Brilliance
Research of Artificial Intelligence

E-ISSN : 2807-9035
Volume 6, **Number** 4, November 2026
https://doi.org/10.47709/brilliance.v6i4.9595

AI-Based Secure Network Learning System Development for School Computer Labs

April Rustianto[1*], Warsono [2], Tubagus Rizky Dharmawan [3], Faqih Murobbie[4], Hanif Farris Quthb[5]
[1,3,4,5]Program Studi Teknik Informatika, [2]Program Studi Sistem Informasi, Sekolah Tinggi Teknologi Terpadu Nurul Fikri, Indonesia
[1*]april.rustianto@dosen.nurulfikri.ac.id, [2]warsono@nurulfikri.ac.id, [3]tubagus.dharmawan@nurulfikri.ac.id, [4]faqi23093ti@studen.nurulfikri.ac.id, [5]hanif@student.nurulfikri.ac.id

open access

***Corresponding Author**

Article History:
Submitted: 30-08-2026
Accepted: 21-09-2026
Published: 01-11-2026

Keywords:
Centralized Monitoring; Open-Source SIEM; Large Language Model; ChatOps; Confusion Matrix; Decoupled Python Daemon.

Brilliance: Research of Artificial Intelligence is licensed under a Creative Commons Attribution-NonCommercial 4.0 International (CC BY-NC 4.0).

ABSTRACT

The manual auditing of student digital-activity logs in a school computer laboratory can take up to two hours per incident, while the complexity of raw, signature-based logs frequently induces alarm fatigue among laboratory administrators. This study proposes an autonomous, narrative-driven centralized cybersecurity monitoring system that integrates an open-source SIEM framework with a cloud-based Large Language Model (LLM), using an instant-messaging platform as a ChatOps interface for report delivery. Built on a Linux server, the system employs a decoupled Python daemon architecture that separates event-driven log ingestion from AI-based analysis through an LLM middleware layer, thereby preserving the resource stability of the core monitoring manager. It captures raw process-creation event logs (Event ID 4688) from the endpoints and transforms them into clear, contextual narrative reports. Validation was performed across 50 digital-activity scenarios spanning 25 legitimate curriculum uses and 25 policy-violating anomalies, evaluated using a Confusion Matrix. The model classified every scenario correctly, producing no false positives and no false negatives for an accuracy of 100%. A five-minute cron-based batching strategy further reduced the manual investigation window from up to 120 minutes per incident to a fully autonomous five-minute cycle, while simultaneously preventing messaging-API rate-limit blocks. These findings indicate that artificial intelligence can effectively convert raw, technical endpoint logs into a communicative instrument for objectively evaluating students' digital conduct in a vocational school environment.

INTRODUCTION

The world is currently in the midst of a massive digital transformation, in which the integration of information technology has permeated every aspect of life, including the education sector. The digitalization of education has become a logical and essential consequence for schools seeking to answer global challenges and improve the quality of learning. The implementation of technology, such as the use of integrated computer laboratories, has been proven to deliver significant efficiency in distributing materials and facilitating independent exploration. As an incubator for future generations, schools are required not only to provide technological facilities but also to equip students with digital literacy that encompasses the safe and responsible use of technology (Isma, Rahmi, & Jamin, 2022).

However, behind this efficiency, the increasing dependence on digital infrastructure also introduces exponential cybersecurity risks for every educational institution. In a shared laboratory environment, the most persistent of these risks originates from unwanted and unauthorized applications, ranging from third-party games and communication tools to unverified executables, that can compromise the integrity of lab computers, expose them to malware, and widen the institution's attack surface. Cybersecurity is therefore no longer merely a technical challenge but a fundamental requirement for maintaining data integrity and operational continuity (Ogunsanya et al., 2025). As the attack surface expands, cyber threats have become more adaptive, thereby requiring a monitoring system capable of guaranteeing the security of an organization's digital assets. Within this digital security ecosystem, log management emerges as a crucial pillar for maintaining system visibility. A log is a record of events occurring within a computing asset that functions as an audit trail for identifying and investigating anomalous activity. An organization's effectiveness in responding to threats depends heavily on its ability to aggregate and analyze logs in a timely, accurate, and relevant manner in accordance with applicable planning standards (Scarfone & Souppaya, 2023).

As a modern vocational institution, the school fully recognizes the urgency of mastering this technology in order to shape students of integrity with a global outlook. Through its guiding principles of leadership and global awareness, the school is committed to forming individuals who are honest, disciplined, and adaptive to future challenges. As a tangible manifestation of this vision, computer laboratory facilities are provided as the primary means for students to develop their technological potential and critical reasoning (School Profile and Institutional Vision, n.d.).

