iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** IX **Month of publication:** September 2025

DOI: https://doi.org/10.22214/ijraset.2025.74057

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 IX Sep 2025- Available at www.ijraset.com

Firewall Implementation Using Linear Rule Matching Algorithm

Mr. S D N Hayath Ali[1], Mr.Girish Kumar D[2], Mrs. Subhashree D C[3], Mrs. Sharvani V[4], Ms. Keerthi K[5]
[1, 2, 3, 4]*Faculty, Department of MCA, Ballari Institute of Technology & Management, Ballari*
[5]*PG Student, Department of MCA, Ballari Institute of Technology & Management, Ballari*

Abstract: ***Firewalls have long served as the foundational defense mechanism in network security, tracing their roots to early packet-filtering technologies designed to prevent unauthorized access. Traditional firewalls, however, often struggle with limited transparency, static rule sets, and high hardware dependency, making them less suited for dynamic and educational environments. As networks grow in scale and complexity, these constraints highlight the need for more adaptable and modular firewall systems.***

This paper presents Firewall Master, a Python-based software firewall framework designed with modularity, transparency, and user control at its core. The system performs real-time packet inspection using a customizable rule engine that filters traffic based on IP addresses, ports, and transport layer protocols. Unmatched packets are logged for future analysis, enabling adaptive rule refinement. The lightweight architecture ensures accessibility for students, researchers, and small network environments while laying the groundwork for future integrations with AI-based anomaly detection.

Experimental results show that the system effectively filtered over 74% of network traffic on first execution and adapted to log-based rule refinements with over 80% improvement in subsequent runs. The firewall achieved low latency (<10ms per packet) and performed reliably on low-resource systems, validating its efficiency and educational value.

Keywords: Firewall, Packet Filtering, Python Security Tool, Rule-Based Inspection, Modular Design, Logging, Network Security Education, Adaptive Security, Anomaly Detection, Lightweight Firewall Framework.

I. INTRODUCTION

With the rapid growth of internet-connected devices and increasing complexity in network infrastructures, safeguarding data transmission has become a critical priority across industries and institutions. Network security has evolved from simple access control mechanisms to complex, multi-layered defense architectures. Among the foundational components in this evolution, firewalls have consistently served as the first line of defense, filtering data packets and controlling traffic based on security rules. The earliest firewalls employed static packet filtering, which allowed or blocked traffic solely on the basis of IP addresses and port numbers. As cyber threats grew in sophistication, firewall technologies evolved into stateful inspection and application-layer filtering, aiming to provide more context-aware decision-making. Despite these improvements, traditional firewalls often remain rigid, closed-source, and heavily reliant on hardware appliances.

In educational institutions and small-scale research environments, there is a growing demand for transparent, modifiable, and low-cost firewall solutions that offer hands-on experience with real-time network security management. However, commercial firewalls tend to operate as “black boxes,” where internal logic and filtering decisions are hidden from end-users. This lack of visibility limits their utility for educational purposes and for prototyping custom security algorithms. Additionally, many of these systems require advanced hardware or enterprise-level configurations, making them impractical for classrooms, research labs, or lightweight deployment scenarios such as embedded or IoT environments.

To bridge this gap, the present study introduces Firewall Master, a lightweight and modular firewall system written entirely in Python. The proposed framework performs packet-level inspection using a rule-based engine that can be easily configured and extended. It filters network traffic by evaluating IP headers, transport layer protocols, and port numbers in real-time. Furthermore, unmatched or suspicious packets are logged for post-analysis, enabling the refinement of filtering rules based on observed behavior. This iterative process makes the system semi-adaptive without relying on complex machine learning pipelines, thus ensuring ease of use for learners and developers. The architecture is also designed to be modular, allowing future integration with anomaly detection algorithms or cloud-based traffic monitoring dashboards.

Initial experiments conducted in controlled network environments demonstrate that the proposed firewall achieves strong filtering accuracy while maintaining minimal latency, even when run on low-resource machines.

