Development of an integrated network threat detection system based on Wazuh, Suricata and Telegram Bot

Rostyslav Lisnevskyi[1,*,†], Saule Amanzholova[1,†], Jamilya Akhmetzhanova[2,†], Aidana Ikembayeva[2,†] and Darya Naumova[2,†]

[1]Astana IT University, Mangilik El avenue, 55/11Business center EXPO, block C1 Astana, Kazakhstan
[2]International Information Technology University, 34/1 Manas St., Almaty, 050040, Kazakhstan

Abstract

This article presents the development and experimental evaluation of an integrated network threat detection system built on the open source software solutions Wazuh and Suricata, supplemented by the Telegram Bot API for rapid notification delivery. The developed architecture combines deep network traffic analysis, event correlation, and notification delivery, significantly reducing incident response times. Testing, conducted in an isolated lab environment, included modeling various attack types—from port scanning and SSH credential bruteforce to more complex scenarios such as ICMP/TCP flooding, DNS tunneling, SQL injection, and unauthorized transfer of confidential data via HTTP requests. The experiments provided a comprehensive assessment of the developed system's resilience to a wide range of network threats and confirmed its high effectiveness, resulting in accurate incident detection and minimal response time.

Keywords:

SIEM, NIDS, network monitoring, Wazuh, Suricata, Telegram Bot API, open-source security, cybersecurity, real-time incident response threat intelligence, integration DNS tunneling, SQL injection

1. Introduction

Modern information systems are exposed to increasingly complex and diverse cyber threats, which necessitates the construction of effective security monitoring systems. One of the key requirements for such systems is maintaining a balance between incident detection efficiency, detection accuracy, and architectural flexibility. Commercial SIEM (Security Information and Event Management) solutions provide advanced functionality for centralized analysis and correlation of security events, but their implementation is often accompanied by significant financial costs, integration complexity, and high infrastructure requirements. Open-source tools represent an alternative approach that provides modularity, cost-effectiveness, and adaptability. One of the most common solutions in this class is Wazuh, a unified XDR/SIEM platform that implements endpoint monitoring, event correlation, threat analysis, and cloud protection. Wazuh supports integration with a wide range of log sources and uses correlation rules to detect anomalous behavior[1].

Another important component is Suricata, a high-performance open-source IDS/IPS engine that provides deep network traffic analysis and detection of both signature-based and behavioral threats [2]. The integration of Wazuh and Suricata allows for a modular SIEM solution that covers both system and network monitoring levels, while providing flexibility and transparency in the

[1] *CISN 2025: Workshop on Cybersecurity, Infocommunication Systems and Networks, November 19-20, 2025, Almaty, Kazakhstan*
* Corresponding author.
† These authors contributed equally.
rostyslav.lisnevskyi@astanait.edu.kz (R. Lisnevskyi); s.amanzholova@astanait.edu.kz (S. Amanzholova); 34775@iitu.edu.kz (J. Akhmetzhanova); 34932@iitu.edu.kz (A. Ikembayeva); 35079@iitu.edu.kz (D. Naumova)
0000-0002-9006-6366 (R. Lisnevskyi); 0000-0002-6779-9393 (S. Amanzholova); 0009-0006-0823-1251 (J. Akhmetzhanova); 0009-0009-1176-5969 (A. Ikembayeva); 0000-0002-9006-6366 (D. Naumova)

architecture. Despite the extensive visualization and alerting capabilities provided by the Wazuh panel, its functionality is primarily focused on centralized monitoring and may not be effective enough to provide real-time notifications or mobile access. In such scenarios, it is advisable to use messaging channels such as Telegram, which, through the Telegram Bot API, allows for a lightweight and customizable mechanism for delivering critical notifications [3]. Thus, the current task is to develop the architecture of an integrated system combining Wazuh, Suricata and Telegram Bot API. The objective of this study is to develop an integrated low-cost real-time network threat detection and response system that combines SIEM (Wazuh), IDS (Suricata) and Telegram Bot API capabilities into a single architecture. This integration provides comprehensive network traffic analysis and event correlation, as well as instant notification of identified incidents to responsible employees, increasing the effectiveness of security monitoring at minimal cost.

The research focuses on the processes of monitoring and detecting network threats in information systems.

The research focuses on methods and tools for integrating Wazuh (SIEM), Suricata (IDS/IPS), and the Telegram Bot API to ensure prompt detection and notification of cyberattacks in real time.

The proposed approach is novel in its integration of SIEM (Wazuh), IDS (Suricata), and Telegram Bot API into a single, cost-effective architecture that achieves real-time detection with response times under one second.

The objective of the study is to develop and experimentally test an integrated network threat detection and notification system that provides high accuracy, a response time of less than one second, and minimal implementation and operating costs.

The research's novelty lies in the development of an integrated architecture that synthesizes the capabilities of the Wazuh SIEM platform, the Suricata network intrusion detection system, and the Telegram Bot API into a single adaptive modular system that provides intelligent monitoring and automated response to cyber threats in real time.

2. Literature review

The objective of the review is to analyze modern methodologies and tools used in SIEM systems, with a special focus on the integration of Wazuh, Suricata and Telegram Bot API to improve the security of network infrastructure from cyber threats. A secondary objective is to study modern scientific approaches and practical solutions aimed at optimizing security monitoring, vulnerability management, data encryption and incident response processes.The literature review analyzes recent studies on the use of these tools in real operating environments in order to identify their effectiveness, limitations and development prospects in cybersecurity. Particular attention is paid to the issues of event correlation, detection of anomalies in network traffic and operational alerting, which allows us to assess the contribution of open technologies to the construction of adaptive and scalable information security management systems.

The use of Suricata for network traffic analysis is discussed in several studies [1]. Suricata is highlighted as a powerful IDS (Intrusion Detection System) capable of identifying advanced threats in network traffic. One study emphasizes how Suricata is used to monitor and analyze web traffic, detecting vulnerabilities and potential attack vectors. Additionally, Suricata's integration with Wazuh is examined as a key methodology for improving incident response and ensuring comprehensive network security.