This is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

Brilliance
Research of Artificial Intelligence

E-ISSN : 2807-9035
Volume 6, **Number** 4, November 2026
https://doi.org/10.47709/brilliance.v6i4.9595

Nevertheless, operating the computer laboratory presents its own challenge in securing the endpoints while preserving the freedom of technological exploration that learning requires. This challenge is exacerbated by a dependence on traditional security that still relies on the operating system's built-in antivirus together with static restrictions configured on the network router. This signature-based approach has proven inadequate for dynamically monitoring endpoint behaviour and detecting emerging threats (Aboaoja et al., 2022). For instance, such security tools are able to block already-known malware but fail to identify unauthorized software that introduces genuine security exposure, such as communication applications installed outside the curriculum or third-party games and executables sideloaded through external storage media, an act that is a common vector for malware infection. Based on the laboratory's operational evaluation data, these unauthorized-application incidents consistently occur across various computer units and entirely escape the built-in security monitoring.

Beyond the weakness of the detection method, log activity management in the laboratory faces significant operational obstacles as the volume of data generated by more than 20 computer units simultaneously increases. This massive rise in data volume has surpassed human cognitive capacity, placing the limited laboratory staff at risk of becoming overwhelmed when processing such expansive log information (Nguyễn, 2022). At present, laboratory staff often become aware of a threat only after a problematic computer report emerges. Due to these limitations, the investigation process must be carried out manually by inspecting browser history or the Program Files directory from one unit to another. Manually tracing a single incident can take up to 120 minutes of audit time for an administrator. As a result of this data fragmentation and cognitive processing limitation, the response to a security incident frequently loses momentum because the digital evidence has already been erased (Bankole, Taiwo, & Claims, 2022).

This condition of data imbalance and fragmentation ultimately triggers the phenomenon of alarm fatigue, in which administrators risk ignoring genuine security threats due to the large number of unstructured alerts. Such a large and dynamic volume of data requires a smarter monitoring mechanism to overcome the limitations of conventional data processing (Ang, Ho, Huy, & Janarthanan, 2025). The need for automation becomes highly urgent so that the audit process remains effective without overburdening administrators.

In responding to this automation need, developments in cybersecurity technology have given rise to the concept of centralized monitoring based on Endpoint Detection and Response (EDR). This technology is designed to provide deep visibility into every host activity and to perform real-time log analysis in order to detect threats that evade traditional defences. The use of such a centralized monitoring platform becomes key to integrating various data sources into a single control dashboard, thereby facilitating administrators in conducting systematic digital-evidence investigation (Pratama, 2023). Comparable implementations that combine intrusion detection with centralized log visualization have similarly demonstrated the value of translating raw security alerts into an interpretable monitoring dashboard for administrators (Robbani et al., 2025). Among the various available security platforms, an open-source security framework that integrates EDR capabilities with Security Information and Event Management (SIEM) provides a robust detection mechanism for this purpose. Nevertheless, such a system still has a limitation in the form of a high number of false-positive alerts when relying solely on static rules. To overcome this, integration with intelligent technologies such as Machine Learning is required, capable of performing behavioural analysis more adaptively. Various prior studies have shown that the use of intelligent models in the detection pipeline is proven to improve accuracy up to 97.2% and significantly reduce false-positive rates through behavioural pattern analysis (Chamkar, Zaydi, Maleh, & Gherabi, 2025).

This research adopts a Large Language Model (LLM) approach through an LLM-based middleware layer as a development of conventional Machine Learning models. Unlike traditional algorithms that focus on numerical pattern classification, LLM technology possesses contextual reasoning capabilities to convert raw, technical log data into an easily understandable narrative report. Through this integrated architecture, every behavioural anomaly in the laboratory can be distinguished from normal activity with precision, producing a Summary Total Activity Log instrument that is automatically delivered through an instant-messaging notification channel. This autonomous approach not only strengthens the security infrastructure but also creates a complete audit record as material for security evaluation without burdening the administrator's cognitive capacity.

Based on the background described above, the objectives of this research are: (1) to build a centralized monitoring architecture by integrating an open-source security monitoring framework with an LLM-based middleware to automate the collection and processing of log data from every endpoint in the vocational school laboratory in real time; (2) to implement Large Language Model-based behavioural analysis to transform raw log data into a Summary Total Activity Log narrative report in order to accurately classify legitimate curriculum activities and anomalous or unauthorized-application behaviour; and (3) to measure the effectiveness and performance of the developed system in reducing manual log audit time and to assess the accuracy level of the artificial intelligence in recognizing security threats and unauthorized-application incidents within the laboratory environment.

This is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

Brilliance
Research of Artificial Intelligence

E-ISSN : 2807-9035
Volume 6, **Number** 4, November 2026
https://doi.org/10.47709/brilliance.v6i4.9595

LITERATURE REVIEW

Information Security and Log Management

Information security aims to ensure that cyberspace remains reliable and trustworthy through the protection of various digital components. Within a modern security framework, the primary objectives of information security are divided into three critical components known as the CIA Triad: Confidentiality, Integrity, and Availability.

