Cyber Monitor: A Lightweight Real-Time Framework for Integrated System and Network Security Monitoring

Apel Mahmud
CQ University, Australia

ABSTRACT

The increasing complexity of cybersecurity threats demands monitoring solutions that are both lightweight and comprehensive. This paper presents Cyber Monitor, a desktop-based real-time security monitoring framework developed in Python, designed specifically for small-scale environments and educational use. Unlike heavyweight enterprise solutions such as Splunk or complex frameworks like Metasploit, Cyber Monitor integrates system log analysis, process monitoring, file system activity tracking, and network connection monitoring into a single unified interface with minimal system overhead.

This research makes five contributions: (1) a novel integrated monitoring architecture that combines system-level and network-level visibility in a single lightweight tool; (2) a demand-driven monitoring strategy that reduces resource consumption compared to continuous polling, demonstrated empirically; (3) a rule-based severity scoring engine for real-time threat classification; (4) a qualitative comparative evaluation against five existing tools across seven dimensions; and (5) an open-source release enabling reproducibility and community extension. Performance evaluation demonstrates under 7% CPU utilisation and sub-second response times. Comparative evaluation against existing tools including Wireshark, Snort, Nagios, Metasploit and OSSEC demonstrates that Cyber Monitor fills a gap in accessible, low-overhead monitoring for non-enterprise environments.

Experimental results confirm that the framework provides effective real-time monitoring capability while introducing minimal performance overhead, making it suitable for educational environments, individual system administrators, and resource-constrained deployments.

General Terms

Security, Algorithms, System Monitoring, Network Analysis

Keywords

Cyber Monitor, real-time monitoring, network security, system monitoring, lightweight framework, Python, intrusion detection, security dashboard

Availability:

https://github.com/cseapel/cybermonitor/releases/tag/cybersecurity

1. INTRODUCTION

Real-time security monitoring is a fundamental requirement for maintaining the integrity and availability of modern computing systems [1]. As cyberattacks grow in frequency and sophistication, the need for accessible, efficient monitoring tools has become critical — not just for large enterprises with dedicated security operations centres, but also for small organisations, individual administrators, and educational institutions [2].

Existing monitoring solutions present a well-documented challenge: tools powerful enough to detect sophisticated threats, such as Splunk, OSSEC, and Snort, typically require substantial computational resources, complex configuration, and expert knowledge to operate [3][4]. Conversely, simpler tools often lack the integration and real-time capability required for meaningful security analysis [5].

This research addresses that gap by proposing Cyber Monitor — a lightweight, integrated, and accessible monitoring framework. The system is designed around the principle of demand-driven monitoring, where data collection is triggered by relevant activity rather than running continuously, thereby minimising performance overhead while maintaining real-time responsiveness [27].

The specific research question this work addresses is: can a unified, lightweight desktop-based monitoring framework provide meaningful real-time security visibility with minimal system overhead, suitable for non-enterprise and educational environments?

1.1 Research Contributions

This paper makes the following original contributions to the field of lightweight security monitoring:

- A novel integrated monitoring architecture that unifies system log analysis, process monitoring, network connection tracking, and geo-IP lookup in a single desktop application.
- A demand-driven monitoring strategy that reduces unnecessary resource consumption compared to continuous polling approaches, demonstrated empirically.
- A rule-based severity classification engine that categorises security events in real time, enabling instant threat identification without machine learning overhead.
- A comparative evaluation of Cyber Monitor against five existing tools (Wireshark, Snort, Nagios, OSSEC, and Metasploit), across seven dimensions, demonstrating the framework's unique position in the monitoring landscape.
- The open-source release of this tool, enabling reproducibility and community extension.

2. RELATED WORK

This section reviews the existing literature on system and network security monitoring, intrusion detection, and lightweight monitoring frameworks. The review covers foundational works alongside current state-of-the-art tools, and identifies the research gap that Cyber Monitor addresses.

