iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** III **Month of publication:** March 2026

DOI: https://doi.org/10.22214/ijraset.2026.78013

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 III Mar 2026- Available at www.ijraset.com

Honeypot- Enabled Authentication System

Mr. R. Rajasekhar[1], M. Lalitha Anusha[2], M. Shriya Goud[3], Y. Vinay[4]
Computer Science & Engineering, Department of CS, ECE & IoT, Malla Reddy University, Hyderabad, India

Abstract: ***Honey Shield represents a deception-driven cybersecurity paradigm focused on deceiving, redirecting, and tracking malicious activities targeting web application authentication systems. The paradigm provides a centralized security service interface that enables the use of multiple defense strategies, such as bot protection, honeypot redirection, and SQL injection protection. Upon the detection of malicious activity, such as repeated login attempts within a certain time period, the malicious user is automatically redirected to a decoy system, thus effectively isolating malicious traffic from the actual web application. All security incidents, including suspicious access events, timestamps, request patterns, and activity types, are recorded and stored in a secure database for ease of analysis after the incident. An interactive dashboard is available for the visualization of login attempts, blocked attacks, and selected security services. By combining deception-driven defense and centralized logging, HoneyShield improves authentication security, attack analysis, and provides an ethical platform for advancing web application security best practices.***

Keywords: ***Honeypot-Enabled Authentication, Web Application Security, Bot Protection, Brute-Force Attack Protection, Credential Stuffing Protection, Deception-Driven Cybersecurity, Real-Time Threat Activity Monitoring, MongoDB Forensic Logging, Node.js Security Platform, Authentication Intrusion Detection System.***

I. INTRODUCTION

In today's modern age of rapidly growing online digital services, web-based login interfaces have become the primary entry point to secure user data, financial transactions, and organizational resources. As more and more organizations and institutions rely on onlineWithin the context of platforms, the login screen has evolved from a simple entry point to a highly critical security gateway. However, this increased dependency has also introduced a number of threats to cybersecurity. The most common of these threats are bot attacks, brute-force login attacks, credential stuffing, and phishing simulations. These threats are a result of poor password policies and login intervals, which can result in unauthorized access, breaches, and disruptions. The traditional security solutions available for these threats include CAPTCHA challenges, account lockout policies, and firewall filters, which are normally passive and do not have actual attack behavior data. In order to fill this security loophole, HoneyShield is proposed as a deception-oriented, honeypot-based authentication protection mechanism that aims to deceive, deflect, and dissect malicious login traffic in real-time. Unlike other solutions that only aim to drop malicious traffic, HoneyShield is designed to deliberately deceive potential attackers into a controlled decoy environment that simulates a genuine login page. In this isolated environment, malicious user activity is safely captured and logged without compromising the production environment. Using trap field detection, brute-force attacks, and behavior analysis, HoneyShield is able to detect bot traffic and credential attacks. With real-time notifications and threat analysis, HoneyShield assists administrators in monitoring malicious activity and making informed security decisions before a breach occurs.
The configuration of HoneyShield is a modular web stack.e. The backend, which is built using Node.js and Express, is responsible for handling login traffic, identifying anomalies, and handling the redirection process. For secure and organized long-term storage of forensic information, a MongoDB persistence layer is used to store IP addresses, timestamps, user-agents, and login activity, which allows for threat analysis. The backend is complemented by an interactive frontend built using HTML, CSS, and JavaScript, which provides administrators with a dynamic interface to analyze login attempts in real-time and historical information. By structurally analyzing malicious login traffic, HoneyShield promotes a proactive and research-oriented defense strategy that greatly improves web application security in today's increasingly hostile online environment.

II. LITERATURE SURVEY

Web application security research has gained momentum with the growing number of online authentication systems. Initially, research focused on improving password security storage—cryptographic hashes, secure database practices—to protect data at rest.Current trends indicate that former approaches are no longer keeping up with automated login attacks on authentication portals. Even if storage is secure, attackers can still use poor password security or reuse leaked login information using brute force and credential stuffing attacks. Recently, bot-based attacks have become a significant threat to login pages.

