Experimental Web Server Incident Detection Based on Wazuh-Snort using SPDLC Method

Agus Wijayanto[1], Ahmad Yazid Bustomi[1], Syaddam[2] and Tri Sudinugraha[3]
[1,1]Information Technology, Faculty of Computer Science, Universitas Mulia, Balikpapan, Indonesia
[2]Information System, Politeknik Bisnis Kaltara, Tarakan, Indonesia
[3]Faculty of Computer Science and Information Technology, Universiti Malaysia Sarawak, Sarawak, Malaysia

Corresponding author: Agus Wijayanto (e-mail: aguswijayanto@universitasmulia.ac.id).

ABSTRACT Web servers are essential but vulnerable to network attacks such as SYN Flood and port scanning. This study designs and implements an incident detection system by integrating Wazuh, Snort (IDS), and Telegram notifications following the Security Policy Development Life Cycle. Snort functions as a network sensor generating alerts forwarded to Wazuh for centralized log management and correlation; critical alerts are pushed to administrators via a Telegram bot for real-time response. The experimental environment uses virtualized machines that host a Wazuh server, a Snort IDS, a web server, and an attacker node. Evaluation metrics include detection accuracy, false-negative rate, and notification response time under two attack scenarios: SYN Flood and port scanning. Results indicate detection accuracy of 41.34% and a false-negative rate of 58.6% for SYN Flood attacks, with Snort-to-Wazuh latency of 13.559 ms and Telegram delivery of 0.4 s. Port scanning was detected with 100% accuracy and 0% false-negative rate, recording Snort-to-Wazuh latency of 1.490 ms and Telegram delivery of 1.33 s. The integration enhances centralized visibility and accelerates administrator awareness, yet it exhibits limitations for high-volume DoS traffic due to log buffering and throughput constraints. Recommendations include increasing Wazuh agent buffer capacity, optimizing Snort rules, and deploying higher-capacity hardware to improve detection under heavy attack loads.

KEYWORDS Intrusion Detection, Security Policy Development Life Cycle, Snort, Wazuh

I. INTRODUCTION

The rapid development of information technology has positioned web applications at the forefront of interactions between organizations and users. Web servers, as the primary infrastructure responsible for handling client requests, play a vital role in ensuring the availability of information [1]. However, their exposure to public networks makes them prime targets for various cyberattacks [2][3]. According to recent cybersecurity reports [4], s attacks targeting web infrastructure continue to increase in both frequency and complexity, threatening the confidentiality, integrity, and availability (CIA) of data [5].

One of the most significant threats to web server availability is Distributed Denial of Service attacks, particularly SYN Flood attacks [6][7]. This type of attack overwhelms the server with forged connection requests, exhausting server resources and preventing legitimate users from accessing services. In addition, Port Scanning attacks are commonly employed during the reconnaissance phase to identify open ports and potential security vulnerabilities [8]. Failure to detect such activities at an early stage can have severe consequences for organizational operations [9].

To mitigate these threats, the use of Intrusion Detection Systems (IDS) such as Snort has become a widely adopted standard. Snort is well known for its capability to perform deep packet inspection and detect signature-based attacks [10]. Nevertheless, standalone IDS implementations often face challenges in log management. Snort generates a large volume of log data that is difficult to analyze manually, making it challenging for administrators to quickly derive meaningful insights into the overall security posture of the network [11].

On the other hand, Security Information and Event Management (SIEM) technologies such as Wazuh provide solutions for centralized log management and event correlation [12][13]. Wazuh can collect data from multiple agents and present it through informative dashboard visualizations. Several previous studies have demonstrated the effectiveness of Wazuh in monitoring security compliance and file integrity [14][15][16]. However,

Wazuh's detection capabilities are often limited to operating system and application-level logs, making it less sensitive to specific network traffic anomalies unless they are integrated with external network monitoring tools.

Compared to previous studies that focused on Wazuh's file integrity [14], log-based attack detection [15], or SOC implementation with ELK [16], the present study makes several distinct contributions. First, instead of relying solely on operating system and application logs, our system integrates Snort as a network-based intrusion detection sensor, enabling packet-level analysis for SYN Flood and port scanning attacks. Second, we incorporate Telegram-based real-time alerting, thereby reducing the delay between attack occurrence and administrator awareness. Third, we provide quantitative evaluation using detection accuracy, false negative rate, and response time metrics under two distinct attack scenarios, whereas previous works primarily focused on qualitative or compliance-oriented assessments. Fourth, the adoption of the SPDLC methodology ensures that security policies are systematically aligned with risk analysis. Finally, our experimental environment uses a physical network topology with a MikroTik router and dedicated LAN, providing a more realistic testbed compared to fully isolated virtual networks.