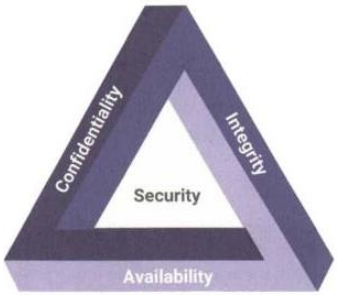


Figure 1. CIA Triad

Confidentiality refers to the protection of information from unauthorized access or disclosure. Integrity refers to safeguarding information against unauthorized modification or destruction in order to guarantee accuracy, completeness, authenticity, and non-repudiation; the integrity of digital assets can be controlled and verified through the use of logging, digital signatures, hashes, and encryption. Availability refers to the guarantee of timely and reliable access to information and resources for legitimate users (ISACA, 2021). These pillars are supported by the fundamental principles of Need-to-know and the Principle of Least Privilege (PoLP), in which users are granted only the minimum level of access required to perform their duties.

As an instrument for fulfilling the objectives of integrity and availability, Log Management plays an important role in recording every system event. According to the NIST SP 800-92 standard, effective log management includes the systematic processes of generation, transmission, storage, and analysis (Scarfone & Souppaya, 2023). Through the integration of the CIA Triad theory and centralized log management, an organization can achieve digital forensic readiness, in which the entire audit trail is instantly available for reconstructing past events to support policy evaluation in the educational environment (Bankole, Taiwo, & Claims, 2022).

SIEM and EDR as a Monitoring Platform

The increasingly sophisticated development of cyber threats requires organizations to shift from manual supervision to an integrated monitoring system in order to achieve broad visibility in identifying high-risk areas. The two core technologies used in a modern security framework are Security Information and Event Management (SIEM) and Endpoint Detection and Response (EDR).

SIEM is a comprehensive system that functions as a central platform within modern Security Operations Centers. Fundamentally, a SIEM system has the capacity to collect, aggregate, store, and correlate events generated by the managed infrastructure. The basic components of a functional SIEM architecture comprise six main integrated blocks: the Source Device, Log Collection, Parsing & Normalization, the Rule & Correlation Engine, Log Storage, and Monitoring & Visualization (González-Granadillo, González-Zarzosa, & Diaz, 2021).

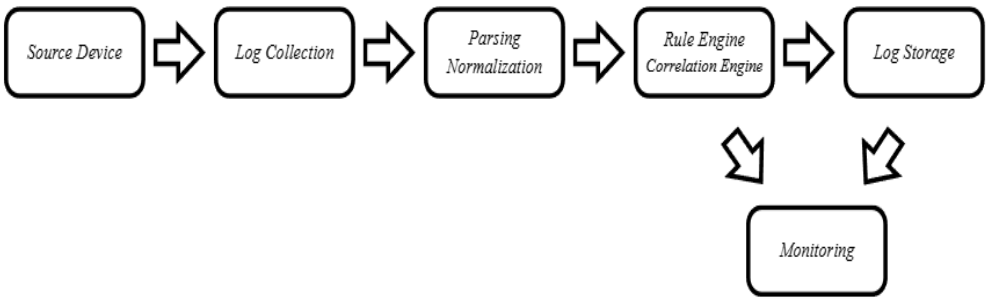


Figure 2. Basic Components of SIEM

If SIEM focuses on the broad aggregation of logs, EDR works specifically to provide deep visibility into activity at the individual host level. EDR is designed to detect threats that may evade traditional defences through continuous monitoring of system behaviour. The integration of EDR into a SIEM platform creates a synergy in which the organization benefits from compliance reporting as well as deeper and more accurate threat detection and response (González-Granadillo, González-Zarzosa, & Diaz, 2021).

This is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

Brilliance
Research of Artificial Intelligence

E-ISSN : 2807-9035
Volume 6, **Number** 4, November 2026
https://doi.org/10.47709/brilliance.v6i4.9595

Artificial Intelligence and Contextual Analysis

Conventional security detection methods that rely on static rules (rule-based) have limitations in facing dynamic and adaptive threats. To overcome the blind spots often missed by traditional signature-based monitoring tools, the integration of intelligent technology capable of understanding user behaviour in depth is required (Aboaoja et al., 2022). A Large Language Model (LLM) represents the latest evolution in the field of artificial intelligence, specifically in the branch of Natural Language Processing (NLP), trained using massive-scale text parameter datasets. Unlike conventional Machine Learning models such as clustering algorithms that focus on the statistical grouping of numbers, an LLM utilizes the Transformer architecture with a self-attention mechanism. This mechanism enables the model to examine the relationships between data variables within a sequence of log events comprehensively, thereby producing logical reasoning about the intent or purpose of a series of digital activities.

In the operational context of cybersecurity and behavioural auditing, an LLM can function as a high-level reasoning engine. The LLM does not merely read static labels or Event IDs on Windows, but performs contextual extraction from log metadata (such as process names, file directory structures, network interactions, and timestamps) to be assembled into situational understanding. Recent benchmarking research further confirms that large language models can be effectively applied to security log analysis, achieving high classification accuracy while producing interpretable, human-readable insights that surpass conventional log-processing techniques (Karlsen, Luo, Zincir-Heywood, & Heywood, 2024). This intelligent analysis process is optimized through Prompt Engineering, in which the LLM is given a structured system prompt containing the school's curriculum-policy parameters. Through this method, the system is not only able to classify anomalies in a binary manner (normal or dangerous), but is revolutionarily able to convert stacks of raw, technical log data into a representative, comprehensive, and easily interpretable narrative report format for non-technical administrative staff (González-Granadillo, González-Zarzosa, & Diaz, 2021). In this research, the artificial intelligence model specifically adopted is Google Gemini 2.5 Flash Lite, chosen for its high computational efficiency, low response latency, and massive context-window processing capability, making it ideal for handling large batch analyzes of security logs without sacrificing contextual reasoning quality.