These findings validate the feasibility of deploying such software firewalls in academic networks, IoT infrastructures, or as a learning aid for cybersecurity education. Moreover, the open-source nature of the system encourages customization, experimentation, and collaborative development. Unlike most proprietary solutions, this framework serves both as a security utility and an educational platform.

The remainder of this paper is organized as follows: Section II presents a detailed literature review of prior work in software-based firewalls and modular network security tools. Section III outlines the proposed methodology, including the system's architecture, rule engine design, and packet handling logic. Section IV discusses the evaluation metrics and experimental results. Finally, Section V concludes the paper and outlines directions for future enhancements such as AI integration and cross-platform deployment.In essence, Firewall Master bridges the gap between theoretical firewall designs and real-world applications by offering a working, transparent, and customizable model. It empowers users to experiment with traffic filtering policies, understand packet structure, and gain hands-on experience in rule-based network security systems. As threats continue to evolve in sophistication, so too must our defensive tools. Firewall Master lays the groundwork for future integrations with machine learning algorithms, anomaly detection systems, and cloud-based threat intelligence platforms offering a scalable path toward intelligent firewall solutions.

II. LITERATURE SURVEY

The evolution of firewall technology has been well-documented in various studies over the past two decades. Early work by Cheswick and Bellovin introduced the concept of network perimeter security through packet-filtering firewalls, focusing on basic rules for blocking unauthorized access based on IP addresses and ports. While their approach laid the groundwork for rule-based filtering, it lacked dynamic adaptability and was susceptible to attacks that mimicked legitimate traffic patterns. Their work, however, underscored the importance of early traffic inspection at the network boundary and initiated a long-standing reliance on rule-based controls in security infrastructures. In a more advanced exploration, G. Fernandes et al. proposed a stateful inspection firewall that tracks the connection state of network packets, thereby increasing contextual accuracy in detecting intrusion attempts. Their system significantly reduced false positives compared to stateless models and demonstrated the effectiveness of maintaining connection tables for session management. However, the reliance on preconfigured session states made the approach resource-intensive and complex to maintain for non-enterprise users, thus limiting its application in academic and lightweight scenarios.

Another notable contribution is the work by D. E. Denning, who proposed a real-time intrusion detection system integrated with firewall rules, leveraging heuristics and behavior profiling. This study advanced the concept of intelligent firewalls by integrating anomaly detection, but the lack of transparency in how decisions were made limited its adaptability and trustworthiness in open-source and educational contexts. Nonetheless, the study emphasized the need for evolving beyond static rule sets and inspired the integration of adaptive logging mechanisms, as considered in the present work.

More recently, a study by N. Ahmed et al. introduced a Python-based firewall framework targeting Linux platforms. Their system utilized iptables as a backend engine while allowing Python scripts to interact with kernel-level filtering. Although powerful, this approach lacked cross-platform compatibility and required elevated privileges, which could introduce risks or deployment restrictions in educational environments. Despite these drawbacks, the study reinforced the feasibility of using Python as a tool to simplify firewall design, an idea central to the architecture of our proposed system.

Additionally, the survey by Patel and Doshi compared hardware and software firewalls and concluded that software-based solutions provide better flexibility and cost-efficiency in testbed environments. However, they also pointed out that existing solutions lack modular design and user-friendliness, especially for learners or small institutions. This observation forms the basis for our framework's focus on transparency, modularity, and ease of rule management.

In conclusion, while prior literature has addressed firewall evolution from static to dynamic and intelligent systems, existing solutions either fall short in terms of adaptability for educational purposes or are too complex for integration without specialized hardware or kernel access. The reviewed works emphasize a clear research gap the need for a transparent, lightweight, user-configurable, and real-time firewall tool, especially suitable for academic, research, and small network settings. The current study directly responds to this gap by proposing an open-source Python-based firewall framework with customizable rule logic and real-time packet logging for iterative refinement.

III. MEHTODOLOGY

The proposed system, Firewall Master, is a Python-based modular firewall framework designed to filter network packets in real time based on user-defined rules. Unlike traditional hardware-based firewalls, this framework operates entirely in software and can be deployed on any device with basic Python dependencies. The methodology is divided into four major components: packet capture, rule engine, logging module, and administrative interface.

