JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Threat Intelligence Processing and Behavioral Anomaly Detection for Automated Security Monitoring

[1]Moksha Patel, [2]Anusha PP
[1]MSc CS Specialization in Cybersecurity, [2]Assistant Professor
[1,2]Department of Advanced Computing
[1,2]Nagindas Khandwala College
[1,2]University of Mumbai, Maharashtra, Mumbai

Abstract - Cyber threats continuously evolve, making timely identification of malicious activities an important requirement for modern security monitoring. Threat Intelligence (TI) provides information about known malicious entities such as IP addresses, domains, and other Indicators of Compromise (IOCs), while anomaly detection helps identify suspicious behavior that may not be present in threat intelligence databases. This research presents a lightweight Threat Intelligence Feed Processor and Anomaly Detector designed for local security monitoring. The proposed system collects malicious IP indicators from an external threat intelligence source, processes and stores them in a SQLite database, and compares incoming network and log events against the stored indicators. Events that match known malicious indicators are identified as threats, while events without an IOC match are further examined using behavioral anomaly detection. The system generates alerts with low, medium, or high-risk levels and stores relevant evidence for analysis. A Flask-based dashboard provides centralized visibility into IOCs, alerts, network activity, system activity, evidence, and security scores. In the experimental implementation, 1,000 AbuseIPDB indicators were processed, resulting in a total IOC database containing 2,736 IP records. The system analyzed 16 unique IP addresses, generated seven security alerts, and detected five behavioral anomalies in a controlled environment. The results demonstrate that combining threat intelligence with behavioral analysis can provide a practical approach for local security monitoring.

Keyword: *Cyber Threat Intelligence, Threat Intelligence Feed, Indicators of Compromise, Anomaly Detection, Malicious IP Detection, Security Monitoring, Risk Scoring, Automated Threat Detection.*

1. INTRODUCTION

The increasing number of cyberattacks has created a strong requirement for systems capable of identifying malicious activities at an early stage. Attackers may use malicious IP addresses, compromised systems, brute-force attempts, suspicious emails, and abnormal network behavior to target computer systems. Traditional security monitoring approaches that depend only on known signatures may fail to identify new or previously unseen activities.

Cyber Threat Intelligence (CTI) provides structured information about known threats and their associated Indicators of Compromise (IOCs). IOCs may include IP addresses, domain names, URLs, file hashes, email addresses, and other artifacts associated with malicious activities. By maintaining a database of known indicators, security systems can compare observed activity against previously identified threats.

However, IOC-based detection alone has limitations. An attacker may generate abnormal activity using an IP address that is not present in an existing threat intelligence database. Therefore, behavioral analysis and anomaly detection are useful complementary techniques. Abnormal request frequency, repeated authentication attempts, and unusual network activity can indicate suspicious behavior even when no known IOC is detected.

This research proposes a Threat Intelligence Feed Processor and Anomaly Detector that combines IOC-based detection with behavioral anomaly analysis. The system collects malicious IP indicators, processes them into a local SQLite database, monitors network and log activity, detects IOC matches, analyzes abnormal request behavior, and generates security alerts.

The proposed system is designed as a lightweight local security monitoring solution using Python, SQLite, and Flask. It provides a web-based dashboard that allows an analyst to view threat intelligence indicators, alerts, evidence, network events, system activity, and security scores.

2. LITERATURE REVIEW

Many studies have explored Cyber Threat Intelligence, IOC processing, SIEM integration, and anomaly detection as methods to enhance security monitoring.

2.1 Threat Intelligence and IOC Processing

Threat Intelligence systems provide information that can be used to identify and investigate cyber threats. IOC processing is an important part of CTI because it converts raw threat information into structured indicators that can be searched and correlated with observed activity.

Rastogi et al. presented TINKER, a framework for open-source cyber threat intelligence that supports the processing and analysis of threat information [1]. Liao et al. investigated automatic discovery and analysis of IOCs from open-source cyber threat intelligence [2]. These studies demonstrate the importance of automatically extracting and processing threat indicators.

Froudakis et al. examined IOC extraction from threat reports and highlighted challenges associated with obtaining accurate indicators from unstructured sources [3]. Chen et al. proposed the use of STIX graphs for improving IOC quality and relationships between indicators [4].

The proposed research follows the general concept of automated IOC processing but implements a lightweight local database-driven approach suitable for a controlled monitoring environment.

