address, attack type, and timestamp, which is useful for further analysis and monitoring. The email alert system also functioned effectively, providing instant notifications for security breaches.

Overall, the results demonstrate that the system is capable of providing efficient and reliable protection against common web-based attacks. The implementation is lightweight, cost-effective, and suitable for deployment in real-world applications such as educational portals, administrative systems, and small-scale web services. The system's performance validates its effectiveness in enhancing application security and preventing unauthorized access.

VI. CONCLUSION

This project presents a robust and practical solution for improving web application security through real-time detection and prevention of brute force and injection attacks. By integrating multiple security mechanisms such as input validation, attack detection, IP blocking, logging, and email alert systems, the proposed solution ensures a secure authentication process.

The system successfully demonstrates how common vulnerabilities in login systems can be mitigated using simple yet effective techniques. It enhances the reliability of web applications by preventing unauthorized access and protecting sensitive user data. The use of Flask and MongoDB makes the system lightweight, scalable, and easy to deploy in different environments.

Additionally, the system improves administrator awareness through real-time alerts and detailed logging, enabling quick response to potential threats. The approach followed in this project emphasizes proactive security measures rather than reactive solutions.

In conclusion, the developed system serves as a strong foundation for building secure web applications and highlights the importance of integrating security mechanisms at the application level. It contributes to creating a safer digital environment and demonstrates the practical implementation of cybersecurity concepts.

VII. ACKNOWLEDGMENT

The successful completion of this project would not have been possible without the valuable support and guidance of several individuals. I would like to express my sincere gratitude to my project guide and faculty members of the Department of Computer Science and Engineering for their continuous encouragement, technical guidance, and constructive feedback throughout the development of this project.

I also extend my appreciation to my friends and colleagues who contributed their ideas and assisted in testing the system under various scenarios. Their support played a significant role in improving the functionality and performance of the project.

Finally, I would like to acknowledge the open-source communities and developers behind technologies such as Flask, MongoDB, and Python, whose tools and documentation greatly contributed to the successful implementation of this project.

REFERENCES

[1] W. Stallings, *Cryptography and Network Security: Principles and Practice*, Pearson, 2017.

[2] OWASP Foundation, "OWASP Top 10 Web Application Security Risks," 2023.

[3] W. G. Halfond, J. Viegas, and A. Orso, "A Classification of SQL Injection Attacks and Countermeasures," IEEE, 2006.

[4] J. Clarke, *SQL Injection Attacks and Defense*, Syngress, 2012.

[5] J. Viega and G. McGraw, *Building Secure Software*, Addison-Wesley, 2002.