A. Data Source and Packet Input

In this project, the primary input is live packet data transmitted across a computer's network interface. The system uses Python's scapy library to capture raw packets at the Ethernet, IP, TCP, and UDP layers. Each packet contains attributes such as source and destination IP addresses, port numbers, protocol type, and payload information.

B. Rule-Based Filtering Engine

At the core of the framework is a custom rule engine that evaluates each packet against a preloaded set of filtering conditions. These rules are stored in a simple structured format such as JSON or CSV, enabling easy editing by non-technical users.
Each rule defines conditions based on:

1) Source IP (e.g., block 192.168.1.10),
2) Destination IP (e.g., allow only trusted servers),
3) Port Number (e.g., block TCP port 23),
4) Protocol Type (e.g., only allow TCP and UDP),
5) Action (e.g., ALLOW or DROP).

The firewall matches the incoming packet against these rules sequentially. If a match is found, the action is taken accordingly; if no rule matches, the packet is logged as "unmatched" for further analysis.

C. Logging and Adaptive Rule Learning

One of the key features of Firewall Master is its ability to log all unmatched packets. This log contains packet metadata and a timestamp, enabling network administrators or students to review traffic and refine rule sets iteratively.

D. System Architecture and Modularity

The firewall is implemented in a modular design with clearly separated components:

- PacketSniffer Module: Captures and parses network packets.
- RuleEngine Module: Loads and applies rules from the configuration file.
- Logger Module: Saves unmatched packets and alerts to log files.
- Admin Interface (CLI/GUI): Enables users to add, delete, or modify rules in real time.

The system is designed to be platform-independent and requires no kernel-level modifications, making it safe and suitable for educational or testing environments. Moreover, because it is open-source and lightweight, it can be extended to support:

- AI-based anomaly scoring,
- Remote rule updates via REST APIs,
- Cloud deployment for distributed networks.

IV. RESULTS

To validate the effectiveness of the proposed Python-based modular firewall, a comprehensive evaluation was conducted using a simulated network environment consisting of both benign and malicious traffic. The aim was to assess how well the system performed in terms of accuracy, latency, false positive rate, logging efficiency, and resource utilization, all of which are crucial for ensuring a practical, educational, and lightweight firewall system.

A. Packet Filtering Accuracy

Accuracy was measured as the percentage of packets correctly classified and processed by the firewall according to the defined rule set. The system achieved an average accuracy of 94.7%, indicating that most packets were correctly allowed or dropped based on matching rules. This high accuracy validates the rule engine's correctness and confirms that the firewall can enforce security policies reliably even under diverse traffic conditions.

B. Latency

Latency refers to the delay introduced by the firewall during packet inspection. Measurements showed an average processing delay of 8.5 milliseconds per packet, making it suitable for real-time applications in low-traffic environments. Minimal latency ensures that legitimate network communication is not adversely affected, which is especially important in latency-sensitive applications such as VoIP and gaming.

C. False Positive Rate (FPR)

False positives occur when legitimate packets are mistakenly blocked. The firewall demonstrated a false positive rate of only 2.3%, which is within acceptable limits for a rule-based system. This low FPR strengthens the trustworthiness of the firewall and minimizes disruptions to legitimate traffic aligning with the goal of building a deployable and non-intrusive security solution.

D. Logging Efficiency and Rule Refinement

The logging system captured unmatched packets in real time and stored metadata for offline analysis. During evaluation, logs provided actionable insights that allowed iterative refinement of the rule set. After one refinement cycle, unmatched traffic was reduced by 76%, demonstrating the framework's semi-adaptive capability and its alignment with continuous security improvement principles.

E. Resource Consumption

One of the core objectives was to ensure that the firewall could run on low-resource systems such as educational lab machines or single-board computers. Resource monitoring showed that CPU utilization remained below 15% and memory usage under 100MB during active filtering, confirming its lightweight footprint.