2.2 SIEM and Threat Feed Integration

Security Information and Event Management (SIEM) systems combine security events from multiple sources and support centralized analysis. Modern security monitoring solutions increasingly integrate external threat intelligence feeds to improve detection.

Siam et al. studied real-time threat feed integration with Wazuh SIEM for automated malware detection and response [5]. Bezas and Filippidou compared open-source SIEM systems and discussed their capabilities for security monitoring [6]. Sheeraz et al. investigated correlation mechanisms for multi-layered attack detection [7].

Unlike full-scale enterprise SIEM platforms, the proposed system focuses on a smaller local environment. It provides basic threat intelligence integration, event analysis, alert generation, and dashboard visualization, while using SQLite as the local database.

2.3 Log Anomaly Detection

Log data contains valuable information about system and network behavior. Analysis of logs can identify repeated requests, authentication failures, abnormal traffic patterns, and other suspicious activities.

Landauer et al. proposed a framework for extracting cyber threat intelligence from raw log data [8]. Pithode and Patheja investigated log anomaly detection using deep learning techniques [9]. Han et al. studied few-shot log anomaly detection using matching networks [10].

The proposed system uses a lightweight behavioral approach instead of complex deep learning models. This makes the implementation easier to deploy in a local, controlled environment while still providing an additional layer of detection beyond IOC matching.

3. RESEARCH GAP

Existing research demonstrates the effectiveness of threat intelligence processing, IOC extraction, SIEM integration, and log anomaly detection. However, many advanced security monitoring platforms require significant infrastructure, configuration, and computational resources.

There is a need for a lightweight implementation that demonstrates the complete flow from threat intelligence collection to IOC storage, network monitoring, anomaly detection, alert generation, evidence storage, and visualization.

The proposed system addresses this gap by combining threat intelligence processing and behavioral anomaly detection in a single Python-based local monitoring environment. The system also provides a dashboard for viewing the results without requiring a complete enterprise SIEM deployment.

4. PROBLEM STATEMENT

Traditional IOC-based detection systems mainly identify threats that are already known and recorded in threat intelligence databases. Such systems may not detect suspicious behavior when an unknown IP address generates excessive requests or abnormal activity.

The problem addressed in this research is to develop a lightweight security monitoring system capable of:

1. Collecting and processing threat intelligence indicators.
2. Maintaining a local database of malicious IP addresses.
3. Monitoring network and system log activity.
4. Matching observed IP addresses against known IOCs.
5. Detecting behavioral anomalies when an IOC match is not available.
6. Generating alerts based on detected threats.
7. Assigning risk levels to detected events.
8. Maintaining evidence and alert history.
9. Providing centralized visualization through a web dashboard.

5. OBJECTIVES OF THE RESEARCH

The major objectives of the proposed research are:

1. To collect and process malicious IP indicators from an external threat intelligence source.
2. To maintain a structured local IOC database.
3. To process at least 1,000 malicious IP indicators.
4. To monitor network and log activity in a controlled environment.
5. To identify known malicious IP addresses using IOC matching.
6. To detect suspicious behavioral patterns using anomaly detection.
7. To generate alerts with appropriate risk levels.
8. To maintain evidence and historical security events.
9. To provide a web-based dashboard for security monitoring.
10. To evaluate the effectiveness of combining IOC detection with anomaly detection.

6. PROPOSED METHODOLOGY AND SYSTEM ARCHITECTURE

6.1 Overall Methodology

The proposed methodology consists of multiple stages. First, the system obtains malicious IP indicators from an external threat intelligence source. The collected information is validated, cleaned, normalized, and stored in a local SQLite database.

Network and log events are then monitored by the system. The system extracts the source IP address from each relevant event and compares it with the stored IOC database. If the IP address matches a known malicious indicator, the event is classified as a known threat and an alert is generated.

If there is no IOC match, the system performs behavioral analysis. The number and pattern of requests are examined to identify excessive or abnormal activity. If suspicious behavior is detected, an anomaly alert is generated.

Detected events are assigned risk levels and stored as alerts and evidence. Finally, the Flask dashboard presents the collected information to the analyst.

6.2 System Architecture

The architecture consists of the threat intelligence source, collection and processing modules, IOC database, monitoring components, detection engines, alert management, and dashboard.

The threat intelligence collector obtains malicious IP information and forwards it for validation and normalization. The processed indicators are stored in the SQLite IOC database.

Network and log monitoring provides observed events to the IOC Matching Engine. Known malicious indicators are identified through database matching. Events without an IOC match are passed to the Anomaly Detection Module.

