iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** VI **Month of publication:** June 2026

DOI: https://doi.org/10.22214/ijraset.2026.83477

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VI June 2026- Available at www.ijraset.com

TyphoCrypt: A Django-Based Simulation Platform for Demonstrating and Defending Against Salt Typhoon Credential Attacks

Amrithesh S[1], Magna Y[2]
Department of Cyber Security, Dr. MGR Educational and Research Institute

Abstract: ***Protecting user credentials from increasingly sophisticated cyber threats has become a major challenge for organizations and online platforms in today's rapidly evolving digital environment. Among the emerging threats, the Salt Typhoon attack has gained considerable attention due to its ability to exploit weaknesses in password storage and database security mechanisms. This advanced password-cracking technique takes advantage of predictable salting practices, outdated hashing algorithms, weak access controls, poor password policies, and SQL injection vulnerabilities to gain unauthorized access to sensitive authentication data. To address these concerns, this paper presents TyphoCrypt, a web-based simulation platform developed using Django that demonstrates both offensive and defensive cybersecurity techniques through realistic attack scenarios. The platform simulates SQL injection and brute-force attacks targeting salted password hashes while incorporating modern security measures such as Argon2 password hashing, dynamic salting, peppering, and AES encryption. By providing both secure and insecure authentication environments, TyphoCrypt enables users to understand the practical impact of different security approaches and highlights the importance of implementing robust credential protection mechanisms. The platform serves as a bridge between theoretical cybersecurity concepts and their real-world application, offering a hands-on learning experience for students, researchers, and security professionals.***

Index Terms: ***Salt Typhoon Attack, Password Security, Python-Django Authentication, SQL Injection Simulation, Brute Force Attack, Argon2 Hashing, AES Encryption, Cybersecurity Education, Credential Protection, Web Application Security.***

I. INTRODUCTION

In today's digital world, usernames and passwords remain the most widely used method of authentication for web applications. These systems play a critical role in protecting sensitive user information and controlling access to online services. However, despite growing awareness of cybersecurity risks, many authentication systems continue to suffer from poor design and implementation practices. As a result, they are often vulnerable to attacks such as SQL injection, brute-force password guessing, credential stuffing, and automated bot activity. Common security weaknesses include inadequate input validation, the absence of human verification mechanisms, insecure password storage methods, and limited monitoring of login activities.

Among the more advanced threats facing modern authentication systems is the Salt Typhoon attack, a sophisticated password-cracking technique that targets databases containing salted and encrypted passwords. This attack exploits weaknesses such as predictable salting methods, outdated hashing algorithms, and insufficient database access controls. By taking advantage of these vulnerabilities, attackers can bypass security measures that would otherwise protect user credentials. The consequences can be severe, ranging from unauthorized access to individual accounts to large-scale data breaches that result in financial losses and damage to an organization's reputation.

A major challenge in cybersecurity education is that many students and developers have limited exposure to real-world authentication failures. Traditional learning approaches often focus heavily on theoretical concepts while providing few opportunities for practical experimentation. As a result, learners may understand security principles in theory but struggle to recognize vulnerabilities and their impact in actual applications. This gap in practical knowledge can lead to the deployment of insecure authentication systems in production environments.

To address this issue, this paper presents TyphoCrypt, a web-based simulation platform designed to provide hands-on experience with both secure and insecure authentication systems. The platform consists of four interactive modules. The first module demonstrates a secure authentication system that employs dynamic salting and key derivation techniques during user registration and login.

©IJRASET: All Rights are Reserved

The second module simulates attacker activities, illustrating how database compromises and password-cracking attacks can occur when security best practices are ignored. The third module provides an administrative dashboard for monitoring brute-force attacks, analyzing attack effectiveness, and managing security configurations. The fourth module visually demonstrates how modern cryptographic techniques can protect sensitive user data, even in situations where portions of that data have been exposed.