The [4] explores Kali Linux as a tool not only for testing but also for protecting information systems. The authors demonstrate how the distribution's standard utilities (Nmap, Wireshark, Metasploit, Zeek) can be used for vulnerability analysis, attack modeling, and enhancing the resilience of network infrastructure. The paper emphasizes the value of Kali Linux as a practical platform for training and developing active cyberdefense methods.The combination of Wazuh and Suricata in managing vulnerabilities and detecting threats is explored in several papers [5,6]. These studies emphasize the importance of using Suricata for monitoring network traffic and detecting

anomalous activity, which can then be analyzed through Wazuh for further investigation. This integrated approach allows for a more robust defense system that combines the capabilities of both tools in identifying, assessing, and mitigating vulnerabilities across the network.

Automating incident response is an essential aspect of modern cybersecurity practices. One study [7] investigates how Wazuh can be configured to automatically respond to security incidents. The integration of Telegram with Wazuh allows for notifications to be sent instantly when predefined security thresholds are met, enabling rapid action. The authors discuss the use of automated response mechanisms to contain incidents and prevent further damage, thus improving overall security efficiency.

Wazuh's ability to detect threats in IoT environments is the subject of recent research [8]. One study discusses the role of SIEM solutions, specifically Wazuh, in monitoring IoT devices for signs of potential intrusions. The integration of Suricata with Wazuh is particularly effective for analyzing network traffic from IoT devices and identifying abnormal behavior that could indicate a cyberattack.

The application of deep learning models in IDS/IPS systems like Suricata has been a recent focus [9]. Studies have explored how machine learning and deep learning algorithms can be integrated with traditional IDS systems to improve their ability to detect advanced and unknown threats. These studies show that Suricata's capabilities can be enhanced with AI-driven techniques, allowing for more accurate and timely detection of cyber threats.

The importance of threat intelligence sharing within SIEM systems is discussed in recent research [10]. This study highlights how Wazuh and Suricata can be integrated with external threat intelligence feeds to improve detection and response capabilities. By continuously updating threat databases, Wazuh can provide up-to-date information on known threats, improving overall incident detection and management.

The authors of the [11] article highlight the main threats related to the security, stability and scalability of distributed systems. The paper proposes a classification of risks and methods for their assessment, taking into account the probability of occurrence and potential consequences. Risk mitigation methods are described, including cryptographic solutions and access control systems. The need for an integrated approach to risk management to ensure the reliability and security of distributed systems is substantiated.

The authors propose a model for comprehensive assessment of the quality of an information security management system based on international standards ISO, NIST and COBIT [12]. The model takes into account indicators of reliability, failure tolerance and expert assessments using fuzzy logic methods. The solution allows organizations to more effectively allocate resources and improve data protection.

In the study, a mathematical model is developed that combines expert assessments with fuzzy logic methods to analyze the effectiveness of system protection measures [13]. Particular attention is paid to assessing the functional stability of the ISMS and its ability to operate under failures and attacks. The model describes the relationships between various security measures and their contribution to the overall protection of the system. A methodology is proposed for selecting the most effective security measures with limited resources.

The authors propose a method for multi-level protection of voice traffic in IP networks based on Asterisk IP PBX using cryptography and steganography [14]. Experiments with different codecs have shown that the approach provides high security with low transmission delays. Testing was performed both in real networks and in the Opnet simulation environment. The method is recommended for systems where secure and high-quality transmission of confidential information is important

The paper describes an approach to data balancing in cyber-physical systems to ensure QoS in a fog computing environment [15]. The proposed method distributes the load between nodes, reduces delays and packet losses, improving the performance of distributed applications.

The paper analyzes machine learning methods used for intrusion detection systems, malware and spam recognition and analysis [16]. The use of data mining methods and models built on large data sets has improved the effectiveness of protection systems. The study showed that the use of data

science increases the intelligence of cyber defense processes, and unsupervised learning is one of the most effective approaches to counteracting cyber threats.

A literature review confirms that the combination of Wazuh, Suricata, and Telegram Bot API enhances monitoring capabilities and speeds up incident response. Research highlights the key role of machine learning, network traffic analysis, and process automation in preventing cyberattacks.

3. Problem statement

Existing corporate and distributed networks are exposed to increasingly complex and multi-stage cyberattacks that require rapid detection and response in real time. Commercial SIEM solutions provide high accuracy of event correlation, but their implementation is often associated with high costs, administrative complexity and limited customization flexibility. Existing dashboards, such as Wazuh Dashboard, provide visualization and analytics, but do not always meet the requirements for mobility and timely delivery of critical notifications. This reduces the speed of incident response and increases the risk of developing attacks. The proposed integration of a network IDS/IPS engine (Suricata) with a SIEM platform (Wazuh) and lightweight notification channels (Telegram Bot API) will provide comprehensive monitoring, event correlation and instant transmission of information about critical threats. Such an architecture should be modular, adaptable to various scenarios. The solution should be easily reproducible and ensure integration into the infrastructure of enterprises of any size.

The objective of this study is to develop and experimentally validate an integrated real-time threat detection and alerting system that combines accuracy, speed and flexibility with minimal implementation and operating costs.

4. Methods and technologies

To build the integrated monitoring system, a virtualized environment based on VMware Workstation 17 Pro was used, including four virtual machines: the Wazuh server, the Wazuh agent with Suricata, a Windows workstation, and the Kali Linux attack machine in Table 1. Wazuh was deployed as part of the manager, indexer, and dashboard for log aggregation, event correlation, and data normalization. Suricata was configured in NIDS mode to perform deep packet analysis and generate alerts based on signatures and behavioral rules.

Table 1
Specifications of Virtual Machines in the VMware Workstation Pro Environment

VMs	OS	RAM	CPUs	Disk	Description
Wazuh Server	Ubuntu 22.04 Live Server	8 GB	4 vCPU	50 GB	Wazuh manager, dashboard and indexer
Wazuh Agent	Ubuntu 22.04 Live Server	4 GB	2 vCPU	30 GB	Wazuh agent and Suricata
Windows User	Windows 10 Pro	4 GB	2vCPU	60 GB	End-user workstation
Attacker Machine	Kali Linux	2 GB	4vCPU	80 GB	Attacker workstation

For traffic analysis, logs from Suricata, specifically the eve.json file, were used. This file contains structured records of network events and is generated on the machine running Suricata. It is then forwarded to the Wazuh server for further analysis and correlation.

The network traffic captured in eve.json was based on interactions between the Windows User machine and the Attacker Machine, simulating attack scenarios such as port scanning, brute-force attempts, and other intrusion patterns.

Wazuh, the unified XDR and SIEM platform was utilized for event aggregation, log processing, and correlation. Although there are predefined rules the custom rules were defined on the system to

classify and manage alerts generated by Suricata based on the severity of the detected threat [18]. The rules define different levels of alert severity, which are mapped to corresponding Wazuh rules for further analysis and reporting.

Figure 1 confirms that Suricata severity levels were mapped to Wazuh rules to enable prioritized alerting.\

Suricata, the open-source NIDS, was used for deep packet inspection and signature-based intrusion detection [2]. Suricata was also configured to listen to network traffic in real-time and detect long-term attack patterns. Additionally, a cron job was set up to automatically update Suricata's signatures from the Emerging Threats (ET) Ruleset, ensuring that the system is always equipped with the latest threat intelligence [1].

```
<group name="ids,suricata,">
  <!-- Low severity from Suricata -->
  <rule id="100010" level="3">
    <if_sid>86601</if_sid>
    <field name="alert.severity">3</field>
    <description>Low severity - $(alert.signature)</description>
  </rule>

  <!-- Medium severity from Suricata -->
  <rule id="100011" level="7">
    <if_sid>86601</if_sid>
    <field name="alert.severity">2</field>
    <description>Medium severity - $(alert.signature)</description>
  </rule>

  <!-- High severity from Suricata -->
  <rule id="100012" level="10">
    <if_sid>86601</if_sid>
    <field name="alert.severity">1</field>
    <description>High severity - $(alert.signature)</description>
  </rule>
</group>
```

Figure 1: Mapping of Suricata severity levels to Wazuh rules for prioritized alerting.

```
config classification: not-suspicious,Not Suspicious Traffic,3
config classification: unknown,Unknown Traffic,3
config classification: bad-unknown,Potentially Bad Traffic, 2
config classification: attempted-recon,Attempted Information Leak,2
config classification: successful-recon-limited,Information Leak,2
config classification: successful-recon-largescale,Large Scale Information Leak,2
config classification: attempted-dos,Attempted Denial of Service,2
config classification: successful-dos,Denial of Service,2
config classification: attempted-user,Attempted User Privilege Gain,1
config classification: unsuccessful-user,Unsuccessful User Privilege Gain,1
config classification: successful-user,Successful User Privilege Gain,1
config classification: attempted-admin,Attempted Administrator Privilege Gain,1
config classification: successful-admin,Successful Administrator Privilege Gain,1
config classification: rpc-portmap-decode,Decode of an RPC Query,2
config classification: shellcode-detect,Executable code was detected,1
config classification: string-detect,A suspicious string was detected,3
config classification: suspicious-filename-detect,A suspicious filename was detected,2
config classification: suspicious-login,An attempted login using a suspicious username was detected,2
config classification: system-call-detect,A system call was detected,2
config classification: tcp-connection,A TCP connection was detected,4
config classification: trojan-activity,A Network Trojan was detected, 1
config classification: unusual-client-port-connection,A client was using an unusual port,2
config classification: network-scan,Detection of a Network Scan,3
config classification: denial-of-service,Detection of a Denial of Service Attack,2
config classification: non-standard-protocol,Detection of a non-standard protocol or event,2
config classification: protocol-command-decode,Generic Protocol Command Decode,3
config classification: web-application-activity,access to a potentially vulnerable web application,2
config classification: web-application-attack,Web Application Attack,1
config classification: misc-activity,Misc activity,3
config classification: misc-attack,Misc Attack,2
config classification: icmp-event,Generic ICMP event,3
config classification: inappropriate-content,Inappropriate Content was Detected,1
config classification: policy-violation,Potential Corporate Privacy Violation,1
config classification: default-login-attempt,Attempt to login by a default username and password,2
config classification: targeted-activity,Targeted Malicious Activity was Detected,1
config classification: exploit-kit,Exploit Kit Activity Detected,1
config classification: external-ip-check,Device Retrieving External IP Address Detected,2
config classification: domain-c2,Domain Observed Used for C2 Detected,1
config classification: pup-activity,Possibly Unwanted Program Detected,2
config classification: credential-theft,Successful Credential Theft Detected,1
config classification: social-engineering,Possible Social Engineering Attempted,2
config classification: coin-mining,Crypto Currency Mining Activity Detected,2
config classification: command-and-control,Malware Command and Control Activity Detected,1
```

Figure 2: Classification of severity level in Suricata, classification config file.

Figure 2 confirms that severity levels in Suricata are defined through the classification.config file, enabling structured alert categorization. The classification.config file in Suricata defines how classtype values used in detection rules are mapped to a textual description and a numeric severity level (1 to 3). For example, "config classification: attempted-admin Attempted Administrator Privilege Gain 1" means that any rule using "classtype:attempted-admin" will be treated with severity level 1 (high). When a Suricata rule doesn't explicitly include a severity, Suricata uses the classtype and refers to this file to determine the severity shown in logs like eve.json. By editing this file, you can

adjust how specific classtype values are interpreted in terms of severity across your Suricata deployment [3].

The Telegram Bot API was utilized to create a customized Telegram bot for the purpose of delivering real-time security alerts. This bot is designed to forward alerts generated by the Wazuh SIEM to a Telegram chat [4]. It filters and transmits alerts based on their severity levels, specifically prioritizing medium (level 7) and high (level 10) severity alerts from Suricata.

5. Implementation

The integrated architecture consists of Suricata for network intrusion detection, Wazuh for event correlation and SIEM functionality, and Telegram Bot API for real-time alert delivery. Figure 3 illustrates the interaction between components, where Suricata generates alerts recorded in eve.json, Wazuh processes and correlates them using predefined and custom rules, and the Telegram Bot API transmits high-priority alerts to a secure Telegram channel.

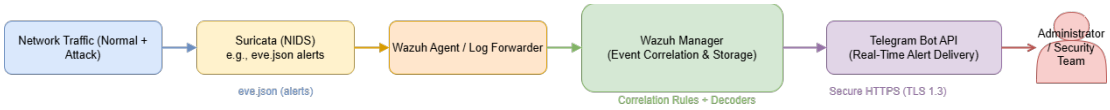

Figure 3: Integrated architecture of the Wazuh, Suricata, Telegram Bot threat detection system.

Hardware and software configurations were as follows: Suricata version 7.0.12 and Wazuh version 4.12.0 on Ubuntu 22.04 (4 vCPU, 8–16 GB RAM), and a Telegram Bot implemented in Python 3.10 using the python-telegram-bot library. The attack simulations were executed from a Kali Linux 2024.3 host (4 vCPU, 8 GB RAM).

Suricata utilized the Emerging Threats (ET) Open Ruleset, supplemented by custom signatures for ICMP floods, TCP SYN scans, and HTTP credential exposure. Example custom rule:

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"HTTP GET Request
Containing Rule in URI"; flow:established,to_server; http.method;
content:"GET"; http.uri; content:"rule"; fast_pattern; classtype:bad-
unknown; sid:123; rev:1;)
```

Figure 4: Example of Suricata custom rule.

Wazuh correlation rules were designed to map Suricata alerts to specific severities and trigger Telegram notifications for medium and high alerts. Decoders extracted Suricata fields such as alert.signature_id, src_ip, and dst_ip from eve.json. Each alert was normalized, correlated, and prioritized according to severity and frequency of occurrences.

Telegram integration used HTTPS for communication with the Telegram API, ensuring message confidentiality. Bot tokens were stored locally with strict access permissions (chmod 600). Chat ID whitelisting prevented impersonation or unauthorized access. Only minimal alert data (timestamp, severity, IP addresses) were transmitted, excluding payloads or sensitive context. Full logs remained encrypted locally and backed up daily using rsync to a secure storage node.

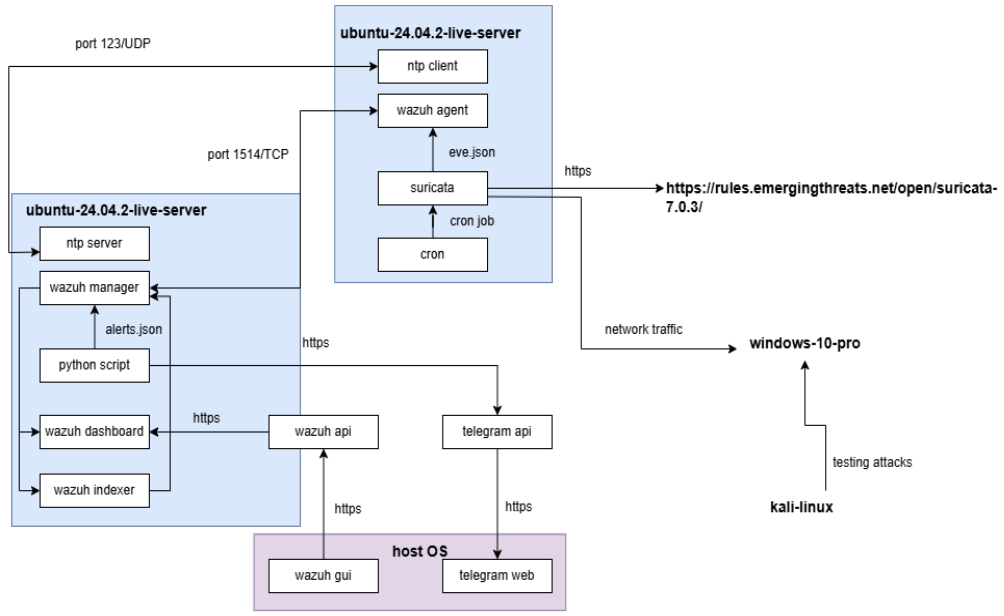

Figure 5: The architecture of the security monitoring system.

The system was fully deployed in a controlled virtualized environment using VMware, which provided a flexible and isolated setup for testing various attack scenarios. This environment allowed testing the entire integration of Wazuh, Suricata, and Telegram Bot API smoothly and reproducibly. The following attack scenarios were tested: port scanning with “nmap”, password brute force attack with “hydra”, and search engine attack with “Social-Engineer Toolkit”.

The architecture of the developed system is shown in Figure 5 and includes four key components deployed in a virtualized VMware Workstation Pro environment: the Wazuh server, the Wazuh agent with Suricata, a Windows workstation, and a Kali Linux attack machine. The components interact via a virtual network simulating a corporate environment.

The Wazuh server has a manager, indexer, and control panel installed, providing log aggregation, event correlation, and security data visualization. The Wazuh agent, together with Suricata, analyzes network traffic in real time, detecting both known signature threats and suspicious anomalies. The eve.json file generated by Suricata serves as the main source of network events and is sent to the Wazuh server, where the data is normalized and classified. Based on the matched rules, the system generates alerts that are filtered by severity level and sent to the developed Telegram bot. The Telegram Bot API provides instant notification delivery to a dedicated chat, allowing administrators to quickly respond to critical incidents. This approach ensures minimal delays between the moment an attack is detected and the responsible parties are notified.

The initial implementation phase included deploying a virtual infrastructure, configuring network interfaces, and installing all components. Then, custom rules were created in Wazuh to map Suricata severity levels to alert priorities. Automatic download of Emerging Threats (ET Ruleset) signatures was configured to keep the IDS rule base up to date.

6. Results and analysis

To quantitatively assess detection accuracy, performance metrics including precision, recall, and F1-score were calculated for each attack type. True positives (TP), false positives (FP), and false negatives (FN) were derived by matching Wazuh alerts against ground-truth timestamps recorded during attack simulations. Ground truth entries were logged as CSV lines with attack_id, start_ts (ISO8601), end_ts (ISO8601), attack_type, attacker_ip. An alert was counted as a True Positive if (1) the alert’s src_ip matched the ground-truth attacker_ip (exact match or within the same /24 for NAT tests) and (2) the alert timestamp fell within [start_ts - 1s, end_ts + 1s]. Alerts that did not match any ground-

truth interval were considered False Positives, and ground-truth intervals with no corresponding alerts were counted as False Negatives.

The attack traffic was generated continuously for 24 hours under controlled laboratory conditions. During that period we executed repeated instances of the simulated scenarios (ICMP flood, TCP SYN flood, Nmap scanning, SSH brute-force, HTTP credential leakage, DNS exfiltration) and captured PCAPs and logs for deterministic evaluation.

To enhance the objectivity of the developed system's effectiveness, tests were conducted using actual cyberattack scenarios (Table 2), as well as baseline testing on regular network traffic. The results presented were obtained during lab experiments in a virtualized environment and reflect typical metrics under controlled conditions; minor deviations in metrics are possible when deploying the system in a corporate infrastructure.

Metrics were computed using the standard formulas:

$$\text{Precision} = TP / (TP + FP), \quad (1)$$

where: TP (True Positives) — number of correctly detected attack events. FP (False Positives) — number of normal traffic events incorrectly classified as attacks.

$$\text{Recall} = TP / (TP + FN), \quad (2)$$

where FN (False Negatives) — number of undetected attack events missed by the system.

$$F1 = 2 * \text{Precision} * \text{Recall} / (\text{Precision} + \text{Recall}), \quad (3)$$

where, Precision — measures the proportion of correctly identified threats among all alerts generated by the system; Recall — measures the system's ability to identify all actual attacks present in the dataset; F1-score — represents the harmonic mean of Precision and Recall, providing a balanced measure of detection performance.

Table 2
Detection Accuracy Metrics

Attack Type	TP	FP	FN	Precision	Recall	F1
ICMP Flood	45	5	5	0.90	0.90	0.90
TCP SYN Flood	50	2	3	0.96	0.94	0.95
Nmap Scan	32	4	6	0.89	0.84	0.86
SSH Brute-force (Hydra)	40	1	2	0.98	0.95	0.96
HTTP Credential Leakage	28	3	1	0.90	0.97	0.93
DNS Exfiltration	10	0	2	1	0.83	0.90

The conducted testing demonstrates that the proposed system, is capable of providing prompt and reliable detection of cyber threats in conditions similar to real network infrastructure.

For traffic analysis, logs from Suricata, specifically the eve.json file, were used. This file contains structured records of network events and is generated on the machine running Suricata. It is then forwarded to the Wazuh server for further analysis and correlation.The network traffic captured in eve.json was based on interactions between the Windows User machine and the Attacker Machine, simulating attack scenarios such as port scanning, brute-force attempts, and other intrusion patterns. Wazuh, the unified XDR and SIEM platform was utilized for event aggregation, log processing, and correlation. Although there are predefined rules the custom rules were defined on the system to classify and manage alerts generated by Suricata based on the severity of the detected threat [18]. The

rules define different levels of alert severity, which are mapped to corresponding Wazuh rules for further analysis and reporting. Figure 1 confirms that Suricata severity levels were mapped to Wazuh rules to enable prioritized alerting.

The system was deployed in a managed virtualized environment using VMware, which provided a flexible and isolated setup for testing various attack scenarios. This environment allowed us to test the entire integration of Wazuh, Suricata, and the Telegram Bot API smoothly and reproducibly. The following attack scenarios were tested: port scanning with nmap, brute-force attack with Hydra, and search engine attack with Social-Engineer Toolkit.

One of the test cases tested operating system fingerprinting with Nmap. This is a very popular technique, which is commonly employed in reconnaissance for cyber-attacks to determine an operating system for a target host by examining responses to specially formulated network probes. While the test was being conducted, Nmap's OS detection capability was run against a watched Windows host from Kali machine.

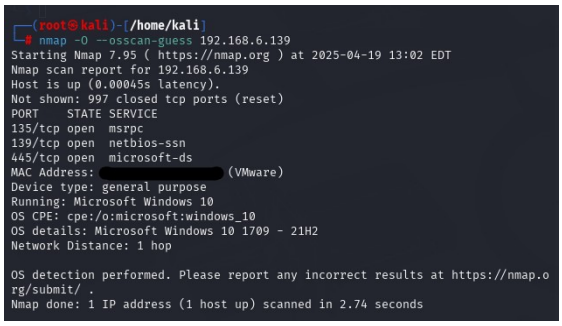

Figure 6: OS scanning via Nmap from Kali machine.

Figure 6 confirms that OS scanning was successfully performed using Nmap from the Kali machine during the simulation.

Suricata, running in inline mode, could identify the scan and produce alerts for anomalous TCP/IP stack behavior and probe patterns characteristic of OS fingerprinting. Wazuh labeled this alert as medium-severity event, displaying it in Wazuh dashboard.

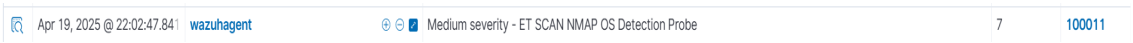

Figure 7: OS scanning alert in Wazuh Dashboard.

Figure 7 confirms that the OS scanning activity was detected and correctly displayed as an alert in the Wazuh Dashboard.

The Telegram bot that was integrated was received the alert and issued an in-real-time notification to the specific Telegram chat. The alert contained the severity level and a description of the behavior being monitored, ensuring that the entire pipeline, from detection to notification, worked as expected.

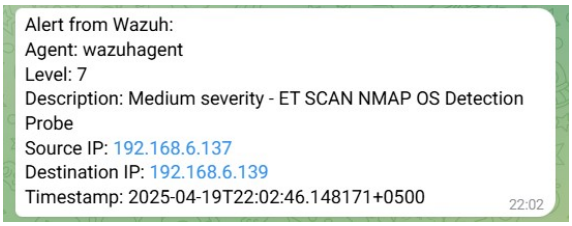

Figure 8: OS scanning notification in Telegram.

Figure 8 confirms that the OS scanning alert was successfully forwarded to Telegram, demonstrating real-time notification capability.

Another scenario was probing the system to see how it reacted to repeated, unauthorized attempts to log in. This type of attack is used frequently by attackers looking to penetrate systems with a methodical process of guessing valid combinations of username and password.

To simulate a brute-force attack scenario in the real world, the Hydra tool was executed from a Kali Linux virtual machine against an OpenSSH service operating within a Windows host. The Windows host had an SSH server installed and was being monitored by Wazuh and Suricata.