Detected threats and anomalies are sent to the Alert Manager. Risk scoring is then applied before storing alerts and evidence. The Flask Dashboard provides the final visualization layer for the analyst.

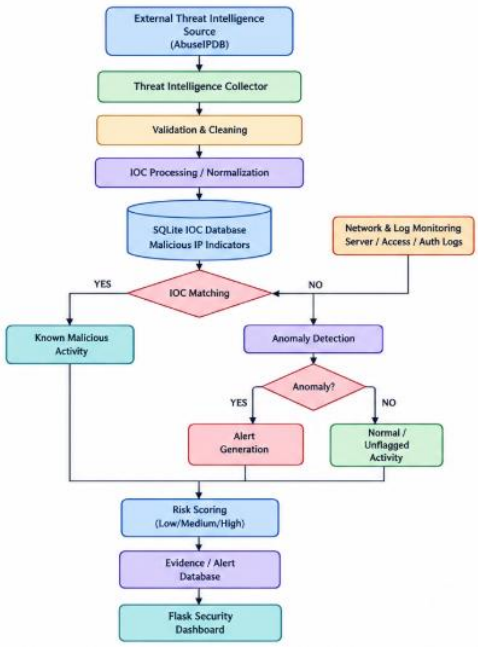

Fig 6.2.1: Threat Intelligence Feed Processor and Anomaly Detector – System Architecture

6.3 System Components

The proposed system contains the following major components:

1.Threat Intelligence Collector: Collects malicious IP indicators from the selected external threat intelligence source.

2.IOC Processing Module: Validates, cleans, normalizes, and prepares indicators for database storage.

3.SQLite IOC Database: Stores malicious IP indicators and associated information such as source, confidence, country, indicator type, and timestamp.

4.Network and Log Monitoring: Collects activity from network requests, server logs, access logs, and authentication-related logs.

5.IOC Matching Engine: Compares extracted IP addresses against known malicious indicators.

6.Anomaly Detection Module: Analyzes behavior when an IOC match is not available and identifies excessive or unusual activity.

7.Alert Manager: Creates alerts for detected threats and maintains alert history.

8.Risk Scoring Module: Assigns Low, Medium, or High risk levels according to the detection result.

9.Evidence Management: Stores relevant evidence and supporting information associated with security events.

10.Flask Dashboard: Provides centralized visualization of IOCs, alerts, network activity, system activity, evidence, and security scores.

6.4 Detection Decision Flow

The detection process uses two complementary mechanisms: IOC matching and behavioral anomaly detection.

For every monitored event, the source IP address is extracted and checked against the IOC database. If a match is found, the event is considered a known malicious IOC and an alert is generated.

If the IP address is not present in the IOC database, behavioral analysis is performed. Excessive requests or other suspicious patterns can result in an anomaly alert. Events that do not satisfy the anomaly conditions are treated as normal activity.

This approach allows the system to detect both known threats and suspicious behavior that may not yet be represented in the threat intelligence database.

6.5 System Workflow

The complete workflow begins when a network event is received. The source IP address is extracted and submitted to the IOC lookup process. The system determines whether the IP address exists in the malicious IOC database.

If the IOC is found, the system identifies the event as known malicious activity and generates an alert.

If no IOC match is found, behavioral analysis is performed. The system checks whether the

observed behavior satisfies the anomaly detection conditions.

The resulting event is assigned a risk level, stored as evidence or an alert, and displayed on the Flask dashboard.