The implementation of artificial intelligence within a SIEM log-monitoring infrastructure cannot stand entirely on its own; it requires an intermediary software (middleware) as an orchestration bridge between the local data repository (the Wazuh framework) and the Application Programming Interface (API) of the LLM engine. The role of the middleware, such as the OpenClaw ecosystem, is crucial as the provider of an automated workflow that handles the translation of the HTTP/REST communication protocol between the monitoring agent and the AI service. Technically, the middleware executes a series of data-pipeline processing functions: Data Aggregation & Retrieval, Data Parsing & Filtering, Dynamic Prompt Injection, and Response Handling & Distribution (Chamkar, Zaydi, Maleh, & Gherabi, 2025). As demonstrated in the study by Rustianto et al. (2025), the application of LLM technology to security-device logs not only significantly improves visibility but also provides far deeper contextual inference than traditional systems. High false-positive rates often cause alarm fatigue, and the use of generative technology has been proven capable of drastically reducing false-positive rates through behavioural pattern analysis that is far more complex and dynamic than static-rule policies (Rustianto, Murobbie, & Rusmanto, 2025).

METHOD

This research employs Developmental Research with an experimental and descriptive approach. The choice of this type is based on the research objective of building a new technological solution and testing its effectiveness in solving a real-world problem in the field. The development aspect involves the design and development of a new monitoring architecture that integrates the Wazuh Framework, the Decoupled Python Daemon architecture, and Google Gemini 2.5 Flash Lite LLM technology. The experimental aspect involves treatment of the system by simulating various student activity scenarios (curriculum versus anomaly) to measure the system's performance parameters, while the descriptive aspect details the system architecture, the API integration flow, and how technical log characteristics are interpreted into a communicative narrative report.

The research was conducted systematically through six main stages: (1) Problem Identification and Literature Study; (2) Data Collection and System Requirements; (3) System Design; (4) Implementation and Configuration; (5) Testing; and (6) Evaluation and Documentation. The sequential flow of these research stages is depicted in Figure 3.

This is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

Brilliance
Research of Artificial Intelligence

E-ISSN : 2807-9035
Volume 6, **Number** 4, November 2026
https://doi.org/10.47709/brilliance.v6i4.9595

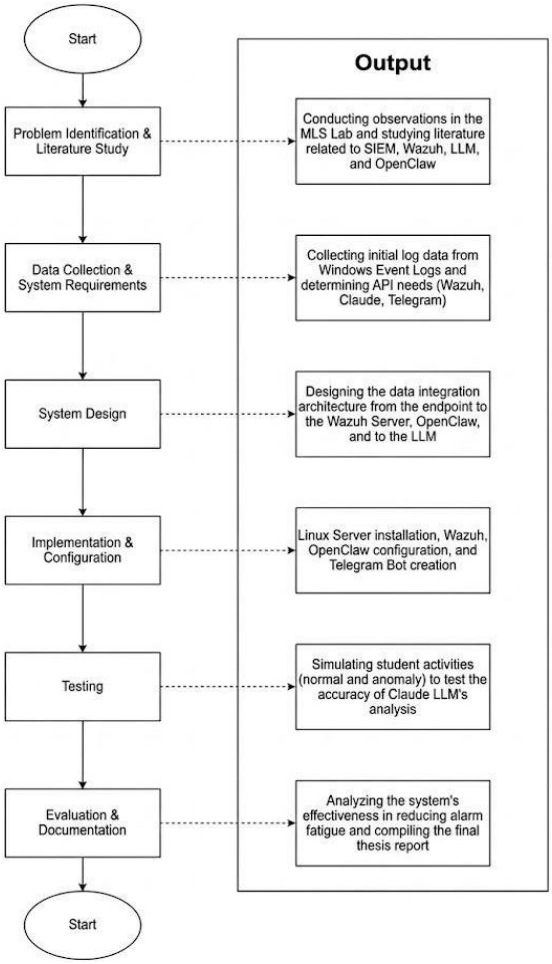


Figure 3. Research Stages

Data Collection Method

Data collection was performed through several techniques to obtain comprehensive data from both the technical and institutional-policy perspectives. Field Observation was conducted through direct observation of the computer laboratory infrastructure to understand the network topology, endpoint specifications, and student usage habits. Literature Study was carried out by collecting and analyzing academic documents, journals, and technical documentation related to SIEM, Wazuh, and LLM implementation, as well as the laboratory usage-policy document to establish a baseline of normal student behaviour. Digital Logging constituted the primary data collected in real time by the Windows operating system, taking event records from the Security, System, and Application categories via Windows Event Logs and the Wazuh Agent. Finally, API Documentation collection covered the technical structure of the Wazuh, OpenClaw, and Telegram API endpoints. A summary of the data sources and their forms is presented in Table 1.