Research has shown that bots can easily evade traditional security measures such as CAPTCHAs and simple rate limiting by using headless browsers and AI automation. Credential stuffing, or the reuse of leaked login information across multiple sites, has been shown to be a significant factor in many account breaches. Most papers have identified that conventional security measures are centered on blocking suspicious traffic but lack detailed forensic information about attack behavior, including login activity, repeated IPs, and timing of requests. To fill these research gaps, researchers have focused on deception-based security, with honeypots being at the forefront. Honeypots originated from network intrusion detection systems but have since developed into web-specific variants that identify malicious activity in application fields. Low-interaction honeypots, such as hidden trap fields in login forms, have been successful against automated bot attacks because human users cannot interact with invisible form fields. Scalable logging systems using databases such as MongoDB are recommended to efficiently process large volumes of attack data. More recent research combines behavioral analysis and machine learning to identify attack patterns based on parameters such as login rate, IP grouping, and user-agent information. This represents a paradigm shift in web security research from static preventive security to dynamic monitoring and intelligence-driven security. HoneyShield draws upon these research findings by integrating hidden trap field detection, brute-force attack monitoring algorithms, structured log retention, and a centralized dashboard for real-time attack visualization. In the process, HoneyShield combines deception technology theory and web authentication security best practices.

III. SYSTEM ANALYSIS

A. Problem Statement

The modern web environment has turned into a dangerous place. Login pages are bombarded by constant bots, brute-force attacks, credential stuffing attacks, and phishing simulations.

As organizations rely more on web applications for digital services, user management, and financial transactions, login pages have turned out to be the most susceptible and attacked entry points. Attackers use automated scripts and stolen credential sets to test for unauthorized entry at scale, often evading detection in the process.

The main issue with the current state of authentication is that most security measures are reactive and not based on profound understanding. CAPTCHAs, login rate limiting, and account lockout mechanisms are all attempts to mitigate malicious activity, but they are only effective at blocking entry and not at understanding attacker behavior. Once an attack is blocked, precious information about IP behavior, login patterns, and attacker behavior is left unexamined and lost. This creates a “block and forget” cycle, where administrators remain blind to ongoing attack attempts and emerging attack trends.

The current authentication infrastructure may not offer a controlled environment to analyze malicious login behavior. Malicious traffic is either completely filtered out or blended with legitimate traffic, which increases the risk of false positives. Without forensic logging and centralized analysis, it becomes difficult for organizations to identify malicious attack attempts, mark high-risk IP addresses, and reverse-engineer automated attack methods. The existing divide between prevention and intelligence gathering indicates a requirement for a more proactive deception strategy.-oriented authentication protection framework.

B. Existing System

The first line of defense for most web authentication systems today is based on things such as CAPTCHA verification, rate limiting login attempts, firewall filtering, and locking out accounts after too many attempts.Although these methods make brute-force attacks slower and less common, they do not provide much information about the intentions and actions of the attacker.

CAPTCHAs, which were believed to be highly effective, are now commonly bypassed by AI-powered bots and automated browsers. Rate limiting can temporarily prevent rapid-fire login attempts, but it does not provide you with detailed metadata information—not a hint about user agent, login credential patterns, or other nuanced information. Firewalls primarily protect the network perimeter and can be fooled by sophisticated credential stuffing attacks that appear to be legitimate at the request level.

One of the major weaknesses in many authentication systems is the absence of centralized forensic intelligence. There are authentication attempt logs, but they are often disorganized, confusing, or not analyzed in real-time. There is no mechanism to deliberately interact with attackers in a controlled manner to gain insight into their behavior in a more detailed way. As a result, organizations are missing patterns of repeated intrusions, abused sources, or coordinated bot activity.

The absence of deception-based isolation and organized threat tracking makes authentication systems vulnerable to continuous automated attacks and makes it difficult to conduct effective forensic analysis.