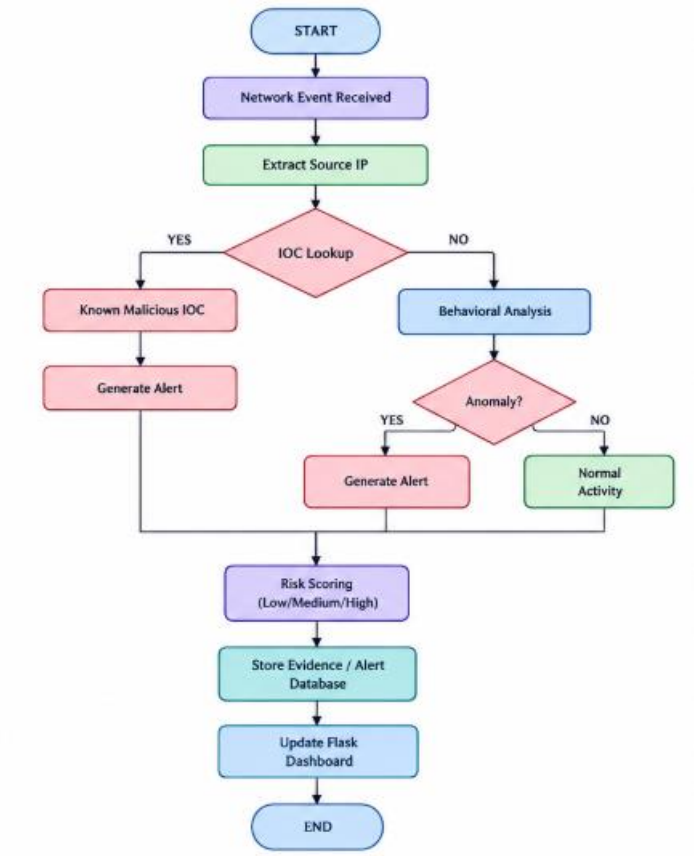

Fig. 6.5.1: Threat Intelligence Feed Processor and Anomaly Detection – Detection Workflow

7. IMPLEMENTATION DETAILS

7.1 Threat Intelligence and IOC Database

The system uses a local SQLite database for storing threat intelligence indicators. The system obtains malicious IP information from an external threat intelligence source.

The collected indicators are processed before insertion into the database. Duplicate and incomplete information is handled during processing. Each IOC record may contain the IP address, source, confidence information, country information, indicator type, and timestamp.

The implementation processed 1,000 malicious IP indicators from AbuseIPDB. After integration with the existing IOC records, the database contained 2,736 entries.

SQLite was selected because it is lightweight and does not require a separate database server. This makes it suitable for a local cybersecurity research prototype.

7.2 Network and Log Monitoring

Network and server activity is monitored through log files generated by the test environment. The system extracts source IP addresses and other relevant information from monitored events.

Access logs and authentication-related logs provide information about requests and login activity. These events are forwarded to the detection modules for IOC matching and behavioral analysis.

7.3 IOC Matching

IOC matching is performed by comparing an extracted source IP address against the malicious IP records stored in the SQLite database.

When an IP address matches an IOC, the system records the detection and generates a security alert. The alert contains information such as the detected indicator, detection source, risk level, timestamp, and relevant request information.

IOC matching provides fast identification of known malicious addresses without requiring complex behavioral analysis for every known threat.

7.4 Anomaly Detection

IOC matching cannot detect every suspicious event because an unknown attacker may use an IP address that has not yet been included in a threat intelligence feed.

The anomaly detection module therefore analyzes behavioral activity. One of the implemented conditions is excessive request activity. When the number of requests from an IP address crosses the configured detection threshold, the system identifies the behavior as anomalous.

This provides a second detection layer for previously unknown or unlisted sources.

7.5 Alert Management and Risk Scoring

The system generates alerts whenever a known malicious IOC or behavioral anomaly is detected.

Alerts are assigned risk levels such as Low, Medium, and High. The risk level depends on the detection type and observed activity.

The alert database maintains historical information including the alert identifier, alert type, indicator,

risk level, request count, source, status, and timestamp.

7.6 Web-Based Dashboard

A Flask-based dashboard provides centralized visibility into the security monitoring system.
The dashboard contains sections for:

- Dashboard
- IOC Management
- Email Analysis
- Network Analysis
- System Analysis
- Alerts
- Evidence
- Security Score
- Reports

The dashboard displays threat intelligence statistics, analyzed IP addresses, active alerts, behavioral anomalies, and other monitoring information.

7.7 Software Modules

The implementation consists of multiple Python modules responsible for different security functions. Major modules include the database management, IOC management, and email analysis modules, anomaly detector, brute-force detector, alert manager, scoring module, and Flask dashboard.
This modular design allows individual detection components to operate independently while contributing their results to the central monitoring system.

7.8 Threat Detection Logic and Algorithm

The main detection algorithm can be summarized as follows:

1. Receive a network or log event.
2. Extract the source IP address.
3. Search the IOC database.
4. If an IOC match exists:
   - Identify known malicious activity.
   - Generate security alert.
5. If an IOC match does not exist:
   - Perform behavioral analysis.
6. If the system detects anomalous behavior:
   - Generate anomaly alert.
7. Assign risk level.
8. Store alert and evidence.
9. Update the dashboard.
10. Continue monitoring.