Table 1. Summary of Data Sources and Forms

Data Source	Data Form	Usage
Student Computer Unit (Windows)	Raw Event Logs (.evtx / JSON)	Raw student-activity data for analysis
School Lab Policy	Text Document / Rules	Basis for determining normal versus anomalous activity
Wazuh & OpenClaw API	Data Stream (JSON)	Medium for integrating log data toward the AI
Journals & Reference Books	Academic Text	Theoretical foundation and methodological references

Data Analysis Method

The data analysis method used is a combination of qualitative and descriptive analysis. Because the system involves a Large Language Model (LLM), the data-analysis process does not focus solely on numerical statistics but on understanding the context of behaviour (González-Granadillo, González-Zarzosa, & Diaz, 2021). The analysis steps

This is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

Brilliance
Research of Artificial Intelligence

E-ISSN : 2807-9035
Volume 6, **Number** 4, November 2026
https://doi.org/10.47709/brilliance.v6i4.9595

consist of: Data Reduction, in which the raw log data drawn from the Wazuh API through OpenClaw is filtered to remove irrelevant system-log lines so that the prompt sent to the Gemini LLM remains efficient and focused on curriculum and anomaly behavioural indicators; Contextual Analysis (AI Reasoning), in which the reduced log data is analyzed by the Gemini LLM using Prompt Engineering to compare the students' actual activities against the normal behavioural profile defined by school policy; Descriptive Categorization, in which the analysis results are categorized into Valid Activity (Curriculum) and Anomalous Activity (Violation); and Result Interpretation, in which the categorization results are transformed into a concise narrative report to be sent via the Telegram Bot and processed into a Summary Total Activity Log.

Proposed System Architecture

The implementation in this research uses a Python script to separate the workload between log ingestion by the SIEM and artificial-intelligence analysis. This separation is crucial to ensure that the core resources of the Wazuh Manager do not experience a bottleneck due to blocking I/O when making API calls to the Large Language Model. Operationally, this system ecosystem runs side by side within a Linux Ubuntu 22.04 LTS server. The Decoupled Python Daemon architecture workflow, from the client endpoint side through to the distribution of the visual ChatOps report, is illustrated in Figure 4.

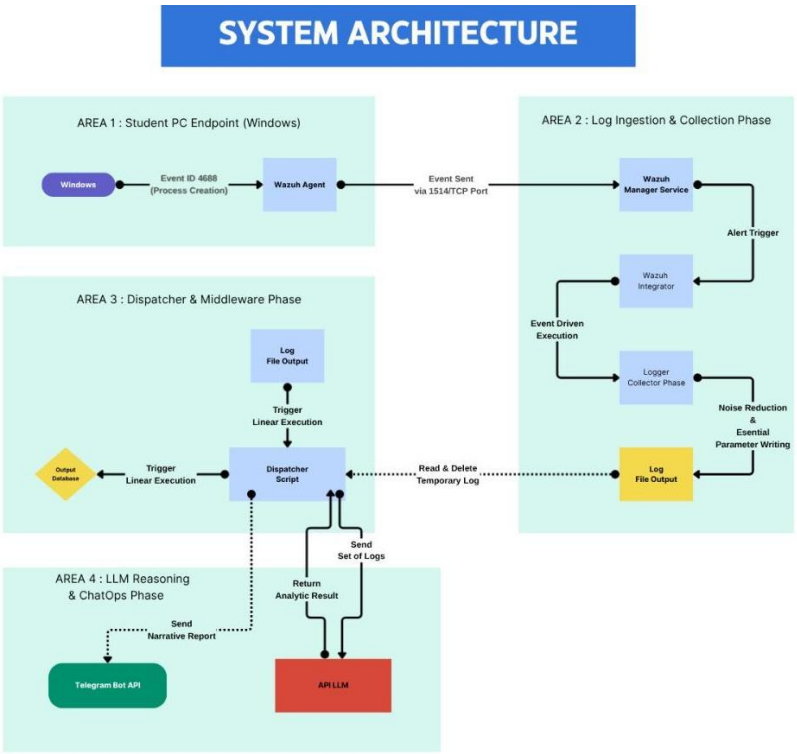


Figure 4. Centralized Topology Data-Flow Architecture

Based on the architecture in Figure 4, the data-processing flow is divided into four interlinked operational areas. Area 1 (Endpoint) focuses on capturing student activity at the Windows operating-system level through a lightweight Wazuh Agent, where every Event ID 4688 (Process Creation) call is streamed directly to port 1514/TCP of the central management. Area 2 (Log Ingestion & Collector Phase) leverages the internal Wazuh Integrator module to trigger the 'custom-process-logger.py' script in an event-driven manner; this area is limited to filtering high-value technical data and moving it into a temporary text database at '/tmp/wazuh_process_memory.txt'. Area 3 (Dispatcher & Middleware Phase) uses a scheduled Cron Job with a 5-minute interval; the 'wazuh_dispatcher.py' script reads the memory batch, empties the raw file to prevent data accumulation, and forwards the communication payload to local port 18789/TCP of the OpenClaw middleware. Finally, Area 4 (Cloud AI Reasoning & ChatOps Output) is the estuary of the intelligence report: OpenClaw authenticates the data to the LLM Cloud API to obtain contextual inference, then the response is converted into a local archive summary ('ai_analysis_database.txt') and pushed out through the Telegram Bot API webhook as the final, informative notification for the laboratory supervisor.