TyphoCrypt is developed using Python, Django, HTML, CSS, and JavaScript for the application layer, with MySQL or SQLite serving as the database backend. By combining realistic attack simulations with modern defensive techniques, the platform functions as both an educational resource and a practical engineering tool for learning, testing, and understanding secure authentication systems.

II. PROBLEM DEFINITION AND MOTIVATION

One of the major challenges in cybersecurity education is the lack of practical and ethical environments where learners can explore both secure authentication practices and the risks associated with poorly designed login systems. While many educational resources explain security concepts and vulnerabilities, they often do so in a theoretical manner, without demonstrating how these weaknesses can be exploited in real-world scenarios. As a result, students and novice developers may struggle to fully understand the importance of implementing effective security controls.

TyphoCrypt was developed to address this gap by providing an interactive platform that allows users to examine both secure and vulnerable authentication systems within a controlled environment. Rather than simply studying security principles, learners can observe how authentication mechanisms behave under different conditions and gain a deeper understanding of the factors that contribute to system security.

The need for such a platform is increasingly important as web-based services continue to expand across industries. Secure authentication has become a fundamental requirement for protecting user data, maintaining trust, and ensuring the reliability of digital services. However, many organizations—particularly small businesses, startups, and educational institutions—often lack dedicated cybersecurity expertise and are therefore more vulnerable to credential-based attacks. Even minor implementation mistakes can create opportunities for attackers to compromise sensitive information. TyphoCrypt is motivated by two primary goals. First, it aims to help future developers build practical security awareness by exposing them to realistic attack and defense scenarios. Second, it seeks to demonstrate how seemingly small design flaws can lead to serious security breaches when appropriate safeguards are not in place. To achieve these objectives, the platform combines a securely designed authentication system with intentionally vulnerable simulation environments. Users can observe how SQL injection attacks bypass authentication mechanisms when input validation is inadequate and how brute-force attacks become effective when protections such as rate limiting and account lockout policies are missing. By directly experiencing both successful attacks and effective defenses, learners gain a more meaningful understanding of authentication security. This hands-on approach transforms cybersecurity education from passive learning into an engaging experience that emphasizes the real-world consequences of security decisions.

III. PROJECT OBJECTIVES

A. *Primary Objectives*

The primary goal of TyphoCrypt is to develop a secure and reliable authentication system using Python and Django that follows established cybersecurity best practices. The system is designed to incorporate multiple layers of protection, including secure password hashing through algorithms such as Argon2id or PBKDF2, CAPTCHA-based human verification, account lockout mechanisms after repeated failed login attempts, IP-based access restrictions, and detailed logging of authentication activities. Together, these security measures help protect user accounts from common threats while providing clear demonstrations of how modern authentication defenses operate in practice. Another key objective is to create a controlled and ethical learning environment where users can directly observe the effectiveness of these security mechanisms. By allowing learners to compare secure and insecure authentication implementations, the platform highlights the importance of proper security design and demonstrates how different protection strategies can prevent real-world attacks. This contrast forms a central component of the educational value offered by TyphoCrypt.

B. *Secondary Objectives*

A secondary objective of the project is to simulate intentionally vulnerable authentication environments that omit essential security controls. These environments provide a safe setting for experimenting with attack techniques such as SQL injection and brute-force password guessing without posing any risk to real systems, users, or sensitive data. All simulations are conducted within clearly defined ethical boundaries and are intended solely for educational and research purposes.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VI June 2026- Available at www.ijraset.com

In addition, TyphoCrypt seeks to improve cybersecurity awareness by offering users immediate feedback on attack outcomes and security responses. Following each simulation, the platform provides summaries, explanations, and educational resources that help users understand the underlying causes of vulnerabilities, the methods attackers use to exploit them, and the security practices required to prevent similar weaknesses. Through this approach, the platform encourages a deeper understanding of both offensive and defensive aspects of authentication security.