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

C. Proposed System

HoneyShield challenges conventional web login security with a deception-based strategy. Instead of just preventing suspicious login activity, this system intelligently identifies possible danger and tricks attackers into a plausible, contained honeypot. This safeguards the actual production environment until the attacker's activity is observed and recorded for later analysis.

On the server side, using Node.js with Express, login requests are handled using methods to prevent abuse. In particular, hidden honeypot fields and a brute-force protection system are used.The malicious bots that fill up the hidden honeypot fields are quickly detected, and further login attempts from the same IP address are then tracked and assessed against set thresholds. Once malicious activity is detected, users are automatically diverted to a decoy dashboard that accurately mimics the look and feel of a successful login, thus halting malicious activity targeting the actual application.

For persistence, a MongoDB tier is used to store forensic information in a structured manner, including IP addresses, timestamps, user agents, login credentials, and threat levels. This setup allows the system to assess past activity patterns and identify past malicious activity. An admin dashboard, designed with the latest web tech, offers real-time visualizations of attack data, suspicious IP address groupings, and login attempt patterns.

With the incorporation of deception technology, logging, and real-time monitoring, HoneyShield utilizes authentication-driven intrusion attempts as a rich source of threat intelligence. This approach not only secures web applications against unauthorized access but also improves situational awareness, forensic analysis, and the development of a proactive defense strategy in line with the modern digital landscape.

IV. METHODOLOGY

A. Architecture Module & Tiered Framework

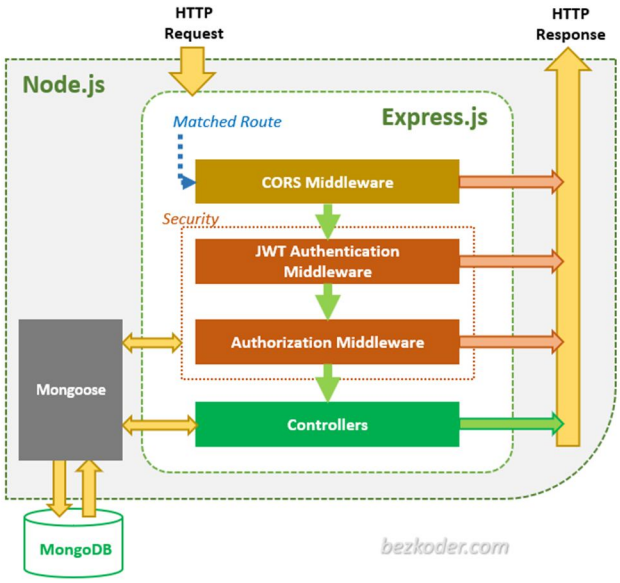


Figure 1: HoneyShield System Architecture Diagram

The HoneyShield architectural design uses a structured multi-tier web application architecture that promotes modularity, security isolation, and efficient processing of authentication requests. As shown in the system architecture diagram, the system architecture is divided into four main tiers: the Presentation Tier, the Logic Tier, the Data Tier, and the Security Tier.

At the heart of the system architecture is the Presentation Tier, which comprises the Honeypot Login Interface and the Administrative Dashboard.. The login interface is designed to mimic a genuine authentication interface while incorporating hidden trap fields that are invisible to human users. The administrative dashboard provides real-time visualization of malicious login activity, IP address monitoring, and attack analytics.

The Logic Tier, developed using Node.js and Express, is the system's processing engine. This system backend handles incoming login requests, performs bot detection analysis, evaluates brute-force attack thresholds, and manages dashboard redirection. It is the system's decision-making component that determines the legitimacy, suspiciousness, or maliciousness of a login attempt.

To facilitate data persistence and organized logging, the Data Tier utilizes MongoDB as the document-oriented database. This system tier is responsible for storing comprehensive authentication attempt logs, which include IP address, timestamp, user-agent, attempted credentials, and threat type.

Finally, the Security Tier utilizes deception techniques such as hidden honeypot trap fields and simulated dashboard redirection. By trapping malicious users in a controlled environment, the system protects production authentication services from being affected while allowing for secure forensic analysis.