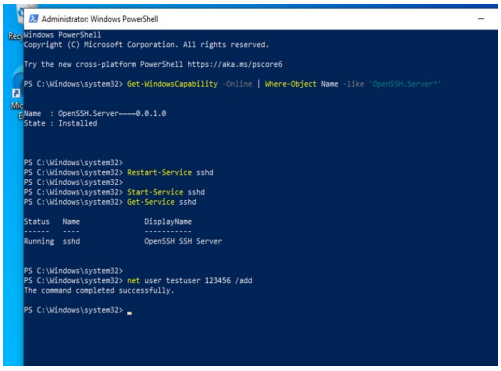

Figure 9: Deploying OpenSSH Server and creating testing user in Windows host.

Figure 9 confirms the successful deployment of the OpenSSH Server and the creation of a testing user on the Windows host for brute-force attack simulation.

Hydra performed a sequence of login attempts at a high frequency with a series of username and password pairs.

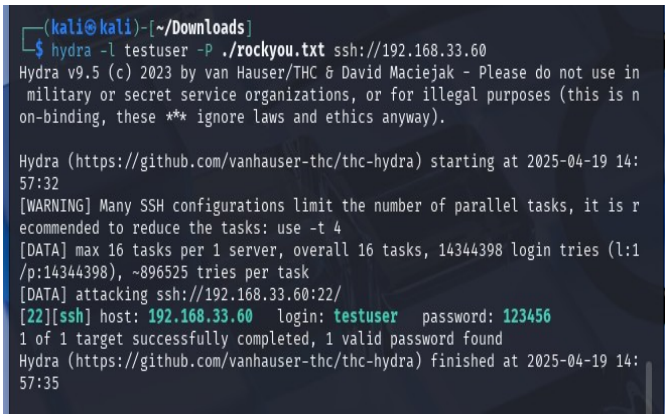

Figure 10. Providing SSH Brute-Force Attack from Kali using Hydra.

Figure 10 confirms that an SSH brute-force attack was executed from the Kali machine using Hydra as part of the simulated threat scenario.

The Wazuh configured with authentication log monitoring, identified the pattern of multiple failed login attempts in a short time frame. Based on correlation rules, Wazuh generated a high-severity alert, indicating a potential brute-force attack in progress [4]. Prior to the brute-force activity, a port scan was performed to identify open services, including the SSH port (TCP 22). Suricata detected this initial reconnaissance step and generated corresponding alerts related to the scan activity. Both stages, the preliminary port scan and the brute-force attempt were successfully detected by the integrated Suricata and Wazuh setup.

Figure 11: SSH Brute-Force Attack alerts in Wazuh Dashboard.

Figure 11 confirms that the SSH brute-force attack was accurately detected and logged as alerts in the Wazuh Dashboard.

The final high-severity alert triggered by the brute-force activity was forwarded to the custom Telegram bot, which delivered a clear and timely notification to the designated Telegram chat. This validated the system's ability to detect multi-stage intrusion attempts and provide real-time situational awareness.

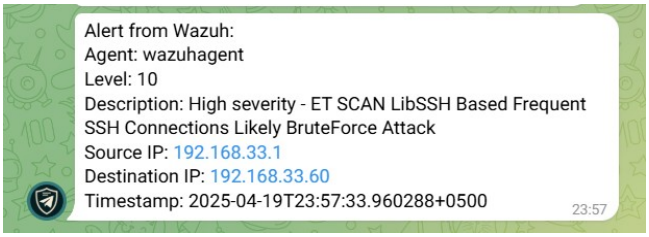

Figure 12: SSH Brute-Force Attack notification in Telegram.

Figure 12 confirms that the SSH brute-force attack alert was successfully forwarded to Telegram, providing real-time notification.

In this attack scenario, an HTTP-based attack was emulated with the use of the Social-Engineer Toolkit on a Kali virtual machine against a Windows host. During the attack, a straightforward exploitation of a web application vulnerability was employed where a user could enter his or her login credentials. During the attack, the login credentials of the user, including a password, were sent in plain text through the use of the HTTP protocol.

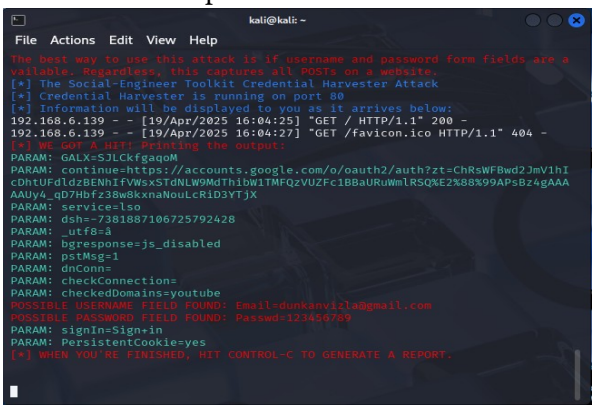

Figure 13: SE Attack using SE Toolkit on Kali machine.

Figure 13 confirms the execution of a SE attack using the SE Toolkit on the Kali machine during the simulation.

Suricata, monitoring the network traffic, detected the unencrypted transmission of sensitive data. It identified the HTTP request containing the login information, including the cleartext password, and generated an alert based on this suspicious traffic.

Wazuh, configured to process network-based alerts from Suricata, correlated the event with a rule designed to detect the transmission of unencrypted passwords. Upon detection, Wazuh issued a high-severity alert indicating that sensitive credentials had been sent in cleartext.

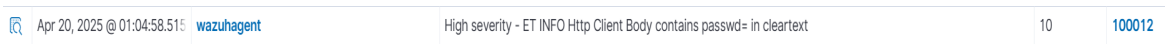

Figure 14: SE Attack alert in Wazuh Dashboard.

Figure 14 confirms that the SE attack was detected and displayed as an alert in the Wazuh Dashboard.

The warning was then sent to the custom Telegram bot, which, in automatic mode, alerted the specified Telegram chat with information on the incident and severity level. This was evidence of

success of the combined system in identifying unencrypted transmission of credentials and alerting administrators of possible security risk in advance.

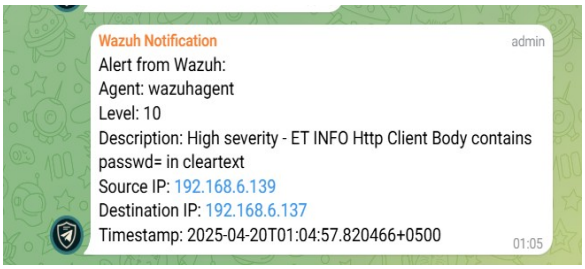

Figure 15: SE Attack notification in Telegram.

Figure 15 confirms that the SE attack alert was successfully forwarded to Telegram, providing real-time notification to the user.

Throughout the test cycle, the system performed consistently. Suricata and Wazuh integration ensured proper detection of all attack vectors with. Alerts were raised and sent to the Telegram chat within seconds of detection, thus ensuring timely security intrusion reporting. The system was optimal within the virtualized environment without noteworthy resource overhead.

Table 3 shows the performance metrics by attack types. The developed system was evaluated according to key metrics: the average notification delivery time was less than one second, all test attack scenarios were successfully detected without false positives, and the load on resources remained low, which confirms the suitability of the solution for low-cost infrastructure.

Table 3
Performance metrics table by attack type

Attack Scenario	Detection Tool	Alert Severity	Telegram Delivery Delay (s)	Detection Result	False Positives	Notes
Port Scanning	Suricata + Wazuh	Medium	0.094	Detected & Alert Sent	0	OS fingerprinting detected in real time; Telegram alert delivered faster than dashboard update
SSH Brute-force	Suricata + Wazuh	High	0.983	Detected & Alert Sent	0	Brute-force activity correctly correlated and prioritized; real-time alert validated
Malicious HTTP	Suricata + Wazuh	High	0.102	Detected & Alert Sent	0	Unencrypted password transmission detected and alerted
Port Scanning	Splunk Enterprise Security	Medium	within 2-3	Detected & Alert Sent	0	Splunk Enterprise Security identified port scanning activity and issued a medium-severity alert using its network traffic analysis and correlation mechanisms [18].
SSH Brute-force	Splunk Enterprise Security	High	within 2-4	Detected & Alert Sent	0	The system correlated multiple failed logins attempts and generated a high-severity alert, effectively indicating an ongoing brute-force attack [18].

Malicious HTTP	Splunk Enterprise Security	High	within 1-2	Detected & Alert Sent	0	Splunk detected the unencrypted transmission of user credentials and classified it as a high-severity incident, providing detailed context for subsequent incident response [18].

The comparison showed that the developed system based on Wazuh, Suricata and Telegram Bot demonstrates a level of detection accuracy comparable to commercial SIEM solutions (Splunk, QRadar), while remaining significantly more cost-effective. The notification delivery time was less than one second, which exceeds the speed of notifications in traditional systems using dashboards or e-mail. The key advantages of the solution are flexible configuration of correlation rules and lack of vendor dependence, which makes it the best option for organizations with a limited budge.

The comparison showed that the developed system based on Wazuh, Suricata and Telegram Bot demonstrates a level of detection accuracy comparable to commercial SIEM solutions (Splunk, QRadar), while remaining significantly more cost-effective. The notification delivery time was less than one second, which exceeds the speed of notifications in traditional systems using dashboards or e-mail. The key advantages of the solution are flexible configuration of correlation rules and lack of vendor dependence, which makes it the best option for organizations with a limited budge.

The results of the experiment show that the proposed Wazuh + Suricata + Telegram Bot API architecture achieves detection performance levels comparable to leading commercial SIEM solutions such as Splunk or IBM QRadar, which often require significant investments in infrastructure and licensing [19,20]. Moreover, the open-source nature of the solution allows for significant savings: implementation and maintenance costs are significantly lower compared to commercial offerings, making continuous monitoring cost-effective for small and medium-sized organizations [21]. These results confirm that a modular, open-source approach can provide enterprise-grade threat detection capabilities while maintaining visibility, flexibility, and cost-effectiveness.

Using Telegram introduces data security considerations. All communications occur via HTTPS, and the bot validates messages through a preconfigured chat_id whitelist. Sensitive information such as payloads or credentials is never transmitted over Telegram. Data retention and backup policies ensure that all logs and PCAPs are encrypted using LUKS and rotated every 30 days. Backups are automatically mirrored to a secure off-site repository. This minimizes risks associated with external platform integration.

7. Conclusion

The study addressed a key scientific objective: developing and experimentally validating the effectiveness of an integrated architecture for network threat detection and alerting. This architecture combines the functionality of Wazuh (SIEM), Suricata (IDS/IPS), and the Telegram Bot API into a single modular platform. The research's novelty lies in the development of a comprehensive approach that combines event correlation, network traffic analysis, and instant alerting mechanisms within a single adaptive solution. For the first time, an architecture has been proposed that provides an optimal balance between incident detection accuracy, system response speed, and configuration flexibility with minimal operational costs. The results of experimental testing confirmed the proposed architecture's high effectiveness in detecting various types of cyberattacks—from ICMP/TCP floods and network scanning to SSH brute-force attacks, SQL injections, and DNS exfiltration. The calculated results demonstrated a precision of 0.90–0.98 and a recall of 0.83–0.97, with an average F1 value of 0.93, indicating high threat classification accuracy and resistance to false alarms. The average delay in notification delivery via the Telegram Bot API was approximately one second, ensuring a real-time system response.

A comparative analysis with commercial SIEM platforms (Splunk, QRadar) showed comparable accuracy and performance. Thus, the use of open source software confirms the feasibility of creating

effective, scalable, and cost-effective enterprise-grade security solutions without relying on proprietary technologies. The practical value of this work lies in the development of a reproducible architecture adapted for use in research laboratories, educational institutions, and resource-constrained organizations.