IV. TECHNOLOGY STACK

TyphoCrypt integrates a carefully selected set of tools and frameworks across its application layers. Table I summarises the complete technology stack.

TABLE I
Technology Stack of TyphoCrypt

Layer	Tools / Frameworks
Frontend	HTML5, CSS3, JavaScript, Bootstrap / Tailwind CSS
Backend	Python, Django
Database	MySQL or SQLite
Encryption	AES (cryptography / pycryptodome)
Hashing	Argon2id / PBKDF2 (Django password hashers)
Authentication	Django auth framework with customizable User model
Simulation	Custom Django views and scripts for brute-force simulation
Visualization	Chart.js / Plotly for real-time simulation graphs
Security	CSRF Protection, Input Validation, Secure Query Practices

V. LITERATURE REVIEW

SQL injection continues to be one of the most common and dangerous threats in web application security, largely due to its simplicity and the widespread presence of vulnerable code in real-world applications. Clarke-Salt [1] provides an in-depth examination of how these attacks are constructed against insecurely designed login systems and highlights parameterized queries and ORM-based database interactions as the most effective countermeasures. Similarly, the OWASP Top Ten [5] identifies injection attacks and broken authentication as some of the most critical security risks facing modern web applications, recommending practices such as account lockout mechanisms, CAPTCHA verification, and secure password storage as essential baseline protections.

Stuttard and Pinto [2] further expand on authentication vulnerabilities by categorizing a broader range of weaknesses, including poor session management, predictable authentication tokens, and inadequate input validation. Their findings emphasize that authentication security cannot rely on a single safeguard; instead, it must be built using multiple overlapping layers of defense to reduce the likelihood of successful exploitation.

©IJRASET: All Rights are Reserved

The foundational principles of password security were first clearly established by Morris and Thompson [4], who demonstrated that storing plaintext passwords is fundamentally insecure and proposed the use of one-way cryptographic transformations for credential storage. This work laid the groundwork for modern password hashing techniques, which now incorporate adaptive algorithms and unique per-user salts. Contemporary frameworks such as Django implement these principles through built-in hashing systems designed to resist brute-force attacks and precomputed rainbow table exploits.

Schneier [6] and Stallings [9] provide comprehensive discussions of the cryptographic foundations that underpin secure authentication systems, including hashing functions, salting strategies, key derivation methods, and symmetric encryption techniques. These concepts collectively form the theoretical basis for the defensive mechanisms implemented in TyphoCrypt.

In addition, research on CAPTCHA and reCAPTCHA systems shows that they significantly reduce automated login attempts by distinguishing between human users and bots, making them a standard component in modern authentication pipelines.

Finally, recent studies in cybersecurity education highlight the effectiveness of simulation-based and experiential learning approaches. These studies consistently show that learners develop a deeper and more lasting understanding of security concepts when they are able to interact with realistic attack and defense scenarios rather than relying solely on theoretical instruction. TyphoCrypt is designed based on this pedagogical insight, allowing users to directly observe both successful attacks and defensive mechanisms within a controlled environment.

VI. SYSTEM DESIGN

A. Architectural Overview

TyphoCrypt is built using Django's Model–View–Template (MVT) architectural pattern. The Model layer is responsible for defining the database structure, including user accounts, login attempt logs, brute-force activity records, blocked IP addresses, and account lockout states. The View layer handles the core application logic, such as authentication processing, CAPTCHA validation, attack detection, simulation execution, event logging, and user account management. The Template layer manages the presentation of the application through HTML, CSS, and JavaScript, providing intuitive interfaces for both regular users and administrators in the form of forms and dashboards.

B. User Flow

The user journey begins with registration, where new users provide basic details such as name, email, phone number, and a chosen password. For security purposes, passwords are immediately hashed using Django's built-in make_password() function before being stored in the database.