A fundamental weakness of using IDS or SIEM systems independently lies in response time [17]. In many cases, security systems merely record attack events in log databases without providing immediate alerts to administrators [18]. This creates a significant time gap between the occurrence of an attack and the administrator's awareness of the incident, particularly when attacks occur outside operational working hours. Such delays provide attackers with additional time to further exploit the system before mitigation actions can be taken.

Therefore, an integrative approach is required to combine the strengths of Snort's packet inspection capabilities with Wazuh's log analysis and visualization features. This integration enables Snort to function as a network sensor that forwards alerts to the Wazuh agent for further analysis. As a result, administrators gain comprehensive visibility across multiple layers, ranging from network packets to operating system-level events, within a single centralized interface.

Furthermore, to address notification delays, the detection system must be equipped with a real-time alert mechanism [19] . The utilization of instant messaging platforms such as Telegram through a Bot API offers an effective and efficient solution. With this approach, high-priority alerts detected by the integrated Wazuh–Snort system can be instantly delivered to the administrator's mobile device, enabling significantly faster incident response compared to manual dashboard monitoring.

In designing such a complex security system, a structured approach is essential to ensure that implemented security policies are appropriately targeted. This study adopts the Security Policy Development Life Cycle (SPDLC) methodology [20] [21]. Unlike general software development methods, SPDLC focuses on the lifecycle of security policies, including asset identification, risk analysis, policy design, implementation, testing, and evaluation. The use of this method ensures that the integration of security tools is not only technically sound but also aligned with previously identified risk mitigation requirements.

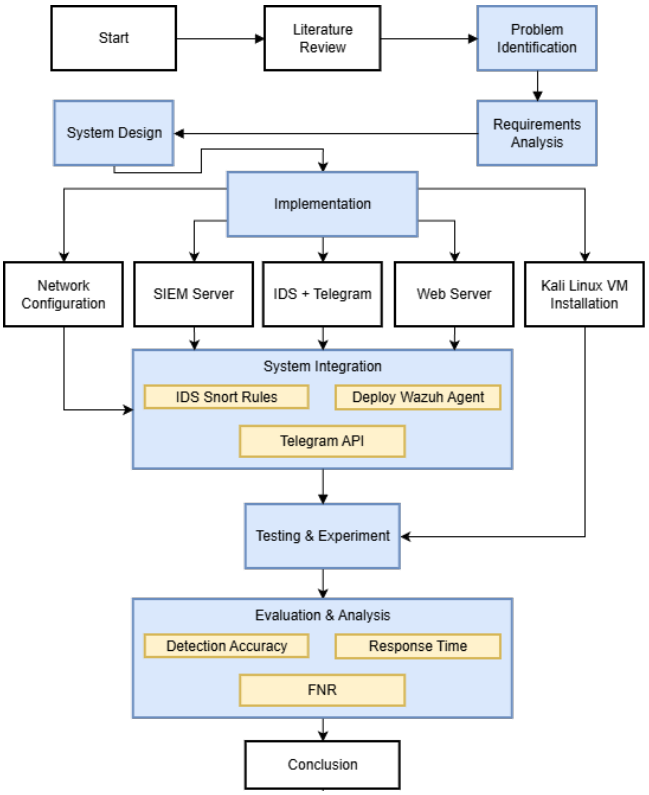

Figure 1. Research methodology based on Security Policy Development Life Cycle (SPDLC) consisting of six phases: Identification, Analysis, Design, Implementation, Testing, and Evaluation

Based on this background, this study aims to design and implement an incident detection system for web servers by integrating Wazuh, Snort, and Telegram using the SPDLC method. The primary focus of the research is to evaluate the effectiveness of the system in detecting SYN Flood and Port Scanning attacks, as well as to measure system performance based on detection accuracy and notification response time received by administrators.

II. METHODOLOGY RESEARCH

This study employs a descriptive quantitative research design to evaluate the performance of the integrated Wazuh, Snort, and Telegram detection system. The Security Policy Development Life Cycle (SPDLC) framework was adopted to guide the systematic development and testing process, comprising six stages: identification, analysis, design, implementation, testing, and result analysis.

Figure 1 illustrates the research methodology based on the Security Policy Development Life Cycle (SPDLC). The methodology is structured into six main phases: identification, analysis, design, implementation, testing, and evaluation.

Each phase is conducted sequentially to ensure that the developed security system aligns with identified threats, security requirements, and evaluation objectives. The SPDLC framework was selected because it emphasizes security policy alignment throughout the system lifecycle, making it suitable for intrusion detection system research.

A. IDENTIFICATION PHASE

The identification phase focuses on determining critical assets and system components that require protection. In this study, the primary asset is the web server that provides public-facing services and is exposed to network-based attacks. Supporting assets include network services, open ports, and log data generated by security monitoring components. This phase establishes the security scope and defines the protection objectives based on confidentiality, integrity, and availability (CIA) principles.

B. ANALYSIS PHASE

During the analysis phase, potential security threats and risks related to the identified assets are examined. Two types of network attacks are selected for evaluation: SYN Flood attacks and Port Scanning attacks. SYN Flood attacks represent availability-based threats that aim to exhaust server resources, while Port Scanning attacks are commonly used during reconnaissance to identify open services and vulnerabilities. These attack scenarios were chosen to evaluate the system's capability to detect both high-volume traffic attacks and reconnaissance activities.

C. DESIGN PHASE

The design phase defines the architecture of the proposed detection system. Snort is deployed as a network-based intrusion detection system (NIDS) to perform packet inspection and signature-based attack detection. Wazuh is implemented as a Security Information and Event Management (SIEM) platform to centralize logs, correlate events, and visualize security incidents. Telegram is integrated as a real-time alert delivery mechanism to notify administrators immediately after an attack is detected. The interaction between these components is designed to ensure continuous monitoring and rapid incident awareness.

D. IMPLEMENTATION PHASE

The implementation phase involves deploying the experimental environment and configuring each system component according to the design specifications. A virtualized environment is prepared to simulate real-world network conditions. The virtualized environment was built using VMware Workstation 17.6.0 on a host laptop with an Intel Core i7-1185G7 processor, 32 GB of RAM, and a 1 Gbps USB LAN adapter connected to a MikroTik RB952 router configured as a DHCP server. Four virtual machines (VMs) were created with the following specifications: (1) Wazuh server VM: 2 vCPUs, 5.7 GB RAM, 50 GB HDD, running Ubuntu 20.04.6 LTS; (2) Snort IDS and Telegram VM: 2 vCPUs, 5.7 GB RAM, 50 GB HDD, running Ubuntu 20.04.6 LTS; (3) Web server VM: 2 vCPUs, 4 GB RAM, 50 GB HDD, running Ubuntu Server 22.04.4 LTS with Nextcloud as the target web application; (4) Attacker VM: 2 vCPUs, 4 GB RAM, 42 GB HDD, running Kali Linux 2024.3. All VMs were connected to the same local area network using bridge mode and obtained IP addresses from the DHCP server in the range 192.168.1.0/28. This configuration ensures reproducible experimental conditions for other researchers. Snort detection rules are configured to identify SYN Flood and Port Scanning activities. The Wazuh agent is deployed to collect and process security logs, while the Wazuh manager handles event correlation and alert generation. Additionally, a Telegram Bot API is configured to enable automated alert notifications when predefined security thresholds are met.

E. EVALUATION PHASE

To evaluate the effectiveness of the proposed web server incident detection system, several performance metrics were employed. These metrics are commonly used in intrusion detection system (IDS) evaluation to assess detection capability, reliability, and system responsiveness. The evaluation focuses on three main aspects: detection accuracy, response time, and false negative rate (FNR).

- **DETECTION ACCURACY**

Detection accuracy is used to measure the system's ability to correctly classify network events as either attack or normal traffic. This metric is derived from the confusion matrix, which is a standard evaluation method in classification-based security systems, including IDS and SIEM platforms. Based on these parameters, detection accuracy is calculated using the following formula [22]:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \; x \; 100\% \qquad (1)$$

A higher accuracy value indicates that the detection system performs well in distinguishing malicious traffic from legitimate network activity. In the context of this research, accuracy reflects the effectiveness of the integrated Wazuh–Snort system in detecting SYN Flood and Port Scanning attacks.

- **RESPONSE TIME**

Response time refers to the time interval between the moment an attack is detected by the intrusion detection mechanism and the delivery of an alert notification to the administrator. This metric is critical in security monitoring systems, as faster response times allow administrators to take immediate mitigation actions and reduce potential damage. In this research, response time is measured in two stages: Detection latency and Notification Latency.

- **FALSE NEGATIVE RATE**

False Negative Rate (FNR) is a crucial metric in intrusion detection evaluation, as it represents the proportion of attacks that are not detected by the system. A high FNR indicates that the system fails to recognize malicious activities, which may allow attackers to exploit vulnerabilities without being noticed. The FNR is calculated using the following formula [23]:

$$FNR = \frac{FN}{FN+TP} \; x \; 100\% \qquad (2)$$

This metric is particularly important in the evaluation of Denial-of-Service–related attacks, such as SYN Flood, where high traffic volume may overwhelm detection components and cause missed alerts. In this study, FNR analysis helps identify limitations in the log processing capacity and detection throughput of the integrated Wazuh–Snort system.

To evaluate the proposed system, two attack scenarios were conducted. The experimental evaluation was carried out by launching attack traffic from a Kali Linux machine targeting a web server within a virtualized environment. Snort functioned as a network-based intrusion detection system to identify malicious traffic patterns, while Wazuh processed and correlated detection logs. Alert notifications were then

delivered to the administrator through Telegram. Specifically, SYN Flood attacks were generated using the hping3 tool with the --flood option, sending TCP SYN packets to port 80 of the web server. Port Scanning attacks were performed using the Legion tool to discover open ports on the target. Each attack scenario was executed under controlled network conditions, and quantitative metrics were recorded for analysis

III. *RESULTS AND DISCUSSION*

This section presents the findings from the implementation and testing of the integrated Wazuh-Snort detection system. The evaluation focuses on the system's ability to detect SYN Flood and Port Scanning attacks, as well as its performance in terms of detection accuracy, response time, and false negative rate.