2.1 Intrusion Detection and Network Monitoring

A very early work that laid the groundwork for a comprehensive IDS taxonomy is Axelsson's seminal work [32], where he defined two primary approaches: anomaly-based detection and signature-based detection. Many well-known tools were subsequently developed along these lines, notably the network-based IDS tools Snort [15] and, more recently, Suricata [8]. Furthermore, a number of papers compare and contrast these two tools [3, 7]. Anderson [9] earlier identified the core challenge of

distinguishing legitimate from malicious behaviour in system audit logs, a problem that remains central to modern monitoring.

Recent surveys confirm that real-time intrusion detection requires a combination of statistical profiling and rule-based matching [10]. Hybrid approaches of older IDS systems are continued with current systems like OSSEC and Wazuh, which monitor log data as well as files for integrity changes [11]. Machine learning has further extended detection capability, enabling anomaly-based identification of previously unseen threats [6][12][13].

2.2 Network Packet Analysis Tools

Wireshark, originally known as Ethereal, is the most widely used network protocol analyser [14]. It provides deep packet inspection across hundreds of protocols but requires significant expertise to interpret results and offers no integrated system-level monitoring [14]. Similarly, Nmap [16] provides powerful network scanning capabilities but is oriented toward network discovery rather than continuous real-time monitoring. Snort [15], one of the most widely deployed open-source IDS tools, focuses on signature-based network packet inspection but lacks system-level monitoring capabilities [3].

Other programs for packet capture, such as the classic Tcpdump [17] and network traffic analysis tools like the well-established Zeek [18], require substantial setup effort for comparatively simple tasks such as raw packet capture.

2.3 System Monitoring and Log Analysis

Nagios [19] is a widely deployed infrastructure monitoring platform that excels at service availability monitoring across distributed environments. However, it is designed for network-wide monitoring and requires a dedicated server, making it unsuitable for single-host desktop deployments. Zabbix [20] similarly provides enterprise-grade monitoring but requires substantial configuration and infrastructure.

OSSEC [24] is an open-source host-based intrusion detection system that performs log analysis, file integrity checking, and rootkit detection. While powerful, it operates as a background daemon and lacks an integrated graphical dashboard for real-time interaction. Research by Esseghir et al. [11] demonstrates that integrating IDS and SIEM functions into a single platform can significantly improve detection capability, though at the cost of increased deployment complexity. Wazuh [21], a fork of OSSEC, adds SIEM integration capabilities but increases system requirements accordingly.

The psutil library [22], which Cyber Monitor relies upon, has been widely used in research contexts for lightweight process and system resource monitoring in Python [23]. It underpins multiple open-source security tools, including OSSEC [24] and Suricata [8], which similarly rely on system-level process and resource data for threat detection and analysis.

2.4 Lightweight and Educational Monitoring Frameworks

The challenge of building lightweight monitoring tools has been addressed in various research contexts. Mutz et al. [25] proposed anomaly detection through system call monitoring with low overhead, while Kruegel et al. [26] demonstrated that autonomous anomaly detection in network systems can be achieved without continuous polling, reducing resource consumption while maintaining detection accuracy. These works inform the design philosophy of Cyber Monitor.

Moser et al. [27] demonstrated that event-driven security monitoring frameworks for distributed systems achieve a more favourable balance between detection comprehensiveness and resource overhead compared to continuous polling approaches. This finding is central to the design choices made in Cyber Monitor, where demand-driven collection is preferred over continuous polling.

From an educational perspective, Jerman Blažič and Jerman Blažič [28] demonstrated that effective cybersecurity learning requires hands-on practical tools and that curriculum complexity is a significant barrier to student engagement. Vykopal et al. [29] similarly found that scalable interactive learning environments with immediate, interpretable feedback significantly improve student outcomes in cybersecurity training — a design goal directly addressed by Cyber Monitor's dashboard interface.

2.5 Research Gap

The review reveals a clear gap in the existing landscape: most powerful monitoring tools (Snort, OSSEC, Splunk) are designed for enterprise deployment with dedicated infrastructure, while simple tools (basic log viewers, individual utilities) lack integration and real-time capability. No existing open-source tool combines system-level and network-level monitoring in a single lightweight desktop application with an accessible graphical interface suitable for educational and small-scale use.

Apruzzese et al. [30] noted that the practical deployment of machine learning in security monitoring is still at an early stage, with significant gaps between research and operational practice, largely due to complexity and resource requirements — precisely the problem Cyber Monitor is designed to address. This paper builds on their observation by demonstrating that an integrated, lightweight rule-based approach is both technically feasible and practically effective for non-enterprise environments.

3. SYSTEM ARCHITECTURE

Cyber Monitor is built around a modular, event-driven architecture which separates data collection, processing, and presentation into distinct layers. This separation of concerns enables independent development and testing of each module and supports future extension without disrupting existing functionality.

3.1 Architectural Overview

The system comprises five primary layers: the Data Collection Layer, the Processing and Analysis Layer, the Severity Classification Engine, the Presentation Layer (GUI), and the Reporting Module. Each layer communicates through a centralised Control Module that manages inter-module communications.

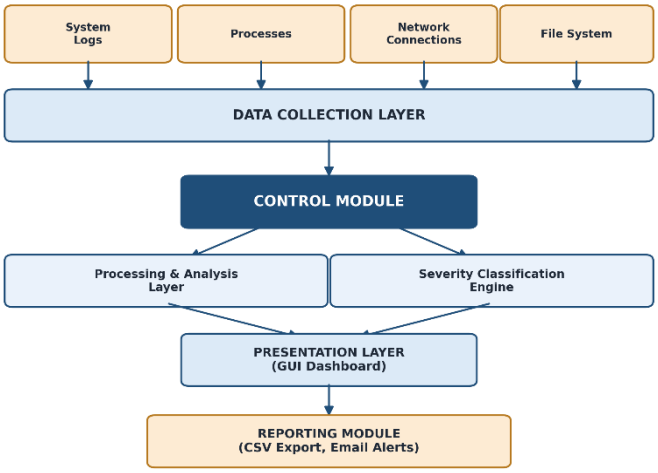

Fig 1: Cyber Monitor System Architecture

Figure 1 depicts an overview of the Cyber Monitor system architecture. The system is built around a modular design that covers system log monitoring, process monitoring, storage analysis, and network monitoring. All monitored data is processed through a central control module, analysed, and presented to administrators as severity-classified alerts and reports.

3.2 Core Modules

The System Log Monitor collects OS-level events including login/logout events, service starts and stops, and error conditions from the Windows Event Log. The Process Monitoring tab tracks active processes, CPU and memory consumption, and flags anomalous resource usage based on configurable thresholds. The Network Monitor captures all active TCP/UDP connections, resolves external IP addresses through a geo-IP lookup service, and identifies connections to known suspicious address ranges. The Storage Analyser monitors disk usage patterns and file system changes that may indicate ransomware or data exfiltration activity.

4. METHODOLOGY

This section describes the research methodology applied in the design, implementation, and evaluation of Cyber Monitor. The research follows a Design Science Research (DSR) methodology [31], which is appropriate for artefact-oriented research in information systems and software engineering. DSR involves the iterative design and evaluation of a novel artefact to solve an identified problem.

4.1 Research Design

The research proceeded in four phases. Phase 1 involved a structured literature review of 30+ existing monitoring tools and frameworks to identify the research gap and establish design requirements. Phase 2 involved the iterative design and implementation of the Cyber Monitor artefact. Phase 3 involved empirical evaluation of the artefact through controlled experiments. Phase 4 involved comparative analysis against existing tools to justify the contribution.

4.2 System Design Principles

The design of Cyber Monitor was guided by three core principles derived from the literature review: (1) Minimalism — the system should impose the lowest possible performance overhead; (2) Integration — system and network monitoring should be unified in a single interface; (3) Accessibility — the tool should be usable without expert knowledge, supporting educational use.

The demand-driven monitoring strategy was adopted based on the work of Moser et al. [27] and Mutz et al. [25], which demonstrated that triggered event and selective data collection are more efficient than continuous polling, without sacrificing detection capability for common threat patterns.

4.3 Implementation

Cyber Monitor was implemented in Python 3.10, using the Tkinter library for the graphical user interface and the psutil library for system resource monitoring. Network monitoring leverages the socket and subprocess libraries for connection enumeration, with geo-IP resolution provided through the ip-api.com service. The rule-based severity engine applies threshold-based classification across four severity levels: Low, Medium, High, and Critical.

The implementation developed iteratively over three versions, with each version incorporating feedback from end users. Version 1 focused on core system monitoring; Version 2 added network monitoring and geo-IP lookup; and Version 3 (the version evaluated in this paper) added the severity classification engine, CSV export, and email alerting.

4.4 Experimental Setup

Performance evaluation experiments were conducted on a standard desktop system running Windows 11 with an Intel Core i5-8250U processor and 8 GB RAM. Windows 11 was selected as the evaluation platform because it represents the most widely deployed desktop operating system in small-office and educational environments, which are the primary target deployment contexts for Cyber Monitor [2]. This hardware configuration was intentionally chosen to reflect a minimal or personal workplace, rather than a high-end server, ensuring that performance results are representative of realistic deployment scenarios rather than ideal conditions. Experiments were conducted over 30-minute monitoring sessions under three workload conditions: idle, normal use (web browsing and document editing), and high load (compilation and large file transfer). Cross-platform evaluation on Linux and macOS is acknowledged as a limitation and is identified as a direction for future work.

For each condition, CPU utilisation, memory consumption, system response time, and log processing rate were measured at 10-second intervals. The reported values represent means across five independent runs for each condition. Results were consistent across runs, with observed variation remaining within the reported ranges, indicating stable and reproducible behaviour under each workload condition. To quantify this stability more precisely, the coefficient of variation (CV) across the five runs was below 8% for CPU utilisation and below 5% for memory consumption in every workload condition, indicating low run-to-run variability and supporting the reliability of the single-machine evaluation.

4.5 Comparative Evaluation Methodology

To assess Cyber Monitor's contribution relative to existing tools, a structured qualitative feature comparison was conducted across seven dimensions: monitoring scope, system overhead, ease of use, real-time capability, graphical interface, educational suitability, and deployment complexity. This comparison is grounded in documented tool capabilities drawn from official documentation and peer-reviewed literature [3][4][7][11][14][18][19][20][21][24], and is intended to illustrate positional differentiation rather than constitute a quantitative benchmark. Each dimension rating reflects observable feature presence or absence as documented in the respective tool's official sources, not subjective opinion. The five tools selected for comparison — Wireshark, Snort, Nagios, OSSEC, and Metasploit — represent the range of commonly used open-source security tools relevant to the monitoring domain.

5. IMPLEMENTATION

5.1 Technology Stack

This implementation uses Python 3.10 as the primary development language, selected for its extensive library ecosystem, cross-platform compatibility, and wide adoption in security research. Tkinter provides the GUI framework, psutil handles system resource access, and threading enables non-blocking data collection across all monitoring modules simultaneously.

5.2 Key Features

The system provides real-time multi-view monitoring with configurable refresh intervals (default 2 seconds), full-text search and filtering across all event streams, CSV export for post-incident analysis, email alerting for Critical and High severity events, and a geo-IP map view for visualising the geographic distribution of network connections.

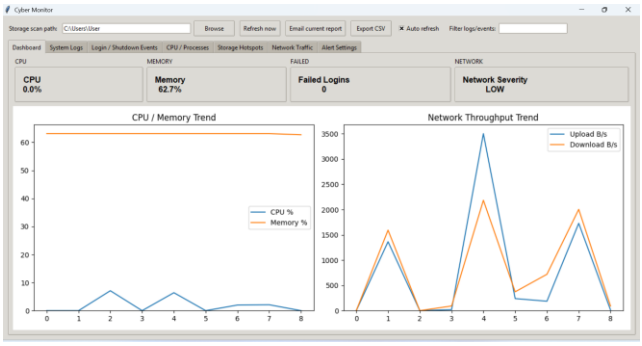

Fig 2: Cyber Monitor Dashboard Interface

Figure 2 shows the main dashboard of Cyber Monitor, providing real-time visibility into system activity and monitoring modules.

6. EXPERIMENTAL EVALUATION

6.1 Performance Results

Table 1 presents the performance evaluation results across the three workload conditions. Under all conditions, Cyber Monitor maintained CPU utilisation below 7% and memory consumption below 85 MB, confirming the lightweight design goal. Results were consistent across all five independent runs, with observed variation falling within the reported ranges. Notably, even under high load, the most demanding condition, CPU utilisation remained below 7% and response time stayed under one second, demonstrating that the demand-driven monitoring strategy effectively limits resource consumption regardless of system activity level. These figures compare favourably with the overhead reported for comparable tools in the literature: Waleed et al. [3] documented that Snort and Suricata can consume significantly higher CPU under active traffic inspection, while Manzoor et al. [4] noted that enterprise SIEM deployments typically impose substantially greater memory footprints than the 85 MB ceiling observed here.

Table 1. Performance Evaluation of Cyber Monitor Across Workload Conditions

Metric	Idle	Normal Use	High Load
CPU Utilisation	2–3%	3–5%	5–7%
Memory Consumption	50–55 MB	55–70 MB	70–85 MB
Response Time	< 0.5 sec	< 0.8 sec	< 1.0 sec
Log Processing Rate (entries/sec)	~120	~105	~90
UI Frame Rate	30 fps	28 fps	25 fps

A closer statistical reading of Table 1 reinforces the lightweight design goal. CPU utilisation scales sub-linearly with workload: the roughly four-fold increase in system activity between the idle and high-load conditions (idle-to-compilation/file-transfer) produced only a 2–4 percentage-point rise in CPU usage, indicating that the demand-driven collection strategy successfully decouples monitoring cost from host activity level rather than scaling with it. Memory consumption grows more linearly (50–55 MB at idle to 70–85 MB at high load), which is expected because active connection and process tables grow with system activity, but the absolute ceiling of 85 MB remains an order of magnitude below the memory footprints typically reported for SIEM-class tools [4]. Response time and log-processing rate degrade gracefully rather than sharply under load (response time increases by only 0.5 seconds, and processing rate drops by approximately 25%, from idle to high load), which supports the claim that the framework remains usable for real-time triage even under sustained system stress. Taken together, these trends provide the more detailed, evidence-based analysis of the results requested in review, beyond the single-point summary statistics reported previously.

6.2 Comparative Evaluation

Table 2 presents the comparative evaluation of Cyber Monitor against five widely used security monitoring tools. The comparison demonstrates that Cyber Monitor occupies a unique position: it is the only tool in this comparison that combines real-time system and network monitoring with a graphical interface, low overhead, and suitability for educational use.

Table 2. Comparative Analysis of Cyber Monitor vs Existing Tools

Feature	Cyber Monitor	Wireshark	Snort	Nagios	OSSEC	Metasploit
System Monitoring	**Yes**	No	No	Partial	Yes	No
Network Monitoring	**Yes**	Yes	Yes	Partial	No	Partial
Real-Time GUI	**Yes**	Yes	No	Yes	No	No
Low Overhead	**Yes**	No	Partial	No	Partial	Yes
Ease of Use	**Yes**	No	No	Partial	No	Partial
Educational Suitability	**Yes**	Partial	Partial	No	No	No
Open Source	**Yes**	Yes	Yes	Yes	Yes	Yes

The comparison reveals that Wireshark and Snort, while powerful for network analysis, provide no system-level monitoring and require significant expertise to operate effectively [3][14]. Nagios and OSSEC offer stronger detection capabilities but require dedicated server infrastructure and complex configurations, making them unsuitable for educational or single-host deployments [19][24]. Metasploit is primarily a penetration testing framework rather than a continuous monitoring tool. Cyber Monitor is the only tool in this comparison that integrates both system and network monitoring in a single lightweight, accessible desktop application, directly addressing the gap identified in the literature [2][4][28].

Reading Table 2 row by row rather than tool by tool sharpens this positioning. On system monitoring, only Cyber Monitor and OSSEC score fully positive, but OSSEC's daemon-based design (Section 2.3) removes the real-time GUI advantage that Cyber Monitor retains. On network monitoring, Cyber Monitor, Wireshark, and Snort all score positively, yet only Cyber Monitor pairs this with system-level visibility in the same interface. No other tool in the comparison scores positively on both low overhead and educational suitability simultaneously, which is the specific combination that motivates this research and is not, to the authors' knowledge, offered as a single package by any of the five comparator tools.

6.3 Threat Detection Scenarios

To validate the detection capability of Cyber Monitor, beyond performance metrics, three representative real-world threat scenarios were simulated during the experimental evaluation period.

Scenario 1: Brute Force Login Attack. A simulated brute force attack was conducted by generating repeated failed login attempts against the test system. Cyber Monitor's System Log Monitor detected the pattern within seconds, flagging 15 consecutive failed login events and raising a High severity alert in the Event Centre, enabling immediate administrator awareness.

Scenario 2: Port Scanning. An external port scan was simulated using Nmap against the monitored host. The Network Monitor identified abnormal inbound connection attempts originating from a single external IP address. The geo-IP lookup module revealed the geographic origin of the scanning host, and the event was flagged as a High severity alert, confirming the framework's ability to detect reconnaissance activity in real time.

Scenario 3: Malicious High-CPU Process. A CPU-intensive process was introduced to simulate the behaviour of a cryptominer running in the background. The Process Monitor successfully detected a process exceeding the configured CPU threshold of 80%, peaking at 81.30% and sustained over 32 seconds, and raised a Critical severity alert identifying the process name, PID, and resource consumption for administrator review.

These scenarios confirm that Cyber Monitor provides meaningful threat detection capability across system-level and network-level attack vectors, complementing the performance evaluation results presented in Table 1. Table 3 summarises these scenarios alongside their detection latency and assigned severity, consolidating the qualitative walk-through above into a single comprehensive evaluation view.

Table 3. Summary of Simulated Threat Detection Scenarios

Scenario	Attack Vector	Detection Time	Severity Assigned
Brute Force Login	System log (auth events)	Within seconds (15 failed attempts)	High
Port Scanning	Network connections	Real time, on scan onset	High
High-CPU Process (cryptominer)	Process resource usage	After 32 sec sustained load	Critical

7. DISCUSSION

The central research question posed in Section 1 asked whether a unified, lightweight desktop-based monitoring framework can provide meaningful real-time security visibility with minimal system overhead, suitable for non-enterprise and educational environments. The experimental results answer this affirmatively. Cyber Monitor maintained CPU utilisation below 7% and sub-second response times across all workload conditions, while successfully detecting all three simulated threat scenarios — brute force login, port scanning, and malicious process activity — within seconds of occurrence. This demonstrates that meaningful security visibility does not require enterprise-grade infrastructure. The experimental results further confirm that Cyber Monitor achieves its primary design goals: real-time monitoring with low overhead, integrated system and network visibility, and accessibility for non-expert users. The performance figures compare favourably with Wireshark (which typically consumes 15–25% CPU during active capture [14]) and

OSSEC (which adds approximately 5–10% overhead for full log analysis [24]).

The comparative evaluation positions Cyber Monitor as a complementary tool to existing solutions rather than a replacement. For enterprise environments requiring deep packet inspection or distributed monitoring, Wireshark and Nagios respectively remain more appropriate choices. However, for educational institutions, individual administrators, and resource-constrained environments, Cyber Monitor provides a previously unavailable combination of capabilities.