During login, the system first verifies the CAPTCHA response to ensure the request is initiated by a human user. It then proceeds to validate the credentials. Each failed login attempt is recorded and contributes to a counter maintained by the system. If the number of failed attempts exceeds a predefined threshold, the account is temporarily locked for a configured duration. Once authentication is successful, the user is granted access to the platform's attack simulation modules.

C. Admin Flow

Administrators access the system through a dedicated and secure login interface, after which they are redirected to an administrative dashboard. This dashboard provides a comprehensive overview of system activity, including user login logs, lists of blocked IP addresses, and summaries of security-related events.

From this interface, administrators can manage user accounts by suspending or reactivating them, manually unlocking accounts that have been locked due to suspicious activity, and reviewing detailed histories of login attempts and simulation events. In addition, the system is configured to send automated email alerts whenever critical security events are detected, such as potential SQL injection attempts or repeated brute-force attacks exceeding predefined thresholds.

D. Security Design

The security architecture of TyphoCrypt is built on multiple layered protections. These include adaptive password hashing with unique per-user salts, CAPTCHA-based human verification at the authentication boundary, configurable account lockout mechanisms, IP-based blocking for repeated malicious activity, and detailed logging of authentication events including timestamps, IP addresses, and user agents.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

To enhance operational security, sensitive configuration values such as email credentials and reCAPTCHA keys are stored securely within Django's configuration settings rather than being hard-coded in the application source code. This approach reduces the risk of accidental exposure and improves maintainability.

E. Database Design

The system's data layer is implemented using Django's Object-Relational Mapping (ORM), ensuring structured and secure interaction with the underlying relational database. Separate tables are maintained for user accounts, administrator accounts, login activity logs, attack simulation records, and account lockout tracking.

This separation of data into distinct logical entities helps maintain clear boundaries between operational data and audit data. It also improves query efficiency and supports the reporting and visualization features of the administrative dashboard.

VII. SYSTEM METHODOLOGY

The development of TyphoCrypt followed a modular, security-focused methodology structured into eight distinct phases. This phased approach ensured that both functional requirements and security objectives were systematically addressed throughout the development process.

The first phase, Requirement Anal**ysis**, involved identifying common authentication threats and defining the educational goals of the platform. This phase helped establish what the system needed to demonstrate in terms of both secure authentication practices and potential vulnerabilities.

The second phase, System Design, focused on developing the overall architecture of the application. This included defining the Model–View–Template (MVT) structure, designing database schemas, and outlining system workflows that guided implementation.

The third phase involved implementing the secure authentication core, which included password hashing, CAPTCHA verification, and account lockout mechanisms. This formed the foundation of the system's defensive capabilities.

In the fourth phase, the attack simulation modules were developed. These modules intentionally replicate insecure coding practices within isolated and controlled environments, allowing users to safely observe how vulnerabilities such as SQL injection and brute-force attacks can be executed.

The fifth phase established the logging and monitoring infrastructure, enabling detailed tracking of authentication attempts, attack simulations, and system events.

The sixth phase focused on building the administrative control module, which provides tools for user management, security monitoring, and system oversight.

The seventh phase involved functional and security testing, where all modules were evaluated using controlled test cases and manual attack simulations to ensure both usability and resilience against common threats.

The final phase concentrated on educational output and refinement, ensuring that each simulation is followed by clear explanations, security insights, and references for further learning. This phase reinforced the project's primary goal of not only demonstrating attacks and defenses but also improving the user's understanding of underlying security principles.

VIII. MODULE DESCRIPTION

A. User Modules

The User Registration Module is responsible for collecting basic user information and securely storing credentials. Passwords are immediately hashed upon submission to ensure that no plaintext credentials are stored in the database.

The Secure Login and Authentication Module handles user authentication by verifying submitted credentials against stored hashed values. As an additional security layer, CAPTCHA verification is enforced before access is granted, helping to prevent automated login attempts.

The Account Lock and Recovery Module continuously monitors failed login attempts and triggers temporary account lockouts when a predefined threshold is exceeded. Users are informed about the lockout duration, and recovery is facilitated through an administrative approval process.