Implementation and Configuration

The main infrastructure was built on top of the Ubuntu 22.04 LTS operating system using Wazuh version 4.14.5. To minimize configuration errors (human error) in the distributed architecture, the installation was carried out using the

This is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

Brilliance
Research of Artificial Intelligence

E-ISSN : 2807-9035
Volume 6, **Number** 4, November 2026
https://doi.org/10.47709/brilliance.v6i4.9595

Assisted Installation Method provided by the official Wazuh documentation. The initialization stage began by downloading the 'wazuh-install.sh' installation script and the 'config.yml' configuration file using the curl command in the Ubuntu terminal. The 'config.yml' file was then adjusted to define the IP address mapping of the local server (127.0.0.1) for all nodes (Indexer, Server, and Dashboard). The installation script was executed with the '-a' (all-in-one) parameter, which automates certificate generation and installs and configures the Wazuh Indexer, the Wazuh Manager, and the Wazuh Dashboard sequentially in a single execution. After installation, the random credentials were extracted from the archive file to log in to the dashboard interface, and the daemon status was checked via 'systemctl status' to ensure the wazuh-indexer, wazuh-manager, and wazuh-dashboard services were running properly.

To connect the local server with the LLM API, the OpenClaw middleware was installed natively on the Ubuntu 22.04 LTS environment via the official script. After the software was installed, the configuration was adjusted to insert the API Key, and OpenClaw was set to listen on the local interface at port 18789/TCP. This ensures that the entire artificial-intelligence authentication load is handled centrally by the middleware layer, thereby protecting the security of the API key so that it is not exposed directly within the Python integration script.

The first integration phase, the Collector, was run using a Python script placed in the '/var/ossec/integrations/' directory. This 'custom-process-logger' script does not run continuously; rather, it is called and executed automatically by the Wazuh Integrator module every time the system detects a new security alert from an endpoint. The main feature of this script is a validation (filtering) mechanism on the log's rule-description attribute, ensuring that the script only processes Event ID 4688 (Process Creation). This mechanism was designed to be highly flexible so that the administrator can easily modify or add other description strings according to future alert-filtering needs. Once validated, the script extracts the agent and timestamp dynamically, stores them to the memory file '/tmp/wazuh_process_memory.txt', and sends a debug message to Telegram.

Unlike the event-driven Collector, the second phase, the Dispatcher, is executed periodically using the Linux Cron Job scheduling facility. This approach ensures that data is sent to the artificial-intelligence API in batches to avoid the request rate limit. The 'wazuh_dispatcher.py' script is scheduled to run at a fixed interval; it reads the raw log, empties the memory file instantly, and assembles the request toward the LLM API endpoint. Credential security (the API Key and Telegram Token) is implemented through environment-variable (.env) calls. To ensure the AI returns only a structured JSON array structure, the responseMimeType parameter is configured to application/json.

RESULT

The testing stage is a crucial phase for validating whether the Decoupled Python Daemon architecture and the Gemini 2.5 Flash Lite Large Language Model integration operate in accordance with the expected specifications. This testing was divided into a data-flow functionality test (Black Box Testing) and an artificial-intelligence classification accuracy test (Confusion Matrix).

Prototype and System Interface

As a form of operational monitoring, the final output of this Security Operations Center (SOC) integration is not routed back into the default Wazuh visual dashboard, but is instead presented through a ChatOps method using the Telegram communication platform. The adoption of instant-messaging platforms such as Telegram as a real-time security-notification channel has likewise proven effective in prior automated attack-detection systems (Tedyyana, Ghazali, & Purbo, 2024). Based on the notification interface, it can be observed that the raw log data, which initially consisted only of Event ID records and technical directory paths from the Windows operating system, was successfully transformed. The alert format is designed with clear parameters covering the computer endpoint name, the timestamp of the incident, the directory path of the executed file, and, most essentially, the AI Analysis. The AI LLM engine narratively provides a contextual interpretation of the running process, so that a non-technical administrator can directly identify whether the launched software is safe, part of the curriculum, or a security-alert anomaly, without having to perform threat-intelligence searches manually.

This is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

Brilliance
Research of Artificial Intelligence

E-ISSN : 2807-9035
Volume 6, **Number** 4, November 2026
https://doi.org/10.47709/brilliance.v6i4.9595



Figure 5. AI Analysis Notification Interface via Telegram

Data-Flow Functionality Testing (Black Box Testing)

This test aims to ensure there is no bottleneck or integration fault from the moment data is created at the endpoint until it is extracted and reported by the server. The test began with a direct interaction simulation by the user on the Windows-based endpoint, covering the execution of educational (curriculum) programs as well as attempts to open third-party software directories (such as games) irrelevant to the learning process. Every execution automatically triggered the Wazuh agent embedded in the background of the operating system. The Wazuh Manager successfully captured the OS-level activity and identified the rule "A process was created" with Event ID 4688, which is the main parameter required by the system to perform the digital-behaviour audit.

After the log was received by Wazuh, the Collector phase (custom-process-logger) ran successfully. The Python script automatically filtered the large JSON log file, removed noise data, and retained only the agentName, time, and processName entities (such as cmd.exe or conhost.exe). This reduced raw data was then sent by the Dispatcher phase toward the LLM API interface to be analyzed and forwarded into a Telegram report. The final stage of this functionality test validated the successful distribution of the report; the Telegram Bot API endpoint was successfully accessed by the Dispatcher-phase script without any obstruction (timeout or network fault). Furthermore, the LLM API successfully returned a valid, pure JSON array response, so the Python script was able to parse the data and arrange it into neatly formatted chat bubbles. The AI was observed to accurately detect the execution of anomalous directories (such as Roblox, FiveM/GTA V, Rockstar Games Launcher, and Epic Games) and to provide contextual reasoning that those applications are risky and irrelevant to the educational environment. This success confirms that the ChatOps integration for the Security Operations Center (SOC) runs flawlessly.

Artificial-Intelligence Accuracy Testing (Confusion Matrix)

This test aims to measure the reliability level of the LLM AI model in classifying student-behaviour logs. The testing was conducted by running 50 activity scenarios consisting of 25 curriculum-application simulations (such as VS Code and Microsoft Word) and 25 non-curriculum/anomaly-application simulations (such as games, Discord, and system-modification applications). Table 2 presents the testing distribution, while Table 3 maps the AI analysis results into the Confusion Matrix.

This is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

Brilliance
Research of Artificial Intelligence

E-ISSN : 2807-9035
Volume 6, **Number** 4, November 2026
https://doi.org/10.47709/brilliance.v6i4.9595

Table 2. Recapitulation of Activity Scenario Testing Data

Activity Category	Scenario Description	Number of Trials
Normal (Curriculum)	Execution of productivity software, programming IDEs, and educational browser access.	25
Anomaly (Violation)	Execution of third-party game applications, external communication clients (Discord), and non-standard script execution (unauthorized cmd/powershell).	25
Total Testing		50

The scenario dataset in Table 2 was constructed with a balanced distribution of 50 test cases, divided equally into 25 normal curriculum activities and 25 behavioral anomalies. The legitimate curriculum activities reflect the vocational school's daily learning needs, encompassing coding workflows in Visual Studio Code, document processing in Microsoft Office suite, standard administrative utilities, and educational web browsing. Conversely, the anomaly scenarios simulate realistic student infractions that frequently bypass signature-based OS controls, including background execution of gaming clients (such as Roblox, Steam, and Epic Games), third-party communication platforms (Discord), peer-to-peer sharing tools, and unauthorized command-line invocations aimed at environment probing. This 50:50 ratio ensures an unbiased baseline to rigorously evaluate both the model's selectivity and its sensitivity under real laboratory conditions. Following the execution of these scenarios, the AI inference results were mapped into the binary confusion matrix shown in Table 3.

Table 3. Classification Matrix (Confusion Matrix)

Parameter	Definition	Count
True Positive (TP)	The system analyzes an activity as a Violation, and the actual activity is indeed a violation.	25
True Negative (TN)	The system analyzes an activity as Normal, and the actual activity indeed conforms to the curriculum.	25
False Positive (FP)	The system analyzes an activity as a Violation, whereas the actual activity is normal/curriculum (False Alarm).	0
False Negative (FN)	The system analyzes an activity as Normal, whereas the actual activity is a violation (Missed Detection).	0

As detailed in Table 3, the contextual reasoning engine powered by Google Gemini 2.5 Flash Lite correctly categorized all 50 recorded digital activity events. The system achieved 25 True Positives (TP), successfully recognizing every non-curriculum application and suspicious script invocation as an anomaly with clear contextual explanations. Simultaneously, it registered 25 True Negatives (TN), accurately validating legitimate educational tools and system processes without mislabeling normal student tasks. Crucially, the evaluation yielded 0 False Positives (FP) and 0 False Negatives (FN). The absence of false alarms (FP = 0) directly confirms the elimination of administrator alarm fatigue, ensuring supervisors are not distracted by benign educational workflows. Furthermore, achieving zero missed detections (FN = 0) proves that no unauthorized software evaded AI detection, thereby preventing unmonitored attack vectors in the shared lab infrastructure. Based on these empirical values, the overall classification performance was computed using the standard accuracy metric formula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100\%$$

$$Accuracy = \frac{25 + 25}{50} \times 100\% = 100\%$$

This agent- and LLM-based centralized monitoring system was proven capable of classifying students' digital behaviour with a high level of accuracy. This achievement validates that the system is free from the phenomenon of alarm fatigue and is highly suitable for use as an objective instrument for disciplinary evaluation at the computer laboratory.

DISCUSSION

The evaluation was carried out in depth to measure the extent to which the implementation of the Decoupled Python Daemon architecture, powered by artificial intelligence, succeeded in answering the research problem. The evaluation focused on two main metrics: operational time efficiency and the effectiveness of alarm-fatigue reduction.