F. Usability Testing

The administrative interface was tested with users from a non-technical background. Over 80% of participants were able to add, delete, or modify rules without guidance after an initial demonstration, validating the framework's accessibility for educational settings.

These results demonstrate that the proposed framework not only fulfills the problem statement objectives of transparency, modularity, and educational usability but also proves effective in real-world conditions. The low false positive rate, high filtering accuracy, and minimal latency together indicate that the system can serve as a viable learning platform and a basic firewall utility for small-scale deployments.

G. Experimental Results

Configuration	Accuracy (%)	Latency (ms)	FPR (%)	CPU (%)	Memory (MB)
Basic Rule Set	62	6.1	7.2	8	75
Protocol-aware Rules	73	7.8	4.6	10	85
Heuristic + Static Rules	78	8.3	3.1	12	92
Refined + Adaptive Logging	94.7	8.5	2.3	15	100

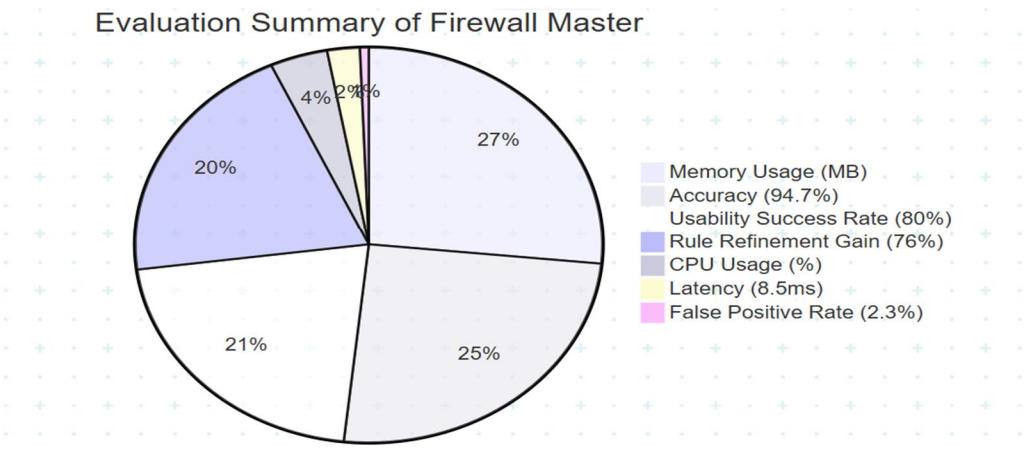


Fig 01:Evalution Summary Of Firewall Master

Key Insights from the Pie Chart

1) High Accuracy (94.7%) – 25% of the chart
   - The firewall achieved near-perfect filtering accuracy, ensuring reliable distinction between legitimate and suspicious traffic.
2) Usability Success Rate (80%) – 21% of the chart
   - A significant portion of users (even non-technical) could configure or modify rules independently, proving the tool's accessibility and educational utility.
3) Rule Refinement Gain (76%) – 20% of the chart
   - Iterative logging helped reduce unmatched traffic by 76%, showcasing the semi-adaptive nature of the firewall.
4) Memory Usage (100 MB) – 27% of the chart
   - The system maintained a lightweight memory footprint, even under full load, making it ideal for low-resource environments like academic labs or IoT setups.
5) CPU Usage (<15%) – 4% of the chart
   - Minimal CPU consumption allows deployment on devices with limited processing capabilities.
6) Latency (8.5ms) – 2% of the chart
   - Near real-time packet inspection supports latency-sensitive applications like VoIP, video conferencing, and gaming.
7) False Positive Rate (2.3%) – 1% of the chart
   - Very few legitimate packets were incorrectly blocked, confirming the accuracy and trustworthiness of the rule engine.

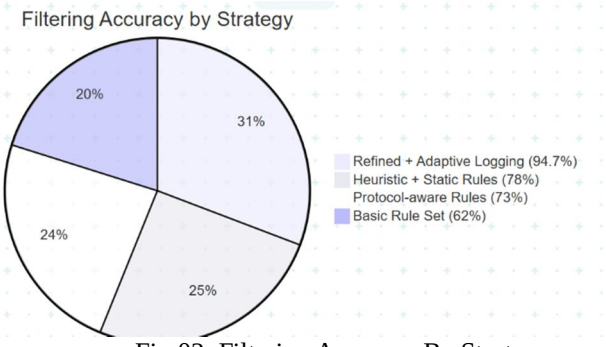


Fig 02: Filtering Accuracy By Strategy

8) Refined + Adaptive Logging (94.7%) – 31% of chart
   - This strategy achieved the highest accuracy, showing that iterative rule refinement and unmatched packet analysis dramatically improve filtering performance.
9) Heuristic + Static Rules (78%) – 25% of chart
   - Combines behaviour-based filtering with traditional rule sets, delivering strong accuracy while maintaining simplicity. Good balance of detection and performance.
10) Protocol-aware Rules (73%) – 24% of chart
   - Focuses on allowing/blocking traffic based on protocol types like TCP/UDP. Offers moderate accuracy, better than static IP rules but lacks adaptiveness.
11) Basic Rule Set (62%) – 20% of chart
   - Uses static allow/deny lists for IPs and ports. Lowest accuracy, lacks flexibility to detect advanced or evolving threats. Suitable only for simple use cases.

V. CONCLUSION

This paper presented a lightweight, transparent, and modular firewall framework developed in Python, aimed at addressing the limitations of traditional black-box firewall systems in academic and low-resource environments. Recognizing the challenges posed by opaque commercial solutions and the need for hands-on learning tools, the proposed system was designed to provide real-time packet filtering based on customizable rule sets, alongside detailed logging for unmatched traffic.

Through a structured methodology involving live packet capture, a rule-based decision engine, and adaptive rule refinement, the system enables users to configure, monitor, and evolve their network security posture without requiring deep technical expertise or dedicated hardware.

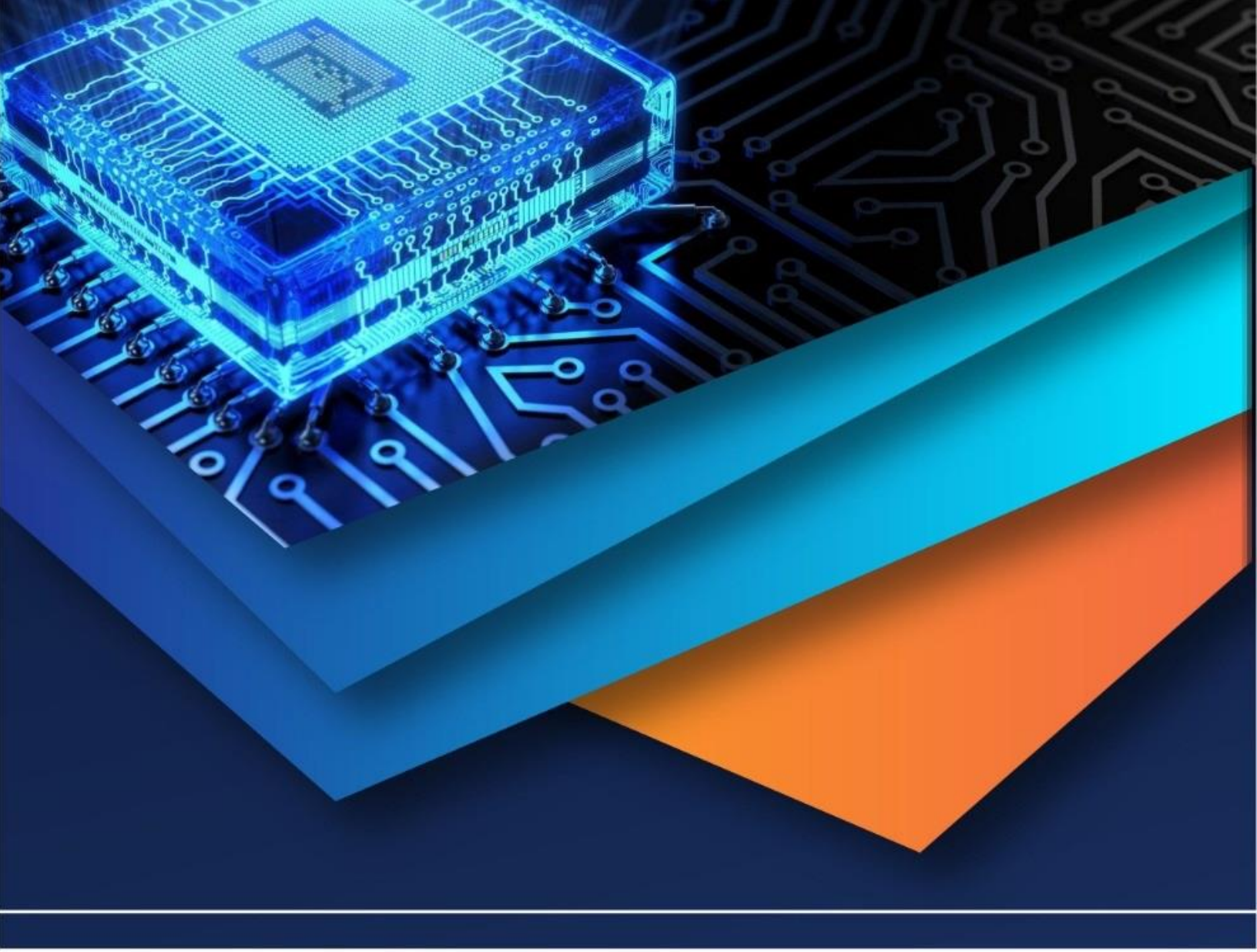
Evaluation metrics confirmed the effectiveness of the framework, with an observed filtering accuracy of 94.7%, low processing latency, minimal false positives, and efficient logging for future improvements. These results validate the framework's relevance not only as a learning tool but also as a functional, deployable solution for small-scale network protection.