The SQL Injection Simulation Module provides a controlled environment where a deliberately vulnerable login form is used to demonstrate unsafe database query construction. Users can observe how SQL injection attacks succeed and are provided with explanations of the underlying vulnerability along with recommended fixes.

The Brute Force Attack Simulation Module demonstrates password guessing attacks by systematically testing user-provided password lists against a simulated credential database. This module highlights how quickly weak passwords can be compromised when no protective mechanisms are in place.

The Awareness and Learning Module concludes each simulation by presenting clear explanations, security insights, and curated recommendations for further reading. This ensures that users not only observe attack behaviour but also understand how such vulnerabilities can be prevented.

B. Admin Modules

The Admin Authentication Module provides a secure and restricted login interface for authorised administrators.

The User Management Module allows administrators to view user accounts, and perform actions such as suspension or reactivation when necessary.

The Account Unlock and Control Module enables administrators to reset lockout counters and restore access for legitimate users who may have been incorrectly or temporarily blocked.

The Logging and Monitoring Module provides searchable and structured logs of all authentication-related events, including login attempts and system actions, along with relevant metadata for analysis.

The Attack Detection and Response Module identifies suspicious activity patterns and triggers automated protective actions such as IP blocking when necessary.

The Alert and Notification Module sends automated email notifications to administrators when critical security events are detected, enabling timely review and response to potential threats.

IX. SYSTEM FLOW ALGORITHM

The TyphoCrypt system operates through a sequence of ten logical steps that govern both authentication and simulation workflows.

At startup, the application initializes its configuration settings and establishes a connection with the database. During the registration process, user inputs are first validated to ensure uniqueness of account details, after which the password is securely hashed before being stored in the database.

During login, the system follows a layered verification process. It begins by validating the CAPTCHA response to confirm human interaction. It then checks whether the account is currently locked. If the account is active, the system proceeds to verify the provided password against the stored hash. On successful authentication, the failed-attempt counter is reset and a user session is created. On failure, the system increments the failed-attempt counter and triggers an account lock if the predefined threshold is exceeded. Once authenticated, users are redirected to the dashboard, where they can access either of the simulation modules. In the SQL injection simulation module, user input is processed through a deliberately vulnerable query mechanism. When injection patterns are detected, the system simulates a successful attack outcome and provides a detailed explanation of the underlying vulnerability along with corrective measures.

In the brute-force simulation module, the system iterates through a list of user-provided password candidates until a valid match is found or the list is fully exhausted. This process demonstrates how weak passwords can be exploited when proper security controls are absent.

All authentication attempts and simulation activities are recorded in the system logs, including metadata such as IP address, timestamp, and operation outcome. The administrative workflow manages user accounts, system logs, and locked sessions, providing oversight and control over system activity.

Finally, when a user logs out, the session is securely terminated and the user is redirected back to the login page.

X. REALISTIC CONSTRAINTS

A. Legal and Ethical Constraints

TyphoCrypt is developed strictly for academic learning and cybersecurity awareness within a controlled, consent-based environment. All attack demonstrations are performed using dummy data and simulated systems, with no interaction with real users, live websites, or production databases.

The techniques illustrated in the platform are intended solely for educational purposes. Attempting to apply similar methods on real systems without explicit authorization is illegal and punishable under applicable cybersecurity laws, including the Information Technology Act (India) and comparable international regulations. To reinforce this boundary, the user interface clearly communicates that all attack scenarios are simulated and intended for learning only.

In addition, automated notifications and alerts are limited to security demonstration and monitoring purposes within the system and are not intended for external or real-world operational use.

B. *Technical Constraints*

The simulation modules for SQL injection and brute-force attacks are built using predefined datasets and controlled scenarios, which means they do not fully replicate the complexity and unpredictability of real-world attack environments. The system currently focuses on only two primary attack types, while more advanced threats such as zero-day exploits, distributed denial-of-service (DDoS) attacks, and side-channel attacks are outside the scope of the platform.