B. Implementation of the Threat Detection Engine

The core functionality of HoneyShield revolves around detecting malicious authentication behavior and converting it into structured security intelligence. The backend continuously processes login submissions and evaluates them using predefined detection algorithms.

The first detection mechanism involves hidden honeypot trap fields embedded within the login form. Human users cannot interact with these invisible inputs; however, automated bots typically auto-fill all available fields. If a hidden field contains data upon submission, the system immediately flags the request as bot activity and initiates logging procedures.

The second level of detection is focused on the observation of brute-force attacks.The backend service keeps track of login attempts related to a particular IP address.If the rate of login attempt attempts exceeds a certain limit within a certain time frame, the login attempt is labeled as suspicious or malicious.Instead of launching an immediate blocking process, HoneyShield redirects the malicious traffic to a fake dashboard, thus tricking the attackers into thinking that the login attempt has been successful. This approach ensures that the legitimate login attempt does not reach the system while continuing to harvest data.

C. Data Persistence and Forensic Logging

To overcome the limitations of temporary or fragmented log storage, HoneyShield integrates a dedicated MongoDB database module for structured forensic retention. This persistence layer is responsible for maintaining comprehensive authentication logs generated during suspicious interactions.

Each log message contains rich metadata like IP address, timestamp, number of login attempts, login credentials, browser details, and threat level. The document model of MongoDB makes it easy to store and retrieve high-velocity login messages with flexibility in schema, which is a hallmark of relational databases. This logging feature helps the administrators track patterns of attacks and also helps them to monitor the trends of suspicious activities. The backend dynamically interacts with the database to fetch historical data, which helps

D. Risk Assessment and User Notification

HoneyShield incorporates a structured threat classification framework to simplify complex authentication data for administrative review. Based on the analysis performed in the Logic Tier, login attempts are categorized into predefined risk levels such as normal, suspicious, or high-risk.

For example, a single failed login attempt may be treated as standard behavior, whereas repeated rapid attempts from the same IP address trigger elevated risk classification. Similarly, detection of honeypot trap field interaction immediately marks the request as automated bot activity.

These categories are then relayed to the administrative dashboard in real time.The administrative dashboard organizes login statistics, listing suspicious IP addresses, providing graphs of attempt frequency and threat alerts in a readable format. This proactive monitoring ensures that administrators are aware of the constant threats to authentication and are in a position to take informed measures as necessary.

E. Behavioral Analysis for Attack Identification

The intelligence of HoneyShield lies in its behavioral monitoring approach to authentication-based intrusion detection.Instead of relying solely on fixed rule enforcement, the system evaluates patterns such as login attempts, IP grouping, and attempts to use credentials. By keeping a structured log history, the system can detect repeated attempts to intrude from the same network source.Suspicious IP addresses can be flagged for administrative review or optional blacklist implementation.

This behavioral model allows HoneyShield to distinguish between legitimate user errors and coordinated automated attacks. By analyzing request timing and submission patterns, the system improves the reliability of detection while minimizing false positives.

F. Experimental Deployment Configuration

To evaluate system performance, HoneyShield was deployed within a controlled laboratory environment using standard web development infrastructure.The backend server was set up using Node.js and Express, and MongoDB was run locally to enable organized log storage. Automated login simulation scripts were used to test and replicate brute-force and bot attack simulations.
The system successfully detected hidden trap field interactions and identified repeated login attempts exceeding defined thresholds. All suspicious activities were logged in real time and reflected immediately within the administrative dashboard.
This controlled testing environment validated the effectiveness of deception-based authentication monitoring and demonstrated the system's ability to operate efficiently without impacting legitimate user interactions. The results confirm that HoneyShield provides a practical and scalable solution for enhancing web application authentication security through structured forensic intelligence and controlled attacker isolation.