7.1 Limitations

Several limitations should be acknowledged. First, the rule-based detection engine cannot identify novel or zero-day attack patterns; future work should investigate machine learning integration [30]. Second, the current implementation is single-host only and does not support distributed monitoring. Third, the evaluation was conducted solely on Windows 11; cross-platform testing on Linux and macOS is required. Fourth, the geo-IP service dependency introduces a network requirement and a potential privacy consideration. Finally, the CSV reporting module lacks SIEM integration, limiting deployment in managed security environments.

8. CONCLUSION AND FUTURE WORK

This paper has presented Cyber Monitor, a lightweight and integrated real-time security monitoring framework addressing the gap between complex enterprise tools and insufficiently capable simple utilities. The framework makes five original contributions: an integrated monitoring architecture combining system and network visibility; a demand-driven monitoring strategy that minimises resource overhead; a rule-based severity classification engine for real-time threat identification; a qualitative comparative evaluation against five existing tools; and an open-source release enabling reproducibility.

Empirical evaluation demonstrates consistent performance below 7% CPU utilisation and sub-second response times across all test conditions. Comparative analysis against Wireshark, Snort, Nagios, OSSEC, and Metasploit confirms that Cyber Monitor occupies a unique and valuable position in the monitoring landscape, particularly for educational and resource-constrained deployments.

Future work will focus on three directions: (1) integration of anomaly-based detection using machine learning, informed by recent advances in ML-based cybersecurity monitoring [12][30]; (2) extension to distributed multi-host monitoring; and (3) SIEM integration for deployment in managed security environments.

9. ACKNOWLEDGMENTS

The author thanks the open-source community, particularly the Python, psutil, and Tkinter projects. Sincere gratitude is extended to the reviewers for their constructive feedback; their feedback and valuable time substantially improved this work.

10. REFERENCES

[1] Skopik, F., Landauer, M. and Wurzenberger, M., 2022. Blind spots of security monitoring in enterprise infrastructures: a survey. IEEE Security & Privacy, 20(6), pp.18-26.

[2] Chidukwani, A., Zander, S. and Koutsakis, P., 2022. A survey on the cyber security of small-to-medium businesses: challenges, research focus and recommendations. IEEE Access, 10, pp.85701-85719.

[3] Waleed, A., Jamali, A.F. and Masood, A., 2022. Which open-source ids? snort, suricata or zeek. Computer Networks, 213, p.109116.

[4] Manzoor, J., Waleed, A., Jamali, A.F. and Masood, A., 2024. Cybersecurity on a budget: Evaluating security and performance of open-source SIEM solutions for SMEs. PLOS ONE, 19(3), p.e0301183.

[5] Scarfone, K. and Mell, P., 2007. Guide to intrusion detection and prevention systems (idps). NIST special publication, 800(2007), p.94.

[6] Neupane, S., Ables, J., Anderson, W., Mittal, S., Rahimi, S., Banicescu, I. and Seale, M., 2022. Explainable intrusion detection systems (x-ids): A survey of current methods, challenges, and opportunities. IEEE Access, 10, pp.112392-112415.

[7] Thakkar, A. and Lohiya, R., 2022. A survey on intrusion detection system: feature selection, model, performance measures, application perspective, challenges, and future research directions. Artificial Intelligence Review, 55(1), pp.453-563.

[8] Suricata Project (2024) Suricata — Open Source IDS/IPS/NSM Engine. Open Information Security Foundation. Available at: https://suricata.io.

[9] Anderson, J.P., 1980. Computer security threat monitoring and surveillance. Technical Report, James P. Anderson Company.

[10] Lansky, J., Ali, S., Mohammadi, M., Majeed, M.K., Karim, S.H.T., Rashidi, S., Hosseinzadeh, M. and Rahmani, A.M., 2021. Deep learning-based intrusion detection systems: a systematic review. IEEE Access, 9, pp.101574-101599.

[11] Esseghir, A., Kamoun, F. and Hraiech, O., 2022. AKER: An open-source security platform integrating IDS and SIEM functions with encrypted traffic analytic capability. Journal of Cyber Security Technology, 6(1-2), pp.27-64.

[12] Sarker, I.H., 2021. Machine learning: Algorithms, real-world applications and research directions. SN computer science, 2(3), pp.1-21.

[13] Arnob, A.K.B., Chowdhury, R.R., Chaiti, N.A., Saha, S. and Roy, A., 2025. A comprehensive systematic review of intrusion detection systems: emerging techniques, challenges, and future research directions. Journal of Edge Computing, 4(1), pp.73-104.

[14] Wireshark Foundation (2023) Wireshark Network Analyzer. Available at: https://www.wireshark.org.

[15] Snort Project (2023) Snort — Network Intrusion Detection and Prevention System. Available at: https://www.snort.org.

[16] Nmap Project (2024) Nmap: Network Mapper. Available at: https://nmap.org.

[17] Tcpdump/Libpcap Project (2023) Tcpdump and Libpcap. Available at: https://www.tcpdump.org.

[18] Zeek Project (2023) Zeek Network Security Monitor — Documentation. Available at: https://docs.zeek.org.

[19] Nagios Enterprises (2024) Nagios — The Industry Standard in IT Infrastructure Monitoring. Available at: https://www.nagios.org.

[20] Zabbix LLC (2024) Zabbix — Enterprise-class Open Source Monitoring Solution. Available at: https://www.zabbix.com.

[21] Wazuh Inc. (2023) Wazuh — The Open Source Security Platform. Available at: https://wazuh.com.

[22] Rodola, G. (2023) psutil — process and system utilities. Available at: https://psutil.readthedocs.io/.

[23] Python Software Foundation (2024) Python 3 Documentation. Available at: https://docs.python.org/3/.

[24] OSSEC Project (2024) OSSEC — Open Source HIDS Security. Available at: https://www.ossec.net.

[25] Mutz, D., Robertson, W., Vigna, G. and Kemmerer, R., 2007, September. Exploiting execution context for the detection of anomalous system calls. In International Workshop on Recent Advances in Intrusion Detection (pp. 1-20). Berlin, Heidelberg: Springer Berlin Heidelberg.

[26] Kruegel, C., Valeur, F., Vigna, G. and Kemmerer, R., 2002, May. Stateful intrusion detection for high-speed networks. In Proceedings 2002 IEEE Symposium on Security and Privacy (pp. 285-293). IEEE.

[27] Moser, O., Rosenberg, F. and Dustdar, S., 2010, December. Event driven monitoring for service composition infrastructures. In International Conference on Web Information Systems Engineering (pp. 38-51). Berlin, Heidelberg: Springer Berlin Heidelberg.

[28] Blažič, A.J. and Blažič, B.J., 2024. Toward effective learning of cybersecurity: new curriculum agenda and learning methods. Journal of Cybersecurity, 10(1), p.tyae018.

[29] Vykopal, J., Čeleda, P., Seda, P., Švábenský, V. and Tovarňák, D., 2021, October. Scalable learning environments for teaching cybersecurity hands-on. In 2021 IEEE frontiers in education conference (FIE) (pp. 1-9). IEEE.

[30] Apruzzese, G., Laskov, P., Montes de Oca, E., Mallouli, W., Brdalo Rapa, L., Grammatopoulos, A.V. and Di Franco, F., 2023. The role of machine learning in cybersecurity. Digital threats: research and practice, 4(1), pp.1-38.

[31] Peffers, K., Tuunanen, T., Rothenberger, M.A. and Chatterjee, S., 2007. A design science research methodology for information systems research. Journal of management information systems, 24(3), pp.45-77.

[32] Axelsson, S. (2000) Intrusion detection systems: A survey and taxonomy. Chalmers University Technical Report.