Certain technical limitations also affect system behavior. For example, Google reCAPTCHA has reduced effectiveness in localhost or development environments, which limits real-world testing accuracy. Similarly, IP-based blocking can be bypassed in real-world scenarios through VPNs or proxy rotation, which is not fully addressed in this implementation.

The brute-force simulation operates in a sequential manner, whereas real-world attacks often use parallel processing to increase speed and efficiency. Additionally, the system is designed for a single-server architecture and does not account for distributed or load-balanced deployments.

Finally, performance may degrade when handling large volumes of log data, as the platform is optimized for educational demonstration rather than high-throughput production environments.

XI. CONCLUSION AND FUTURE WORK

A. *Conclusion*

TyphoCrypt demonstrates that a single integrated platform can effectively combine secure authentication practices with a safe, controlled environment for exploring how insecure systems are exploited. By bringing together essential security mechanisms such as password hashing, CAPTCHA-based human verification, account lockout policies, detailed logging, and administrative monitoring, alongside carefully designed SQL injection and brute-force simulations, the platform delivers a comprehensive, dual-purpose learning experience.

Rather than simply describing vulnerabilities in theory, TyphoCrypt allows users to observe how weak design choices lead to real security failures within a risk-free environment. Each simulation is followed by clear explanations that break down why the attack succeeded and how it could have been prevented using proper security controls. This approach helps bridge the gap between theoretical knowledge and the practical decision-making required in real-world application development, making the learning experience more intuitive and impactful.

B. *Future Work*

Future development of TyphoCrypt will focus on expanding both the range of attack simulations and the strength of defensive mechanisms. Planned enhancements include the addition of new vulnerability modules such as Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), and session hijacking, allowing users to explore a broader spectrum of web application security threats.

On the defensive side, the authentication system will be improved with support for multi-factor authentication (MFA) and more advanced CAPTCHA implementations. The platform will also be extended with real-time threat intelligence integration and lightweight intrusion detection features to better reflect modern production security environments.