8. EXPERIMENTAL EVALUATION AND RESULTS

8.1 Experimental Setup

The proposed system was implemented and tested in a controlled local cybersecurity environment using Kali Linux.

The main technologies included:

- **Operating System:** Kali Linux
- **Programming Language:** Python
- **Web Framework:** Flask
- **Database:** SQLite
- **Threat Intelligence Source:** AbuseIPDB
- **Monitoring:** Server and network logs
- **Visualization:** Flask-based web dashboard

The experiments were designed to evaluate IOC, anomaly, and brute-force detection, malicious email analysis, alert generation, and dashboard reporting.

Test Case	Description	Expected Result	Observed Result
TC-01	Malicious IP Matches IOC	IOC alert generated	Successful
TC-02	Unknown IP with normal activity	No threat alert	Successful
TC-03	Excessive requests	Behavioural anomaly detected	Successful
TC-04	Repeated login/request activity	Brute-force alert	Successful
TC-05	Malicious email indicator	Email alert generated	Successful
TC-06	IOC-based suspicious activity	High-risk alert	Successful
TC-07	Dashboard monitoring	Security information displayed	Successful

Table I. Experimental Test Cases and Results

8.2 Test Case Analysis

The first test case verified that the IOC database could identify known malicious IP addresses. The system successfully matched malicious indicators and generated alerts.

The second test case verified normal activity. When the IP address did not match a known IOC and did not exhibit abnormal behavior, the system did not generate a threat alert.

The third test case evaluated behavioral anomaly detection. The system identified excessive requests from an IP address as abnormal and generated an anomaly alert.

The fourth test case evaluated repeated requests or authentication activity and verified the brute-force detection functionality.

The fifth test case evaluated the email analysis component using malicious email indicators. The system identified the suspicious indicator and generated an alert.

The remaining test cases verified IOC-based detection, risk scoring, and dashboard visualization.

8.3 Overall Results

The experimental implementation processed **1,000 AbuseIPDB malicious IP indicators**. After combining the processed indicators with existing IOC records, the local database contained **2,736 IOC IP records**.

During testing, the system analyzed **16 unique IP addresses** and generated **7 security alerts**. The system identified **5 behavioral anomalies** among the detected events.

These results demonstrate that the proposed system can combine known threat intelligence with behavioral analysis to identify different categories of suspicious activity.

8.4 Implementation Statistics

The main implementation statistics are summarized below:

- Total IOC IP records: **2,736**
- AbuseIPDB indicators processed: **1,000**
- Unique IP addresses analyzed: **16**
- Security alerts generated: **7**
- Behavioral anomalies detected: **5**
- Risk levels: **Low, Medium, High**

The results were obtained from a controlled testing environment and should therefore be interpreted as implementation results rather than large-scale production performance measurements.

9. DISCUSSION

The experimental results indicate that integrating threat intelligence with behavioral analysis provides two complementary detection mechanisms.

IOC matching is useful for identifying known malicious IP addresses quickly. However, it depends on the availability and quality of threat intelligence data. An IP address that has not been previously reported may not produce an IOC match.

Behavioral anomaly detection addresses this limitation by examining activity patterns. The system identifies excessive requests even when the source IP is not present in the IOC database.

The combination of these techniques provides broader detection coverage than using either mechanism independently. The Flask dashboard further improves visibility by bringing IOC information, alerts, evidence, and monitoring statistics into one interface.

The current implementation is intentionally lightweight and suitable for a controlled research environment. It demonstrates the fundamental security monitoring workflow without requiring the infrastructure of a complete enterprise SIEM platform.

10. ADVANTAGES OF THE PROPOSED SYSTEM

The proposed system provides the following advantages:

1. **Multiple detection techniques:** IOC matching and behavioral anomaly detection operate together.
2. **Lightweight architecture:** Python, SQLite, and Flask reduce infrastructure requirements.
3. **Local IOC database:** Threat indicators can be searched quickly without depending on a remote database during detection.

4. **Behavioral detection:** The system identifies suspicious activity even without an IOC match.
5. **Alert history:** The system stores security events for later analysis.
6. **Evidence storage:** The system retains relevant information for investigation.
7. **Web-based monitoring:** Analysts can view security information through a centralized dashboard.
8. **Modular design:** Detection components can be extended independently.
9. **Risk classification:** The system categorizes alerts into Low, Medium, and High risk levels.
10. **Practical implementation:** The system can be demonstrated in a controlled cybersecurity laboratory.

11. LIMITATIONS

The proposed implementation has several limitations.

