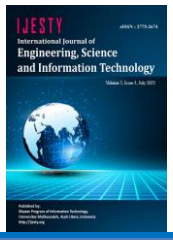
International Journal of Engineering, Science and Information Technology
Volume 5 No. 3 (2025) pp. 220-228
ISSN 2775-2674 (online)
Website: http://ijesty.org/index.php/ijesty
DOI: https://doi.org/10.52088/ijesty.v5i3.901
Research Paper, Short Communication, Review, Technical Paper



Implementation of an Intrusion Detection System Using Snort and Log Visualization Using ELK Stack

Fatih Dien Robbani[1], Emy Haryatmi[1*], Tri Agus Riyadi[2], Riza Adrianti Supono[3], Ary Bima Kurniawan[4], Rosdiana[4]

[1]Magister of Electical Engineering, Gunadarma University, Depok, Indonesia
[2]Department of Informatics, Gunadarma University, Depok, Indonesia
[3]Magister of Information System Management, Gunadarma University, Depok, Indonesia
[4]Department of Information System, Gunadarma University, Depok, Indonesia

**Corresponding author E-mail: emy_h@staff.gunadarma.ac.id*

The manuscript was received on 28 December 2024, revised on 22 January 2025, and accepted on 15 May 2025, date of publication 4 June 2025

Abstract

Cyber threats like malware, ransomware, and DDoS attacks demand fast and integrated detection systems. Traditional network monitoring tools often struggle to identify complex real-time attack patterns. This study evaluates the integration of Snort, an Intrusion Detection System (IDS), with the ELK Stack (Elasticsearch, Logstash, Kibana) to detect and visualize cyberattacks effectively. The system was tested against three attack scenarios: a Windows ping flood, port scanning using Zenmap, and SSH brute force attacks via Nmap Scripting Engine (NSE). Wireshark was employed as a supporting tool to monitor raw network traffic. The results indicate that Snort detected all simulated attacks in real time, and the ELK Stack efficiently processed and visualized the alert data. However, limitations in Kibana's dashboard refresh rate slightly hindered real-time monitoring capabilities. Overall, the integration of Snort and the ELK Stack proves effective for network threat detection and analysis, with room for future improvements in visualization performance and automated response mechanisms.

Keywords: *Snort, ELK Stack, Nmap, NSE, Ping Flood.*

1. Introduction

The rapid growth of digital technologies has improved convenience for individuals and organizations and led to more frequent and sophisticated cyberattacks. Threats such as malware, ransomware, and Distributed Denials of Service (DDoS) can cause financial loss and data breaches [1][2][3], making cybersecurity a growing concern [4]. Some architecture is also used to secure various distributed systems [5]. One challenge in intrusion detection is recognizing complex and stealthy attack patterns [3][6]. Recent surveys have proposed ways to improve the performance of open-source IDS like Snort in high-speed network environments [7], as traditional tools often fail to deliver real-time threat insight.

Traditional network monitoring tools often fail to provide comprehensive and real-time threat insights. Therefore, open-source tools such as Snort and the ELK Stack (Elasticsearch, Logstash, and Kibana) have become effective network security monitoring and analysis solutions. Snort serves as a Network Intrusion Detection System (NIDS), capable of detecting potential attacks by analyzing real-time network traffic. At the same time, the ELK Stack enables efficient storage, search, and visualization of data logs [2][8].

Several previous studies have explored the application of Snort and ELK Stack in securing network infrastructures. Snort was implemented as both an IDS [9][10][11] and IPS to detect and prevent intrusions on an academic network [2]. The ELK Stack was examined for log event management, focusing on monitoring SSH activity and preventing brute force attacks [8]. The effectiveness of Snort in detecting multiple flooding attacks on wireless networks was demonstrated, with detection results visualized using the BASE web interface [12]. Furthermore, the evaluation of wireless LAN security through brute force penetration testing provided insights into vulnerabilities in weak passphrase configurations [1], and the impact of DoS attacks using Hping3 on server performance was also investigated [13].

Building upon these prior works, this study aims to integrate Snort and ELK Stack into a unified system to detect and analyze network cyber threats. Snort captures suspicious activity in real-time, while ELK Stack supports centralized data processing and visualization. This integration is aligned with recent advances that utilize containerized Snort and big data Technologies to enhance scalability and monitoring efficiency [14]. To evaluate their effectiveness, this research simulates common network attacks such as ping floods, brute force, and port scanning.

2. Literature Review

This section presents an overview of previous studies and foundational concepts relevant to detecting cyberattacks using open-source tools. It covers the snort Intrusion Detection System (IDS) and the ELK Stack, as well as recent experimental research on detecting common cyber threats such as flooding, port scanning, and brute-force attacks.

2.1. Overview of Snort and ELK Stack

Snort is an open-source network intrusion detection system (NIDS) developed by Sourcefire. It monitors network traffic in real-time, analyzes packet data, and logs suspicious or potentially harmful activities [3][15]. Snort is capable of identifying various attacks, such as buffer overflow, port scanning, and Denial of Service (DoS) attacks [2][16]. Snort operates based on predefined rules that specify the patterns or characteristics of detectable attacks. Each rule corresponds to a specific type of threat, and once matched, Snort generates alerts and logs the event [17][18]. These customizable rules allow the user to tailor the detection system to their specific needs.
Snort can run in three main modes:

1. Sniffer Mode, Snort display network traffic in real-time
2. Packet Logger, Snort stores network traffic data into log files
3. Intrusion Detection Mode: Snort analyzes packets according to the rule set and generates alerts when anomalies are detected [17][18].

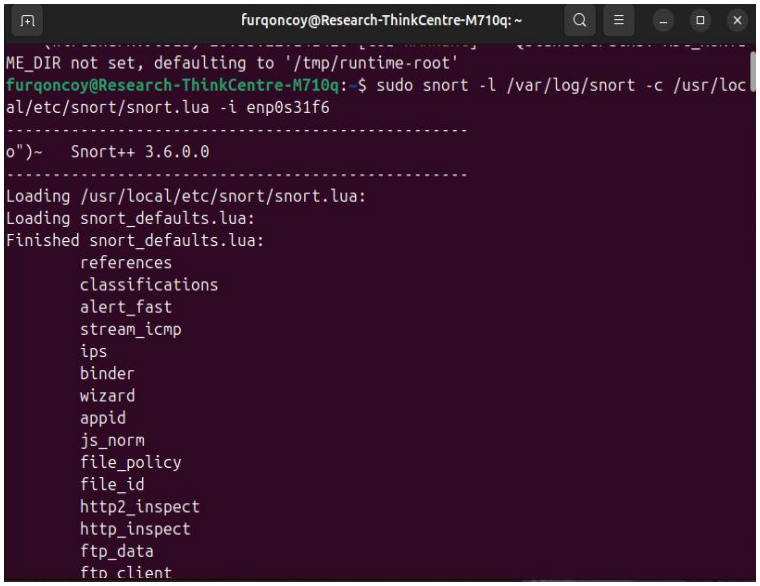


Fig 1. Snort startup process on Ubuntu terminal

Snort's main advantages are its flexibility and availability. However, its detection capability is limited by the quality and completeness of the rules it uses. If the rule is outdated or incomplete, its effectiveness in identifying intrusions may be reduced [18] [19]. These limitations have encouraged various improvements in the deployment of Snort. For instance, a recent study proposed a cloud-based Snort NIDS architecture using containerization and big data processing to address scalability and remote monitoring challenges [14] [20].
The ELK Stack is an open-source tool for data analysis and log management. It consists of three main components: Elasticsearch, Logstash, and Kibana. Together, these tools support collecting, processing, searching, and visualizing large volumes of log data from various sources [21].
Elasticsearch is a distributed search and analytics engine that stores, indexes, and retrieves large datasets in real-time. It operates based on a cluster consisting of nodes and shards. Nodes are individual instances that store data and perform processing, while shards are partitions of data that allow for load distribution and performance optimization. With its ability to scale horizontally, Elasticsearch ensures stable performance even as data grows [22][23]. Its ability to index various data formats and provide integration via APIs makes it suitable for diverse application scenarios [19][21][23].
Logstash is a data processing pipeline that supports various inputs, filters, and output plugins. It allows raw log data to be transformed and standardized before being forwarded to Elasticsearch [8]. For example, unstructured log data can be parsed into a structured format using filters. Filters can perform pattern extraction, value transformation, or data merging from multiple sources. This makes Logstash highly flexible in handling real-time data from various systems [22].
Finally, Kibana provides robust visualization capabilities for the data stored in Elasticsearch. It offers customizable dashboards that display metrics and trends tailored to user needs [21]. With its user-friendly interface, Kibana simplifies analyzing and sharing insights within teams, allowing for efficient collaboration.

2.2. Cyberattack Detections and Experimental Studies

Cyberattacks are deliberate exploitations of systems to access, damage, or disrupt their everyday operations. Common attack types such as flooding, port scanning, and brute-force login attempts aim to compromise system availability or integrity.
In the context of network monitoring, flooding, especially ICMP Flood, is often triggered by repeated ping commands that overwhelm the target's network. Despite its simplicity, this Denial of Service (DoS) attack remains effective, especially in unprotected environments. Snort detects such attacks by identifying excessive ICMP Echo requests within a short interval [2][17].
Port scanning is commonly performed using tools such as Nmap or Zenmap to prove target machines for open or vulnerable ports. This technique often proceeds to a more advanced intrusion. Snort detects port scans by correlating multiple connection attempts from a single

IP address to a range of destination ports. Detection is rule-based, often identifying SYN Packets without a proper TCP Handshake [12], [18].

Brute-force attacks, particularly over SSH, involved repeated attempts to guess username and password combinations to gain unauthorized access. These attacks can be detected by quickly monitoring repeated failed login attempts. Snort flags such behavior by identifying the login failure patterns or abnormal connection frequencies [22], [23]. Similar works, such as [17] and [18], snort was deployed in testbed environments to simulate controlled attacks. The results indicated that detection latency was minimal when alerts were forwarded to Logstash and indexed by Elasticsearch. Kibana provided a real-time dashboard for tracking the status and frequency of each threat. The combination of Snort and ELK offers a reactive alert system and a platform for retrospective log analysis. These capabilities are essential for research involving repeated attack simulation and comparative analysis, as will be conducted in this study.

3. Methods

3.1. Research Design

This study applies an experimental method to evaluate the effectiveness of Snort 3 and the ELK Stack in detecting and visualizing cyberattacks in real-time. The approach is conducted within an isolated local network, where a laptop acts as the attacker, a micro-PC as the target, and a router is used as a switch to connect the devices. The scenario is designed to imitate real-time intrusion detection and log visualization.

3.2. Network Topology

The experimental setup consists of a single subnet of three physical devices: a Windows laptop (attacker), a micro PC running Ubuntu 24.04.1 LTS (Target), and a router acting as a switch. The attacker and target devices are connected to a router that functions solely as a switch without internet access. This configuration allows the simulation of attacks without external interference or risk to public systems. No virtualization is used to maintain a lightweight and real-time environment.

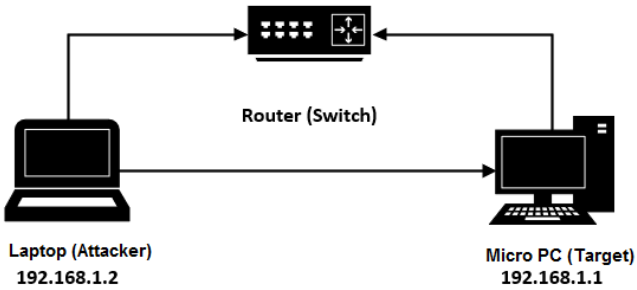


Fig 2. Network topology of the testbed environment

The figure illustrates the logical connection among the devices. The attacker device launches various cyberattacks while the target device runs Snort and ships logs to the ELK Stack for analysis and visualization.

3.3. System Configuration

The target system is an Ubuntu-based machine running Snort 3 as the Network Intrusion Detection System (NIDS). Log data is sent to Logstash, indexed by Elasticsearch, and visualized in Kibana. The attacker uses a Windows laptop with tools including CMD, Zenmap, and Nmap scripting engine. Wireshark is installed on the target device for independent traffic capture and verification during testing.

3.4. Attack Scenarios

Three attack types were selected for testing. Each attack was executed in three separate iterations to assess consistency in detection and logging

1. Ping Flood Attack using Windows CMD to generate ICMP echo requests.
2. Port Scanning using Zenmap (Nmap GUI) to probe open ports.
3. SSH Brute Force Attack using Nmap Scripting Engine to try SSH Logins repeatedly.

3.5. Testing Procedure

For each attack scenario, the following procedure was followed:

1. Launch the attack from the attacker's machine.
2. Capture traffic on the target machine using Snort 3.
3. Monitor packet flow and alert generation using Wireshark.
4. Forward detection logs to ELK Stack for storage and visualization.
5. Compare the results with the expected detection pattern.

The analysis includes observing whether Snort successfully generates alerts for each attack, the time it takes to detect and log the event, and how Kibana displays the data.

Table 1. Summary of Cyberattack Testing Procedures

No	Attack Type	Tools Used	Durations	Iterations	Snort Detection	Kibana Visualization	Notes
1	ICMP Ping Flood	Windows CMD	15 seconds	3	Yes	Yes	High ICMP packet rate
2	Port Scan	Zenmap	~20 seconds	3	Yes	Yes	Sequential TCP SYN
3	SSH Brute Force	Nmap Script	~10 minutes	3	Yes	Yes	Repeated login attempts

Table 1 combines key information from all three attack tests, including attack type, execution tool, duration, iterations, detection status by Snort, and visualization status in Kibana, and notes about what to look for from the test.

4. Results and Discussion

4.1. Analysis of Ping Flood Attack

This subsection discusses the results of a Windows Ping Flood attack executed via the Command Prompt (CMD) application. The attack involves sending large-sized ICMP packets high-frequency to flood the target system's network. The experiment was conducted in three executions using similar parameters to ensure consistency in observation and detection.

Figure 3 illustrates the terminal output during one execution of the Ping Flood attack. The attack used ICMP packets of 65,500 bytes, continuously targeting the victim IP address for 15 seconds. The ping statistics indicate all packets were delivered successfully, with response times ranging between 1 to 2 milliseconds. While the visual output here represents one trial, all three executions resulted in similar packet transmission behavior and duration.

```
Pinging 192.168.1.1 with 65500 bytes of data:
Reply from 192.168.1.1: bytes=65500 time=1ms TTL=64
Reply from 192.168.1.1: bytes=65500 time=2ms TTL=64
Reply from 192.168.1.1: bytes=65500 time=1ms TTL=64
Reply from 192.168.1.1: bytes=65500 time=2ms TTL=64
Reply from 192.168.1.1: bytes=65500 time=1ms TTL=64
Reply from 192.168.1.1: bytes=65500 time=1ms TTL=64
Reply from 192.168.1.1: bytes=65500 time=1ms TTL=64
Reply from 192.168.1.1: bytes=65500 time=1ms TTL=64
Reply from 192.168.1.1: bytes=65500 time=1ms TTL=64
Reply from 192.168.1.1: bytes=65500 time=2ms TTL=64
Reply from 192.168.1.1: bytes=65500 time=2ms TTL=64
Reply from 192.168.1.1: bytes=65500 time=2ms TTL=64
Reply from 192.168.1.1: bytes=65500 time=1ms TTL=64
Reply from 192.168.1.1: bytes=65500 time=2ms TTL=64
Reply from 192.168.1.1: bytes=65500 time=1ms TTL=64
Reply from 192.168.1.1: bytes=65500 time=2ms TTL=64

Ping statistics for 192.168.1.1:
    Packets: Sent = 16, Received = 16, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 2ms, Average = 1ms
```

Fig 3. CMD Output of Ping Flood Attacks

Wireshark was used to capture traffic during the attacks to observe the network impact in more detail. Figure 4 shows the Wireshark capture filtered by ICMP protocol, focusing on one representative execution. The capture includes the main interface window and the IPv4 conversation tab, showing 32 packets sent from the attacker (IP A: 192.168.1.2) to the target (IP B: 192.168.1.1) and 16 replies in return. The consistent number of packets exchanged across the tests confirms symmetrical ICMP communication behavior under flood conditions.

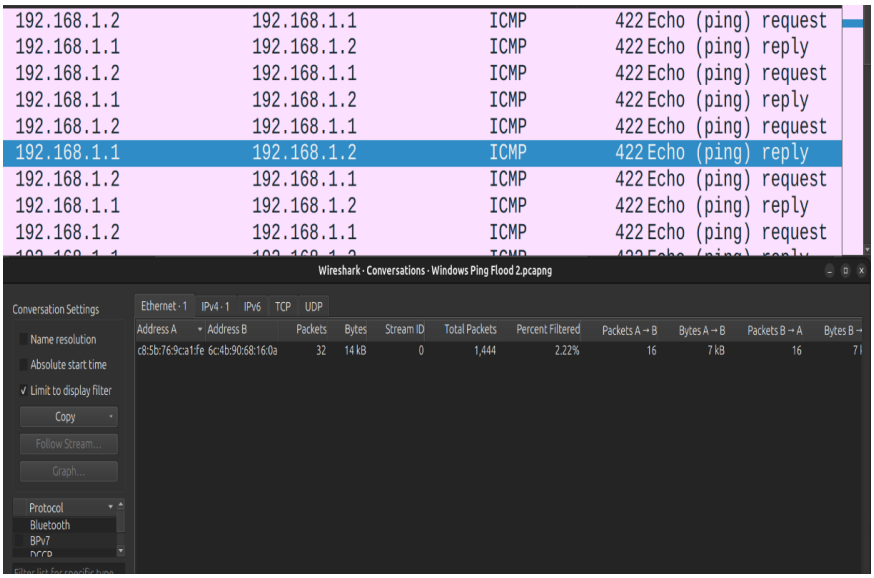


Fig 4. Wireshark Capture of Ping Flood Attacks

Figure 5 presents the Kibana dashboard displaying Snort detection logs related to the Ping Flood attack. The dashboard includes pie and bar charts highlighting detected alert types and their timestamps. All detections occurred around the scheduled attack times, with classification labels such as PROTOCOL-ICMP and SERVER-OTHER. Additional tables on the dashboard list alert details by source/destination IP, protocol, and signature classification.

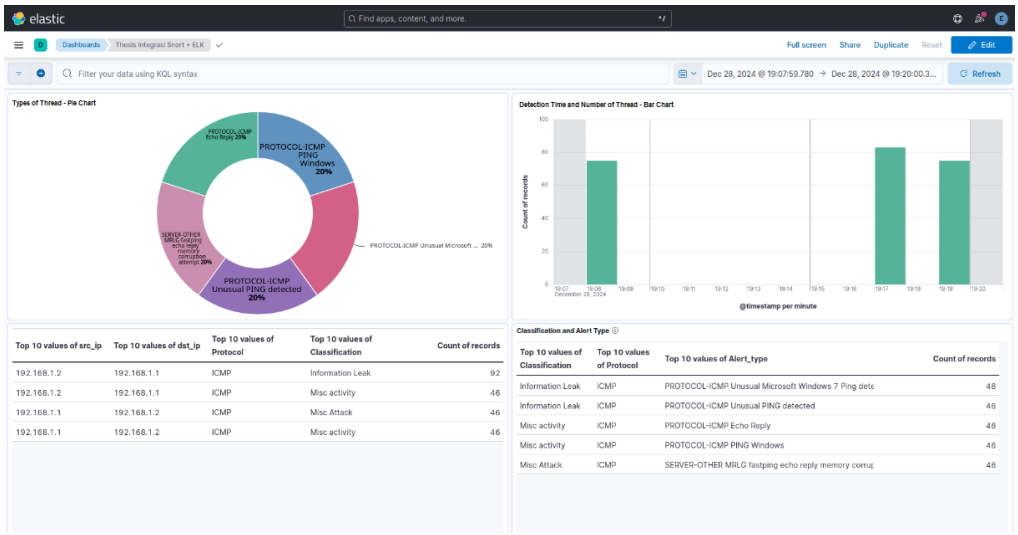


Fig 5. Kibana Dashboard of Ping Flood Attacks

Although Figure 3 and Figure 4 represent a single execution of the attack for visual clarity, the experiment was conducted three times under similar conditions to ensure reliability. The selected screenshots were taken from the second execution, which yielded the highest packet and alert counts among the three trials (32 packets, 83 Snort entries). The summarized results from all three executions are presented in Table 2, which captures key parameters such as execution time, attack duration, packet counts from Wireshark, number of alerts detected by Snort via Kibana, alert classifications, and Snort detection timestamps.

Table 2. Summary of Ping Flood Attack Results

Execution	Time	Duration	Packets (A/B)	Alerts	Main Classification
1	19:08:10	15s	15/15	75	Info Leak, Misc. Activity, Misc. Attack
2	19:17:01	15s	16/16	83	Info Leak, Misc. Activity, Misc. Attack
3	19:19:30	15s	15/15	75	Info Leak, Misc. Activity, Misc. Attack

Note: A = Attacker (192.168.1.2), B = Target (192.168.1.1)

The data in Table 2 confirms that the attack produced consistent results across all three executions. Each trial lasted 15 seconds and generated approximately 30–32 ICMP Packets. The Snort detection logs ranged from 75 to 83 entries, dominated by Information Leak and Misc classifications. Activity, with the unusual ping rule contributing significantly to the alert count. These results verify that the system was able to identify and log the Ping Flood behavior accurately

4.2. Analysis of Port Scan Attack

This test simulates a port scanning attack using Zenmap from the attacker's laptop to identify open services on the target system. The scan was executed using the Intense Scan profile, which aggressively probes for open ports and gathers service information. According to

the result displayed in Figure 6, Zenmap successfully identified that TCP port 22 (SSH) and port 9200 (Elasticsearch) were open on the target micro PC.

```
Nmap scan report for 192.168.1.1
Host is up (0.00080s latency).
Not shown: 998 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 9.6p1 Ubuntu 3ubuntu13.5 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   256 e8:6d:34:6e:06:8d:64:01:e4:3a:44:06:13:e4:c1:45 (ECDSA)
|_  256 a7:81:ae:36:29:d8:96:9d:5e:b2:f0:45:74:4f:46:91 (ED25519)
9200/tcp open  http    Elasticsearch REST API 7.0 or later (Shield
plugin; realm: security)
|_http-title: Site doesn't have a title (application/json).
| http-methods:
|   Supported Methods: GET DELETE HEAD OPTIONS
|_  Potentially risky methods: DELETE
| http-auth:
| HTTP/1.1 401 Unauthorized\x0D
|   Basic realm=security charset=UTF-8
|_  ApiKey
MAC Address: 6C:4B:90:68:16:0A (LiteON)
```

Fig 6. Zenmap Port Scan Output

During the attack, Wireshark was used to capture and analyze network traffic on the target. As shown in Figure 7, one scan execution generated over 1,000 TCP conversations. This burst of activity is characteristic of a port scanning attempt, primarily when performed using a profile with aggressive timing and probing options.

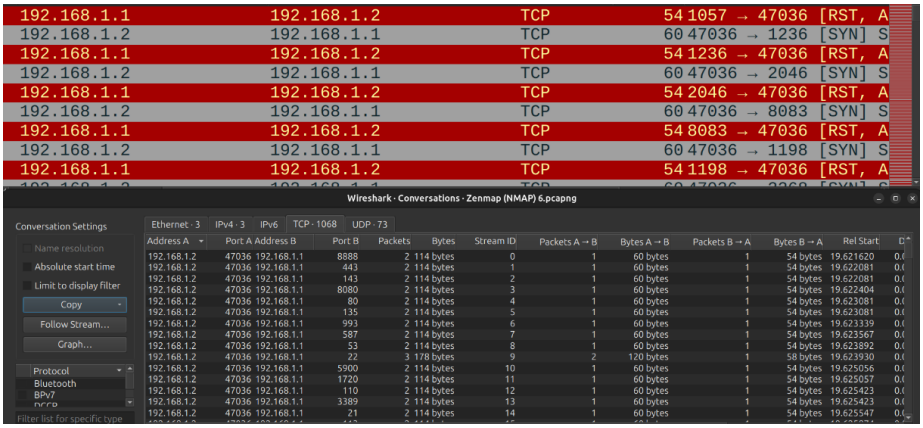


Fig 7. Wireshark TCP Conversations During Scan

A packet filter was applied to isolate packets with the SYN and ACK flags enabled to confirm which ports responded as open. This allows analysts to identify SYN-ACK responses typically returned by services listening on open ports. As shown in Figure 8, visible SYN-ACK packets from ports 22 and 9200 indicate that these services were active and responsive during the scan, confirming the findings observed in Zenmap.

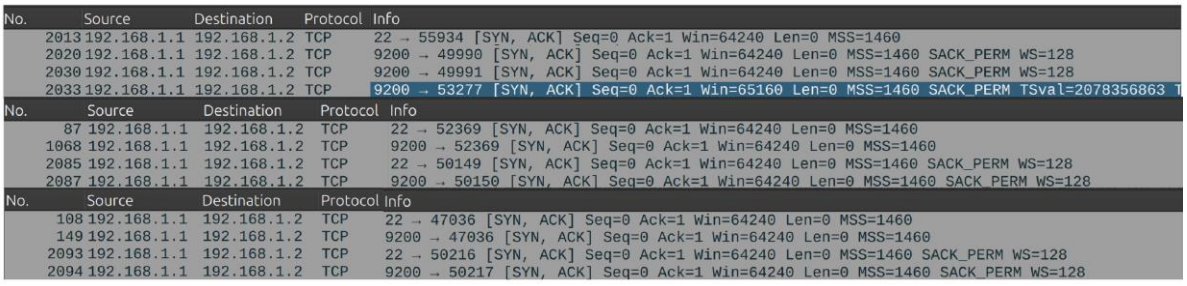


Fig 8. Filtered SYN-ACK Packets in Wireshark

The visualization in Figure 9 shows the detection result in Kibana. The Snort-generated alerts were successfully indexed into Elasticsearch and displayed in the dashboard. Port scan activities were identified and grouped by timestamp, alert signature, and source IP. The pie chart and bar graph components clearly illustrate the frequency and distribution of alerts. In contrast, the data table lists alert types such as "TCP Ports Scan" and their associated metadata.

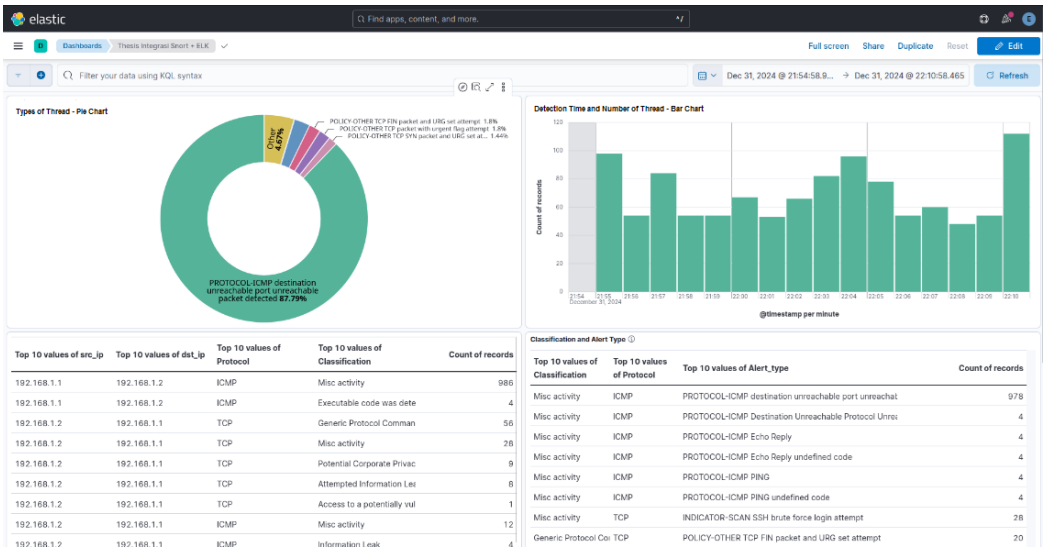


Fig 9. Kibana Dashboard Visualization of Port Scans

The detection and response data collected throughout this experiment are summarized in Table 3, including packet statistics and the number of alerts generated. This shows Snort and ELK Stack's capability to detect port scanning in real time and present the findings in a visually structured format.