V. RESULTS

A. User Interface and System Entry

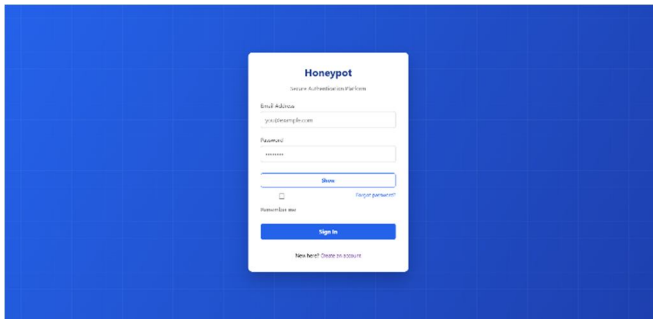


Fig. 1: HoneyShield Login Interface

The HoneyShield landing interface establishes the application as a secure authentication monitoring platform.Figure 1 shows a login page designed to resemble a typical login page in order to create realism while using deception-based security mechanisms in the background. The login page contains elements of a typical login page, including an email entry field, a password entry field, a "Remember Me" checkbox, and account recovery links. Behind this typical interface, the system uses honeypot trap fields that are not visible to real users but can be detected by bots. The interface has two main functions:
1.To enable real users to gain access.
2.Bot Interaction Monitoring – Captures automated scripts that attempt to auto-fill all form fields.
3.Credential Behavior Tracking – Records repeated rapid login attempts for brute-force detection.
By maintaining a professional and minimalistic design, the system ensures usability while operating as a deception-based monitoring gateway

B. Dashboard: Operational Overview

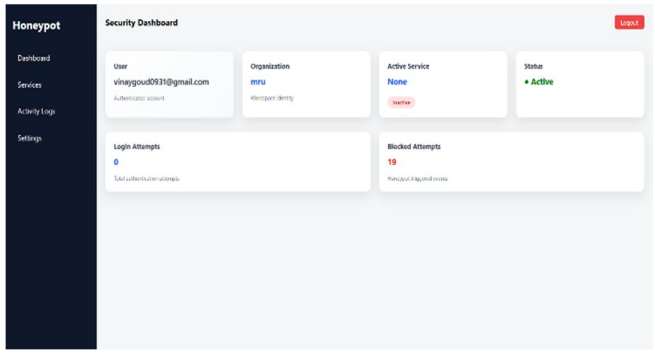


Fig. 2: HoneyShield Security Dashboard

The Security Dashboard is the central command center of the HoneyShield system. As shown in Fig. 2, the security dashboard combines authentication telemetry into organized real-time metrics for administrative visibility.
The dashboard is divided into multiple monitoring components:
1.User Information Panel – Displays authenticated account details and organization context.
2.System Status Indicator – Shows active security posture (e.g., Active/Inactive).
3.Login Attempts Counter – Tracks total authentication attempts processed by the system.
4.Blocked Attempts Metric – Displays the number of honeypot-triggered or suspicious events.
This consolidated overview enables administrators to quickly assess the security state of the authentication environment without analyzing raw logs

C. Security Services Module

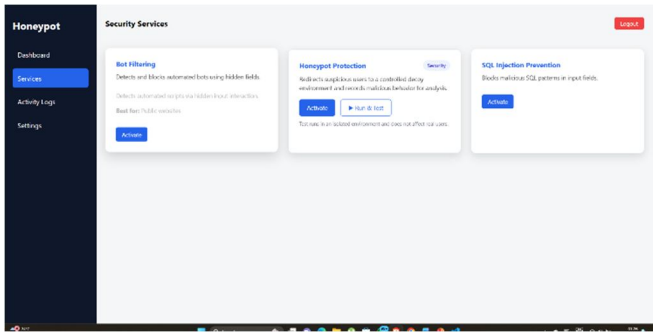


Fig. 3: HoneyShield Security Services Page

The Services module represents the configurable protection layer of the system. As shown in Fig. 3, the interface provides activation controls for a variety of protection mechanisms:
1. Bot Filtering – This module detects automated scripts through interaction with hidden input fields.
2. Honeypot Protection – This module redirects malicious users to a controlled decoy environment for observation.
3. SQL Injection Protection – This module detects malicious input patterns in authentication fields.
Each module provides activation controls to allow administrators to dynamically activate or test protection layers. The “Run & Test” functionality provides a controlled testing environment to conduct experiments without affecting production data.