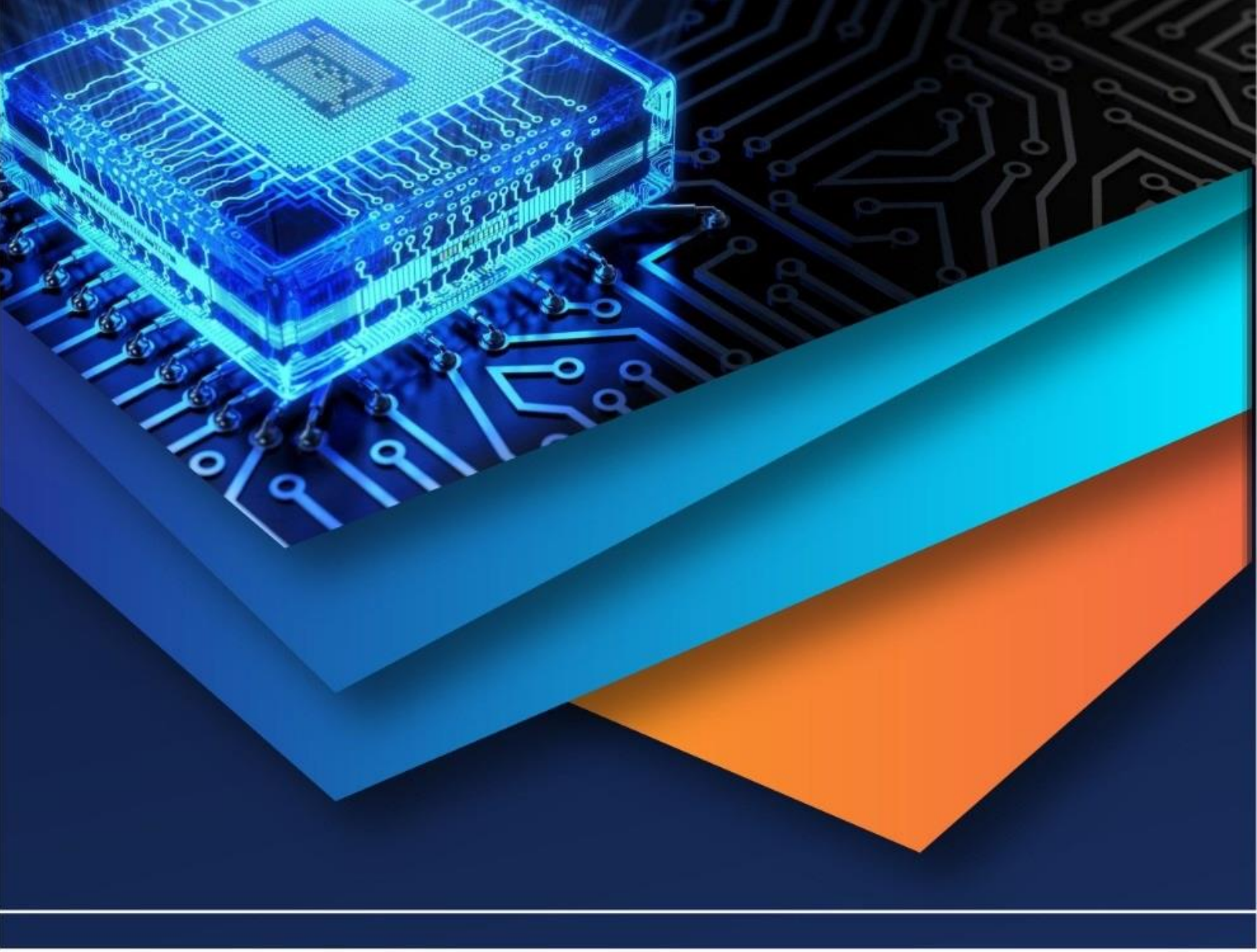
To improve realism and scalability, future versions will explore cloud deployment, enabling more accurate testing of rate limiting and distributed attack scenarios. In addition, enhanced analytics dashboards, role-based access control for administrators, and structured reporting tools will be introduced. These improvements aim to evolve TyphoCrypt into a more complete cybersecurity training platform suitable for academic instruction, practical workshops, and advanced security experimentation.

REFERENCES

[1] J. Clarke-Salt, SQL Injection Attacks and Defense, 2nd ed. Syngress, 2012.
[2] D. Stuttard and M. Pinto, The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws, 2nd ed. Wiley, 2011.
[3] R. Anderson, Security Engineering: A Guide to Building Dependable Distributed Systems, 3rd ed. Wiley, 2020.
[4] R. Morris and K. Thompson, "Password Security: A Case History," Communications of the ACM, vol. 22, no. 11, pp. 594-597, Nov. 1979.
[5] OWASP Foundation, "OWASP Top 10: Web Application Security Risks," 2021. [Online]. Available: https://owasp.org/Top10/
[6] B. Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C, 2nd ed. Wiley, 1996.
[7] W. S. Vincent, Django for Professionals: Production Websites with Python and Django. WelcomeToCode, 2022.

[8] B. Chess and J. West, Secure and Resilient Software Development. CRC Press, 2010.
[9] W. Stallings, Cryptography and Network Security: Principles and Practice, 8th ed. Pearson, 2022.
[10] W. Stallings and L. Brown, Computer Security: Principles and Practice, 4th ed. Pearson, 2018.
[11] OWASP Foundation, "Authentication Cheat Sheet," OWASP Cheat Sheet Series. [Online]. Available: https://cheatsheetseries.owasp.org/
[12] J. Erickson, Hacking: The Art of Exploitation, 2nd ed. No Starch Press, 2008.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)