REAL-TIME FIREWALL EVENT ALERT SYSTEM FOR NETWORK PROTECTION

KAVIYA S, POOJA S, SHRUTHIKA M

1,2,3. B.sc ISCF students, Dr.M.G.R Educational and Research Institute Deemed to be University, Chennai. Corresponding Email ID: shruthikaa2422@gmail.com

4. Professor, Dr.M.G.R Educational and Research Institute, Deemed to be University , Chennai.

5. Assistant Professor, Dr.M.G.R Educational and Research Institute, Deemed to be University, Chennai.

ABSTRACT

The proposed system addresses the limitations of traditional firewalls by introducing real-time alert functionality using Django and Python. It monitors firewall logs to detect suspicious activities like brute-force login attempts, port scans, and large data transfers. Alerts are generated immediately and sent via email, ensuring quick response. The system includes a web dashboard for admin visibility and uses SQLite for lightweight storage. It is affordable, scalable, and suitable for small organizations and educational institutions. By automating detection and alerts, it minimizes manual workload and enhances network security posture. This project showcases the effectiveness of open-source tools in building real-time cybersecurity systems

INTRODUCTION

The modern digital era is characterized by unprecedented connectivity and dependence on internet-based services. As a result, cybersecurity has become a fundamental concern for individuals, businesses, and institutions alike.

Manual inspection of firewall logs is not only tedious and time-consuming but also susceptible to human error. The delay in recognizing threats can have disastrous consequences, especially in cases of fast-moving attacks such as brute-force login attempts or data exfiltration operations. Therefore, the need for a real-time, intelligent alerting system has become critical in the current cybersecurity landscape.

The proposed system addresses these concerns by offering a Django-based real-time firewall alert platform that actively monitors logs and generates alerts based on suspicious behavior patterns. It enables administrators to take immediate action by alerting them via both a web dashboard and email notifications. The integration of automation in threat detection significantly reduces the workload on system administrators and improves the organization’s ability to respond to security incidents promptly.

This paper explores the design, implementation, and impact of such a system, focusing on its feasibility, scalability, and usability. By leveraging widely-used open-source technologies, the project ensures that cost and accessibility barriers are minimized, making advanced network security monitoring viable for a broader audience

2. LITERATURE SURVEY

[1] A. K. Sharma and R. Verma, "An Automated Real-Time Intrusion Detection System for Network Security," in Proc. 9th Int. Conf. on Cyber Security and Protection

of Digital Services (Cyber Security), pp. 56–61, Apr. 2021.

Abstract: This paper presents a real-time Intrusion Detection System (IDS) that analyzes firewall logs using rule-based mechanisms. The system incorporates pattern matching and anomaly detection algorithms to identify malicious activity. Emphasis is placed on real-time detection and the use of automated alerting via email and SMS. The study evaluates the performance of the IDS in a controlled environment and demonstrates its effectiveness in mitigating brute-force and port-scan attacks.

[2] S. R. Iyer and M. T. Thomas, "Enhancing Network Security through Log-Based Threat Detection Using Open Source Tools," Int. J. Comput. Sci. Netw. Secur., vol. 20, no. 9, pp. 145–152, Sep. 2020.

Abstract: The research focuses on using open-source platforms like Snort and Fail2Ban to analyze system and firewall logs for suspicious activities. It discusses how custom rule sets can be created and applied for effective monitoring. The study evaluates the scalability and usability of these tools in small to mid-sized networks. The limitations of manual configuration and lack of GUI are also examined, highlighting the need for web-based log management solutions.

[3] P. Kumar and S. Mehta, "Design and Implementation of a Web-Based Security Alert System for Enterprise Networks," Int. J. Inf. Syst. Appl., vol. 14, no. 4, pp. 23–29, Dec. 2022.

Abstract: This paper explores the design of a web-based alert system built on the Django framework for real-time monitoring of network activity. The system collects firewall data, stores it in a structured database, and applies detection logic to generate alerts. It emphasizes usability, modularity, and integration with email servers for quick notification. The implementation showed that such systems could be deployed in non-enterprise environments with minimal resource requirements.

[4] M. A. Hussain and L. Singh, "Real-Time Network Monitoring and Alert Generation using Lightweight Web Technologies," in Int. Conf. on Advances in Computing, Communication and Control (ICAC3), pp. 102–108, Jul. 2023.

Abstract: This paper introduces a lightweight real-time monitoring system that leverages web technologies such as Python, Flask, and SQLite for continuous tracking of network events. The system features a simple web interface and supports customizable alert logic, including failed login attempts and unauthorized port access. The authors demonstrate how such systems can be effectively deployed in educational and SME networks, emphasizing low cost, portability, and extensibility. The paper concludes by comparing the system's performance with more complex SIEM solutions, showing comparable detection accuracy at a fraction of the resource usage.

3. EXISTING SYSTEM

Most firewall monitoring systems act after an attack occurs, rather than preventing it in real time. Administrators are required to analyze firewall logs manually, which consumes time and heightens the chance of overlooked threats. Because firewalls create a huge amount of log data, it is hard to identify real threats among normal events without an automated filtering system.

Most current systems lack instant alerting capabilities, hence administrators are not immediately alerted when something unusual occurs. There are also systems lacking a web interface, implying that monitoring can be accomplished using specific computers only, resulting in delays within cases of emergencies or remote work.

DRAWBACKS

- The system relies on predefined rules, limiting its ability to detect new and unknown attack patterns.

- Real-time monitoring may lead to high system resource consumption, especially in large networks.
- False positives can occur, leading to unnecessary alerts and potential alert fatigue for administrators.
- It does not prevent attacks but only alerts administrators, requiring manual intervention for mitigation.
- The system requires continuous updates and maintenance to remain effective against evolving threats

4. PROPOSED SYSTEM

The suggested system overcomes the shortcomings of current solutions through an intelligent real-time alert mechanism. It incorporates various components in a web-based dashboard, where administrators can view network activity in an efficient manner. The system continually monitors firewall logs and provides alerts based on predetermined security rule.

Developed with Django for backend processing and SQLite for light data storage, the system provides fast and dependable performance. Django processes requests, sessions, and dynamic content, and administrators can safely log in to see real-time alerts whenever there are suspicious activities.

The system is scalable for future upgrades, including email/SMS notifications, machine learning-based threat detection, and support for varied firewalls. With its easy, responsive interface and low setup needs, it provides an efficient cybersecurity tool without expensive investments.

ADVANTAGES

- Provides real-time alerts, enabling quick response to potential security threats.
- Automates firewall log analysis, reducing manual workload and improving efficiency.
- Enhances network security by identifying suspicious activities based on predefined rules.
- Helps organizations comply with security policies by ensuring continuous monitoring and logging.

5. REQUIREMENT SPECIFICATIONS

HARDWARE REQUIREMENTS

- Processor: Intel i3 or above

- RAM: 4 GB minimum

- Hard Disk: 500 GB or more

- Network Interface Card (NIC)

SOFTWARE REQUIREMENTS

A django is a high-level Python web framework that simplifies the development of secure and scalable web applications. It follows the Model-ViewTemplate (MVT) architecture, making it easier to manage database interactions, business logic, and user interfaces separately.

Key Features:

- Real-time Log Monitoring – Fetches and processes firewall logs dynamically
- Web-Based Dashboard – User-friendly interface for viewing alerts
- Automated Alerts – Instant notifications via email (SMTP integration)
- Custom Security Rules – Configurable detection for suspicious activities

- Scalability – Can be extended with AI-based threat detection

6. IMPLEMENTATION

STEP 1: Setting up Django: Initializing Django, configuring URLs, settings, and models.

STEP 2: Admin Interface: Creating secure login with password validation.

STEP 3: Log Parser: A Python script to read and parse firewall logs.

STEP 4: Views and Templates: HTML templates render data fetched from the database dynamically.

STEP 5: Real-Time Updates: Alerts are shown on the dashboard when logs meet predefined criteria.

STEP 6: Database Integration: SQLite stores all fetched logs with time-stamps. This lightweight system can be deployed on any Linux/Windows machine and provides a user-friendly solution for real-time firewall monitoring.

7. MODULES DESCRIPTION

Admin Login Module: This module ensures that only authorized users can access the system. It employs secure authentication using Django's built-in user authentication framework, including password hashing and session management. It is the gateway to the dashboard and protects the system from unauthorized access.

Log Fetching Module: This module reads log entries from the firewall and parses them in realtime. The log files are continuously scanned, and relevant details such as IP addresses, ports, and access types are extracted. This module is crucial in detecting potential security threats based on access patterns or irregular behaviors.

Alert Generation Module: The heart of the system, this module analyzes parsed log data and checks for entries that match predefined suspicious patterns or anomalies—like repeated failed access attempts or blacklisted IPs. When such entries are detected, the module generates real-time alerts and stores them in the database for immediate admin visibility.

Dashboard Module: This front-end module is built with HTML, CSS, and Django templates. It dynamically displays logs, alerts, and user information in an intuitive and organized layout. Real-time updates are reflected on the dashboard, and colorcoded alerts help administrators quickly assess the severity of threats.

Database Module: This module ensures all events and alerts are stored efficiently using SQLite. Each log entry is timestamped and saved in an organized structure to allow easy querying, reporting, and future reference. The database acts as the system's memory and supports audit trails. These modules work together as a cohesive unit to provide a secure, automated, and responsive alert system for network protection

8. RESULT

- **Real-time Detection**: The system successfully detected predefined threats like brute-force login attempts, port scans, and large data transfers in real time.
- **Instant Alerts**: Email alerts were generated and sent immediately upon detection of suspicious activities, enabling quick administrator response.
- **Dashboard Efficiency**: The web-based dashboard updated dynamically, displaying logs and alerts with timestamps for clear visibility and faster decision-making.

- **Secure Access**: Admin access was securely managed using Django’s authentication system, ensuring only authorized users could view or manage alerts.
- **Low Resource Usage**: The system ran smoothly on basic hardware configurations (Intel i3, 4GB RAM), proving it to be lightweight and efficient.
- **Automation of Log Monitoring**: Automated parsing of firewall logs eliminated the need for manual log analysis, reducing human error and workload.
- **Accuracy of Detection Rules**: The predefined rules accurately identified specific suspicious behaviors without missing significant events.
- **Cross-platform Compatibility**: Tested on both Windows and Linux, the system functioned effectively across different operating environments.
- **Scalability**: The modular architecture allowed for easy upgrades, such as future ML integration or SMS alerts via APIs.
- **Suitability for SMEs and Education**: The system proved ideal for educational institutions and small businesses that need strong, real-time security without enterprise-level costs.

9. CONCLUSION

The real-time firewall event alert system presented in this study addresses a critical need in cybersecurity: the capability to monitor and respond to threats as they happen. Traditional firewalls, while essential, fall short in providing timely insights or actionable intelligence. Our system bridges this gap through the use of Django and Python to automate log analysis, detect anomalies, and deliver immediate notifications to administrators.

The project showcases how a lightweight, modular architecture can deliver high-value functionality with low overhead. Its successful detection of failed logins, port scanning, and large file transfers demonstrates its utility in real-world network scenarios. Moreover, the system's user-friendly interface and web-based dashboard make it accessible to non-expert users, further expanding its usability across diverse organizational contexts.

From an economic standpoint, the reliance on open-source tools like Django, Python, and SQLite ensures cost-effectiveness. This makes the system particularly attractive to institutions and small businesses that might otherwise lack the resources to invest in commercial security solutions.

Looking ahead, the system lays the groundwork for future enhancements such as machine learning integration for dynamic threat detection, multi-device alerting mechanisms, and API support for external intelligence sources. These additions would further transform the tool from a rule-based alerting system into a smart, adaptive cybersecurity platform.

In essence, this project not only contributes to the field of intrusion detection and real-time alerting but also sets a precedent for building practical, scalable, and cost-efficient security systems using modern web technologies.