A. SYN FLOOD DETECTION RESULTS

The SYN Flood attack experiment was conducted to evaluate the system's detection capability under high-volume denial-of-service conditions. This type of attack generates many TCP SYN packets intended to exhaust server resources and disrupt service availability. Figure 2 presents the runtime statistics generated by Snort during the SYN Flood attack experiment. The results indicate that Snort processed more than 1.27 million packets within approximately three minutes, achieving a throughput of 6,834 packets per second. A high packet drop rate was observed, reflecting the impact of excessive SYN traffic on packet inspection capability. Additionally, a total of 525,644 alerts were generated,

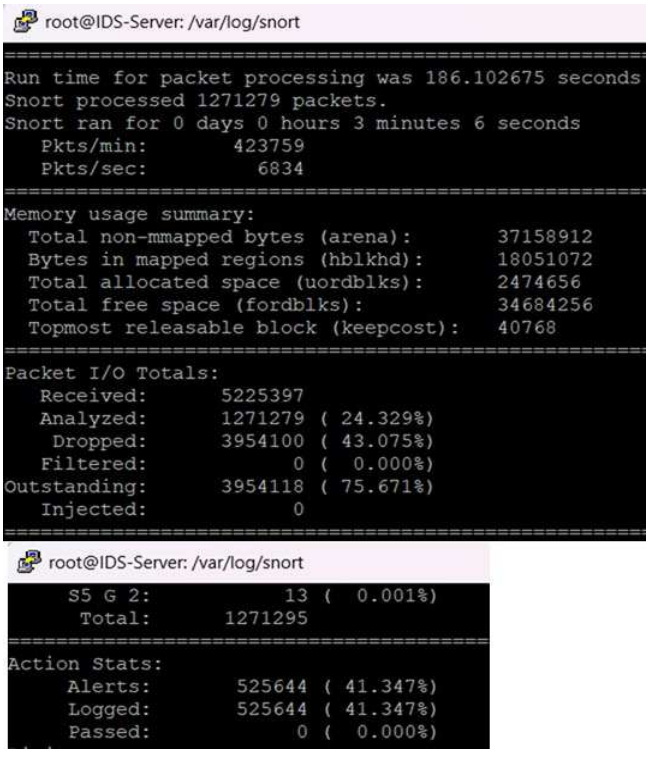

Figure 2. Snort packet processing and alert statistics under SYN Flood attack conditions

demonstrating Snort's ability to detect abnormal traffic patterns under high-load conditions. However, the significant packet drop, and unprocessed traffic contribute to the increased false negative rate observed in this experiment.

TABLE I
SYN FLOOD DETECTION RESULTS

Matric	Value
True Positive (TP)	525.644
True Negative (TN)	0
False Positive (FP)	0
False Negative (FN)	745.635
Detection Accuracy	41.34%
False Negative Rate	58.6%

Table I presents the detection performance of the system during the SYN Flood attack experiment. The detection accuracy of 41.34% indicates that less than half of the attack events were successfully identified by the system. Meanwhile, the high false negative rate of 58.6% shows that a significant portion of attack traffic was not detected. These results highlight the challenges faced by signature-based IDS mechanisms when processing extremely high traffic volumes within a limited time frame.

B. PORT SCANNING DETECTION RESULTS

The Port Scanning experiment was performed to evaluate the system's capability to detect reconnaissance-based attacks. This type of attack is typically used by attackers to identify

open ports and running services before launching further exploitation attempts.

TABLE II
PORT SCAN DETECTION RESULTS

Matric	Value
True Positive (TP)	6
True Negative (TN)	0
False Positive (FP)	0
False Negative (FN)	0
Detection Accuracy	100%
False Negative Rate	0%

Table II shows that the integrated detection system successfully detected all Port Scanning activities without generating false negatives. The perfect detection accuracy demonstrates that signature-based detection is highly effective for reconnaissance attacks, which generate distinct and identifiable traffic patterns.

C. RESPONSE TIME MEASUREMENT RESULTS

Response time analysis was conducted to evaluate how quickly the proposed system delivers alert notifications after an attack is detected. Fast response time is a critical factor in intrusion detection systems, as it enables administrators to respond promptly and minimize the impact of security incidents.

TABLE III
RESPONSE TIME MEASUREMENT RESULTS

Attack Scenario	Snort–Wazuh Detection Latency	Telegram Alert Delivery Time
Syn Flood	13.559 ms	0.4 s
Port Scanning	1.490 ms	1.33 s

Table III summarizes the response time measurements for both attack scenarios. The results indicate that detection latency between Snort and Wazuh remains within milliseconds, while alert notifications are delivered to administrators within seconds. These findings confirm that the integration of Telegram as an alerting mechanism significantly improves real-time incident awareness.

TABLE IV
LOG TIMESTAMP COMPARISON BETWEEN SNORT AND WAZUH

No	Wazuh Log Time	Snort Log Time	Time Difference (ms)
1	02:19:46.377	02:19:32.841	13.536
2	02:19:46.391	02:19:32.841	13.550
3	02:19:46.391	02:19:32.841	13.550
4	02:19:46.395	02:19:32.841	13.554
5	02:19:46.398	02:19:32.841	13.557
6	02:19:46.398	02:19:32.841	13.557
7	02:19:46.404	02:19:32.841	13.563
8	02:19:46.407	02:19:32.841	13.566
9	02:19:46.416	02:19:32.841	13.575
10	02:19:46.423	02:19:32.841	13.582
Average			**13.559**

Table IV presents a detailed comparison between Snort log timestamps and the corresponding events recorded by the Wazuh SIEM. The time difference represents the processing latency required for log forwarding, parsing, and correlation after the detection event is generated by Snort. Based on ten observation samples, the average detection latency was measured at 13.559 milliseconds. The consistency of the measured values indicates stable and reliable communication between the IDS and SIEM components.

From a theoretical perspective, response time in SIEM-based intrusion detection architectures is influenced by log transmission mechanisms, event normalization, and correlation processing. Previous studies indicate that millisecond-level latency is acceptable for near real-time security monitoring. Therefore, the results obtained confirm that the proposed Wazuh, Snort, and Telegram integration fulfill real-time detection and alerting requirements

TABLE V
LOG TIMESTAMP COMPARISON BETWEEN SNORT AND TELEGRAM

No	Telegram Notification Time	Snort Log Time	Time Difference (ms)
1	02:18:42	02:18:42	0
2	02:18:42	02:18:42	0
3	02:18:46	02:18:46	0
4	02:18:46	02:18:46	0
5	02:18:50	02:18:50	0
6	02:18:50	02:18:50	0
7	02:18:54	02:18:53	1
8	02:18:54	02:18:53	1
9	02:18:58	02:18:57	1
10	02:18:58	02:18:57	1
Average			**0.4**

Table V presents the timestamp comparison between Snort detection logs and the corresponding alert notifications delivered via Telegram. The time difference represents the end-to-end alert delivery latency, which includes log forwarding, event processing by Wazuh, and message transmission through the Telegram API. Based on ten observation samples, the average alert delivery time was measured at 0.4 seconds. Most notifications were delivered simultaneously with the Snort detection event, while minor delays of one second occurred due to network and messaging service latency. These results confirm that Telegram provides near real-time alert delivery suitable for security monitoring applications.

When combined with the Snort–Wazuh latency measurements presented in Table IV, the results indicate that most of the response time overhead originates from alert delivery rather than detection processing. According to real-time alerting theory in security monitoring systems, alert delivery delays below one second are considered acceptable for operational incident response. The results demonstrate that the proposed alerting mechanism satisfies real-time requirements without introducing significant delays.

Table VI presents the timestamp comparison between Snort detection logs and the corresponding events recorded by the Wazuh SIEM during the Port Scanning attack experiment. The

measured time difference represents the processing latency required for log forwarding and event correlation. Based on six observation samples, the average detection latency was 1.490 milliseconds. Compared to the SYN Flood scenario, the Port Scanning attack produced significantly lower latency due to reduced traffic volume and simpler detection patterns

TABLE VI
LOG TIMESTAMP COMPARISON BETWEEN SNORT AND WAZUH SIEM (PORT SCANNING)

No	Wazuh Log Time	Snort Log Time	Time Difference (ms)
1	02:09:35.299	02:09:34.572	1.273
2	02:09:39.223	02:09:37.581	2.358
3	02:09:47.232	02:09:46.451	1.219
4	02:09:53.236	02:09:51.844	2.608
5	02:09:55.239	02:09:54.000	1.239
6	02:09:59.244	02:09:59.000	0.244
Average			**1.499**

TABLE VII
LOG TIMESTAMP COMPARISON BETWEEN SNORT AND TELEGRAM (PORT SCANNING)

No	Telegram Notification Time	Snort Log Time	Time Difference (ms)
1	02:09:35	02:09:34	1
2	02:09:39	02:09:37	2
3	02:09:47	02:09:46	1
4	02:09:52	02:09:51	1
5	02:09:56	02:09:54	2
6	02:10:00	02:09:59	1
Average			**1.33**

Table VII shows the comparison between Snort detection timestamps and the alert notifications delivered via Telegram during the Port Scanning attack scenario. The time difference reflects the end-to-end alert delivery latency, including SIEM processing and message transmission through the Telegram API. The average alert delivery time of 1.33 seconds indicates that the alerting mechanism remains responsive and suitable for real-time security monitoring, despite minor delays introduced by external messaging services.

It is important to note that both Table VI and Table VII are derived from the same Port Scanning attack events. When Snort detects an attack, it simultaneously forwards the alert to two different destinations: (1) to the Wazuh SIEM for centralized log management and correlation (Table VI), and (2) to the Telegram bot for real-time administrator notification (Table VII). Therefore, the timestamps in both tables originate from the same Snort detection events but represent two parallel processes. The higher precision in Table VI (milliseconds) is due to Wazuh's detailed log parsing, while Telegram notifications are recorded at second-level granularity as provided by the Telegram Bot API. These two measurements are complementary rather than duplicate, with Table VII reflecting detection latency and Table VII reflecting notification latency.

These detailed measurements validate the response time values summarized in Table III and demonstrate consistent system performance across different attack scenarios.

D. DISCUSSION

This section discusses the experimental results obtained from the implementation of the integrated Wazuh, Snort, and Telegram detection system using the SPDLC methodology. The discussion focuses on detection performance, response time behavior, and system limitations under different attack scenarios.

The experimental results demonstrate that the proposed system can detect both SYN Flood and Port Scanning attacks with varying levels of effectiveness. Port Scanning attacks achieved a detection accuracy of 100%, indicating that Snort signature-based detection performs optimally for reconnaissance-type attacks characterized by low traffic volume and distinct scanning patterns. This finding aligns with intrusion detection theory, where signature-based IDS systems are highly effective in detecting well-defined attack signatures with minimal packet inspection overhead.

In contrast, the SYN Flood attack scenario exhibited lower detection accuracy and a higher false negative rate. This behavior can be attributed to the nature of SYN Flood attacks, which generate a massive volume of TCP SYN packets in a short period. Under such conditions, Snort experiences increased packet processing load, resulting in packet drops and unprocessed traffic. As shown by the Snort runtime statistics, high traffic intensity directly impacts inspection performance, which is a well-documented limitation of signature-based IDS systems when operating under resource exhaustion attacks.

Despite this limitation, the integration with Wazuh SIEM provides additional value by centralizing alerts and correlating security events. Although Wazuh does not directly analyze raw network traffic, its ability to aggregate and visualize Snort alerts enables administrators to maintain situational awareness even when detection accuracy is affected by high traffic volumes.

Response time analysis indicates that the proposed system satisfies near real-time detection and notification requirements. Based on the results summarized in Table III, detection latency between Snort and Wazuh remains within the millisecond range for both attack scenarios. Detailed timestamp comparisons further confirm stable and consistent log forwarding performance.

For the SYN Flood scenario, the average Snort–Wazuh detection latency was measured at 13.559 milliseconds. This relatively higher latency compared to Port Scanning is expected due to increased packet inspection overhead and event generation frequency. Conversely, Port Scanning attacks resulted in significantly lower average detection latency of 1.490 milliseconds, reflecting lighter traffic load and simpler detection patterns.

The measured detection latencies are within the millisecond range, which is considered acceptable for near real-time

intrusion detection. Notably, the higher latency observed in the SYN Flood scenario is directly attributable to the massive packet volume, which increases the processing burden on Snort and the event forwarding queue of the Wazuh agent. In contrast, the Port Scanning scenario involves only a few dozen packets, resulting in near-instantaneous detection. These findings confirm that the proposed Wazuh, Snort, and Telegram integration satisfies real-time requirements for both high-volume and low-volume attack scenarios, with the limitation that extreme traffic volumes can introduce additional latency due to buffering constraints.

Furthermore, Thomas and Bhat [19] reported that Telegram Bot API delivers messages with typical latencies below two seconds under normal network conditions. Our measured notification times are consistent with this range, confirming that Telegram is a suitable real-time alerting channel for intrusion detection systems.

From an SPDLC perspective, the experimental results indicate that the implemented security controls successfully address the risks identified during the analysis phase. The integration of Snort as a network-based IDS fulfills the requirement for packet-level attack detection, while Wazuh provides centralized log management and visualization capabilities. The addition of Telegram-based alerts enhances the response phase by reducing the time gap between attack occurrence and administrator awareness.

The structured SPDLC approach ensures that system design decisions are aligned with security objectives rather than technical implementation. This is evident in the clear mapping between identified threats (SYN Flood and Port Scanning), selected detection mechanisms, and evaluation metrics used to assess system performance.

E. LIMITATIONS

First, the performance of Snort is significantly affected under high-volume SYN Flood attacks, leading to increased packet loss and a high false negative rate (58.6%). Our analysis indicates that the default buffer configuration of the Wazuh agent—specifically the queue_size parameter set to 5000 events—contributes to event dropping when the incoming alert rate exceeds the forwarding capacity. This finding aligns with [17], who identified queue management as an intrinsic weakness of signature-based IDS under volume-based attacks. Similarly, Sheeraz et al [18] demonstrated that SIEM correlation engines require adequate internal buffering to prevent event loss during traffic bursts.

This issue is consistent with findings in networked control systems under DoS attacks, where Kato et al. [24] demonstrated that queue management and buffering mechanisms are critical to prevent packet loss during high-frequency attack intervals. Similarly, in cloud-based intrusion detection, Mishra et al. [25] emphasized that insufficient processing capacity at the virtualization layer can lead to missed detections, and they advocated for adaptive resource allocation—including buffer tuning—to improve robustness against volumetric attacks. Based on these insights, as part of future work, we plan to empirically evaluate the impact of increasing Wazuh agent buffer capacity (e.g., from 5000 to a higher value such as 50000 or 100000 events) on detection accuracy, along with hardware acceleration and hybrid anomaly-based detection methods.

In addition, alert delivery time via Telegram is influenced by external network and service conditions, which are beyond the control of the detection system. Future work may explore alternative or redundant alerting channels to enhance reliability.

F. FUTURE WORK

Based on the limitations identified in this study, several directions for future work are proposed. First, a systematic empirical evaluation of increased Wazuh agent buffer capacity should be conducted to quantify the improvement in detection accuracy under high-volume SYN Flood attacks. Second, optimization of Snort rules—such as reducing false positive rates and prioritizing critical signatures—may enhance detection efficiency. Third, deploying the proposed system on higher-capacity hardware (e.g., multi-core CPUs with faster I/O and increased RAM) could reduce packet loss and improve throughput. Fourth, hybrid detection approaches that combine signature-based Snort with anomaly-based machine learning techniques should be explored to detect zero-day or low-frequency attacks that may evade signature matching. Finally, the scalability of the Wazuh, Snort, and Telegram integration should be evaluated in larger and more complex network environments, such as multi-tenant cloud infrastructures or enterprise networks with hundreds of endpoints.

IV. CONCLUSION

This study successfully designed and evaluated an integrated web server incident detection system based on Wazuh, Snort, and Telegram using the Security Policy Development Life Cycle (SPDLC) methodology. The proposed system was implemented to detect and respond to two common network-based attacks targeting web servers, namely SYN Flood and Port Scanning. Experimental results show that the system is capable of accurately detecting Port Scanning attacks with a detection accuracy of 100%, while SYN Flood attacks exhibit lower detection accuracy due to high traffic volume and packet processing limitations inherent in signature-based IDS systems. Despite this limitation, the integration of Snort with the Wazuh SIEM enables centralized event correlation and visualization, providing administrators with comprehensive situational awareness. Response time evaluation indicates that detection latency between Snort and Wazuh remains within the millisecond range for both attack scenarios. Furthermore, the use of Telegram as an alerting mechanism enables near real-time notification delivery, with average delivery times below two seconds. These results demonstrate that the proposed system effectively reduces the

delay between attack occurrence and administrator awareness, thereby improving incident response readiness. Overall, the integration of network-based intrusion detection, SIEM correlation, and instant messaging alerts provides a practical and efficient solution for web server security monitoring. Future work may focus on enhancing detection performance under high-volume attack conditions, integrating anomaly-based detection techniques, and evaluating system scalability in larger and more complex network environments.

AUTHORS CONTRIBUTION

Agus Wijayanto: Conceptualization, Methodology, Project Administration, Formal Analysis, Original Drafting Writing.
Ahmad Yazid Bustomi: Resources, Acquisition of Data.
Syaddam: Formal Analysis, Validation, Original Drafting Review & Editing Writing.
Tri Sudinugraha: Investigation, Methodology, Supervision, Validation , Review Writing.

COPYRIGHT

REFERENCES

[1] F. K. Ramadhan and A. Solehudin, “Comparative Study of Web Server Performance Testing with and without Docker Based on Virtual Machines,” vol. 8, no. 1, 2024.
[2] P. Chinprutthiwong and Y. Zhang, “The Service Worker Hiding in Your Browser : The Next Web Attack Target ?,” pp. 312–323, doi: 10.1145/3471621.3471845.
[3] A. Wijayanto, I. Riadi, and Y. Prayudi, “TAARA Method for Processing on the Network Forensics in the Event of an ARP Spoofing Attack,” *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 7, no. 2, pp. 208–217, Mar. 2023, doi: 10.29207/resti.v7i2.4589.
[4] T. D. Diwan, “AN INVESTIGATION AND ANALYSIS OF CYBER SECURITY INFORMATION SYSTEMS : LATEST TRENDS AND FUTURE,” vol. 9, no. 2, 2021.
[5] C. K. Yee and M. F. Zolkipli, “Review on Confidentiality , Integrity and Availability in Information Security,” vol. 8, no. 2, pp. 34–42, 2021.
[6] S. R. M. Zeebaree, K. Jacksi, and R. R. Zebari, “Impact analysis of SYN flood DDoS attack on HAProxy and NLB cluster-based web servers,” vol. 19, no. 1, pp. 505–512, 2020, doi: 10.11591/ijeecs.v19.i1.pp505-512.
[7] R. R. Zebari, “Impact Analysis of HTTP and SYN Flood DDoS Attacks on Apache 2 and IIS 10 . 0 Web Servers,” *2018 Int. Conf. Adv. Sci. Eng.*, pp. 156–161, 2018.
[8] S. Ahmed and M. Junaid, “Penetration Testing Active Reconnaissance Phase - Optimized Port Scanning With Nmap Tool,” *2019 2nd Int. Conf. Comput. Math. Eng. Technol.*, pp. 1–6, 2019.
[9] C. P. Beretas, “Information Systems Security , Detection and Recovery from Cyber Attacks,” vol. 1, no. 1, pp. 27–40, 2024.
[10] L. Deri and F. Fusco, “Using Deep Packet Inspection in CyberTraffic Analysis,” in *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, 2021, pp. 89–94. doi: 10.1109/CSR51186.2021.9527976.
[11] H. Al, A. Ghafri, and A. Ghafri, “Implementation of Intrusion Detection System using Snort,” *J. Adv. Comput. Technol. Appl.*, vol. 1, no. 1, pp. 9–16, 2019.
[12] R. Kurnia, F. Widyatama, I. M. Wibawa, and Z. A. Brata, “Enhancing Security Operations Center : Wazuh Security Event Response with Retrieval-Augmented-Generation-Driven Copilot,” pp. 1–26, 2025.
[13] R. Islam and R. Rafique, “Wazuh SIEM for Cyber Security and Threat Mitigation in Apparel Industries,” vol. 9, no. July, pp. 136–144, 2024.
[14] A. Sutanto and A. Rakhman, “Experimental Evaluation of Wazuh-Grafana Integration for Real-Time Cyber Threat Detection in Resource-Constrained Environments,” vol. 9, no. 5, 2025.
[15] S. Stanković, S. Gajin, and R. Petrović, “A Review of Wazuh Tool Capabilities for Detecting Attacks Based on Log Analysis,” no. june, pp. 6–9, 2022.
[16] N. Akshai Sankar and K. A. Fasila, “Implementation of SOC using ELK with Integration of Wazuh and Dedicated File Integrity Monitoring,” in *2023 9th International Conference on Smart Computing and Communications (ICSCC)*, 2023, pp. 350–354. doi: 10.1109/ICSCC59169.2023.10334992.
[17] H. Chaitou, T. Robert, J. Leneutre, and L. Pautet, “Intrinsic weaknesses of IDSs to malicious adversarial attacks and their mitigation,” in *International Conference on Smart Business Technologies*, Springer, 2022, pp. 122–155.
[18] M. Sheeraz, M. H. Durad, M. A. Paracha, S. M. Mohsin, S. N. Kazmi, and C. Maple, “Revolutionizing SIEM security: An innovative correlation engine design for multi-layered attack detection,” *Sensors*, vol. 24, no. 15, p. 4901, 2024.
[19] L. Thomas and S. Bhat, “A Comprehensive Overview of Telegram Services - A Case Study A Comprehensive Overview of Telegram Services - A Case Study,” vol. 6, no. 1, pp. 288–301, 2022.
[20] D. T. Yuwono and S. A. Nuswantoro, “Analysis Performance Intrusion Detection System in Detecting Cyber-Attack on Apache Web Server,” vol. 6, no. 2, pp. 169–178, 2022.
[21] D. Yuliandari, B. K. Raja, R. Ningsih, and A. J. Wahidin, “Simulasi Penerapan Sistem Monitoring Jaringan Snort NIDS Pada Web Server Menggunakan Metode SPDLC,” vol. 5, no. 2, pp. 133–138, 2023.
[22] D. S. Ghazi, “Snort versus suricata in intrusion detection,” vol. 7, no. 2, pp. 73–88, 2024, doi: 10.31987/ijict.7.2.290.
[23] S. Duque and M. Nizam, “Using Data Mining Algorithms for Developing a Model for Intrusion Detection System (IDS),” *Procedia - Procedia Comput. Sci.*, vol. 61, pp. 46–51, 2015, doi: 10.1016/j.procs.2015.09.145.
[24] R. Kato, A. Cetinkaya, and H. Ishii, “Security Analysis of Linearization for Nonlinear Networked Control Systems Under DoS,” vol. 5870, no. c, pp. 1–12, 2021, doi: 10.1109/TCNS.2021.3078130.
[25] P. Mishra, E. S. Pilli, V. Varadharajan, and U. Tupakula, “NvCloudIDS : A Security Architecture to Detect Intrusions at Network and Virtualization Layer in Cloud Environment,” pp. 56–62, 2016.