D. ActivityLogs: Forensic Monitoring

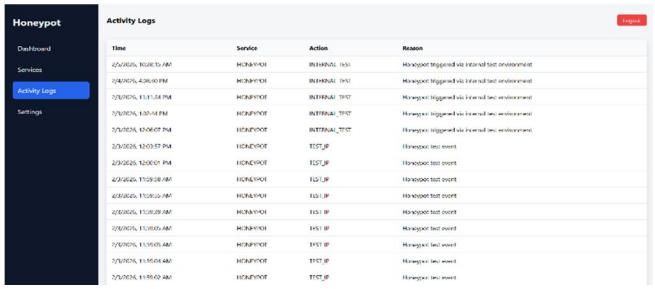


Fig. 4: HoneyShield Activity Logs Interface

The Activity Logs page represents the forensic visualization layer of the system. As shown in Figure 4, each honeypot event is logged with structured metadata. Each log entry contains the following:
1. Timestamp – The date and time of the event.
2. Service Identifier – The service module responsible for the detection (e.g., HONEYPOT).
3. Action Type – The type of event (e.g., INTERNAL_TEST or TEST_IP).
4. Reason Field – The reason for the honeypot event.
The structured logging system allows for the analysis of behavior over time and is useful for security audits.

E. Honeypot Test Environment

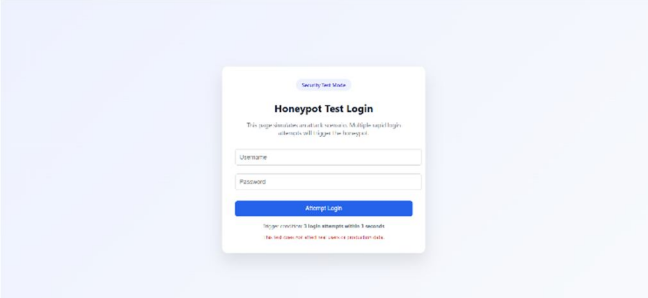


Fig. 5: HoneyShield Honeypot Test Login Page

The Honeypot Test Login page represents the controlled simulation environment of the system. As shown in **Fig. 5**, this interface allows administrators to simulate brute-force scenarios safely.
The test logic is configured with a predefined trigger condition:

- Three login attempts within three seconds activate the honeypot.

Once triggered, the system logs the event and isolates the interaction. Importantly, this test environment does not affect real users or production credentials.
This feature validates the effectiveness of rapid login detection algorithms.

F. Verification and Redirection Mechanism

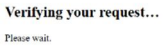


Fig. 6: Honeypot Trigger Verification Screen

After the occurrence of anomalous behavior has been detected, the system moves to a verification phase, as shown in Figure 6. This interface simulates processing as the backend logs the malicious activity. This redirection has two main purposes:

1. It stops any further analysis of the real authentication backend.
2. It allows safe logging and analysis of threat in the database.

The deception response mechanism prevents attackers from staying in a controlled environment without gaining knowledge of the system's defenses.

VI. CONCLUSION

The creation of HoneyShield effectively remedies the critical security deficiency within modern web authentication systems. Throughout this research, it is clear that while login-based services are a critical component of digital platforms, traditional safeguarding solutions are often employed in a reactive manner and lack any real-world visibility into malicious activity. Automated bot attacks, brute-force attacks, and credential-stuffing attacks remain prevalent due to the lack of deception-based monitoring and forensic logging. By leveraging the defense paradigm from simple blocking solutions to a proactive, honeypot-driven solution, this research makes it clear that malicious login activity can now be identified, segmented, and analyzed before the integrity of a web application is compromised. The inclusion of hidden trap fields, brute-force thresholds, and intelligent redirects into a controlled decoy environment confirms the viability of deception-based security solutions.Through the use of a Node.js-based backend solution coupled with a MongoDB-based persistence solution, HoneyShield is able to handle malicious authentication traffic while continuing to handle normal user traffic.The results of the experiment show that the system is capable of detecting malicious activity through automated processes and providing enough logs to facilitate forensic analysis.

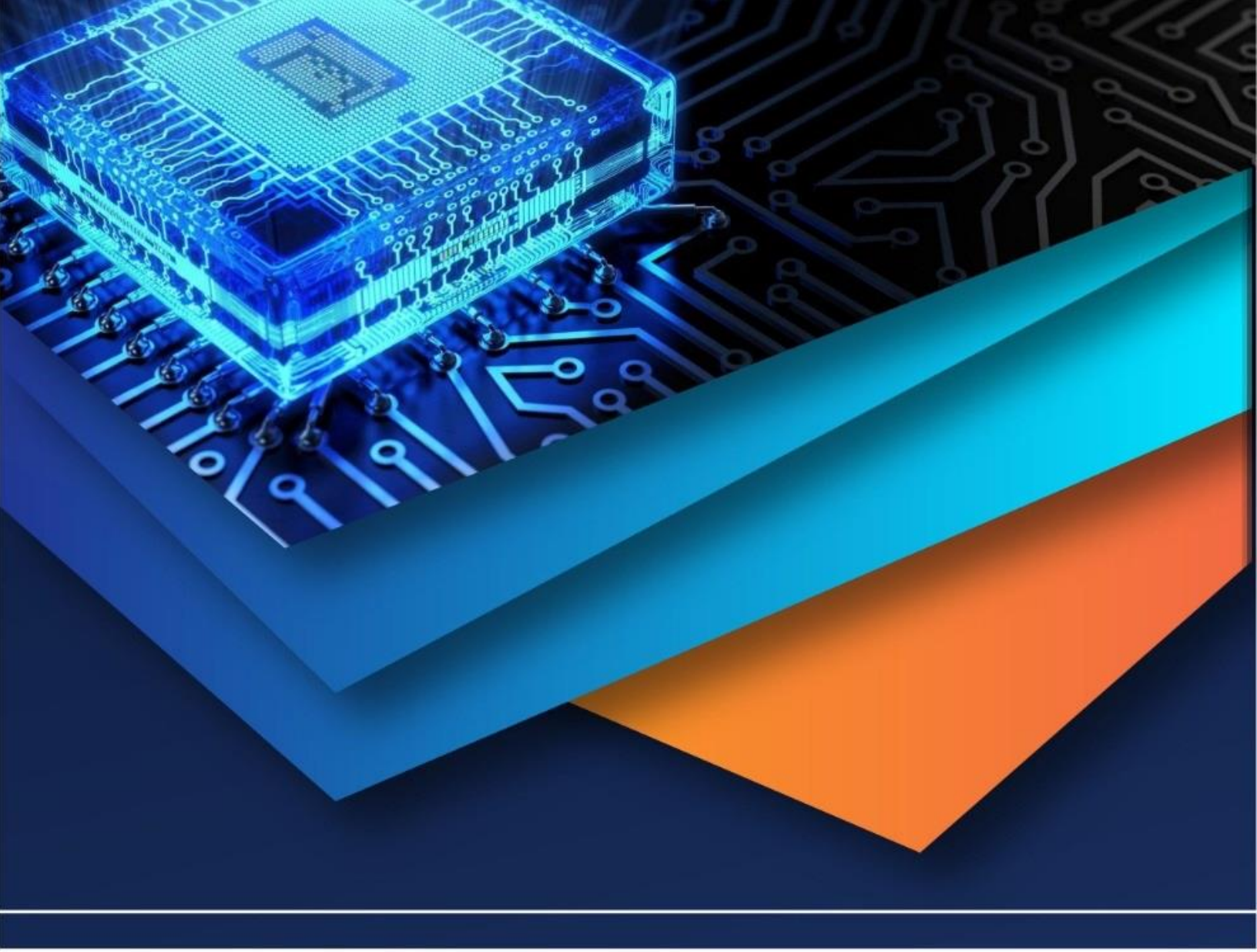
HoneyShield improves the overall security position of web applications by providing real-time monitoring, analysis, and evaluation of the dashboard. The system allows administrators to better understand and counter new intrusion methods by analyzing raw authentication traffic to provide real-time security analysis. This research provides a practical foundation for future innovation in intelligent authentication protection solutions, helping to create a more secure and robust digital environment.

VII. FUTURE SCOPE

One area that could be explored for further development of HoneyShield is the incorporation of Artificial Intelligence (AI) and Machine Learning (ML) to improve its behavioral threat-detection features. With the use of ML algorithms on past authentication data in the MongoDB database, it would be possible to train the system to differentiate between genuine user errors (such as periodic password errors) and malicious automated attack behaviors. This would not only help improve the accuracy of threat detection but also minimize the occurrence of false positives. Moreover, AI-powered anomaly detection would also make it possible to detect minute behavioral anomalies, such as distributed and infrequent credential-stuffing attacks, that might evade traditional threshold-based detection systems. Another critical area where HoneyShield needs to be developed is predictive threat modeling. By leveraging data points such as login attempt frequency, IP clustering patterns, time-based attack patterns, and user-agent patterns, HoneyShield can potentially predict new attack patterns and adjust its defense parameters accordingly. For example, adaptive rate limiting thresholds or automatic IP reputation scoring might be used to improve real-time protection without requiring administrative intervention.

REFERENCES

[1] **Honeypots: Tracking Hackers.**
Lance Spitzner
[2] Virtual Honeypots: From Botnet Tracking to Intrusion Detection.
Niels Provos; Thorsten Holz
[3] Guide to Intrusion Detection and Prevention Systems (IDPS).
Karen Scarfone; Peter Mell
[4] A Survey of Man-in-the-Middle Attacks.
Mauro Conti; Nicola Dragoni; Viktor Lesyk
[5] **Evaluation of Machine Learning Algorithms for Intrusion Detection Systems.**
Mohammad Almseidin; Maen Alzubi; Szabolcs Kovacs; Mousa Alkasassbeh
[6] **Detection of Brute Force Attacks on Web Applications.**
S. R. Chavan; S. P. Patil
[7] **Web Application Security: Attacks and Countermeasures.**
Dafydd Stuttard; Marcus Pinto
[8] **OWASP Top 10 Web Application Security Risks.**
OWASP Foundation
[9] **An Improved Honeypot-Based Intrusion Detection System Using Machine Learning.**
A. M. Alatawi; F. Alsubaei
[10] **Credential Stuffing Attacks: Analysis and Prevention Techniques.**
Akamai Security Intelligence Group
[11] **A Network Attack Blocking Scheme Based on Threat Intelligence.**
Kun Li; Rui Wang; Haiwei Li; Yan Hao
[12] **Intrusion Detection and Prevention Using Blocking and Back Tracking for IP Spoofing.**
Ritesh Kumar; Sunita G; Rajeshwari M
[13] **Deception-Based Cyber Defense Strategies.**
Jeffrey Pawlick; Edward Colbert; Quanyan Zhu
[14] **MongoDB for High-Volume Log Management in Security Applications.**
MongoDB Technical Documentation Team
[15] **Behavioral Analysis for Web-Based Intrusion Detection.**
S. Axelsson
[16] **Detection of Automated Bots in Web Applications Using Hidden Form Fields.**
T. Bursztein; A. Moscicki
[17] **Deep Learning for Intrusion Detection Systems: A Review.**
Yisroel Mirsky; Tom Mahler; Ilan Shelef; Yuval Elovici
[18] **Zero Trust Architecture.**
National Institute of Standards and Technology (NIST
[19] **Modern Web Authentication Threats and Mitigation Techniques.**
SANS Institute Research Team
[20] **Honeypot-Based Authentication Protection Systems.**
K. Alieyan; A. Almomani



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY