Holistic Standalone Application for Log Analysis and Anomaly Detection Consisting of Windows Logs, Network Logs & Syslogs

[1]Dr. Kanti Singh Sangher, [2]Radha Kumari, [3]Dr. Mary Jacintha M
[1]Scientist-E, [2]M. tech student, [3]Scientist-F
[1,3]Eduction and Training,
[1,3]Centre for Development of Advanced Computing, Noida, India
[2]Sant Longowal Institute of Engineering and technology, Punjab, India

Abstract: The growing complexity of online threats requires security surveillance solutions, which can present a single perspective of IT environment of an organization. Analysing disparate host-level and network-level security logs in isolation often fails to detect complex attack patterns. To solve this difficulty, this paper proposes an integrated security monitoring architecture that is fully based on open-source tools, namely the ELK Stack (Elasticsearch, Logstash, Kibana), Winlogbeat, and the Zeek Network Security Monitor. To gather, organize, and analyse structured network traffic logs from Zeek and Windows Event Logs from endpoints, the suggested framework was created and put into use in a virtualized lab environment. The Metasploit Framework was used to simulate a controlled brute-force assault against a Windows target to empirically verify its effectiveness. Through the direct correlation of many unsuccessful logon events (Windows Event ID 4625) on the target host with unusual connection patterns more precisely, an increase in TCP connection attempts from the attacker's IP address recorded in Zeek's conn.log, the results show that the framework successfully detected the attack. This paper presents a scalable architecture of integrated open source threat detection and establishes better security visibility and high fidelity alerting that is achievable with the help of the correlation of heterogeneous sources of data.

Index Terms: Anomaly Detection, ELK Stack, Zeek, Windows logs, Syslogs, Network Logs

1.INTRODUCTION

The contemporary digital environment is subjugated to unrelenting and developing internet threats. Such attacks employ diverse methods both on the network and the host side, leaving traces of inferences in various systems. The classical security installations use the silo-based tools, one to monitor the network traffic, and the other to monitor endpoint activities resulting in a remarkable gap in visibility. As an example, a warning of a network intrusion detection system (NIDS) might contain no host-level context of the effects of that warning and an anomalous process on an endpoint might be overlooked without the network context of its origin. There are various tactics of attackers that hop between the host and network thus leaving evidence in the forensic traces. Using multiple tools to monitor traffic and endpoints covers significant visibility gaps. Industry use of Security Information and Event Management (SIEM) systems has been widespread to close this gap. To provide a single point of view for monitoring and analysis, SIEM platforms primarily serve as data aggregators, centralizing security logs and events from throughout an organization's infrastructure. SIEMs facilitate compliance reporting, forensic analysis, and real-time threat detection by gathering and comparing data from servers, firewalls, network devices, and endpoints. But a lot of small-to-medium businesses (SMEs), academic institutions, and research labs cannot afford the significant operational overhead, complicated deployment processes, and expensive license prices that commercial SIEM solutions frequently have. This has increased interest in using open-source software to create affordable, adaptable SIEM-like solutions [1].

This research work addresses the following problem statement: While mature open-source tools for host and network monitoring are readily available, there is a need for a practical, integrated framework that not only centralizes these heterogeneous data streams but also demonstrates a clear methodology for correlating them to detect specific, common attack patterns. Many open-source SIEM deployments stop at log aggregation, leaving the critical task of event correlation as a complex, manual exercise for the security analyst. This study primarily makes three contributions. First, it details the design and implementation of a complete, end-to-end security monitoring framework using a "best-of-breed" combination of open-source tools: the ELK Stack for scalable log management and visualization, Winlogbeat for granular host event collection, and Zeek for rich network traffic analysis. Second, contribution is to offer an actual assessment of the framework's detection capabilities by simulating a common and persistent threat vector a controlled, repeatable brute-force attack. Third, it presents a detailed analysis of how the direct correlation of host-based artifacts (Windows Event ID 4625) and network-based artifacts (Zeek conn.log entries) within the unified platform leads to high-fidelity, actionable threat detection. The remaining sections of this paper is structured as follows: Section II reviews related work in the field. Section III describes the system architecture and implementation. Section IV details the experimental methodology. Section V presents and analyses the results. Section VI discusses the implications and limitations of the work, and Section VII concludes with a summary and directions for future research.

2.LITERATURE REVIEW

To place the current study within the existing body of literature, this section reviews studies on the application of Zeek for network monitoring, the ELK Stack as a security platform, and standard methods for log-based anomaly detection. For log gathering, management, and analysis, the ELK Stack-which consists of Elasticsearch, Logstash, and Kibana-has emerged as the de facto open-source standard. Its flexibility and scalability are what give it power. Large amounts of data can be indexed by Elasticsearch's distributed, highly accessible search and analytics engine. Before transferring data to a "stash" like Elasticsearch, Logstash acts as a server-side data processing pipeline that can simultaneously ingest data from numerous sources, transform, and enrich it. Kibana offers an effective web interface for data exploration, search, and visualization via dashboards that may be customized. The ELK Stack is useful in making threat monitoring and threat hunting, as pointed out by research and industry reports. It has SIEM[2] core functions such as long-term storage, central logging, and powerful queries. None of which are a complete SIEM, however, the open CE ELK Stack. It does not have an inbuilt intelligence, correlation rules, and automatic alerts, which is a huge weakness. Open source SIEM features will require self-implementation and a lot of engineering by connecting third-party applications, such as Elast-Alert, to auto alert, writing Kibana queries, to manually correlate with other results, and using Logstash settings to write detailed parsing rules. This research work fills this gap by utilizing ELK as it is based on the framework to establish a use of correlation.

Zeek (formerly known as Bro) is a strong and open-source network analysis engine, unlike the traditional signature-based Intrusion Detection Systems (IDS). To my knowledge Zeek generates much higher-level, structured, semantically rich logs since it, in fact, manages to observe and interpret network data rather than simply matching it with a collection of canned signatures. To give a specific example, rather than simply recording raw packets, it generates in-depth logs on specific protocols, such as the basic conn.log (a summary of every TCP, UDP, and ICMP connection), dns.log (which indicates requests and responses) and http.log (1.0 URLs, methods and user agents). Academic studies have confirmed that Zeek can be used effectively to identify as many malicious activities as possible, that is, network scan, as well as more complex malware communications [4]. As its logs are properly structured, they are particularly convenient to be indexed, searched, and compounded with the other sources of data when included in analytics systems, e.g. the ELK Stack.

Large-scale detection of a brutality attack is what to do an excessive level of homework about in classes that study cybersecurity. Security Event Log forms the primary source of information as it were. There is tend to notice brute force attacks a collection of unsuccessful logins attempt available in an interval. Analysis of logs therefore is important in detection. On windows, event 4625, with the message “An account failed to log on provides us with an account of all the unsuccessful attempts. It goes further to guess the targeted username, the type of logon (network, interactive) and most importantly the source IP. There are systems where the set a value of 4625 events of one IP per time, when exceed it, alarm is raised. Large-scale detection of a brutality attack is to do an excessive level of homework about in classes that study cybersecurity. Security Event Log forms the primary source of information as it were. There is tend to notice brute force attacks by finding a collection of unsuccessful logins in an interval. Analysis of logs therefore is important in detection. On windows, event 4625, with the message “An account failed to log on provides us with an account of all the unsuccessful attempts. It goes further to guess the targeted username, the type of logon (network, interactive) and most importantly the source IP. There are systems where by setting a value of 4625 events of one IP per time, when this exceeds, alarm is raised. There is still a lack of thorough, end-to-end case studies in the literature, even though the various components ELK for logging, Zeek for network analysis, and event log monitoring for brute-force detection are thoroughly researched. There aren't many studies that describe the ELK Stack and Zeek's practical integration and then empirically confirm how effective they are together against a shared danger, highlighting the real advantages of connecting their different data sources. By offering a thorough plan and an understandable, fact-based assessment, this study seeks to close that gap [5].

3.EXPERIMENTS & IMPLEMENTATION

A unified data pipeline that flows from dispersed log sources to a centralized analysis platform is intended to be created by the monitoring framework's architecture. This section describes the high-level architecture and the setup of its main elements.

Fig 1, illustrates how logs from multiple sources are forwarded into a single ELK platform, enabling unified storage and analysis. To begin with, the current study does not use logs of hosts or networks individually as was previously done. To combine Windows event data, Syslog and Zeek network data into a single pipeline, and thus can identify anomalies between layers. Second, development of a small engine that converts raw logs into useful security labels, such as Rare Program Execution, Discovery Commands, Brute-force Attempts, Privilege Escalation and Firewall Evasion, gives a clear picture of what is going on, without necessarily joining a series of logs. Third, it can execute tens of thousands of events per second on a relatively inexpensive hardware, which is feasible by schools that cannot afford expensive SIEM tools. Lastly, in contrast to regular ELK systems, which simply render graphs, are literally retrieving anomalies, tagging them, and assessing them numerically, which provides a complete end-to-end detection pipeline [3].

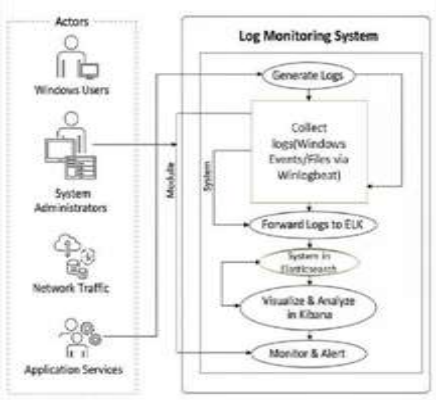

Fig.1 - Centralized Logging

A. High-Level Architecture

An attacker machine, a target machine, and a central monitoring server make up the system, which is set up in a virtualized environment. The architecture shows the logical flow of data and is represented in the conceptual diagram below, which is based on the project report's design.

- **Data Sources:** The network traffic between the Kali Linux Attacker VM and the target, as well as the Windows 10 Target VM, which produces host-level event logs, are the main sources of data.
- **Collection Agents:** To gather and send event logs, Winlogbeat is installed on the Windows target. Operating on the monitoring server, Zeek records and examines network traffic, and Filebeat gathers its output logs.
- **Log Aggregation and Storage:** The core repository is the ELK Stack, which is housed on the monitoring server. All incoming logs are indexed and stored by Elasticsearch, enabling near-realtime searching.
- **Analysis and Visualization:** Kibana offers the user interface for developing dashboards for security monitoring and incident investigation, as well as for querying data in Elasticsearch and producing visualizations.

The unique capabilities of each tool are purposefully combined in this framework. Winlogbeat and Zeek serve as the high-fidelity "sensors" for host events and networks, respectively. Elasticsearch functions as the scalable "data lake," and Kibana serves as the interactive "cockpit" for the security analyst. This modular, best-of-breed approach avoids the limitations of a single tool by creating a system where each component complements the others' weaknesses.

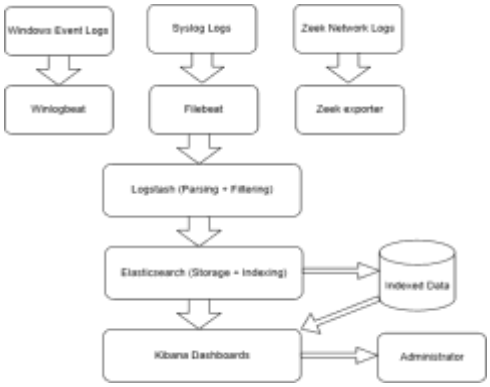

Fig.2. ELK VM connection to Kibana

Fig 2, depicts the setup where the ELK server connects with Kibana, allowing collected logs to be visualized through dashboards.

B. Component Configuration

For the framework to be implemented successfully, each component must be precisely configured to guarantee that data is gathered, parsed, and saved appropriately [6].

C. Log Sources

VirtualBox Workstation was used to build the experimental testbed, which consisted of a collection of virtual computers set up with static IP addresses to guarantee reliable logging and communication.

- **Windows 10 VM (Target):** This computer serves as the victim and is running the default Windows applications. Important host-level security logs originate from it.
- **Kali Linux VM (Attacker):** This system is used to create malicious traffic and mimic assaults against the target virtual machine. It comes with penetration testing tools, such as the Metasploit Framework.\

D. Endpoint Data Collection (Winlogbeat)

On the Windows 10 Target virtual machine, Winlogbeat, a lightweight agent from Elastic, was set up as a service. Winlogbeat.yml, its configuration file, was altered to accomplish two main objectives, it includes:

1. Log Selection: Security, System, and Application are the three main Windows Event Logs that Winlogbeat was set up to monitor and gather events from.
2. Security Event Focus: Within the Security log, Event IDs that correspond directly to MITRE ATT&CK framework methodologies were given attention. The following were given priority in the configuration:
   - Event ID 4625: The account did not successfully log in. Main sign of password-spraying and brute-force attacks
   - Event ID 4624: An account was successfully logged on. Crucial input for tracking lateral movement and identifying unauthorized access.
   - Event ID 4672: A new logon is granted special privileges. This may be a sign of attempts at privilege escalation.
   - Event ID 4648: Explicit credentials were used to log in. This is frequently linked to persistence strategies.

Output Configuration: The output was sent to the Elasticsearch instance that was operating on the IP address and port (192.168.138.100:9200) of the monitoring server.

E. Network Data Generation (Zeek)

To examine network traffic, Zeek was installed on the monitoring server. It can work with pre-captured traffic by processing PCAP files or with current traffic by listening on a network interface. Zeek was used for this project to examine both the real traffic produced during the attack simulation and PCAP files from known malicious activities (such as the Mirai Botnet dataset). Zeek creates a number of tab-separated, structured text logs after processing the traffic [7]. The following are important logs used in this framework

- **conn.log:** Records a summary of every network connection (5-tuple, protocol, duration, bytes transferred, connection state).
- **dns.log:** Records all DNS queries and responses.
- **http.log:** Records details of all HTTP traffic.
- **weird.log:** Records unusual or anomalous traffic that deviates from expected protocol behaviour.

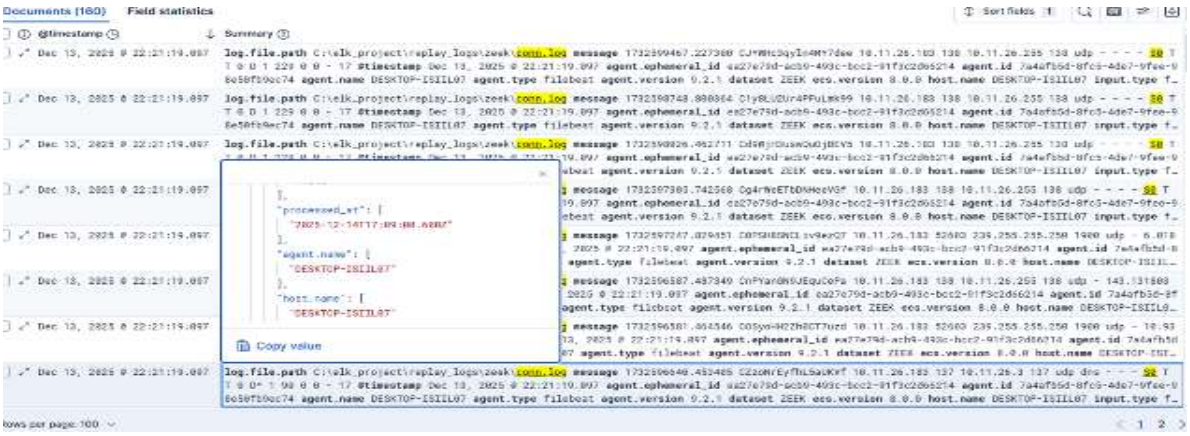

Fig.3. Conn.logs

These logs were written to a local directory on the monitoring server, where they were subsequently picked up by File beat and forwarded to Elasticsearch. Fig 3, displays Zeek's conn.log, which summarizes network connections (IP addresses, ports, protocols, and states) useful for detecting suspicious patterns.

F. Log Aggregation and Storage (ELK Stack)

A dedicated Windows 10 virtual machine was used to install the ELK Stack.

- **Elasticsearch:** Set up as the main data repository, it watches for information coming in from the Beats agents. After receiving the logs, Elasticsearch indexes the fields, parses the JSON-formatted data, and makes the logs queryable.
- **Kibana:** Set up to establish a connection with the nearby instance of Elasticsearch. For every analysis task, it offers a web-based graphical user interface. To enable Kibana to identify and query the recently ingested data from Winlogbeat and Filebeat (Zeek logs), index patterns (such as winlogbeat-, filebeat-) were developed.

4. EXPERIMENTAL EVALUATION

4.1 BRUTE FORCE ATTACK SCENARIO

A controlled experiment simulating a typical network-based attack was created to verify the integrated framework's detection capabilities. Because of its popularity and ability to produce unique, identifiable artifacts at the host and network levels, the experiment concentrated on a brute-force attack [8]. The Next Low-and-Slow Evaluation Scenario. Low-and-Slow Replay Test to test the soundness of our detection pipeline, tested with a low-rate replay test. Rather than firing logs simultaneously, threw out one event at a time to Windows, Syslog and Network datasets on 300 events. The point is to imitate attackers that slow down to evade rate-based intrusion detection mechanisms. Each replayed event was fed through the engine of anomaly -classification, to determine whether it can continue to detect these low volume attacks.

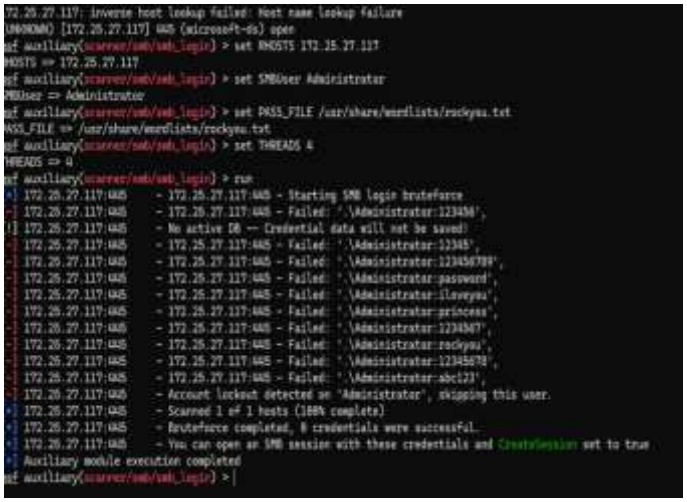

Fig.4. Metasploit SMB brute force attack

Fig 4, shows the Metasploit module (smb_login) being used to perform brute-force login attempts on the Windows target via SMB protocol.

A. Testbed Environment

- **Attacker:** The Kali Linux VM (IP: 10.0.2.4) served as the origin of the attack.
- **Target:** The Windows 10 VM was the intended victim of the brute-force attempt.
- **Monitor:** All network traffic between the attacker and the target was monitored by the ELK/Zeek server, which was also set up to receive event logs from the target using Winlogbeat.

To ensure complete IP-level visibility and enable Zeek to capture and analyze traffic, the virtual machines were connected using NAT or bridged networking mode in Virtual Box.

B. Attack Simulation Methodology

A brute-force assault against the Windows target's Server Message Block (SMB) service was selected as the attack vector. This is a popular technique for getting first access to Windows environments. A module of the Metasploit Framework was used to carry out the attack, guaranteeing a realistic and repeatable simulation.

- **Tool:** Kali Linux virtual machine's Metasploit Framework (msfconsole)
- **Module:** Auxiliary/scanner/smb/smb_login is the module. The purpose of this module is to test SMB logins on various computers and report credentials that work.
- **Parameters:** The following crucial parameters were set up for the module:
  - RHOSTS: Set to the Windows 10 Target VM's IP address.
  - SMB User: Set to the Administrator account, which is the target.
  - PASS_FILE: A comprehensive and extensively used collection of popular passwords, set at /usr/share/wordlists/rockyou.txt.
  - THREADS: To regulate the attack's concurrency, set it to 4.

After that, the attack was initiated, which caused the Metasploit module to repeatedly try to access the target's SMB service using the Administrator account and every password from the rockyou.txt wordlist.

C. Data Collection and Expected Artifacts

The framework's monitoring components operated continuously during the attack scenario. Zeek was examining the network traffic between the attacker and the target, while Winlogbeat was regularly delivering security event logs from the Windows target [9]. Two main categories of forensic artifacts were anticipated to be produced and recorded by the framework, depending on the type of attack:

- **Host Artifact (Windows Event Log):** The Windows security subsystem on the target computer was supposed to create and record a Security Event with Event ID 4625 (An account failed to log on) for each unsuccessful password attempt made by the Metasploit module. The source IP address of the Kali Linux attacker's computer and the target account name, Administrator, should be included in the metadata of each of these events. A significant and quick accumulation of these events in Elasticsearch, coming from the target host, was the anticipated result.
- **Network Artifact (Zeek conn.log):** A TCP connection must be made to the target's SMB port (TCP/445) for every SMB login attempt. Therefore, a high volume of TCP connection attempts from the attacker's IP to the target's IP on port 445 should be indicative of a brute-force attack. With a connection state history showing a failed or reset connection (e.g., REJ for a rejected connection or S0 for only a SYN packet observed), Zeek was likely to record these connections, which corresponded to unsuccessful login attempts, in its conn.log. A significant number of entries in (conn.log) displaying this connection pattern was the anticipated result.

5. RESULTS AND ANALYSIS

Both host-level and network-level artifacts were successfully captured by the proposed framework after the brute-force assault was carried out. A correlated analysis made possible by Elasticsearch's centralized data provided great confidence in the attack's validity.

A. Quantitative Evaluation

While working with a large heterogeneous stack of log data, the findings were quantified. The windows log dataset will have 2,901,154 number of events and among them, 6,378 were failed logon attempts (Event ID 4625) - a loads of unauthorized knocks. These data were used to see the size of the data set and that is why it cannot be simply sift through it manually, there is a need to automate it [10].

Table 1. Performance across multi-source log datasets

Log Type	Total Events	Processing Time (sec)	Event\ Sec	Avg. detection time (sec\event)	Total Anomalies
Windows	2,901,154	280.238	10,352.45	0.00007637	62,756
Syslog	6,083,409	202.926	29,978.40	0.00002616	1,887,222
Network	469,113	312.445	1,501.42	0.00063804	230,782

In proposed research work suggested framework has been tested based on three large and heterogeneous log sets Windows Security Logs, Syslog data, and Zeek-generated network telemetry. **Table 1** is a summary of the processing performance. The system has supported a throughput of 10k to 30k events per second on host and syslog data and approximately 1.5k events per second on network telemetry with the latency of each event to be detected being in the order of microseconds. These findings prove that commodity hardware can be used to provide the anomaly detection engine with almost real-time performance.

B. Detection of Security Events

The anomaly-detection engine placed the events into six large categories. They are high-frequency behavioural anomalies (Rare Program executions and Discovery Commands) and the less frequent but significant attack patterns (Brute Force, Privilege Escalation, and Firewall Evasion). Distribution of detected event:

- Rare Program: 4,209,944
- Discovery Command: 1,682,003
- Successful Logon (context events): 123,503
- Brute Force: 41,846
- Privilege Escalation: 26,017
- Firewall Evasion: 96

Such findings indicate that the system is capable of detecting an expansive spectrum of bizarre actions, including simple deviations and illegal serious actions [11] .

Table 2. Anomaly categories detected by framework

Log Source	Category	Count
Windows Logs	Privilege Escalation	34,860
	Brute Force	26,279
	Rare Event	1,406
	Persistence	128
	Successful RDP Logon	90
	User Management	33
Syslog Logs	Discovery Command	1,682,003
	Successful Logon	123,106
	SSH Brute Force	32,998
	Privilege Escalation	26,017
	Brute Force	22,459
	Rare Program	543
	Firewall Evasion	96
Network Logs	Potential DoS Attack	195,927
	Network Scan	33,614
	Remote Access Port	592
	TOR/Anonymizer Traffic	360
	Botnet Activity (Mirai)	289

As indicated in Table 2, the number of anomalies is substantial. Windows logs contain a high volume of Privilege Escalation and Brute-force events (over 62,000 anomalies), while Syslog data shows a much larger number of Discovery Commands (1.68 million), along with some SSH brute-force attempts. The network telemetry showed large aggregates of potential DoS and scanning and ran into some TOR traffic as well as Mirai-type botnet traffic. All this ascertains that our framework can detect the high-frequency behavioral anomalies as well as the low-frequency yet high-impact attacks using numerous log sources as shown in Fig 5.

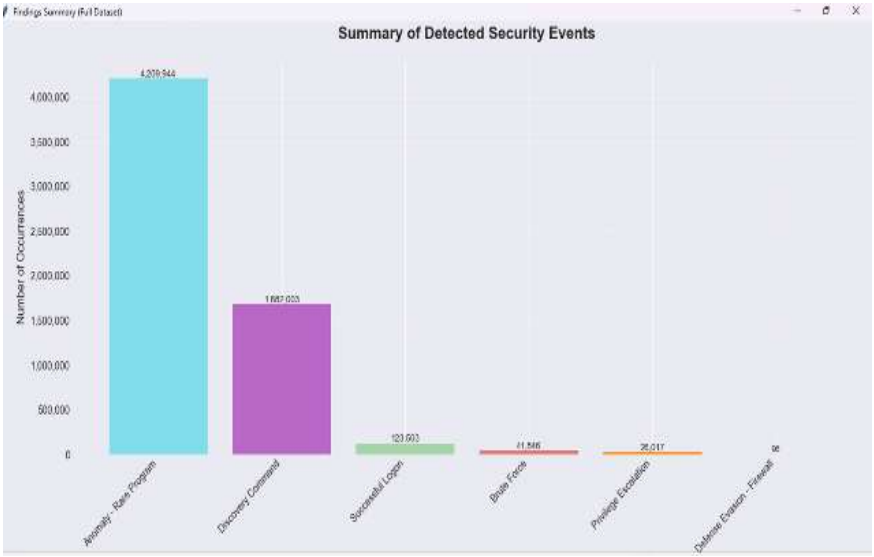

Fig 5. Identified anomaly-detection engine.

Based on the evaluating datasets results **Table 3**, displaying the Low-and-Slow Replay Evaluation Results with detected types of anomalies.

Table 3: Low-and-Slow Replay Evaluation Results

Dataset	Events Replayed	Detections	Dominant Anomaly Types
Windows Logs	300	300	Rare WinEvents, Privilege Escalation
Syslog Logs	300	300	Rare Program Execution, Discovery Commands
Network Logs	300	300	Informational Events

In this discovered that the detection pipeline is rather good when attackers can move relatively slowly and quietly and detect odd things. Detections of unusual login and session related quirks were captured by windows logs, and it is an indication that windows logs are able to detect those minor changes in host utilization even in the absence of a major attack. Syslog logs were good as well with unusual programs being labelled, discovery commands labelled and it indicates that it can fight against those reconnaissance tricks. Networks logs which were primarily simple Zeek telemetry alerts provided normal information, showing that the system does not unnecessarily trigger a high-severity alarm. Overall, these findings provide proof that the framework continues to maintain the level of detection despite attackers dwindling their motion considerably to avoid commonplace IDS.

5.1 Detection of Host-Level Artifacts

The Windows event logs taken from the target computer quickly showed the main sign of the brute-force attack. A Kibana query that examined Windows Security logs for event.code: 4625 showed a sharp increase in unsuccessful logon attempts that coincided with the simulation's duration. Plotting the count of Event ID 4625 in a time-series graphic made it evident that there was a baseline

of almost zero unsuccessful logons, a sharp and prolonged spike to hundreds of events per minute during the attack, and a return to baseline once the attack was over. All the Event ID 4625 logs were structured with concise fields such as source.Ip was equal to the IP of the Kali attacker, user.name was always equal to Administrator, winlog.logon.type was always 3 which was a Network logon and winlog. event data. These were handled using the Elasticsearch status codes e.g. 0xC000006D that means Bad user name or password. As far as the host is concerned, the observation was a high number of incidents that occurred in one source IP against one account was virtually unjustifiable evidence of the attack [12].

5.2 Detection of Network-Level Artifacts

At the same time, a similar set of artifacts were invoked by Zeek after analysing the network data. The data digitized in Zeek and entered via File beat was filtered with the help of a Kibana Query Language (KQL) query that helped to identify the appropriate network activity:

- Data_stream.dataset:“zeek.conn”:source.ip:“10.0.2.4”;destination.port;445
  The result of this query presented a trend that was fully consistent with the findings at the host-level. The connection.log data show several consecutive attempts of connection by the attacker whose IP address (10.0.2.4) to the computer of the victim on TCP port 445 (SMB).
- Nature of these connections were indicative of an automated attack:
  - Short Duration: Bearing in mind, on zeek.connection.duration, most of the connections had a duration shorter than a second.
  - Low Data Volume: It is basically that low data volume comes as a simple attempt at authentication where the source, bytes and destination, bytes fields contain just a few bytes.
  - Connection state: It was browsing through the history of connection and had a hunch, because there was noticing such a pattern frequently, & had REJ (server rejected) or what was known as Connection type of ShADdfF, but in noticed it dropped right after the connection was made. In effect, a legitimate SMB session ought to take longer and really transfer more information than what are experiencing hence making it seem like something is derailing [13].

Visualizes log entries categorized by different source IPs as shown in Fig 6, helps analysts to identify unusual spikes or malicious actors.

Fig 6. Distribution of log records over time

5.3 Correlated Analysis

It is the combined system that is murderous since it allows us to comfortably relate and analyze all these various datasets in unison. And literally, the time and effort saved in the process of validating a threat is terribly reduced, which is a massive relief to any analyst undertaking this job.

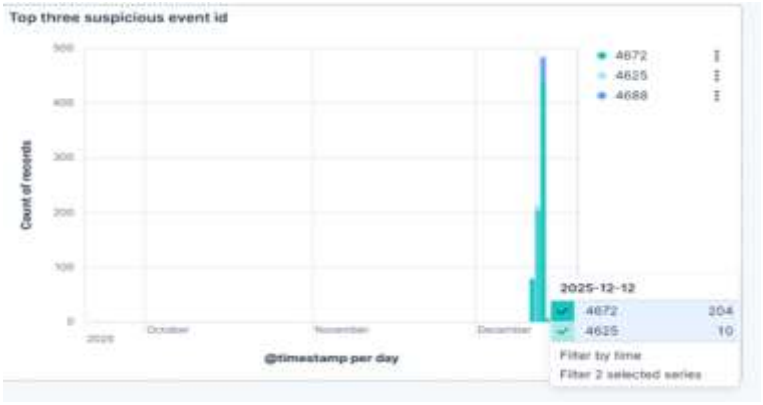

Fig 7. Daily count of Windows security events

A high-level alert, like the unusual surge in unsuccessful logon attempts (Event ID 4625) on a security dashboard, initiates the inquiry. The analyst can quickly determine the top source IP address causing the unsuccessful attempts from this visualization. The analyst can then use this IP address as a global filter across all data indices inside the same Kibana interface. Fig 7, displaying the plots by frequency of key event IDs (4720 user creation, 4625 failed logons, 4688 process creation), highlighting activity trends that may indicate attacks. User Account Creation (4720), Failed Logon (4625), and New Process Creation (4688). This single action instantly pivots the view to include the network-level data from Zeek associated with that same attacker IP. The huge number of brief SMB connections made by the attacker would be visible to the analyst, directly supporting the unsuccessful logon attempts. The host events' timeframe and the network events' timeline would coincide exactly. A host-centric alert ("Many failed logins") is converted into a fully contextualized security incident ("Confirmed network-based SMB brute-force attack from IP 10.0.2.4 against the Administrator account") by this one-step correlation technique. The high-confidence evidence required to start an incident response, like blocking the attacker's IP at the firewall, is provided by this quick validation, which also lowers the mean-time-to-detection (MTTD) [14]. Using the ELK dashboards direct, time-correlated evidence where shown. While the ELK Stack provides a powerful server-based solution for real-time monitoring, security analysts often face the challenge of analysing massive, raw log files (e.g., 7+ GB) on their local machines without relying on a large-scale log management platform. To address this need, a standalone desktop GUI application was developed using Python as a complementary utility. This tool's main goals were to be intelligent, efficient, and user-friendly.

It also needed to be able to handle very big files without crashing and comprehend the unique nested XML structure of Windows logs that Beats agents sent. Python 3 and its built-in Tkinter GUI framework were used to create the application [15]. The use of Python's threading framework to handle background processing was a crucial architectural decision. All laborious operations, such file indexing and searching, are carried out on a different background thread. Fig 8, represents results from the developed standalone Python GUI tool, showing how anomalies in logs are detected and highlighted for analysis.

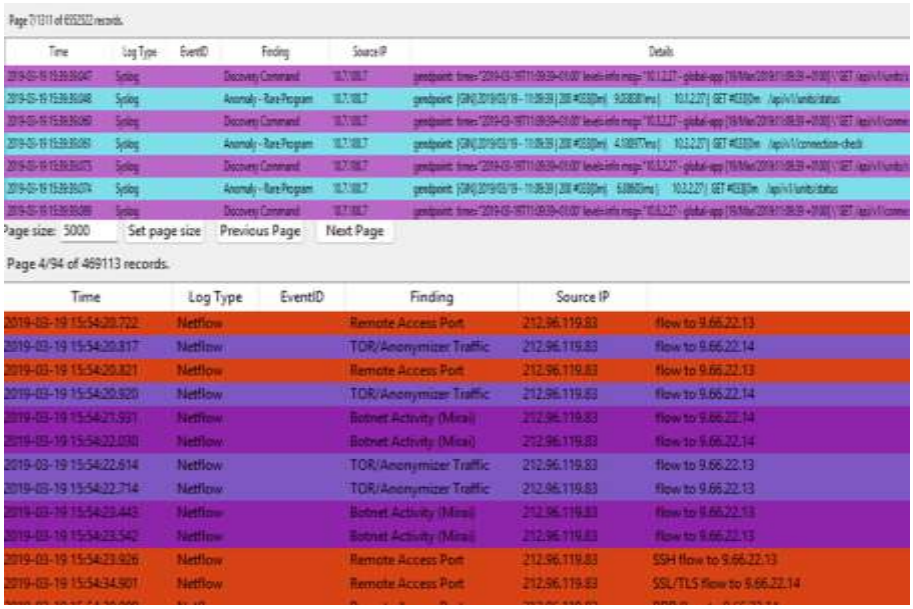

Fig.8. Anomalies Analysis

6. CONCLUSION AND FUTURE WORK

In the research work the design and implementation were addresses as well as evaluation of a full-fledged open-source framework to detect anomalies and track security. The technique bridges the visibility gap between endpoint and network security through drawing network-level traffic logs through Zeek and host-level event logs through Winlogbeat into the ELK stack. It was also found that with Windows failed logon events (Event ID 4625) directly against odd network connection patterns, combination can identify a simulated SMB brute-force attack at a high fidelity. Ideally, the key finding is an effective, SIEM solution system that provides the analysts with a single, context-related perception of security data with which it is easier to identify and analyse threats. Using this base, the second step would be to move away to automated detection of the manual analysis. The future work includes development of real-time correlation rules in Kibana that will automatically trigger SOC analysts into the brute-force pattern (a high number of Event 4625 of a single source IP and large number of TCP connections) and trigger SOC alerts using integrated open sources. There is a need to stress the system with a more expanded array of more intricate threats to design future research model to detect attacks striking from other stages of the MITRE ATT&CK grid, like privilege escalation, lateral movement (as Pass-the-Hash) and data exfiltration, advancing the detection limits still further.

REFERENCES

[1] A. Esseghir, F. Kamoun, and O. Hraiech, "AKER: An open-source security platform integrating IDS and SIEM functions with encrypted traffic analytic capability," *Journal of Cyber Security Technology*, vol. 6, no. 1–2, pp. 27–64, 2022.
[2] J. Su et al., "Large Language Models for Forecasting and Anomaly Detection: A Systematic Literature Review," 2024. [Online]. Available: http://arxiv.org/abs/2402.10350
[3] M. Koca, M. A. Aydin, A. Sertbas¸, and A. H. Zaim, "A new distributed anomaly detection approach for log IDS management based on deep learning," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 25, no. 9, pp. 2486–2501, 2021.
[4] S. A. R. Shah and B. Issac, "Performance comparison of intrusion detection systems and application of machine learning to Snort system," *Future Generation Computer Systems*, vol. 80, pp. 157–170, 2018.
[5] R. Agrawal, A. P. Srivastava, and A. Sugiyama, *International Conference on Sustainability in Digital Transformation Era (SIDTE 2023)*, 2024.
[6] M. Chaumont, "Deep learning in steganography and steganalysis," *Digital Media Steganography*, pp. 321–349, 2020.
[7] C. Smiliotopoulos, G. Kambourakis, and C. Kolias, "Detecting lateral movement: A systematic survey," *Heliyon*, vol. 10, no. 4, 2024.
[8] D. Tovarnˇak, S.´ Spaˇ cek, and J. Vykopal, "Traffic and log data captured during a cyber defense exercise,"ˇ *Data in Brief*, vol. 31, 2020.
[9] C. Islam, M. A. Babar, and S. Nepal, "A multi-vocal review of security orchestration," *ACM Computing Surveys*, vol. 52, no. 2, 2020.
[10] "Infrastructure as Code for Security Automation and Network Infrastructure Monitoring."
[11] M. Catillo, A. Pecchia, and U. Villano, "AutoLog: Anomaly detection by deep autoencoding of system logs," *Expert Systems with Applications*, vol. 191, 2022.
[12] Y. Xie, H. Zhang, and M. A. Babar, "LogGD: Detecting Anomalies from System Logs with Graph Neural Networks," *IEEE Conference on Software Quality, Reliability and Security*, 2022.
[13] A. R. S., "Advancing Protocol Diversity in Network Security Monitoring."
[14] G. Gonzalez-Granadillo, S. Gonz´ alez-Zarzosa, and R. Diaz, "Security information and event management (SIEM): Analysis, ´ trends, and usage," 2021.
[15] S. Buys, "Log Analysis aided by Latent Semantic Mapping," 2013.