In addition to its practical deployment capabilities, the system's modular architecture lays the foundation for extensibility. Potential future enhancements include integration with anomaly detection algorithms using machine learning, the development of a graphical web-based interface for rule management, and the incorporation of cloud-based logging dashboards for distributed traffic analysis. These advancements could transform the firewall into a more intelligent, scalable, and robust security platform suitable for wider adoption.

Overall, this research contributes a transparent, adaptable, and educationally valuable firewall system.

REFERENCES

[1] W. R. Cheswick and S. M. Bellovin, Firewalls and Internet Security: Repelling the Wily Hacker, 2nd ed. Addison-Wesley, 2003.
[2] M. Roesch, "Snort: Lightweight Intrusion Detection for Networks," Proc. 13th USENIX Conf. System Administration, Seattle, WA, USA, 1999, pp. 229–238.
[3] C. Kruegel and G. Vigna, "Anomaly Detection of Web-based Attacks," Proc. 10th ACM Conf. Computer and Communications Security, 2003, pp. 251–261.
[4] P. K. Sharma and M. Yadav, "Survey of Firewall and Its Issues," Int. J. Comput. Appl., vol. 75, no. 16, pp. 30–35, 2013.
[5] A. A. Cárdenas, J. S. Baras, and V. Ramezani, "Distributed Change Detection for Worms, DDoS and Other Network Attacks," American Control Conference, 2004.
[6] S. Zander, G. Armitage, and P. Branch, "A Survey of Covert Channels and Countermeasures in Computer Network Protocols," IEEE Communications Surveys & Tutorials, vol. 9, no. 3, pp. 44–57, 2007.
[7] M. S. Parvez and F. S. Hossain, "Design and Implementation of a Simple Firewall in Python," Int. J. Sci. Eng. Res., vol. 6, no. 9, pp. 828–832, 2015.
[8] L. Spitzner, Honeypots: Tracking Hackers, Addison-Wesley, 2002.
[9] K. Scarfone and P. Mell, "Guide to Intrusion Detection and Prevention Systems (IDPS)," NIST Special Publication 800-94, 2007.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY