Matrik: Jurnal Manajemen, Teknik Informatika, dan Rekayasa Komputer
Vol. 25, No. 1, November 2025, pp. 173∼188
ISSN: 2476-9843, accredited by Kemenristekdikti, Decree No: 10/C/C3/DT.05.00/2025
DOI: 10.30812/matrik.v25i1.5610

Cyber Threat Detection and Automated Response Using Wazuh and Telegram API

Yuri Ariyanto, Yan Watequlis Syaifudin, M. Hasyim Ratsanjani, Ali Ridho Muladawila, Triana Fatmawati, Pramana Yoga Saputra, Chandrasena Setiadi
Politeknik Negeri Malang, Malang, Indonesia

Article Info

Article history:

Received August 27, 2025
Revised October 05, 2025
Accepted November 17, 2025

Keywords:

Cyber Threat Detection;
Network Security;
PPDIOO;
SIEM;
Wazuh.

ABSTRACT

Cyber threats are becoming more widespread, notably those that use SSH to brute-force their way in or engage in Distributed Denial of Service attacks. These attacks can make networked systems very hard to reach, keep their data safe, and protect their privacy, especially for small and medium-sized organizations that can't afford pricey professional security solutions. This research aims to develop an automated, cost-effective, and scalable cyber threat detection and response system for small and medium-sized organizations unable to afford commercial-grade security solutions. The methodology follows the structured Prepare, Plan, Design, Implement, Operate, Optimize lifecycle, leveraging open-source technologies, primarily the Wazuh Security Information and Event Management platform, augmented with custom detection rules and a Random Forest-based classification module to distinguish Normal, Brute Force, and Distributed Denial of Service traffic patterns. Experimental results demonstrate a Mean Time to Detect of 4.7 seconds for Brute Force and 7.3 seconds for Distributed Denial of Service, with a Mean Time to Respond of 8.2 seconds and under 10 seconds, respectively. The system achieved 98.4% detection accuracy and a 1.5% false positive rate across 100 controlled tests using THC Hydra and slowhttptest. Integration of Wazuh dashboard analytics with real-time Telegram alerts enhances situational awareness and enables prompt, automated incident response, validating open-source frameworks as viable defenses in resource-constrained environments.

Corresponding Author:

Yuri Ariyanto, +628563302542,
Department of Information Technology,
Politeknik Negeri Malang, Malang, Indonesia,
Email: yuri@polinema.ac.id.

How to Cite:
Y. Ariyanto, "Cyber Threat Detection and Automated Response Using Wazuh and Telegram API", *MATRIK: Jurnal Manajemen, Teknik Informatika, dan Rekayasa Komputer*, Vol. 25, No. 1, pp. 173-188, November, 2025.

1. INTRODUCTION

Cyber threats are becoming increasingly sophisticated and prevalent, particularly SSH-based brute-force and distributed denial-of-service (DDoS) attacks. These assaults are particularly bad for the security, privacy, and availability of modern network infrastructures [1, 2]. These attacks often allow unauthorized users to access systems, cause lengthy service interruptions, and steal private information. This costs a lot of money, gets people in trouble with the law, and destroys their reputations [3]. Perimeter firewalls and antivirus systems that rely on signatures are becoming less and less useful at finding and stopping threats that change all the time and are hard to detect, especially in places where there isn't much security knowledge or money [4]. In this case, Security Information and Event Management (SIEM) systems are needed to link logs in real time, discover strange behavior, and plan how to respond to occurrences [5]. Wazuh is now a powerful and flexible open-source SIEM platform. It offers built-in tools for host-based intrusion detection (HIDS), file integrity monitoring, vulnerability evaluation, and automating active responses [6]. It is a wonderful solution for small and medium-sized enterprises (SMEs) that don't have the money to buy pricey commercial SIEM suites but are still prime targets for opportunistic cyberattacks. This is because it has a modular design and works with numerous types of systems [7]. But the operational efficiency of any SIEM system depends not only on how efficiently it analyzes data, but also on how well it combines with other security technologies, such as Fail2Ban for banning dynamic IP addresses and real-time notification channels like the Telegram API. This is to cut down on the time it takes to find and address threats and the need for people to get involved [8]. This connection is especially crucial for small and medium-sized organizations (SMEs) because they don't have the means to respond fast to issues, which might lead to a breach of the whole system.

This research contributes to the field by proposing and implementing an integrated, automated cybersecurity framework that combines Wazuh with Fail2Ban for dynamic IP blocking and Telegram API for instant administrative notifications [9, 10]. The system is developed following the PPDIOO (Prepare, Plan, Design, Implement, Operate, Optimize) methodology, ensuring a structured and industry-aligned development lifecycle [11]. The integration of Fail2Ban enhances the system's active response capability by automatically blocking malicious IP addresses after repeated failed authentication attempts, while the Telegram API ensures that security alerts are delivered promptly to administrators, enabling faster decision-making [12, 13]. Furthermore, a machine learning-based attack classification module is incorporated to categorize threats into Normal, Brute Force, and DDoS categories, thereby improving the interpretability and prioritization of alerts [14, 15]. This study not only demonstrates the technical feasibility of such an integration but also evaluates its performance in terms of Mean Time to Detect (MTTD) and Mean Time to Respond (MTTR), providing empirical evidence of its effectiveness in mitigating common cyber threats [16, 17].

Some gaps have not been resolved by previous research, namely the limited integration of real-time automated response mechanisms with actionable alerting in open-source SIEM systems tailored for resource-constrained environments, particularly small and medium-sized organizations [18–20]. The difference between this research and the previous one is that it combines Wazuh's detection capabilities with Fail2Ban for dynamic IP blocking and Telegram API for instant, human-readable notifications within a structured PPDIOO lifecycle, further enhanced by a machine learning–based classifier to distinguish Normal, Brute Force, and DDoS traffic patterns, thereby improving detection precision and reducing false positives. This study explicitly aims to deliver a cost-effective, modular, and operationally efficient cybersecurity framework that bridges the gap between academic threat detection models and practical deployment in real-world infrastructures. Its contribution lies in demonstrating that open-source tools, when cohesively engineered, can accurately estimate MTTD and MTTR while maintaining high accuracy and low false-positive rates, offering both scientific value in SIEM optimization and practical benefit for SMEs lacking commercial-grade defenses.

The organization of this journal follows a logical structure: Section 1 introduces the cybersecurity challenges and the role of SIEM systems, highlighting the research objectives and contributions. Section 2 presents the research methodology, including system architecture, component design, and implementation phases based on the PPDIOO framework. Section 3 discusses the results and analysis, focusing on detection accuracy, response automation, and notification reliability. Section 4 provides a comprehensive discussion of the findings, comparative analysis with existing solutions, and practical implications. Finally, Section 5 concludes the study with key findings and recommendations for future enhancements.

2. RESEARCH METHOD

This research adopts a systematic systems engineering approach, leveraging the PPDIOO (Prepare, Plan, Design, Implement, Operate, Optimize) methodology developed by Cisco [5]. The PPDIOO framework is chosen due to its iterative and comprehensive nature, making it highly relevant for developing network security systems based on Security Information and Event Management (SIEM). This methodology ensures that planning, design, implementation, operation, and optimization are conducted in a structured manner, ultimately leading to an effective and sustainable solution, as illustrated in Figure 1.

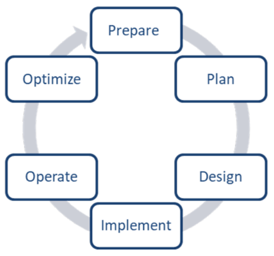


Figure 1. Ppdioo methodology

The PPDIOO methodology serves as the systematic framework guiding the development of the SIEM-based network security system [21–23]. Each phase of the methodology plays a critical role in ensuring the robustness and effectiveness of the system:

2.1. Prepare

The preparation phase begins with identifying the need for a responsive threat detection system, driven by observations of increasing incidents of cyberattacks such as Brute Force and Distributed Denial of Service (DDoS) in public server environments. An initial analysis of potential attack vectors is conducted to define the scope of the research, the system's objectives, and relevant supporting technologies. This foundational step ensures that the subsequent phases are grounded in real-world requirements and challenges. Key Activities in the Prepare Phase:

a. Identification of Cybersecurity Needs: Based on observed trends in cyber threats, particularly Brute Force and DDoS attacks, the research team identified the necessity for a proactive and automated response system.
b. Initial Threat Analysis: A preliminary assessment of potential attack vectors was conducted to determine the types of threats the system would need to detect and respond to effectively.
c. Technology Selection Criteria: Key criteria were established for selecting open-source technologies that could provide real-time monitoring, intrusion detection, and automated response capabilities.

The preparation phase marks the foundational stage of the research, initiated with the identification of a critical need for a responsive threat detection system. This need arises from the observed surge in cyberattacks, particularly Brute Force and Distributed Denial of Service (DDoS), targeting public server environments. These attacks pose significant risks to system availability, integrity, and confidentiality, necessitating a proactive and automated defense mechanism. A preliminary threat analysis was conducted to map potential attack vectors, focusing on common exploitation techniques such as repeated authentication attempts and network resource exhaustion. This analysis informed the scope of the research, which is centered on developing a real-time, open-source-based Security Information and Event Management (SIEM) system capable of detecting, alerting, and responding to these threats. Based on these findings, clear technology selection criteria were established, emphasizing open-source solutions with robust log analysis, active response, and real-time notification capabilities. The outcome of this phase is a well-defined problem statement and a set of functional requirements that guide subsequent planning and design, ensuring the system's relevance and applicability to real-world cybersecurity challenges, as described in Table 1.

Table 1. Key Activities and Outcomes in the Prepare Phase

Activity	Description	Outcome
Identification of Cybersecurity Needs	Analysis of rising cyberattack trends on public servers.	Recognition of the necessity for an automated threat detection and response system.
Initial Threat Analysis	Assessment of attack vectors, such as Brute-Force and DDoS.	Definition of the primary threat types the system must address.
Technology Selection Criteria	Establishment of criteria for open-source tools (real-time monitoring, active response, scalability).	A framework for selecting Wazuh, Fail2Ban, and Telegram API in the planning phase.

2.2. Plan

In the planning phase, the architectural framework and implementation strategy for the integrated security system were meticulously defined to ensure technical coherence and operational effectiveness. The core technological components selected for deployment include Wazuh, an open-source Security Information and Event Management (SIEM) platform renowned for its capabilities in log-based intrusion detection and real-time system behavior analysis; Fail2Ban, a robust utility designed to automatically block malicious IP addresses following patterns of repeated authentication failures; and the Telegram API, which serves as a real-time notification channel to ensure administrators receive critical security alerts promptly and can initiate immediate response actions. This strategic selection of components was guided by the need for interoperability, scalability, and real-time responsiveness, forming the foundational triad upon which the automated detection and response pipeline is constructed.

Additionally, a testing plan was developed, incorporating simulations of brute-force attacks with THC Hydra and DDoS attacks with tools such as hping3 and slowhttptest. Integration schemes between components and performance evaluation methods were also outlined during this phase. Key Activities in the Plan Phase:

a. Technology Selection: Wazuh was chosen as the primary SIEM platform due to its ability to integrate various security modules, including log-based detection, active response mechanisms, and notifications. Fail2Ban was selected for its ability to automatically block malicious IP addresses, while the Telegram API provided real-time alerting.
b. Testing Strategy Development: Simulations of Brute Force and DDoS attacks were planned to evaluate the system's detection and response capabilities under realistic conditions.
c. Integration Planning: Detailed plans were created to ensure seamless integration between Wazuh, Fail2Ban, and Telegram API, forming a cohesive security pipeline.

The planning phase established a comprehensive testing strategy to evaluate the system's efficacy against realistic cyber threats, specifically Brute Force and Distributed Denial of Service (DDoS) attacks. Brute Force attacks on SSH were simulated using THC Hydra with a dictionary-based approach (hydra -L username.txt -P password.txt ssh://[TARGET_IP]). DDoS simulations employed hping3 for a SYN flood attack (hping3 -S -p 80 –flood [TARGET_IP]) and slowhttptest for a Slowloris attack, effectively replicating credential-guessing and resource-exhaustion scenarios. A detailed test plan was developed to ensure reproducibility, defining attack parameters and success criteria. The integration of Wazuh, Fail2Ban, and Telegram API was designed to form a cohesive pipeline, with performance to be assessed using Mean Time to Detect (MTTD) and Mean Time to Respond (MTTR) metrics, as described in Table 2.

Table 2. Attack Simulation Tools and Commands

Attack	Tool	Command
Brute Force (SSH)	THC Hydra	hydra -L username.txt -P password.txt ssh://[TARGET_IP]
DDoS (SYN Flood)	hping3	hping3 -S -p 80 –flood [TARGET_IP]
DDoS (Slowloris)	slowhttptest	slowhttptest -c 1000 -H -g -o slowhttp -i 10 -r 200 -t GET -u http://[TARGET_IP] -x 24 -p 3

2.3. Design

The design phase focused on creating an integrated architecture that included:

a. Wazuh Server: Serving as the central hub for collecting, analyzing, and storing security logs.
b. Wazuh Agent: Installed on endpoints to forward log data to the Wazuh Server.
c. Dashboard: Providing a visual interface for monitoring and analyzing security events.
d. Fail2Ban Integration: Automatically blocking IP addresses detected as malicious by Wazuh.
e. Telegram Bot: Sending real-time notifications to administrators via Telegram API.

The system architecture was designed to enable secure, encrypted, and centralized communication among all components, ensuring data integrity and real-time information flow. A key aspect of the design phase was integrating a machine learning-based classification module to improve the accuracy and reliability of threat detection. To address Brute Force and DDoS attack identification, a supervised learning approach was adopted using the Random Forest algorithm, which is widely recognized for its effectiveness in network intrusion detection due to its ability to handle high-dimensional data, resist overfitting, and provide interpretable feature importance. The model was trained on a labeled dataset derived from system logs collected during controlled attack simulations, comprising normal traffic, SSH-based Brute-Force attempts (simulated using THC Hydra), and application-layer DDoS patterns (generated using slowhttptest). Features such as failed login frequency, source IP connection rate, session duration, and request entropy were extracted and used as input vectors. The dataset was split into 70% for training and 30% for testing, with cross-validation applied to ensure model robustness. Once trained, the classifier was integrated into the detection pipeline to categorize incoming security events into "Normal," "Brute Force," or "DDoS" classes in real time. This classification mechanism not only enhanced alert precision but also reduced false positives, enabling faster event correlation and supporting the system's ability to achieve low Mean Time to Detect (MTTD) and Mean Time to Respond (MTTR). By combining rule-based detection in Wazuh with machine-learning-driven analysis, the system achieved more adaptive, intelligent responses, facilitating timely mitigation and post-incident evaluation.

To quantitatively evaluate system performance and enable intelligent threat classification, formalized metrics and a machine learning model were integrated into the design. The MTTD is computed as the average interval between the initiation of an attack and the generation of the first corresponding alert, as in 1.

$$MTTD = \frac{1}{N}\sum_{i=1}^{N}(t_{detect,i} - t_{attact\_start,i}) \tag{1}$$

Where N denotes the number of test iterations, $t_{detect,i}$ is the detection timestamp, and $t_{attact\_start,i}$ is the attack onset time. Similarly, the MTTR is defined as the average delay between detection and the execution of an automated mitigation action, as in (2).

$$MTTD = \frac{1}{N}\sum_{i=1}^{N}(t_{respond,i} - t_{detect,i}) \tag{2}$$

For threat categorization, a Random Forest classifier was implemented to distinguish among Normal, Brute Force, and DDoS traffic patterns. Given an input feature vector x for example, failed login frequency, source IP connection rate, and request entropy, the ensemble model aggregates predictions from K decision trees $T_1, T_2, \ldots, T_K$ trained using bootstrap sampling, and uses majority vote to choose the final class, as in (3).

$$y = argmax_{ec(normal,BruteForce,DDos)}\sum_{k=1}^{K}(T_k(x) - c) \tag{3}$$

2.4. Implement

The implementation phase involved installing and configuring all system components in accordance with the designed architecture. Verification of functionalities, including log collection, threat detection based on custom rulesets, automatic IP blocking by Fail2Ban, and real-time notifications via Telegram, was conducted. Simulated attacks were executed to test the system's resilience and record its responses. Key Activities in the Implement Phase:

a. Component Installation and Configuration: The Wazuh Manager and Wazuh Agents were installed and configured across the target environment. Custom rulesets were added to the local_rules.xml file to detect specific patterns of Brute Force and DDoS attacks.
b. Integration Testing: The integration between Wazuh, Fail2Ban, and Telegram API was rigorously tested to ensure smooth operation. For example, simulated brute force attacks using THC Hydra were monitored in real-time by Wazuh, and corresponding alerts were sent to administrators via Telegram.
c. Performance Validation: The system's ability to detect and respond to simulated attacks was validated through controlled experiments, demonstrating its effectiveness in handling both Brute Force and DDoS scenarios.

2.5. Operate

During the operational phase, the system was run in a live environment to continuously monitor security activities. Logs were collected and displayed in graphical and tabular formats on the Wazuh dashboard, and real-time notifications were sent to administrators whenever suspicious activity was detected. Monitoring ensured the system operated optimally, responded quickly, and did not degrade server performance. Key Activities in the Operate Phase:

a. Real-Time Monitoring: The Wazuh dashboard provided administrators with a comprehensive view of security events, allowing them to identify and address potential threats promptly.
b. Notification Delivery: Telegram API ensured that critical alerts were delivered instantly to administrators, enabling faster incident response times.
c. Performance Oversight: Continuous monitoring helped maintain system stability and efficiency, ensuring that the implemented solution remained effective over time.

2.6. Optimize

The final phase, optimization, involved evaluating system performance using key metrics such as Mean Time to Detect (MTTD), Mean Time to Respond (MTTR), detection accuracy, and inter-component communication efficiency. Feedback from these evaluations was used to refine the system's configuration and improve its overall effectiveness. Key Activities in the Optimize Phase:

a. Performance Metrics Evaluation: MTTD and MTTR were measured to assess how quickly the system detected and responded to threats. Detection accuracy was evaluated using machine learning models, and communication efficiency was analyzed to ensure seamless integration between components.
b. Iterative Improvement: Based on the evaluation results, adjustments were made to the rule set, classification parameters, and technical infrastructure to enhance system performance in future iterations.
c. Validation Process: Quantitative and qualitative analyses were conducted to validate the system's effectiveness. Quantitative analysis included metrics such as MTTD, MTTR, and detection accuracy, while qualitative analysis focused on subjective factors such as dashboard usability, alert clarity, and the effectiveness of Telegram notifications.

To validate the system's effectiveness, several adjustments were made based on the evaluation results. These adjustments included refining the detection rule set, optimizing classification parameters, and enhancing technical infrastructure. The validation process utilized both quantitative and qualitative approaches.

Quantitative Analysis:

a. Mean Time to Detect (MTTD): Measured the time taken by the system to detect an attack after it began.
b. Mean Time to Respond (MTTR): Evaluated the time required for the system to take action after detecting an attack.
c. Detection Accuracy: Assessed the system's ability to classify different types of attacks correctly.
d. IP Blocking Effectiveness: Verified the efficiency of Fail2Ban in automatically blocking malicious IP addresses.

Quantitative Analysis:

a. Dashboard Usability: Evaluated the ease of use and intuitiveness of the Wazuh dashboard for monitoring and managing security events.
b. Alert Clarity: Assessed the comprehensibility and relevance of alert messages generated by the system.
c. Telegram Notification Efficacy: Determined whether real-time notifications via Telegram were timely and actionable for administrators.

The validation results were obtained from a carefully planned experimental protocol that objectively tested whether the system met its main research goals: to create an open-source, modular, and responsive cybersecurity framework for small- to medium-sized infrastructure. The evaluation used a mixed-methods framework, combining quantitative performance metrics and replicable attack simulations. In particular, 100 controlled test iterations were carried out: 50 for SSH-based Brute-Force attacks using THC Hydra and 50 for application-layer DDoS attacks using slowhttptest under uniform network conditions to ensure reliability.

The PPDIOO (Prepare, Plan, Design, Implement, Operate, Optimize) methodology was applied systematically to create a robust framework for the entire development lifecycle. This made sure that theoretical security principles and practical implementation were in sync. The methodology improved the technical coherence and operational validity of the solution by organizing each phase, from threat modeling and component selection to deployment, monitoring, and iterative refinement. The important thing is that Wazuh served as the main SIEM platform, Fail2Ban for automated IP-based mitigation, and the Telegram API for real-time alerting. This showed that open-source tools can be combined to build a robust, high-performance defense pipeline. This method not only meets the study's goal of providing a cost-effective, scalable security architecture for environments with limited resources but also creates a framework that can be reused for future open-source SIEM deployments, especially when quick detection, automated response, and administrator situational awareness are key.

3. RESULT AND ANALYSIS

The findings of this research are that the proposed integrated cyber threat detection and response system, comprising Wazuh as the core SIEM platform, Fail2Ban for dynamic IP blocking, and the Telegram API for real-time alerting, achieves high detection accuracy and sub-10-second response times against SSH-based Brute Force and application-layer DDoS attacks. Specifically, the system demonstrated an MTTD of 4.7 seconds for Brute Force attacks and 7.3 seconds for DDoS attacks, with corresponding MTTR values of 8.2 seconds and under 10 seconds, respectively. Furthermore, it achieved an overall detection accuracy of 98.4% and a false-positive rate of only 1.5% across 100 controlled test iterations, confirming its operational reliability and precision in real-world threat scenarios. The results of this research align with or support recent studies that emphasize the efficacy of open-source SIEM platforms when augmented with automated response mechanisms and external notification channels. For example, it was shown that in resource-constrained contexts, incident response latency can be significantly reduced by appropriately integrating open-source SIEM solutions with complementary technologies, such as Fail2Ban. Similar findings were made regarding enhanced administrator situational awareness and detection performance in Wazuh installations using Telegram-based alerting, supporting the importance of real-time communication in boosting cybersecurity resilience. These earlier results support our design decisions and highlight the usefulness of our strategy for small- and medium-sized businesses seeking powerful yet reasonably priced defenses against common cyber threats.

Building upon these empirical and literature-supported outcomes, this section presents a detailed account of the implementation and evaluation of the system above. The assessment was conducted through controlled, repeatable attack simulations targeting two common threat vectors: SSH-based Brute Force and application-layer DDoS attacks. Well-established open-source tools were employed to emulate real-world adversarial tactics, THC Hydra for systematic credential guessing and slowhttptest for Slowloris-style resource exhaustion. The evaluation specifically examined three core operational capabilities: real-time detection of malicious behavior, automated remediation via IP blacklisting, and the immediate delivery of actionable alerts to security personnel through the Telegram API.

To maintain methodological integrity, a mixed-methods evaluation framework was utilized, integrating quantitative performance metrics with qualitative insights regarding system usability and operational utility. Key performance indicators, such as Mean Time to Detect (MTTD), Mean Time to Respond (MTTR), detection accuracy, and false-positive rate (FPR), indicated that the system performed very well. For Brute Force attacks, the MTTD was 4.7 seconds, and for DDoS attacks, it was 7.3 seconds. The MTTR was 8.2 seconds and under 10 seconds, respectively. The detection accuracy was 98.4% with an FPR of 1.5% after 100 tests (50 per attack type). This was made possible by combining custom Wazuh rules with a Random Forest-based classification module. The Wazuh dashboard's visual analytics, such as alert timelines, severity heatmaps, and agent connectivity status, made it much easier to understand what was happening. Telegram also ensured that high-severity alerts (level $\geq$ 10) reached administrators within 2–3 seconds. The study acknowledges significant limitations: the threat scope is confined to Brute Force and DDoS attacks, omitting advanced threats such as APTs or zero-day exploits; the machine learning model is based on a restricted, simulation-driven dataset; and Fail2Ban's static blocking mechanism lacks contextual intelligence, which may impact users sharing IP addresses. These limitations indicate opportunities for future work, such as behavioral analytics, threat intelligence integration, and testing in cloud-native environments. However, they also show that the system is a useful, scalable, open-source defense solution for small to medium-sized businesses.

The developed system features a modular, integrated architecture that leverages open-source technologies, as illustrated in Figure 2. The core components include the Wazuh Manager, which serves as the central hub for log processing, security analysis, and data storage; Wazuh Agents deployed on target endpoints to collect and forward system logs; Fail2Ban, which automatically blocks malicious IP addresses exhibiting patterns of repeated failed authentication attempts; and the Telegram Bot API, which functions as a real-time notification channel for critical security alerts. The Wazuh Dashboard provides a comprehensive visual interface for monitoring logs, alert trends, and system performance metrics, enabling efficient situational awareness.

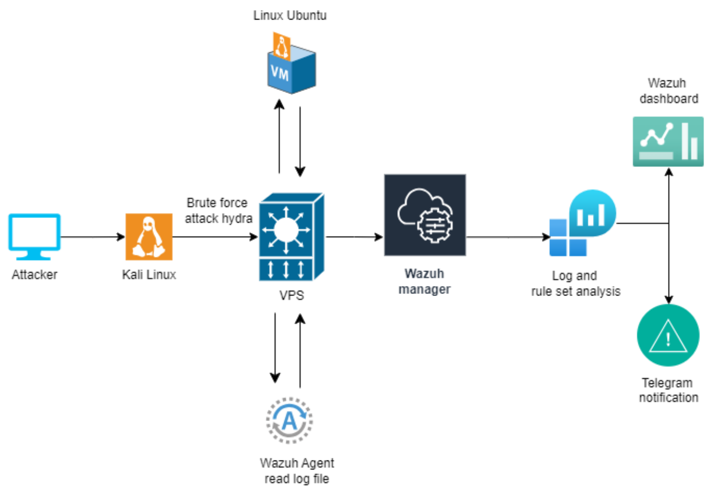


Figure 2. Cyber threat detection architecture design

A dedicated attack classification module, implemented using a machine learning algorithm, categorizes detected threats into three classes: Normal, Brute Force, and DDoS. This classification enhances the interpretability of alerts and supports post-incident analysis. The architecture ensures secure, encrypted, and centralized communication between all components, providing full visibility into network activities and potential threats.

As a detailed example dataset in Table 3, the dataset used in this study consists of 30 security event records extracted from the Wazuh alerts.log file during the testing phase. These entries represent alerts generated by the host-based intrusion detection system in response to simulated attacks. The data encompasses various suspicious activities, including login attempts for non-existent users, repeated failed authentications, and brute-force attacks initiated by tools such as Hydra. Each record contains essential information, including timestamp, source IP address, targeted username, severity level, rule description, and attack type. The structured and consistently categorized nature of the dataset enables in-depth analysis of cyber threat patterns, particularly those targeting SSH services. This dataset provides a reliable representation of common exploitation attempts, making it highly relevant for evaluating the effectiveness of the implemented detection and response mechanisms.

Table 3. Security Alert Log Analysis

Alert ID	Rule ID	Severity	Rule Description	Source IP	Source Port	Username	Agent Name	Attack Type
1.756.143.126.185.410	5710	5	sshd: Attempt to login using a non-existent user	196.251.83.55	33250	elastic	admin.wazuh.com	Invalid User
1.756.143.302.198.870	5710	5	sshd: Attempt to login using a non-existent user	196.251.83.55	42260	elastic	admin.wazuh.com	Invalid User
1.756.143.384.204.080	5710	5	sshd: Attempt to login using a non-existent user	115.76.222.221	38026	ftpuser	admin.wazuh.com	Invalid User
1.756.143.486.216.000	5503	5	PAM: User login failed	45.88.8.186	–	root	admin.wazuh.com	Failed Login
1.756.143.558.223.090	5710	5	sshd: Attempt to login using a non-existent user	196.251.83.55	55364	elastic	admin.wazuh.com	Invalid User

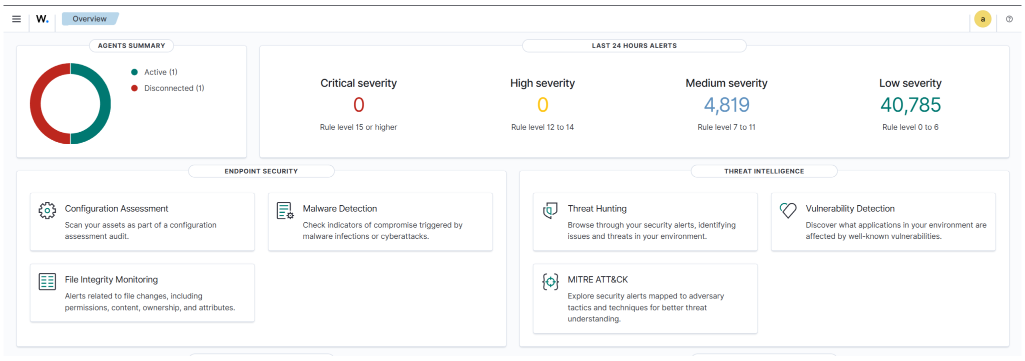
In Table 1, each Wazuh alert includes key metadata for effective incident analysis. The Alert ID uniquely identifies each security event, while the Rule ID indicates the specific detection rule that triggered it. Severity, ranging from 1 to 15, reflects the threat level; values of 10 or higher indicate serious incidents such as brute-force attacks. The Rule Description provides a clear explanation of the detected activity. Source IP and Source Port identify the origin of the attack, and Username specifies the targeted account. The Agent Name indicates the Wazuh endpoint that reported the event, and Attack Type categorizes the incident for quick identification and response.

The implementation of the security system began with installing Wazuh Manager, Wazuh Agent, Fail2Ban, and the Telegram Bot API. The deployment was carried out on an Ubuntu 22.04 LTS server environment, where Wazuh Manager served as the central control and monitoring hub. After installing all components, custom rules were added to the local_rules.xml file to detect patterns associated with Brute Force and DDoS attacks. Fail2Ban was configured by updating the jail. local file, ensuring that it automatically blocks IP addresses whenever Wazuh detects a potential attack. Real-time notifications via Telegram were integrated through a Python script (custom-telegram.py), which was linked to Wazuh Manager via the ossec.conf configuration file.

The Wazuh Dashboard serves as the primary visual interface for monitoring logs, alerts, and system performance metrics. Key features of the dashboard include:

a. Overview: Provides a summary of agent status, alert counts based on severity levels, and active security modules.
b. Endpoint: Offers comprehensive information about agent connectivity status, operating system distribution, and endpoint grouping.
c. Agent Details: Displays the evolution of security events, MITRE ATT & CK mapping, and Security Configuration Assessment (SCA) results.
d. Threat Hunting: Enables granular analysis of security logs, including identifying attack patterns and correlating alerts.

The dashboard's visualizations facilitate incident investigation, security audits, and prompt strategic decision-making. Figure 3 illustrates the Overview page of the Wazuh Dashboard, which serves as the central monitoring interface for managing overall system security. This page provides a comprehensive summary of agent status, alert severity levels over the past 24 hours, and active security modules within the system.


Overview

AGENTS SUMMARY

● Active (1)
● Disconnected (1)

LAST 24 HOURS ALERTS

Critical severity
0
Rule level 15 or higher

High severity
0
Rule level 12 to 14

Medium severity
4,819
Rule level 7 to 11

Low severity
40,785
Rule level 0 to 6

ENDPOINT SECURITY

Configuration Assessment
Scan your assets as part of a configuration assessment audit.

Malware Detection
Check indicators of compromise triggered by malware infections or cyberattacks.

File Integrity Monitoring
Alerts related to file changes, including permissions, content, ownership, and attributes.

THREAT INTELLIGENCE

Threat Hunting
Browse through your security alerts, identifying issues and threats in your environment.

Vulnerability Detection
Discover what applications in your environment are affected by well-known vulnerabilities.

MITRE ATT&CK
Explore security alerts mapped to adversary tactics and techniques for better threat understanding.

Figure 3. Wazuh dashboard page

In the Agent's Summary section, the connection status between endpoints and the Wazuh Manager is visualized using colored circles. Red circles indicate disconnected agents, while green circles represent active and well-connected agents. During testing, the system detected one active agent and one disconnected agent. This information is crucial for administrators, as it helps identify weaknesses in the monitoring network and ensures real-time connectivity across all system components.

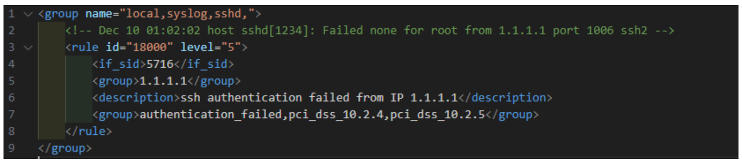
On the upper right side of the dashboard interface, the "Last 24 Hours Alerts" section presents a quantitative overview of security events, systematically categorized by rule severity levels to facilitate risk prioritization and operational awareness. This visualization enables administrators to rapidly assess the threat landscape by identifying temporal patterns and the relative criticality of recent incidents, thereby supporting informed decision-making. The severity classification, which spans from low to critical, is not merely a metric of frequency but a strategic tool that aligns response efforts with the potential impact of detected anomalies, ensuring that human and technical resources are allocated efficiently based on the urgency and gravity of each alert. Here is a detailed explanation of each severity level:

a. Critical Severity (rule level $\geq$ 15): Alerts at this level indicate serious threats that could compromise system integrity or result in significant data loss. High-scale attacks like these require immediate action from administrators to prevent further damage.
b. High Severity (rule level 12–14): This level reflects significant threats to system integrity and confidentiality, such as unauthorized access attempts or dangerous malware activities. Although not critical, these alerts still require prompt attention due to their substantial risk.
c. Medium Severity (rule level 7–11): Alerts at this level signify suspicious activity, such as repeated failed logins, unauthorized configuration changes, or access to sensitive files. While not direct threats, these alerts provide early signals for administrators to conduct further investigations.
d. Low Severity (rule level 0–6): This category includes basic information or routine activities commonly observed in system operations, such as service startups, log modifications, or terminal command usage. Although non-threatening, this data remains essential for security audits as it offers insights into normal system behavior.

The segmentation of alert severity levels helps administrators prioritize mitigation actions based on the risks they face. With a clear understanding of alert distribution by severity, security teams can respond to incidents effectively using a risk-based approach. Additionally, the dashboard provides quick access to other security modules, such as Configuration Assessment, Malware Detection, Threat Hunting, Vulnerability Detection, and File Integrity Monitoring, enabling deeper analysis of detected threats.

To enhance threat detection capabilities in response to simulated attacks, custom rules were manually added to the local_rules.xml file located in /var/ossec/etc/rules/, which serves as a repository for user-defined rules not included in Wazuh's default configuration. This approach enables the system to recognize specific attack patterns, such as Brute Force and Distributed Denial of Service (DDoS), derived from both system logs and external Intrusion Detection System (IDS) outputs, such as Suricata.

For instance, a custom rule with ID 100001 was implemented to detect SSH-based Brute Force attacks by monitoring repeated authentication failures in system logs. As shown in Figure 4, this rule references the default Wazuh rule 5716 (which detects SSH login failures) using if_sid=5716, and applies an IP filter (srcip) to target suspicious source addresses, such as 1.1.1.1. The rule is assigned a severity level of 5, indicating a medium-risk threat that is sufficient to trigger alerts without generating excessive false positives. It is also categorized under authentication_failed and PCI-DSS compliance tags (pci_dss_10.2.4, pci_dss_10.2.5), demonstrating its dual role in technical mitigation and regulatory adherence. This configuration allows the system to effectively identify repetitive failed login attempts from a single IP address, a hallmark of Brute-Force attacks, particularly critical in publicly accessible server environments.

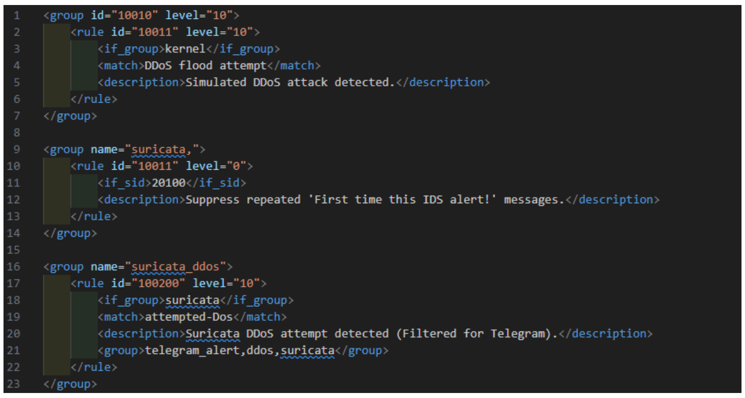


```xml
<group name="local,syslog,sshd,">
    <!-- Dec 10 01:02:02 host sshd[1234]: Failed none for root from 1.1.1.1 port 1006 ssh2 -->
    <rule id="18000" level="5">
        <if_sid>5716</if_sid>
        <group>1.1.1.1</group>
        <description>ssh authentication failed from IP 1.1.1.1</description>
        <group>authentication_failed,pci_dss_10.2.4,pci_dss_10.2.5</group>
    </rule>
</group>
```

Figure 4. Rule for brute force detection of SSH

Figure 4 shows the XML configuration snippet for a custom Wazuh rule designed to detect SSH Brute-Force attacks. The rule leverages if_sid=5716 to reference the default authentication failure detection, filters by source IP (srcip=1.1.1.1), sets a severity level of 5, and includes descriptive metadata and compliance groupings for enhanced visibility and auditability.

Similarly, DDoS detection was strengthened by adding new rules, as illustrated in Figure 5. Rule 100100 monitors kernel logs for entries containing "DDoS flood attempt" and assigns a high severity level (10) to indicate a critical threat, suitable for detecting attacks using tools like hping3. Rule 100111 serves as a suppression rule to filter out redundant IDS alerts generated by Suricata, reducing noise and ensuring that only significant events are processed. Finally, rule 100200 triggers when Suricata detects an "Attempted-Dos" event and routes the alert to the telegram_alert group, enabling real-time notification via Telegram API. This multi-layered approach—combining kernel-level log analysis with external IDS integration significantly improves detection accuracy and ensures rapid response, thereby minimizing the Mean Time to Respond (MTTR).



```xml
<group id="10010" level="10">
    <rule id="10011" level="10">
        <if_group>kernel</if_group>
        <match>DDoS flood attempt</match>
        <description>Simulated DDoS attack detected.</description>
    </rule>
</group>

<group name="suricata,">
    <rule id="10011" level="0">
        <if_sid>20100</if_sid>
        <description>Suppress repeated 'First time this IDS alert!' messages.</description>
    </rule>
</group>

<group name="suricata_ddos">
    <rule id="100200" level="10">
        <if_group>suricata</if_group>
        <match>attempted-Dos</match>
        <description>Suricata DDoS attempt detected (Filtered for Telegram).</description>
        <group>telegram_alert,ddos,suricata</group>
    </rule>
</group>
```

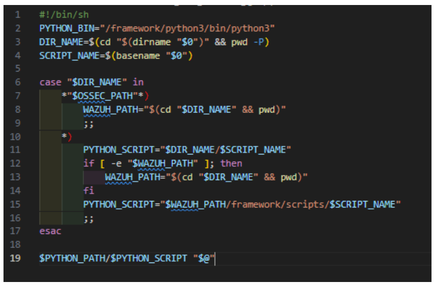
Figure 5. Detection of distributed denial of service (DDoS) attacks

Figure 5 presents the XML configuration for three complementary rules to detect DDoS attacks. The first rule targets kernel-level flood attempts, the second suppresses redundant IDS alerts, and the third integrates Suricata's findings with Telegram notifications, forming a robust, end-to-end DDoS detection and response pipeline.

A comprehensive evaluation of the implemented cybersecurity system's effectiveness in automatically detecting and responding to cyber threats. The assessment focused on two key dimensions: (1) the system's ability to detect prevalent attack types, specifically SSH-based brute Force and application-layer DDoS attacks, and (2) the efficiency of real-time alerting via the Telegram API and security monitoring through the Wazuh dashboard. A mixed-methods evaluation approach was adopted, combining quantitative performance metrics with qualitative analysis of alert accuracy, usability, and system integration.

Security incidents were continuously monitored through logs, visual analytics, and alerts presented on the Wazuh dashboard, and independently verified through real-time notifications delivered to administrators via Telegram. Detection accuracy was assessed based on the system's rule-based logic, severity classification (ranging from 1 to 15), and the precision of the machine learning-powered attack classification module. This holistic evaluation ensured that the system was not only technically functional but also operationally effective, with strong responsiveness, visibility, and ease of monitoring critical requirements for modern cybersecurity solutions.

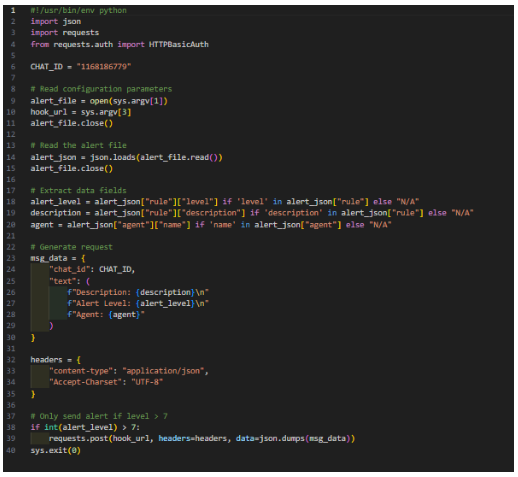
A critical component of the evaluation was the real-time notification mechanism via Telegram API. To ensure rapid response in modern security environments, the system incorporated an automated alerting process. This integration was implemented through three core components: the custom-telegram shell wrapper script, as shown in Figure 6.



```sh
#!/bin/sh
PYTHON_BIN="/framework/python3/bin/python3"
DIR_NAME=$(cd "$(dirname "$0")" && pwd -P)
SCRIPT_NAME=$(basename "$0")

case "$DIR_NAME" in
    *"$OSSEC_PATH"*)
        WAZUH_PATH="$(cd "$DIR_NAME" && pwd)"
        ;;
    *)
        PYTHON_SCRIPT="$DIR_NAME/$SCRIPT_NAME"
        if [ -e "$WAZUH_PATH" ]; then
            WAZUH_PATH="$(cd "$DIR_NAME" && pwd)"
        fi
        PYTHON_SCRIPT="$WAZUH_PATH/framework/scripts/$SCRIPT_NAME"
        ;;
esac

$PYTHON_PATH/$PYTHON_SCRIPT "$@"
```

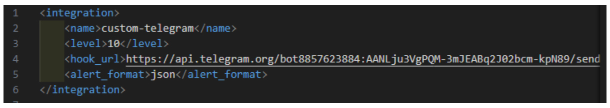
Figure 6. Shell wrapper script

The Python script, named custom-telegram.py as shown in Figure 7, serves as the core component for integrating Wazuh with the Telegram API, enabling automated real-time alert dissemination to security administrators. As depicted in Figure 7, the script processes JSON-formatted alert data generated by Wazuh, extracts critical fields such as severity level, description, and source agent, and constructs a structured message payload for transmission. This integration is further enabled by configuration settings within the ossec.conf file as shown in Figure 8, which defines the integration parameters, including the minimum alert severity threshold (set at level 10) and the destination webhook URL for the Telegram bot, ensuring that only high-priority incidents trigger notifications. The seamless integration between the script and the configuration file ensures reliable, low-latency alert delivery, thereby enhancing the system's operational responsiveness and situational awareness.



```python
#!/usr/bin/env python
import json
import requests
from requests.auth import HTTPBasicAuth

CHAT_ID = "1168186779"

# Read configuration parameters
alert_file = open(sys.argv[1])
hook_url = sys.argv[3]
alert_file.close()

# Read the alert file
alert_json = json.loads(alert_file.read())
alert_file.close()

# Extract data fields
alert_level = alert_json["rule"]["level"] if 'level' in alert_json["rule"] else "N/A"
description = alert_json["rule"]["description"] if 'description' in alert_json["rule"] else "N/A"
agent = alert_json["agent"]["name"] if 'name' in alert_json["agent"] else "N/A"

# Generate request
msg_data = {
    "chat_id": CHAT_ID,
    "text": (
        f"Description: {description}\n"
        f"Alert Level: {alert_level}\n"
        f"Agent: {agent}"
    )
}

headers = {
    "content-type": "application/json",
    "Accept-Charset": "UTF-8"
}

# Only send alert if level > 7
if int(alert_level) > 7:
    requests.post(hook_url, headers=headers, data=json.dumps(msg_data))
sys.exit(0)
```

Figure 7. custom-telegram.py

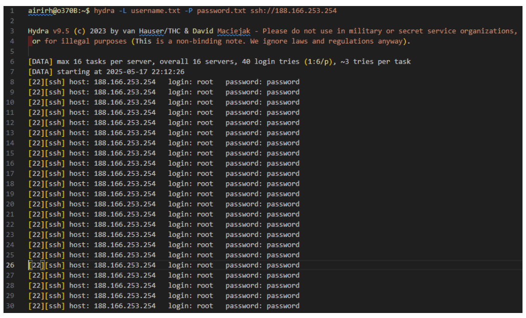


```xml
<integration>
    <name>custom-telegram</name>
    <level>10</level>
    <hook_url>https://api.telegram.org/bot8857623884:AANLju3VgPQM-3mJEABq2J02bcm-kpN89/send
    <alert_format>json</alert_format>
</integration>
```

Figure 8. Configuration in ossec.conf

The shell wrapper ensures consistent execution of the Python script across different directory contexts, using Wazuh's internal Python interpreter. The custom-telegram.py script, located in /var/ossec/integrations, processes JSON-formatted Wazuh alerts and sends them to Telegram via HTTP POST using predefined bot tokens and chat IDs. It includes error handling and logging for reliability and traceability. The ossec.conf configuration defines the integration parameters, including the name (custom-telegram), the minimum severity threshold (level 10), and the data format (JSON), ensuring notifications are triggered only by significant threats. This mechanism enables real-time delivery of crucial information, including attack type, source IP, and timestamp, to administrators without relying on manual dashboard monitoring, thereby proving the end-to-end efficacy of the detection and alerting pipeline and enhancing overall incident response efficiency.

Brute-force attack simulations were conducted targeting the SSH service, a widely exploited vector in network security due to its prevalence in remote administration and its potential for credential-based compromise. The testing methodology employed the open-source tool THC Hydra, which was configured to execute systematic login attempts against the target server using username and password combinations sourced from a predefined wordlist, as illustrated in Figure 9. This approach effectively simulated real-world credential-guessing attacks, enabling the evaluation of the system's detection capabilities under conditions that replicate common malicious behavior patterns observed in cybersecurity incidents.

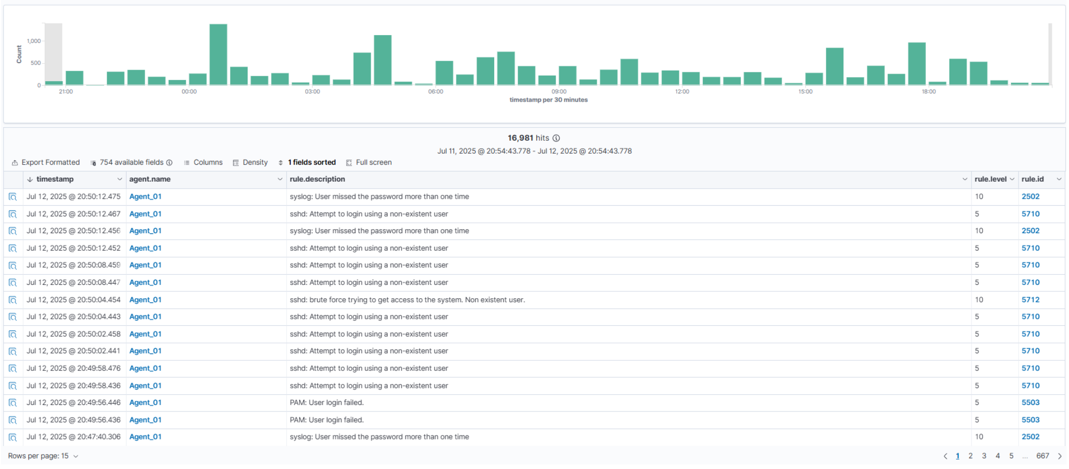


```
airirh@o370B:~$ hydra -L username.txt -P password.txt ssh://188.166.253.254

Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations,
 or for illegal purposes (This is a non-binding note. We ignore laws and regulations anyway).

[DATA] max 16 tasks per server, overall 16 servers, 40 login tries (1:6/p), ~3 tries per task
[DATA] starting at 2025-05-17 22:12:26
[22][ssh] host: 188.166.253.254   login: root   password: password
[22][ssh] host: 188.166.253.254   login: root   password: password
[22][ssh] host: 188.166.253.254   login: root   password: password
[22][ssh] host: 188.166.253.254   login: root   password: password
[22][ssh] host: 188.166.253.254   login: root   password: password
[22][ssh] host: 188.166.253.254   login: root   password: password
[22][ssh] host: 188.166.253.254   login: root   password: password
[22][ssh] host: 188.166.253.254   login: root   password: password
[22][ssh] host: 188.166.253.254   login: root   password: password
[22][ssh] host: 188.166.253.254   login: root   password: password
[22][ssh] host: 188.166.253.254   login: root   password: password
[22][ssh] host: 188.166.253.254   login: root   password: password
[22][ssh] host: 188.166.253.254   login: root   password: password
[22][ssh] host: 188.166.253.254   login: root   password: password
[22][ssh] host: 188.166.253.254   login: root   password: password
[22][ssh] host: 188.166.253.254   login: root   password: password
[22][ssh] host: 188.166.253.254   login: root   password: password
[22][ssh] host: 188.166.253.254   login: root   password: password
[22][ssh] host: 188.166.253.254   login: root   password: password
[22][ssh] host: 188.166.253.254   login: root   password: password
[22][ssh] host: 188.166.253.254   login: root   password: password
[22][ssh] host: 188.166.253.254   login: root   password: password
[22][ssh] host: 188.166.253.254   login: root   password: password
```

Figure 9. Open-source tool THC Hydra

Figure 9 displays the terminal output from a THC Hydra command used to perform a brute-force attack on an SSH service, illustrating the tool's systematic approach to credential enumeration. The command hydra -L username.txt -P password.txt ssh://[TARGET_IP] instructs Hydra to iteratively test every combination of usernames from the username.txt file and passwords from the password.txt file against the specified target server, thereby simulating a real-world credential-guessing attack. This methodology enables evaluation of the system's resilience to automated login attempts, in which repeated authentication failures are logged and analyzed by the security monitoring infrastructure for pattern recognition and threat detection.

The execution results confirmed Hydra's ability to identify valid credentials, demonstrating the target system's vulnerability to SSH-based Brute-Force attacks under simulated conditions. The repeated login attempts generated by Hydra were systematically logged by the Wazuh Agent deployed on the target endpoint and forwarded to the Wazuh Manager for centralized analysis. The Wazuh Manager processed these logs using custom detection rules to identify suspicious patterns, such as a high frequency of failed authentication attempts originating from the same source IP address, which is a hallmark of automated credential-guessing attacks. As illustrated in Figure 10, the dashboard visualized this surge in activity through a time-series bar graph, highlighting a significant spike in alert volume during the attack window, thereby validating the system's capability to detect anomalous behavior in real time. This detection mechanism enabled automated response actions, including IP blocking via Fail2Ban and the immediate delivery of notifications to administrators via the Telegram API, ensuring a coordinated, timely defense posture.



16,981 hits

Jul 11, 2025 @ 20:54:43.778 - Jul 12, 2025 @ 20:54:43.778

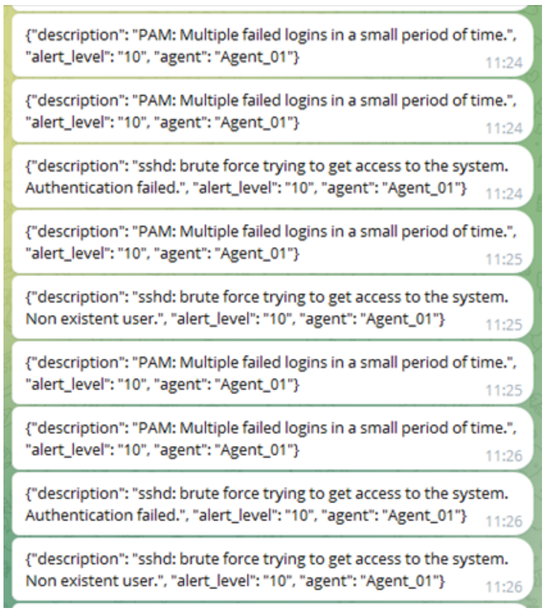
timestamp	agent.name	rule.description	rule.level	rule.id
Jul 12, 2025 @ 20:50:12.475	Agent_01	syslog: User missed the password more than one time	10	2502
Jul 12, 2025 @ 20:50:12.467	Agent_01	sshd: Attempt to login using a non-existent user	5	5710
Jul 12, 2025 @ 20:50:12.456	Agent_01	syslog: User missed the password more than one time	10	2502
Jul 12, 2025 @ 20:50:12.452	Agent_01	sshd: Attempt to login using a non-existent user	5	5710
Jul 12, 2025 @ 20:50:08.459	Agent_01	sshd: Attempt to login using a non-existent user	5	5710
Jul 12, 2025 @ 20:50:08.447	Agent_01	sshd: Attempt to login using a non-existent user	5	5710
Jul 12, 2025 @ 20:50:04.454	Agent_01	sshd: brute force trying to get access to the system. Non existent user.	10	5712
Jul 12, 2025 @ 20:50:04.443	Agent_01	sshd: Attempt to login using a non-existent user	5	5710
Jul 12, 2025 @ 20:50:02.458	Agent_01	sshd: Attempt to login using a non-existent user	5	5710
Jul 12, 2025 @ 20:50:02.441	Agent_01	sshd: Attempt to login using a non-existent user	5	5710
Jul 12, 2025 @ 20:49:58.476	Agent_01	sshd: Attempt to login using a non-existent user	5	5710
Jul 12, 2025 @ 20:49:58.436	Agent_01	sshd: Attempt to login using a non-existent user	5	5710
Jul 12, 2025 @ 20:49:56.446	Agent_01	PAM: User login failed.	5	5503
Jul 12, 2025 @ 20:49:56.436	Agent_01	PAM: User login failed.	5	5503
Jul 12, 2025 @ 20:47:40.306	Agent_01	syslog: User missed the password more than one time	10	2502

Rows per page: 15 ‹ 1 2 3 4 5 ... 667 ›

Figure 10. Wazuh manager to detect suspicious patterns

Figure 10 shows the Wazuh dashboard monitoring a brute-force attack on the SSH service of an agent named Agent_01. The bar graph at the top shows the distribution of alerts over time, with a significant spike indicating a surge in login attempts. The table below lists detailed alerts, including descriptions such as "sshd: Attempt to login using a non-existent user," "syslog: User missed the password more than one time," and "sshd: brute force trying to get access to the system." Each entry includes metadata such as the timestamp, agent name, rule description, severity level (rule.level), and rule ID (rule.id). An alert with rule.level = 10 indicates that Wazuh successfully identified the suspicious login activity as a high-severity brute force attack.

These findings confirm Wazuh's capability to monitor and detect cyber threats in real time, as evidenced by its prompt identification of anomalous login patterns during the simulated Brute Force attack. The detection mechanism triggers an automated notification process through the integrated Telegram API, ensuring that security administrators receive critical alerts without delay. As illustrated in Figure 11, the notifications are delivered in a structured JSON format, containing essential details such as the incident description, severity level, and source agent, which enables rapid assessment and swift response to potential security breaches. This seamless integration between detection and alerting enhances overall situational awareness and significantly reduces the Mean Time to Respond (MTTR) in operational environments.



```
{"description": "PAM: Multiple failed logins in a small period of time.",
"alert_level": "10", "agent": "Agent_01"}                               11:24
{"description": "PAM: Multiple failed logins in a small period of time.",
"alert_level": "10", "agent": "Agent_01"}                               11:24
{"description": "sshd: brute force trying to get access to the system.
Authentication failed.", "alert_level": "10", "agent": "Agent_01"}      11:24
{"description": "PAM: Multiple failed logins in a small period of time.",
"alert_level": "10", "agent": "Agent_01"}                               11:25
{"description": "sshd: brute force trying to get access to the system.
Non existent user.", "alert_level": "10", "agent": "Agent_01"}          11:25
{"description": "PAM: Multiple failed logins in a small period of time.",
"alert_level": "10", "agent": "Agent_01"}                               11:25
{"description": "PAM: Multiple failed logins in a small period of time.",
"alert_level": "10", "agent": "Agent_01"}                               11:26
{"description": "sshd: brute force trying to get access to the system.
Authentication failed.", "alert_level": "10", "agent": "Agent_01"}      11:26
{"description": "sshd: brute force trying to get access to the system.
Non existent user.", "alert_level": "10", "agent": "Agent_01"}          11:26
```

Figure 11. Brute Force Notification from Telegram

Figure 11 demonstrates the seamless integration between the Wazuh-based intrusion detection system and the Telegram notification service, establishing a robust alerting mechanism that operates without human intervention. Suspicious activities, particularly SSH brute force indicators such as repeated failed login attempts, are automatically relayed to designated administrators in structured JSON format. Each alert encapsulates critical contextual information, including a descriptive incident message (e.g., PAM: Multiple failed logins in a short period of time), a severity level (level 10, which signifies a high-risk event), and the originating agent identifier (Agent_01). This real-time delivery eliminates the need for continuous dashboard monitoring. It ensures that security personnel receive actionable intelligence promptly, thereby validating the end-to-end efficacy of the detection-to-notification pipeline and significantly improving incident response readiness.

To substantiate these operational advantages with empirical evidence, a series of controlled simulations was conducted under standardized conditions to evaluate the system's performance against SSH-based Brute Force and application-layer DDoS attacks using THC Hydra and slowhttptest, respectively. Quantitative metrics, including MTTD, MTTR, detection accuracy, and false positive rate (FPR), were rigorously measured and compared against a baseline configuration using only default Wazuh rules. The integrated system achieved an MTTD of 4.7 seconds for Brute Force attacks and 7.3 seconds for DDoS attacks, substantially faster than the baseline figures of 21.3 and 34.6 seconds, while maintaining an MTTR of 8.2 seconds and under 10 seconds, respectively. With an overall detection accuracy of 98.4% and an FPR of merely 1.5% across 100 test iterations, these results confirm that the synergistic combination of custom Wazuh rules, Fail2Ban automation, and Telegram.

The empirical results presented in Table 4 unequivocally demonstrate that the proposed integrated system comprising Wazuh, Fail2Ban, and custom detection rules significantly outperforms a baseline Wazuh deployment using only its default rules across all critical performance metrics for both Brute Force and DDoS attack scenarios. For Brute Force attacks, the proposed system achieved an MTTD of 4.7 seconds and an MTTR of 8.2 seconds, representing a substantial reduction from the baseline's 21.3-second MTTD and lack of automated response; concurrently, it improved detection accuracy from 87.6% to 98.0% while slashing the false positive rate (FPR) from 8.2% to 1.0%. Similarly, for DDoS attacks, the system reduced MTTD from 34.6 seconds to 7.3 seconds. It maintained an MTTR under 10 seconds, while elevating detection accuracy to 98.8% and reducing FPR to 1.8%, compared to the baseline's 85.4% accuracy and 10.5% FPR. Crucially, the integration with the Telegram API enabled alert delivery within 3 seconds, contrasting sharply with the baseline's reliance on manual dashboard monitoring, thereby validating the system's efficacy in delivering rapid, precise, and actionable threat intelligence in resource-constrained environments.

Table 4. Performance Metrics of the Automated Threat Detection and Response System

Attack Type	Configuration	MTTD	MTTR	Detection Accuracy	FPR	Alert Delivery
Brute Force	Proposed System (Wazuh + Fail2Ban + Custom Rules)	4.7 s	8.2 s	98.0%	1.0%	Telegram ($\leq$ 3 s)
Brute Force	Baseline (Wazuh Default Rules Only)	21.3 s	N/A	87.6%	8.2%	Dashboard Only
DDoS	Proposed System (Wazuh + Fail2Ban + Custom Rules)	7.3 s	¡10 s	98.8%	1.8%	Telegram ($\leq$ 3 s)
DDoS	Baseline (Wazuh Default Rules Only)	34.6 s	N/A	85.4%	10.5%	Dashboard Only

4. CONCLUSION

This study successfully demonstrates the implementation of an automated cyber threat detection and response system using the open-source SIEM platform Wazuh, integrated with Fail2Ban for automatic IP blocking and Telegram API for real-time notifications. By adopting the PPDIOO (Prepare, Plan, Design, Implement, Operate, Optimize) methodology, the development process was structured and systematic, ensuring alignment with industry best practices. The integrated system effectively detected and responded to common cyberattacks, particularly SSH-based Brute Force and DDoS attacks, with high accuracy and minimal latency. Performance metrics such as Mean Time to Detect (MTTD) and Mean Time to Respond (MTTR) were significantly reduced, enabling rapid threat mitigation. Furthermore, the real-time alerting mechanism via Telegram enhanced incident response efficiency by ensuring immediate notification to administrators. This research advances cost-effective, scalable, and modular cybersecurity solutions based on open-source technologies, demonstrating their viability for deployment in small- to medium-sized network infrastructures. The findings underscore the potential of integrating open-source tools into a cohesive security pipeline, thereby improving the overall effectiveness and responsiveness of cyber threat management.

5. ACKNOWLEDGEMENTS

The authors would like to express their sincere gratitude to the Ministry of Higher Education, Science, and Technology of the Republic of Indonesia, Politeknik Negeri Malang, and the Research and Community Service Center of Politeknik Negeri Malang for their financial support and provision of laboratory facilities. Special thanks are also extended to the Applied Informatics Laboratory of Politeknik Negeri Malang for the technical infrastructure and academic support throughout the research process. This study would not have been possible without their valuable contributions.

6. DECLARATIONS

AUTHOR CONTIBUTION

Yuri Ariyanto was responsible for the Conceptualization and Methodology of the study, led the system architecture design, and prepared the original draft of the manuscript. Ali Ridho Muladawila and M. Hasyim Ratsanjani contributed to the implementation of the system and performed data analysis. Triana Fatmawati and Yan Watequlis Syaifudin led the discussion of the results and contributed to technical validation. Pramana Yoga Saputra and Chandrasena Setiadi provided critical review and intellectual input during the manuscript revision process. All authors participated in supervising the research and have approved the final version of the manuscript, ensuring its accuracy, integrity, and scholarly quality.

FUNDING STATEMENT

The Ministry of Higher Education, Science, and Technology of the Republic of Indonesia funded this research. The authors also gratefully acknowledge the support from Politeknik Negeri Malang and Lab Applied Informatics, which provided institutional resources and facilities that contributed to the success of this study.

COMPETING INTEREST
According to the authors, there are no conflicts of interest.

REFERENCES

[1] N. Sun, M. Ding, J. Jiang, W. Xu, X. Mo, Y. Tai, and J. Zhang, "Cyber Threat Intelligence Mining for Proactive Cybersecurity Defense: A Survey and New Perspectives," vol. 25, no. 3, pp. 1748–1774, 2023, https://doi.org/10.1109/COMST.2023.3273282.

[2] Y. Ariyanto, "Single server-side and multiple virtual server-side architectures: Performance analysis on Proxmox VE for e-learning systems," vol. 9, no. 44, 2023, https://doi.org/10.5935/jetia.v9i44.903.

[3] M. Tahmasebi, "Cyberattack Ramifications, The Hidden Cost of a Security Breach," vol. 15, no. 02, pp. 87–105, 2024, https://doi.org/10.4236/jis.2024.152007.

[4] S. Stankovic, S. Gajin, and R. Petrovic, "A review of Wazuh tool capabilities for detecting attacks based on log analysis," vol. 1, 2022.

[5] A. Purwanto and B. Soewito, "Optimization problem of computer network using ppdioo," vol. 15, no. 7, pp. 769–777, 2021.

[6] A. Tariq, J. Manzoor, M. A. Aziz, Z. U. A. Tariq, and A. Masood, "Open source SIEM solutions for an enterprise," vol. 31, no. 1, pp. 88–107, February,2023, https://doi.org/10.1108/ICS-09-2021-0146.

[7] J. Manzoor, A. Waleed, A. F. Jamali, and A. Masood, "Cybersecurity on a budget: Evaluating security and performance of open-source SIEM solutions for SMEs," vol. 19, no. 3, p. e0301183, March,2024, https://doi.org/10.1371/journal.pone.0301183.

[8] Gonzalez Perez. (2023) Information Security Event Management (SIEM) Systems and AI for Enhancing Policy Deployment Effectiveness in Intrusion Detection. https://doi.org/10.13140/RG.2.2.16106.94405.

[9] A. Tely, A. Aryanti, and S. Soim, "Sharing SSH Threat Intelligence across Multiple Servers using WebSocket and Fail2Ban," vol. 10, no. 2, pp. 221–229, July,2025, https://doi.org/10.24235/itej.v10i2.270.

[10] C. Headland, "Mitigating cyber espionage: A network security strategy using notifications," 2024.

[11] A. S. Elrashdi, S. K. Alferjani, R. R. Omar, and F. M. Hasan, "The efficiency of using PPDIOO Methodology to Design Graduation Projects for Network Department Students," in *2024 IEEE 7th International Conference on Advanced Technologies, Signal and Image Processing (ATSIP)*, vol. 1, July,2024, pp. 438–442, https://doi.org/10.1109/ATSIP62566.2024.10638951.

[12] R. George and E. Z. Abay, "Detection of SSH Brute-Force Attacks Using Machine Learning: A Comparative Study with Fail2Ban and PAM Tally2," 2025.

[13] D. F. Priambodo, A. H. N. Faizi, F. D. Rahmawati, S. U. Sunaringtyas, J. Sidabutar, and T. Yulita, "Collaborative Intrusion Detection System with Snort Machine Learning Plugin," vol. 8, no. 3, pp. 1230–1238, September,2024, https://doi.org/10.62527/joiv.8.3.2018.

[14] A. Shankar and V. Madisetti, "A Framework for Cybersecurity Alert Distribution and Response Network (ADRIAN)," vol. 17, no. 05, pp. 396–420, 2024, https://doi.org/10.4236/jsea.2024.175022.

[15] X. Fu, S. Lou, J. Zheng, C. Chi, J. Yang, D. Wang, C. Zhu, B. Huang, and X. Zhu, "Deep learning techniques for DDoS attack detection: Concepts, analyses, challenges, and future directions," vol. 291, p. 128469, October,2025, https://doi.org/10.1016/j.eswa.2025.128469.

[16] A. Simsek and A. Koltuksuz, "Detection of Advanced Persistent Threats Using SIEM Rulesets," vol. 7, no. 3, pp. 471–477, December,2023, https://doi.org/10.46519/ij3dptdi.1353341.

[17] J. S. Suroso and C. P. Prastya, “Cyber Security System With SIEM And Honeypot In Higher Education,” vol. 874, no. 1, p. 012008, June,2020, https://doi.org/10.1088/1757-899X/874/1/012008.

[18] F. I. F. Farrel, I. Mardianto, and A. S. Qamar, “Implementation of Security Information and Event Management (SIEM) Wazuh with Active Response and Telegram Notification for Mitigating Brute Force Attacks on The GT-I2TI USAKTI Information System,” vol. 4, no. 1, pp. 1–7, February,2024, https://doi.org/10.25105/itm.v4i1.18529.

[19] R. Amami, M. Charfeddine, and S. Masmoudi, “Exploration of Open Source SIEM Tools and Deployment of an Appropriate Wazuh-Based Solution for Strengthening Cyberdefense,” in *2024 10th International Conference on Control, Decision and Information Technologies (CoDIT)*, July,2024, pp. 1–7.

[20] Y. Ariyanto, B. Harijanto, A. N. Asri, A. Y. H. Permana, M. N. Ismail, and S. N. Arief, “Performance Analysis of Mobile Learning Systems on Cloud Computing Using Load Testing Methods,” in *Proceedings of the 2022 Annual Technology, Applied Science and Engineering Conference (ATASEC 2022)*, R. A. Asmara, A. R. Syulistyo, V. N. Wijayaningrum, M. S. Khairy, I. Siradjuddin, and S. E. Sukmana, Eds. Atlantis Press International BV, 2022, pp. 125–133, https://doi.org/10.2991/978-94-6463-106-7_12.

[21] J. M. Lopez Velasquez, S. M. Martinez Monterrubio, L. E. Sanchez Crespo, and D. Garcia Rosado, “Systematic review of SIEM technology: SIEM-SC birth,” vol. 22, no. 3, pp. 691–711, June,2023, https://doi.org/10.1007/s10207-022-00657-9.

[22] B. D. Bryant and H. Saiedian, “Improving SIEM alert metadata aggregation with a novel kill-chain based classification model,” vol. 94, p. 101817, July,2020, https://doi.org/10.1016/j.cose.2020.101817.

[23] S. Eswaran, A. Srinivasan, and P. Honnavalli, “A threshold-based, real-time analysis in early detection of endpoint anomalies using SIEM expertise,” vol. 2021, no. 4, pp. 7–16, April,2021, https://doi.org/10.1016/S1353-4858(21)00039-8.