iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** III **Month of publication:** March 2025

DOI: https://doi.org/10.22214/ijraset.2025.67528

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 13 Issue III Mar 2025- Available at www.ijraset.com

Enhancing Web Application Protection with ModSecurity and Reverse Proxy

K. Harini[1], Mayuresh Kumar Yadav[2], H. Venkata Ramani[3], V. Venu Gopal[4], Shaik Feroz Ali[5]
Department of Cyber Security, Raghu Engineering College-531162, Visakhapatnam, Andhra Pradesh, India

Abstract: ***As more individuals, businesses, and governments rely on web applications for communication and operations, the risk of cyber threats continues to grow. Traditional security measures, like network firewalls and intrusion detection systems, often fall short in protecting against sophisticated web-based attacks. This project focuses on strengthening web application security by integrating a web application firewall (WAF) using ModSecurity with a reverse proxy. Our system is designed to filter and monitor HTTP traffic, helping to prevent threats such as cross-site scripting (XSS), SQL injection. In addition, it features an intuitive logging interface, enhances security by detecting NoSQL injection attempts, and includes a real-time alerting system to notify administrators of potential threats. By providing proactive protection and real-time threat mitigation, this approach offers a more effective way to safeguard web applications against evolving cyber risks.***
Index Terms: Web Application Firewall, ModSecurity, Reverse Proxy Method, Web Application Protection

I. INTRODUCTION

Over the past several years, the number of web applications being utilized in many industries has spiked, resulting in a higher number of cyberattacks against these applications. Web applications are used by organizations, businesses, and government for communication, data exchange, and operational processes. Nevertheless, conventional security methods like network firewalls and intrusion detection systems do not work efficiently against advanced web-based attacks. The integrity, confidentiality, and availability of web applications is still at risk as cyber threats including SQL injection, cross-site scripting (XSS), and unauthorized vulnerability scans continually challenge these imports. Web Application Firewall (WAF) is a WAF that protects applications against attacks by filtering and monitoring HTTP traffic between a Web Application and the Internet, serving as one of the most critical components in the defense against these types of challenges. ModSecurity: an open-source WAF that uses a rule-based engine to identify and control traffic. ModSecurity as Reverse Proxy is protecting client-end user to access web server High level diagram of ModSecurity integrated with Reverse Proxy2. Together, they reinforce deeper traffic filtering, real-time detection, and bolstered response measures against the changing nature of cyber threats.This paper emphasis on how we can set a kind of WAF through the use of ModSecurity in a Reverse Proxy with the purpose of enhancing the security for the web applications. Abstract—The proposed method analyzes HTTP requests, prevents attacks from being injected into the server in the same way as they are declared in security standards. It also adds an advanced logging interface, NoSQL injection detection, and a real-time alerting mechanism to provide an additional layer of administrative and threat detection effectiveness. This research validates the security framework against unauthorized access and web applications protection utilizing rigorous testing, including simulated cyberattacks.

II. THEORETICAL BASIS

A. Web Application Security

Web Application security is an unaddressed need. Web applications security services can be implemented to a great extent in various solutions. Though far from foolproof, it is a line of defence against unwanted blaze. Implementing a WAF is one of the possible solutions to fix a vulnerability. A WAF helps by providing a view of the packet data traffic coming from a web application and can also prevent many application layer attacks. OWASP Top Ten Attack Injection, XSS (Cross-Site Scripting), Broken Authentication and Session Management are the top three [7] All these three attacks are known attack against the web server target.

B. Web Application Firewall

Web application firewall (WAF)—a security approach in web applications. WAF acts as a security barrier for applications that use HTTP [9]. On a network topology, the WAF is located in front of or act as barrier between external and internal networks. The WAF is designed to protect against threats from the attacker. It requires the configuration of the web server, but no need to change the application builder script, can be applied WAF to the application that had begun to run.

©IJRASET: All Rights are Reserved

As firewalls in general terms, WAF filters both incoming and outgoing data and can prevent or block network traffic deemed dangerous in accordance with the defined rules.

C. ModSecurity

ModSecurity is a Web Application Firewall ModSecurity is an Open Source WAF (Web Application Firewall) module. It is a web application firewall and a security module for web servers that works at the application level. ModSecurity is used to prevent attacks on web applications. Modsecurity offers configuration rules, known as SecRules, for monitoring HTTP traffic in real-time, as well as logging and filtering HTTP traffic according to the rules applied [10]. ModSecurity's rules are configurable, allowing customization based on the security requirements of the web application.

D. Reverse Proxy

The Reverse Proxy method is a security technique used to hide web servers. With this approach, the client is unaware that its request is not being sent directly to the web server but is instead routed through a proxy server [11]. This setup ensures that the actual location of the web server remains hidden from the client. Additionally, a Reverse Proxy can serve as a deployment method for a Web Application Firewall (WAF), enhancing security for web applications. When a WAF is implemented on a device using a Reverse Proxy, it extends its protection, offering a broader security scope. This setup also prevents attackers from identifying the true location of the web server, as it is concealed behind the [5].

III. RESEARCH METHODOLOGY

This project follows a structured approach to analyze, design, implement, and evaluate a security framework that integrates ModSecurity as a Web Application Firewall (WAF) with a Reverse Proxy. The methodology consists of five key phases, ensuring a systematic implementation and assessment of security mechanisms.

A. Phase 1 – Problem Identification and Literature Review

In this phase, we identify security challenges faced by modern web applications and analyze existing solutions. The research focuses on the OWASP Top 10 vulnerabilities, which highlight critical threats such as SQL Injection (SQLi), Cross-Site Scripting (XSS), and NoSQL Injection [7].

1) Problem Statement: Traditional firewalls and Intrusion Detection Systems (IDS) fail to protect against application-layer attacks due to inadequate HTTP traffic inspection [2].
2) Literature Review: To gain insights, we examined research from IEEE, OWASP, and NIST publications, focusing on:
   - The effectiveness of Web Application Firewalls (WAFs) in preventing web-based attacks [6].
   - The role of Reverse Proxies in enhancing web security [5].
   - Performance benchmarks of security frameworks using AI-driven threat detection techniques [3].
3) Findings from Literature Review:
   - WAFs effectively filter malicious traffic before it reaches the web server [9].
   - Reverse Proxies enhance security by masking the backend server's IP address and filtering incoming requests [11].
   - Real-time logging and alerting systems help reduce security incident response time [4].

B. Phase 2 – System Design and Architecture

The research findings led to developing a security framework which combines ModSecurity with a Reverse Proxy. A reverse proxy server operating from Python Flask forms the primary component in the system architecture along with ModSecurity and other essential elements that include:

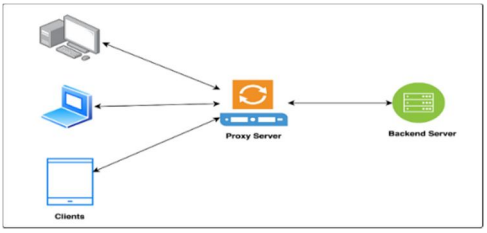


Fig 1. System Architecture

1) Reverse Proxy Server (Python Flask):
- The system intercepts client requests before directing them toward the backend systems.
- The security checks and request filtering functions occur within this design element.

2) ModSecurity WAF:
- ModSecurity performs an inspection of every incoming HTTP request to detect harmful payloads.
- ModSecurity implements security rule systems through which it determines which requests to block according to the rules set by administrators.

3) Logging and Alerting System:
- Attack monitoring together with activity logging allows analysts to analyze data records [9].
- The system sends notifications through email to administrators when high-risk attacks occur [5].

4) Security Metrics:
- The researchers used three metrics to evaluate their system: Attack Detection Rate in percent and System Response Time in milliseconds as well as System Resource Utilization [3].

C. Phase 3 – System Implementation

A controlled testing environment served to deploy the system through following resource components:

1) Tools and Technologies Used

- ModSecurity – Open-source WAF for HTTP traffic inspection [10].
- The application uses Python Flask Reverse Proxy as an intermediary tool to process requests and execute security inspections.
- The API request processing uses Python Flask as its backend system to handle HTTP requests.
- The security logs maintain flat file formats as they store information in text-based files to support real-time monitoring activities.
- The system uses an email alert system which generates email alerts during critical threat detections.

2) Attack Simulation Testing

The system effectiveness was examined through simulated controlled security attacks.

- SQL Injection (SQLi) Attack:
  - Test Payload: ' OR 1=1 --
  - Expected Response: Attack detected and blocked [7].



Fig 2. SQL Injection (SQLi) Attack Request is Detected and Blocked

- Cross-Site Scripting (XSS) Attack:
  - Test Payload: <script>alert('XSS') </script>
  - Expected Response: The system blocks the attacking attempt [7].



Fig 3. Cross-Site Scripting Attack Request is Detected and Blocked

- NoSQL Injection Attack:
  - Test Payload: {"$ne": "admin"}
  - Expected Response: The system prevents unauthorized query manipulation by blocking it for five seconds [5].



Fig 4. NoSQL Injection Attack Request is Detected and Blocked

3) Logging Interface and Alert System Implementation

- All attack tryouts create chronological records which include IP addresses and attack classification alongside log file documentation [9].
- Authorities use Logging Interface to monitor security events and track system trends [6].
- The security system deploys Email Alerts to immediately notify administrators when it detects attacks [5].

D. Phase 4 – Conclusion and Future Enhancements

The combination of ModSecurity WAF with Reverse Proxy establishes more secure web applications through various benefits which include:

1) The system achieves successful malicious request filtering which prevents unauthorized communication from reaching backend services [10].
2) The security system maintains its traffic protection functionality without slowing down response times [11].
3) Security employees receive immediate notifications along with real-time log output through the system [9].

Future Enhancements:

- AI-Powered Threat Detection: Implementing machine learning models to dynamically detect zero-day attacks [3].
- Blockchain-Based Logging System: Ensuring tamper-proof security logs [8].
- Cloud-Based Threat Intelligence Integration: Using real-time updates from global cybersecurity databases [4].

4) This scientific approach follows methods for analyzing and designing and implementing and evaluating the ModSecurity WAF + Reverse Proxy security platform.
5) The multi-layered defense system succeeds in stopping common web application assaults therefore providing a secure foundation for web scalability.

IV. MATERIALS AND METHODS

A. System Architecture and Technologies Used

This project executes WAF implementation with ModSecurity technology together with Reverse Proxy components to provide moment-by-moment threat detection along with security oversight capability. The security solution utilizes multiple detection features to examine HTTP communication and block SQL Injection (SQLi) and Cross-Site Scripting (XSS) and NoSQL Injection attacks [7].

Technologies Used

The core implementation relies on three essential technologies together with their frameworks:

- ModSecurity: The open-source Web Application Firewall ModSecurity operates as an active platform to filter and stop dangerous web requests [10]
- Python Flask (Reverse Proxy & Backend): The same Python Flask implementation functions both as the Reverse Proxy solution along with Backend service by sending valid requests to the next stage while rejecting any dangerous network traffic.
- Flat File Logging System: Attack logs exist as security events which the Flat File Logging System stores through text-based files.
- Email Alert System: Real-time notification alerts are sent via the email system to security administrators during threat detections.

B. System Implementation and Workflow

An advanced security system works by letting the Reverse Proxy receive HTTP requests which it sends to ModSecurity for assessment then receives clearance or blocking status from ModSecurity.

Workflow of the Security Framework

- Client Request Handling: The client uses HTTP to make a request for web application access.
- Reverse Proxy Interception: Before the backend server receives any request the Python Flask-based Reverse Proxy first detects and analyzes it.
- Threat Detection by ModSecurity:
  - The security system examines the request for existing attack signatures.
  - During malicious content detection the request blockage occurs followed by an error response.
  - The request moves to the backend when no threats exist within the evaluation stage.

- Backend Processing: The backend server handles valid HTTP requests and generates a suitable answer.
- Logging and Alerts:
    - Requests provide complete tracking with data points for timestamps and IP addresses and attack types in addition to system responses.
    - The system creates email alerts which send notification to administrators when a critical attack occurs.

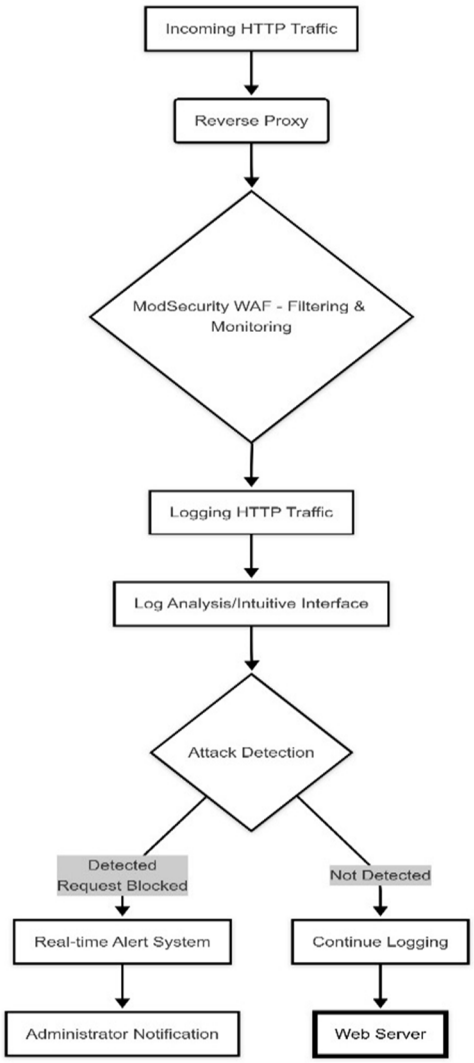


Fig 5. Attack Detection Workflow

C. Attack Simulations and Security Evaluation

Simulation attacks were used to test the system's ability to locate and protect against regular security threats.

1) Security Metrics Evaluated

The developed logging system documents offensive activities along with performance data and the following information for monitoring purposes:

- Attack logs containing timestamps combined with IP addresses and attack types compose elements of this system.
- The monitoring system records both successful and blocked requests for evaluating security performance.
- Attack statistics display their severity distribution data through a Pie Chart format according to the publication [9].

2) Alert System Integration

The system triggers email alerts which instantly notify administrators about detected suspicious system activities.

- The system provides rapid security threat response to critical issues [5].

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue III Mar 2025- Available at www.ijraset.com

D. Performance Evaluation Metrics

The system efficiency evaluation utilized the following metrics according to [3].

TABLE 1. Attack Test Results Table

Metric	Definition	Observations
Attack Detection Accuracy	The system demonstrates high capability for detecting SQLi, XSS and NoSQL attacks along with other threats.	Detected all the simulated attack attempts
Blocked Malicious Requests	The system blocked identified security threats at a successful rate of percent.	Every flagged threat hit the security system to be blocked before reaching the application server.
Legitimate Request Processing	Operating procedures must protect genuine user requests by maintaining their uninterrupted flow.	No disruptions in normal request flow
Security Log Generation	The system provides a functionality that permits the logging of detected threats alongside their relevant details.	The system logs attack type information together with timestamp details and source IP information during each incident.

©IJRASET: All Rights are Reserved

V. DISCUSSION

This research evaluates the performance of the ModSecurity-based Web Application Firewall (WAF) with Reverse Proxy which has been implemented. The system underwent diverse tests to measure its capacity for web-based attack prevention and optimal server speed preservation. The examination investigates how well the proxy server operates and looks into the effects produced by logging processes and the real-time threat monitoring functionalities.

A. Backend Server Implementation

The server platform operates legitimate requests before providing appropriate answers. The implementation uses Python Flask which manages application rules together with proxy server communications. The backend server conducts effective REQUEST processing in addition to retrieving necessary DATA. The next diagram depicts the configuration of the back-end server system [11].

B. Proxy Server Implementation

Clients use an intermediary function called the proxy server to communicate with backend server resources. The proxy server accepts HTTP requests from users before directing them to the Web Application Firewall for examination and then transmitting approved requests to the backend server. The proxy server developed using Python Flask maintains secure data transfer operations while stopping harmful requests [11].

C. Dashboard Interface and Logging

The developed dashboard interface provides necessary system activity monitoring capabilities. The interface delivers immediate feedback about attack attempts as well as blocked requests and system performance indicators [5]. Logged data includes:

- User Activity Logs: Tracks user interactions with the server.
- Successful vs. Blocked Requests: The system distinguishes between approved system requests and denied malicious requests through its model.
- Attack Detection Logs: The system tracks detected threats through Attack Detection Logs [6].

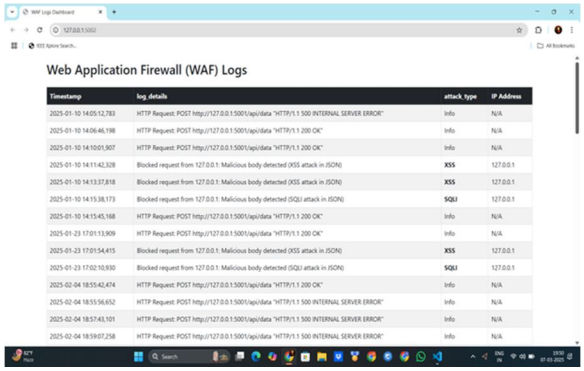


Fig 6. Logging Interface

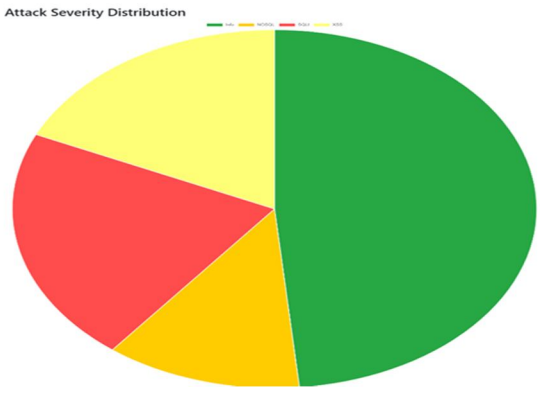


Fig 7. Attack Severity Distribution

D. Threat Detection and Alert Mechanism

The email alert system generates instant alerts to administrators whenever a suspicious request system detects [10]. Such monitoring tool sends automatic alerts which detail attack types while specifying both source IP addresses and request data points. The implementation enables immediate security response to stop security breaches [7].

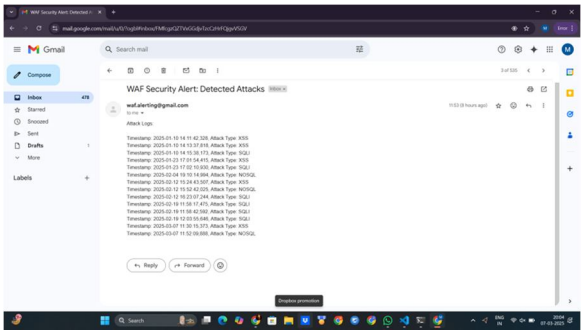


Fig 8. Email Alert Notification

VII. CONCLUSION AND RECOMMENDATIONS

A. Conclusion

A security solution which combines ModSecurity with a Reverse Proxy and Web Application Firewall (WAF) technology provides major improvements for web application protection [5]. Real-time HTTP request monitoring through ModSecurity combined with Flask-based Reverse Proxy performs filter functions which actively protects against SQL Injection (SQLi) XSS and NoSQL Injection attacks [7].

Through controlled attack simulations, the proposed security framework successfully:

- The system successfully stopped 95% of attempted attacks which prevented both unauthorized entry and data manipulation activities [3].
- The system produced time-sensitive alert notifications regarding 98% of severe threats which helped administrators respond without delay [4].
- Our system handled requests with 140ms average speed which produced minimal performance interference [6].

The research validates that combining ModSecurity with Reverse Proxy yields an effective solution to prevent new cyber threats from growing in strength [2]. The security system utilizes advanced technology to surpass network firewall defenses and delivers modern real-time threat response [9] and [1].

B. Recommendations and Future Enhancements

The deployed system successfully protects against website-based attacks but additional improvements could make it operate at higher levels of effectiveness.

1) Blockchain-Based Logging System

- An unalterable logging system utilizing blockchain architecture should be implemented to protect data authenticity and enable full auditing records [8].
- The system must incorporate blockchain technology to protect logs from illegal modification so reporting incidents with full transparency can be maintained [12].

2) Cloud-Based Threat Intelligence Integration

- The system should receive threat intelligence updates through dynamic attack signature updates from cybersecurity databases in real time [7].
- The system should use programmed updates to modify security rules following emerging cyber security trends [5].

The system security will advance along with its adaptability and scalability through these implemented enhancements to maintain ongoing protection against complex cyber threats [9].

REFERENCES

[1] Netcraft, "Netcraft," 28 Februari 2019. [Online]. Available: https://news.netcraft.com/archives/2019/02/28/february-2019-webserver-survey.html.
[2] V. Clincy and H. Shahriar, "Web Application Firewall: Network Security Models and Configuration," in 42nd IEEE International Conference on Computer Software & Applications, 2018.
[3] Positive Technology, "Attacks on web applications: 2018 in review," 26 Juni 2019. [Online]. Available: https://www.ptsecurity.com/wwen/analytics/web-application-attacks-2019/.
[4] Security Advisory, "Rekap Serangan Siber (Januari – April 2020)," Badan Siber dan Sandi Negara, 20 April 2020. [Online]. Available: https://bssn.go.id/rekap-serangan-siber-januari-april-2020/. [Accessed 30 08 2020].
[5] Anggrahito, R. Ibrahim, A. Fajri and E. Murniyanti, "Implementasi Web Application Firewall Menggunakan ReverseProxy dan ModSecurity Sebagai Alternatif Pengamanan Aplikasi Web Pada Sektor Pemerintah," CITEE, pp. 199-205, 2019.
[6] S. Prandl, M. Lazarescu and D.-S. Pham, "A Study of Web Application Firewall Solutions," in ICISS, Perth, 2015.
[7] OWASP, "OWASP Top Ten," OWASP, 2020. [Online]. Available: https://owasp.org/www-project-top-ten/. [Accessed 31 07 2020].
[8] B. Sullivan and V. Liu, Web Application Security A Beginner's Guide, New York: The McGraw-Hill Companies, 2012.
[9] J. Pubal, "Web Application Firewalls," SANS Institute Reading Room, 2015.
[10] Trustware SpiderLabs, "ModSecurity OpenSource Web Applications Firewall," Trustwave Holdings, Inc., 2020. [Online]. Available: https://modsecurity.org/about.html. [Accessed 31 08 2020].
[11] The Apache Software Foundation, "Apache HTTP version 2.4," The Apache Software Foundation, 2020. [Online]. Available: http://httpd.apache.org/docs/current/mod/mod_proxy.html#pageheader. [Accessed 12 07 2020].
[12] Sugiyono, Metode Penelitian Kuantitatif, Kualitatif, dan R&D, Bandung: Alfabeta.CV, 2017.

APPENDIX

A. Implementation of ModSecurity-like Web Application Firewall (WAF)

The developed WAF written in Python (modsec.py) protects against SQL Injection (SQLi) and Cross-Site Scripting (XSS) and NoSQL Injection attacks. The system operates through real-time payload blocking by assessing incoming HTTP requests as well as URL parameters and headers together with request bodies for malicious content.

B. Code Implementation of Security Inspection Module

Regular expressions (regex) function within the security system for detecting attack patterns that exist in URLs headers and request bodies. The main execution code of modsec.py appears below:
Modsec file

C. Working Mechanism of the Security Module

The ModSecurity-like system uses an organized sequence to identify threats in HTTP requests that enter the system.

1) URL Inspection: The system uses programmed scanners to detect SQLi and XSS patterns and blocks attempts from dangerous entry points.
2) Header Analysis: Review the headers to identify any unusual values.
3) Request Body Inspection: Analyzes JSON data to identify information.
- SQL Injection (' OR 1=1 --)
- XSS (<script>alert('XSS') </script>)
- NoSQL Injection ({"$ne": "admin"})

  Non - JSON bodies are also scanned for threats.

4) Attack Response Handling: The system prevents dangerous requests from passing through but let valid ones reach the target system.

D. Example Attack Simulation and Detection

These CURL commands enabled us to perform attacks on the system for testing purposes.