Future research opportunities include integrating machine learning algorithms [22] for intelligent anomaly detection, expanding automated incident response capabilities, and adapting the system to hybrid and cloud infrastructures to improve predictability, scalability, and resilience against multi-stage cyberattacks.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] M.H. Nguyen. 2024. *Security and Threat Detection through Cloud-Based Wazuh Deployment.* International Journal of Computer Applications, 182(4), 25–31. DOI: 10.1109/KHI-HTC60760.2024.10482206**.**

[2] Rahmawati, T. (2024). Enhancing Network Security Through Real-Time Threat Detection with Intrusion Prevention System (Case Study on Web Attack). Jurnal Ilmiah Teknik Elektro Komputer dan Informatika, 10(4), 1004–1020.

[3] Satrio, R. (2022). Send OpenSearch Dashboard Alert via Telegram Messenger. ITSEC Asia Research & Technology.

[4] Lisnevskyi R., Askarbekova N., Lisnevskyi V., Babenko T., Alin G., Ihor D. Using Kali Linux as a Method of Defense Against Attacks (2025) SIST 2025 - 2025 IEEE 5th International Conference on Smart Information Systems and Technologies, Conference Proceedings, doi: 10.1109/SIST61657.2025.11139305.

[5] Babenko T., Kolesnikova K., Lisnevskyi R., Makilenov S., Landovsky Y. Definition of Cryptojacking Indicators (2024) CEUR Workshop Proceedings, 3680, https://www.scopus.com/inward/record.uri?eid=2-s2.0-85192508442&partnerID=40&md5=3dfe1914b48d27c693ac1db293ed15c5.

[6] Setiawan, H., & Sulistyo, W. (2023). SIEM (Security Information Event Management) Model for Malware Attack Detection Using Suricata and Evebox. International Journal of Engineering Technology and Natural Sciences, 5(2), 138–147.

[7] Wasswa, H., Lynar, T., Nanyonga, A., & Abbass, H. (2025). IoT Botnet Detection: Application of Vision Transformer to Classification of Network Flow Traffic.

[8] Alanda, A. (2023). Real-time Defense Against Cyber Threats: Analyzing Wazuh's Effectiveness in Server Monitoring. Jurnal Teknologi Informasi dan Ilmu Komputer, 7(2), 56–62. DOI: 10.25077/jitce.7.2.56-62.2023.

[9] T. Davies, M.H. Eiza, N. Shone, and R. Lyon. 2025. *A Collaborative Intrusion Detection System Using Snort IDS Nodes.* arXiv preprint arXiv:2504.16550. Available at: https://arxiv.org/abs/2504.16550.

[10] D. Palko, H. Hnatiienko, T. Babenko, and A. Bigdan, “Determining key risks for modern distributed information systems,” CEUR Workshop Proceedings, 2021. [Online]. Available: https://www.scopus.com/record/display.uri?eid=2-s2.0-85120940899.

[11] T. Babenko, H. Hnatiienko, and V. Vialkova, “Modeling of the integrated quality assessment system of the information security management system,” CEUR Workshop Proceedings, 2021. [Online]. Available: https://www.scopus.com/record/display.uri?eid=2-s2.0-85104030744.

[12] Babenko T., Amanzholova S., Lisnevskyi R., Abylgazy A. Cybersecurity-level assessment models (2025) CEUR Workshop Proceedings, 3966, https://www.scopus.com/inward/record.uri?eid=2-s2.0-105006928135&partnerID=40&md5=9e153080855fae60730f5fdf2d73067b.

[13] Yakubova, M.; Serikov, T.; Manankova, O. Development and Research of a Method for MultiLevel Protection of Transmitted Information in IP Networks Based on Asterisk IP PBX Using Various Codecs. *Int. J. Adv. Comput. Sci. Appl.* 2024, *15*(7), 724–731.
[14] M. Almiani, A. Razaque, B. Alotaibi, S. Amanzholova and A. Alotaibi, "An Efficient Data-Balancing Cyber-Physical System Paradigm for Quality-of-Service (QoS) Provision over Fog Computing," *Applied Sciences*, vol. 12, no. 1, p. 246, 2022, doi: 10.3390/app12010246.
[15] Mubarakova, S. R., Amanzholova, S. T., & Uskenbayeva, R. K. (2022). Using machine learning methods in cybersecurity. *Eurasian Journal of Mathematical and Computer Applications*, 10(1), 69–78.
[16] ENISA Threat Landscape 2024, European Union Agency for Cybersecurity. DOI: 10.2824/401373.
[17] IBM QRadar Security Intelligence Platform. Product Overview, 2024.
[18] Claise, B., et al. (2023). Comparative Evaluation of Snort and Suricata IDS in Enterprise Networks. Journal of Network Security, 15(3), 102–117.
[19] B. Pandey, K. Kumar, P. Pandey, L. Aldasheva, B. Altaiuly, and W. A. W. A. Bakar, Cryptography tools in ethical hacking. 2025. doi: 10.1201/9781003508632-15.
[20] Amanzholova, S., Galimkair, M., Муханов, C., Olga, U., & Razaque , A. (2025). AI-Powered System for Network Activity Monitoring and Detection of SQL Injection Attacks Using Zabbix and Grafana. International Journal of Information and Communicatin Technologies, 6(3), 61–83. https://doi.org/10.54309/IJICT.2025.23.3.004.
[21] Ussatova, O., Makilenov, S., Karyukin, V., Razaque, A., Amanzholova, S. The Development of an Evaluation Model for User Authentication Methods with Security, Usability, and Usage Frequency // Eastern-European Journal of Enterprise Technologies. — 2025. doi: 10.15587/1729-4061.2025.333720.
[22] Hamada, M., Abiche, A., & Hamada, G. (2023). Using the machine learning models to optimize time management in logistics and supply chain management systems. https://ceur-ws.org/Vol-3966/W3Paper6.pdf.