Table 3. Summary of Port Scan Attack Results

Execution	Time	Packets (A/B)	Alerts	Main Classification
1	21:55:00	1356/1256	98	Misc. Activity, Info leak, Privacy Violation
2	22:04:01	1358/1267	96	Misc Activity, Info Leak, Vulnerable Web App
3	22:10:01	1353/1262	112	Misc Activity, Info Leak, Executable Code

Note: A = Attacker (192.168.1.2), B = Target (192.168.1.1)

Based on the experimental results, the port scanning activity was executed as planned using Zenmap. The open ports detected by Zenmap were confirmed through Wireshark analysis, and the packet behavior observed matched the expected scanning pattern. Furthermore, Snort successfully identified the scanning attempt, and the alerts were properly visualized in Kibana. This confirms that the detection pipeline functioned effectively during this test scenario.

4.3. Analysis of Brute Force Attack

This brute force attack simulates mass SSH login attempts to port 22 of the target device. Each of the three executions lasted approximately ten minutes, as specified in the test plan. The goal is to generate high-volume network traffic and trigger Snort to detect login attempt patterns as Misc Activity over TCP and ICMP protocols. Figure 10 illustrates the attack execution using the NSE script ssh-brute, which serves as the starting point for this analysis.

```
NSE: [ssh-brute 192.168.1.1:22] Trying username/password pair: sysadmin:oliver

NSE: [ssh-brute 192.168.1.1:22] usernames: Time limit 10m00s exceeded.
NSE: [ssh-brute 192.168.1.1:22] usernames: Time limit 10m00s exceeded.
NSE: [ssh-brute 192.168.1.1:22] passwords: Time limit 10m00s exceeded.
Completed NSE at 14:58, 603.12s elapsed
Nmap scan report for 192.168.1.1
Host is up (0.0010s latency).

PORT   STATE SERVICE
22/tcp open  ssh
| ssh-brute:
|   Accounts: No valid accounts found
|_  Statistics: Performed 2465 guesses in 603 seconds, average tps: 4.0
MAC Address: 6C:4B:90:68:16:0A (LiteON)

NSE: Script Post-scanning.
Initiating NSE at 14:58
Completed NSE at 14:58, 0.00s elapsed
Read data files from: D:\PKebutuhan Thesis\Nmap
Nmap done: 1 IP address (1 host up) scanned in 604.88 seconds
           Raw packets sent: 2 (72B) | Rcvd: 2 (72B)
```

Fig 10. Brute Force SSH Attack using Nmap SSE through CMD terminal

The attack targeted the IP address 192.168.1.1 on TCP port 22 by attempting various common username-password combinations, including root: root, admin: admin, and guest:123456789. Each attempt was displayed in real-time, continuing for ten minutes per session. Wireshark captured packet exchanges between the attacker and the target to observe traffic flow during the attack, as shown in Figure 11.

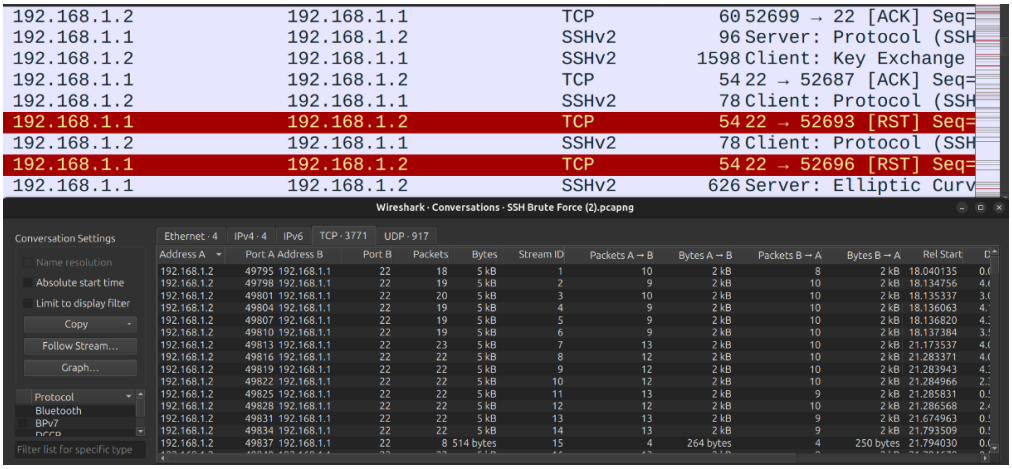


Fig 11. Wireshark Capture of Brute Force SSH Traffic

Based on the capture in Figure 11, one execution session recorded approximately 66,086 packets, with 35,919 from the attacker (A) to the target (B) and 30,167 packets in response. All outbound packets from the attacker targeted port 22. The packet flow exhibits a dominant and structured pattern from A to B, indicative of continuous brute-force login attempts. Detection results were visualized via the Kibana dashboard to understand the alert patterns and detection intervals. Figure 12 presents Snort's detection activity for the three attack executions.

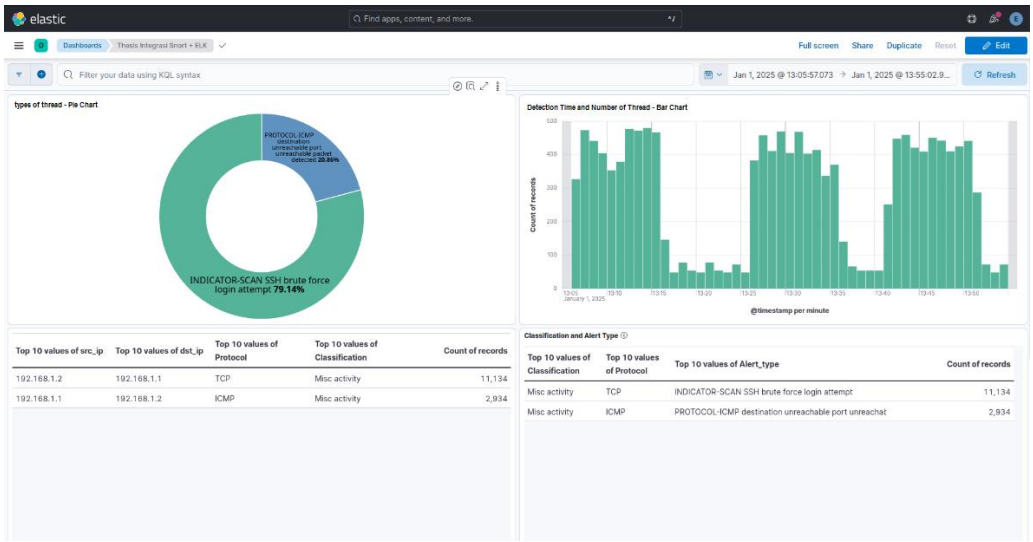


Fig 12. Kibana Dashboard Visualization of Brute Force Attack

Snort recorded detections across three ten-minute sessions. Alerts were mostly triggered by the "INDICATOR-SCAN Brute Force Login Attempt" signature, which accounted for 79.14% of the alert types. The remaining 20.86% were related to ICMP traffic, mainly from unreachable responses triggered by connection attempts. A summary of the results from Wireshark and Kibana for the three brute force sessions is presented in Table 4.

Table 4. Summary of Brute Force Attack Results

Execution	Time	Duration	Packets (A/B)	Alerts	Main Classification
1	13:06:01	10 minute	35919/30167	4395	Misc. TCP, Misc. ICMP
2	13:26:01	10 minute	34490/28925	4234	Misc. TCP, Misc. ICMP
3	13:41:20	10 minute	36216/30399	4413	Misc. TCP, Misc. ICMP

The experimental results confirm that each execution produced significant traffic, most of which originated from the attacker. Snort identified and classified these attempts accurately, recording over 4,200 alerts in each session. The consistent detection patterns demonstrate Snort's capability to recognize brute force behavior based on signature matching. Overall, this test validates the effective integration of Snort and the ELK Stack in identifying SSH brute force attacks. The visualizations in Kibana offer a clear, real-time representation of the threat activity, and the system's performance meets the detection targets established in the experimental design.

5. Conclusion

This research demonstrates that the system successfully detected three types of cyberattacks: Windows Ping Flood, Port Scan using Zenmap, and SSH Brute Force via the Nmap Scripting Engine. Each attack execution produced consistent detection results aligned with the test plan regarding duration, traffic pattern, and alert classification. Integrating Snort and the ELK Stack proved effective for real-time intrusion detection and analysis, delivering structured and analyzable attack logs through the Kibana interface. Overall, the system achieved its intended goals and supports its log-based network security monitoring application.

Acknowledgment

The author would like to sincerely thank the academic advisor, Dr. Emy Haryatmi, SKom., MEngSc., MT, for their invaluable guidance and support throughout this research. Special thanks also go to Fatahillah Furqon Abdul Aziz, who contributed significantly to preparing and writing the report and this article. Their assistance was crucial in completing the experimental work and documentation process.

References

[1] Suroso and Sriyanto, "Evaluasi Keamanan Wireless Local Area Network Menggunakan Metode Penetration Testing pada RSUD Alimuddin Umar Di Lampung Barat," *J. IndraTech*, vol. 5, no. 1, pp. 32–46, 2024, doi: 10.56005/jit.v5i1.138.

[2] H. Suhendi and W. D. Cahyo, "Perancangan Dan Implementasi Keamanan Jaringan Menggunakan Snort Sebagai Intrusion Prevention System (IPS) Pada Jaringan Internet STEI ITB," *NARATIF(Jurnal Ilm. Nas. Ris. Apl. dan Tek. Inform.*, vol. 03, no. 2, pp. 60–68, 2021, doi: 10.53580/naratif.v3i02.137.

[3] G. Pradita and A. Pramono, "Implementasi Monitoring Keamanan Jaringan Pada Server Ubuntu Menggunakan Snort Intrusion Detection Prevention System (IDPS) Dan Telegram Bot Sebagai Media Notifikasi Di PT SS UTAMA," *J. Mhs. Tek. Inform.*, vol. 8, no. 4, pp. 5827–5834, 2024, doi: 10.36040/jati.v8i4.10069.

[4] Maulidar, E. Wanda, and M. Hijriatin, "Cybersecurity Awareness In HR: Protecting Employee Data in the Digital Era," *Int. J. Eng. Sci. Inf. Technol.*, vol. 5, no. 2, pp. 237–242, 2025, doi: https://doi.org/10.52088/ijesty.v5i2.819.

[5] S. Akter, M. A. Hossain, and M. M. Rahman Redoy Akanda, "A Noble Security Analysis of Various Distributed Systems," *Int. J. Eng. Sci. Inf. Technol.*, vol. 1, no. 2, 2021, doi: 10.52088/ijesty.v1i2.101.

[6] V. Gustina and A. Ananda, "Kecerdasan Buatan untuk Security Orchestration, Automation and Response: Tinjauan Cakupan," *J. Komput. Terap.*, vol. 10, no. 1, pp. 36–47, 2024, doi: 10.35143/jkt.v10i1.6247.