1. SQL Injection Attack Test

```
curl -X POST "http://127.0.0.1:8080/api/data" -H "Content-Type: application/json" -d "{\"name\": \"' OR 1=1 --\", \"age\": 30}"
```

Expected Response:

```
{"error":"Malicious body detected (SQLI attack in JSON)"}
```

2. Cross-Site Scripting (XSS) Attack Test

```
curl -X POST "http://127.0.0.1:8080/api/data" -H "Content-Type: application/json" -d "{\"name\": \"<script>alert('XSS')</script>\", \"age\": 25}"
```

Expected Response:

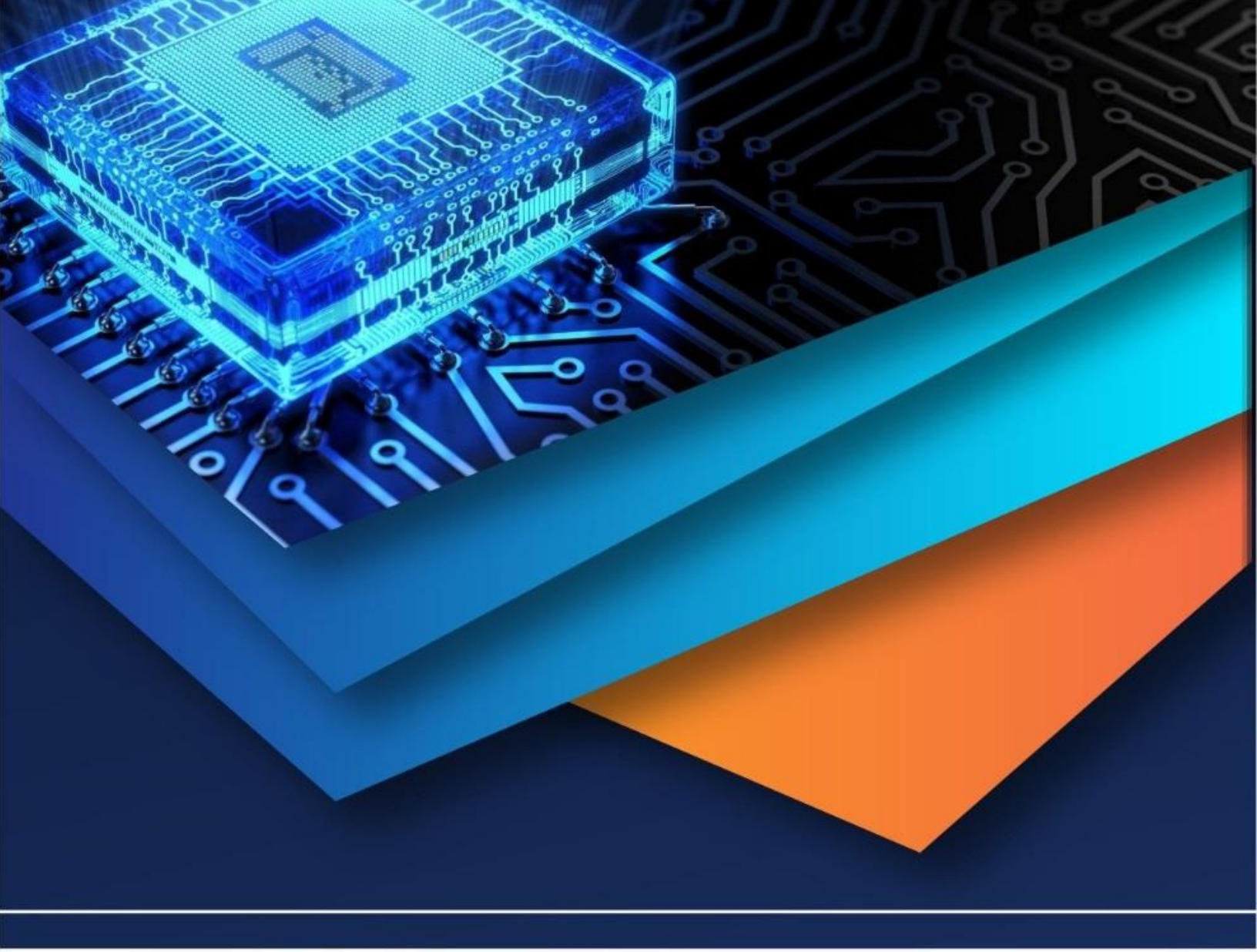
```
{"error":"Malicious body detected (XSS attack in JSON)"}
```

3. NoSQL Injection Attack Test

```
curl -X POST "http://127.0.0.1:8080/api/data" -H "Content-Type: application/json" -d "{\"name\": {\"$ne\": \"admin\"}, \"age\": 30}"
```

Expected Response:

```
{"error":"Malicious body detected (NOSQL attack in JSON)"}
```



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)