First, the system primarily focuses on IP-based IOC detection and does not provide comprehensive support for every IOC type. Second, the anomaly detection mechanism uses predefined behavioral conditions rather than advanced machine learning models.

The system was evaluated in a controlled local environment with a limited number of events. Therefore, the results cannot directly represent performance in a large enterprise network.

The accuracy of IOC-based detection also depends on the quality, freshness, and coverage of the external threat intelligence feed. False positives and false negatives may occur depending on the characteristics of the monitored activity.

12. FUTURE WORK

Future improvements can extend the system in several directions.

The IOC database can be expanded to support domains, URLs, file hashes, email addresses, and other IOC types. Multiple threat intelligence feeds can also be integrated to improve indicator coverage.

Machine learning techniques can be introduced for more advanced behavioral anomaly detection. Automated IOC expiration and reputation updates can improve the freshness of stored intelligence.

The system can also be integrated with enterprise SIEM platforms such as Wazuh. Automated response mechanisms could be added to block malicious IP addresses or isolate suspicious events.

Future versions may also include real-time streaming, advanced correlation between multiple events, improved visualization, and larger-scale evaluation using realistic network traffic.

13. CONCLUSION

This research presents a lightweight Threat Intelligence Feed Processor and Anomaly Detector that combines IOC-based threat detection with behavioral anomaly analysis.

The proposed system collects malicious IP indicators, processes and stores them in a SQLite database, monitors network and log activity, performs IOC matching, detects anomalous behavior, generates security alerts, stores evidence, and presents the results through a Flask-based dashboard.

The implementation processed 1,000 AbuseIPDB indicators and maintained a total of 2,736 IOC IP records. During controlled testing, 16 unique IP addresses were analyzed, 7 security alerts were generated, and 5 behavioral anomalies were detected.

The results demonstrate that combining known threat intelligence with behavioral analysis can provide a practical and lightweight approach to local security monitoring. Although the system has limitations in scale and detection complexity, it provides a foundation that can be extended through additional threat intelligence sources, machine learning-based anomaly detection, and enterprise security integrations.

REFERENCES

[1] N. Rastogi, S. Dutta, A. Gittens, M. J. Zaki, and C. C. Aggarwal, “TINKER: A Framework for Open-Source Cyberthreat Intelligence,” 2022. https://doi.org/10.1109/TrustCom56396.2022.00225

[2] X. Liao, K. Yuan, X. Wang, Z. Li, L. Xing, and R. Beyah, “Acing the IOC Game: Toward Automatic Discovery and Analysis of Open-Source

Cyber Threat Intelligence," 2016. https://doi.org/10.1145/2976749.2978315

[3] E. Froudakis, A. Avgetidis, S. T. Frankum, R. Perdisci, M. Antonakakis, and A. D. Keromytis, "Revealing the True Indicators: Understanding and Improving IoC Extraction From Threat Reports," 2025. https://arxiv.org/abs/2506.11325

[4] S.-S. Chen, R.-H. Hwang, A. Ali, Y.-D. Lin, Y.-C. Wei, and T.-W. Pai, "Improving Quality of Indicators of Compromise Using STIX Graphs," 2024. https://doi.org/10.1016/j.cose.2024.103972

[5] M. Landauer, F. Skopik, M. Wurzenberger, W. Hotwagner, and A. Rauber, "A Framework for Cyber Threat Intelligence Extraction From Raw Log Data," 2019. https://doi.org/10.1109/BigData47090.2019.9006328

[6] K. Pithode and P. S. Patheja, "A Study on Log Anomaly Detection Using Deep Learning Techniques," 2022. https://doi.org/10.1109/ICAAIC53929.2022.9793238

[7] A. A. Siam, M. Hassan, A. K. M. Masum, and T. Bhuiyan, "Automating Malware Detection and Response via Real-Time Threat Feed Integration With Wazuh SIEM," 2025. https://doi.org/10.1109/COMPAS67506.2025.11381876

[8] K. Bezas and F. Filippidou, "Comparative Analysis of Open Source Security Information and Event Management Systems (SIEMs)," 2023. https://doi.org/10.33022/ijcs.v12i2.3182

[9] 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," 2024. https://doi.org/10.3390/s24154901

[10] C. Han, B. Guan, T. Li, D. Kang, J. Qin, and Y. Wu, "Few-Shot Log Anomaly Detection Based on Matching Networks," 2024. https://doi.org/10.1109/TNSM.2024.3363626