[7] A. Gupta and L. Sen Sharma, "A categorical survey of state-of-the-art intrusion detection system-Snort," *Int. J. Inf. Comput. Secur.*, vol. 13, no. 3–4, pp. 337–356, 2020, doi: 10.1504/IJICS.2020.109481.

[8] F. S. Mukti and R. M. Sukmawan, "Integration of Low Interaction Honeypot and ELK Stack as Attack Detection Systems on Servers," *J. Penelit. Pos dan Inform.*, vol. 11, no. 1, pp. 19–29, 2021, doi: 10.17933/jppi.v11i1.336.

[9] A. Erlansari, F. F. Coastera, and A. Husamudin, "Early Intrusion Detection System (IDS) using Snort and Telegram approach," *SISFORMA - J. Inf. Syst.*, vol. 7, no. 1, pp. 21–27, 2020.

[10] D. Satin S, Wahyuddin, A. Kautsar, and A. Setyawan, "Intrusion Detection System Menggunakan Snort dan Telegram Sebagai Media Notifikasi," *SisInfo J. Sist. Inf. dan Inform.*, vol. 7, no. 1, pp. 40–49, 2025.

[11] S. S. Sari and A. Tedyyana, "Analisis Efektivitas Rule Snort dalam Mendeteksi Serangan Jaringan," *Repeater Publ. Tek. Inform. dan Jar.*, vol. 2, no. 4, pp. 1–15, 2024, doi: https://doi.org/10.62951/repeater.v2i4.194.

[12] P. P. Insani, I. Kanedi, and A. Al Akbar, "Implementasi Snort Sebagai Alat Pendeteksi Keamanan Jaringan Wireless Menggunakan Linux Ubuntu," *J. Komputer, Inf. dan Teknol.*, vol. 3, no. 2, pp. 443–458, 2023, doi: 10.53697/jkomitek.v3.2.

[13] W. Haniyah, M. C. Hidayat, Z. F. I. Putra, V. A. Pertama, and A. Setiawan, "Simulasi Serangan Denial of Service (DoS) menggunakan Hping3 melalui Kali Linux," *J. Internet Softw. Eng.*, vol. 1, no. 2, pp. 1–8, 2024, doi: 10.47134/pjise.v1i2.2654.

[14] F. A. Saputra, M. Salman, J. A. N. Hasim, I. U. Nadhori, and K. Ramli, "The Next-Generation NIDS Platform: Cloud-Based Snort NIDS Using Containers and Big Data," *Big Data Cogn. Comput.*, vol. 6, no. 1, p. 19, 2022, doi: 10.3390/bdcc6010019.

[15] V. Wineka Nirmala, D. Harjadi, and R. Awaluddin, "Sales Forecasting by Using Exponential Smoothing Method and Trend Method to Optimize Product Sales in PT. Zamrud Bumi Indonesia During the Covid-19 Pandemic," *Int. J. Eng. Sci. Inf. Technol.*, vol. 1, no. 4, 2021, doi: 10.52088/ijesty.v1i4.169.

[16] S. Oktarian, S. Defit, and Sumijan, "Clustering Students' Interest Determination in School Selection Using the K-Means Clustering Algorithm Method," *J. Inf. dan Teknol.*, vol. 2, pp. 68–75, 2020, doi: 10.37034/jidt.v2i3.65.

[17] H. Awal and A. P. Gusman, "Implementasi Intrusion Detection Prevention System Sebagai Sistem Keamanan Jaringan Komputer Kejaksaan Negeri Pariaman Menggunakan Snort dan IPtables Berbasis Linux," *J. Sains Inform. Terap. E-ISSN*, vol. 2, no. 2, pp. 74–80, 2023, doi: 10.62357/jsit.v2i1.184.

[18] C. D. Alviani, A. S. Padi, and N. Puspitasari, "Keamanan Siber di Masa Depan : Tantangan dan Teknologi yang Dibutuhkan," *Semin. Nas. AMIKOM SURAKARTA 2024*, vol. 2, pp. 1247–1254, 2024.

[19] W. Sholihah, S. Pripambudi, and A. Mardiyono, "Log Event Management Server Menggunakan Elastic Search Logstash Kibana (ELK Stack)," *JTIM J. Teknol. Inf. dan Multimed.*, vol. 2, no. 1, pp. 12–20, 2020, doi: 10.35746/jtim.v2i1.79.

[20] S. Sapriadi, Y. Yunus, and R. W. Dari, "Prediction of the Number of Arrivals of Training Students with the Monte Carlo Method," *J. Inf. dan Teknol.*, vol. 4, pp. 1–6, 2022, doi: 10.37034/jidt.v4i1.168.

[21] A. Setiyawan, A. Pinandito, and W. Purnomo, "Pengembangan Sistem Informasi Log Management Server Monitoring Menggunakan ELK (Elastic Search, Logstash dan Kibana) Stack pada Aplikasi Padichain di PT. Bank Rakyat Indonesia," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 7, no. 5, pp. 2142–2151, 2023.

[22] S. B. Dahal and M. Aoun, "Architecting Microservice Frameworks for High Scalability: Designing Resilient, Performance-Driven, and Fault-Tolerant Systems for Modern Enterprise Applications," *J. Intell. Connect. Emerg. Technol.*, vol. 8, no. 4, pp. 58–70, 2023.

[23] A. Oussous and F. Z. Benjelloun, "A Comparative Study of Different Search And Indexing Tools For Big Data," *Jordanian J. Comput. Inf. Technol.*, vol. 8, no. 1, pp. 72–86, 2022, doi: 10.5455/jjcit.71-1637097759.