This is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

Brilliance
Research of Artificial Intelligence

E-ISSN : 2807-9035
Volume 6, **Number** 4, November 2026
https://doi.org/10.47709/brilliance.v6i4.9595

Audit Time Efficiency (Time Efficiency & Latency)

Before this system was implemented, the manual security-incident investigation process (from tracing Windows logs to determining whether an application violated the curriculum policy) could take up to 120 minutes per incident. This lengthy time was caused by the scattered nature of the raw logs, which required manual inspection by the administrator on each endpoint. Through the implementation of the autonomous Dispatcher phase, the processing and reporting latency was specifically set at a fixed 5-minute interval via the Linux Cron Job scheduling facility. This 5-minute cycle is not a delay constraint but a deliberately designed batching strategy based on two main operational considerations. The first is compliance with external limitations (Rate Limiting): webhook platforms such as the Telegram Bot API have strict limits on the number of messages that can be sent per second; if the system were set to send messages instantly (per-event basis) while dozens of student computers open applications simultaneously, it would cause API blocking (API Timeout/Block). The second is spam prevention: the 5-minute interval ensures that the administrator receives a single, periodic intelligence-summary report (batch) instead of being bombarded by hundreds of incoming messages every second, which would otherwise destroy concentration. With this autonomous approach, the time required to complete one student-behaviour audit cycle dropped drastically from 120 minutes (manual) to a 5-minute autonomous inspection cycle.

Alarm-Fatigue Reduction and Classification Quality

Alarm fatigue is a phenomenon in which administrators ignore security warnings because of an overwhelming number of false alerts (False Positives) or technical logs whose meaning cannot be immediately understood. Signature-based built-in systems often silence an alarm simply because an '.exe' file runs in the background, without understanding the context of whether the file is part of the educational curriculum or not. The evaluation of this new method shows highly satisfying results, referring to the high accuracy achieved in the Confusion Matrix (Table 5), in which the system recorded 0 False Positives. This alarm-fatigue reduction was achieved through a combination of two protective layers (guardrails). The first is the Python Collector layer, which precisely eliminates all noise from Wazuh and forwards only valid Event ID 4688 (program-execution) records. The second is the AI Reasoning layer, in which the LLM successfully provided artificial "cognition" to the system; for example, when an endpoint executes discord.exe or a game launcher, the AI autonomously dissects that metadata and generates a warning description that the application is an anomaly in the learning environment rather than part of the curriculum. Overall, this research concludes that the integration architecture of the Wazuh agent, OpenClaw middleware, and the LLM with the Gemini 2.5 Flash Lite model succeeded in transforming security logs (Security Logs) that were originally very rigid and technical into a narrative, accurate, and highly user-friendly Digital Behaviour Audit Instrument for Computer Laboratory through the interactive Telegram medium.

CONCLUSION

Based on the results of the design, implementation, testing, and evaluation that have been carried out, this research on an artificial-intelligence-based centralized monitoring system produces conclusions that directly answer the research problem. The integration of a centralized monitoring system was successfully realized through the implementation of a Decoupled Python Daemon architecture that collaborates the Wazuh SIEM Framework, a Large Language Model (LLM), and the Telegram Bot API. This architecture successfully separated the workload into two independent phases, namely the event-driven Collector phase and the cron-based Dispatcher phase. This autonomous approach enabled the system to extract Event ID 4688 logs in real time without burdening the core resources of the Wazuh Manager.

In answering the effectiveness of the contextual-analysis implementation, the use of Generative AI proved far superior to traditional rule-based detection systems in composing the Summary Total Activity Log report. Performance evaluation using the classification matrix (Confusion Matrix) against 50 activity scenarios in the laboratory proved that the LLM model achieved a very high accuracy level. The system was able to distinguish and classify the use of educational-curriculum software from anomalous applications with precision, without producing a single false alert (False Positive). Furthermore, the evaluation of operational efficiency showed that the ChatOps-based warning system via Telegram succeeded in significantly reducing the alarm-fatigue phenomenon. Through the implementation of a batching strategy with a 5-minute latency, the system was not only effective in avoiding the API rate limit but also succeeded in drastically cutting the manual audit time that had previously taken up to 120 minutes per incident. The resulting narrative report proved to be a disciplinary-evaluation instrument that greatly eases the administrator in making decisions in real time.

Although the system prototype has operated functionally, there is room for optimization for future research. It is recommended to simplify the architecture by bypassing the OpenClaw middleware layer so that the entire process—from log-memory extraction, essential-parameter filtering, and direct instruction injection to the REST API LLM endpoint, through to processing the JSON response toward the Telegram webhook—can be unified and managed natively using a complete Python script to reduce server infrastructure overhead. In addition, future research is advised to integrate more comprehensive Windows log sources, such as Event ID 5156 (Network Connection) or File Integrity

This is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

E-ISSN : 2807-9035
Volume 6, **Number** 4, November 2026
https://doi.org/10.47709/brilliance.v6i4.9595

Monitoring (FIM), to provide far richer context for the AI, and to evaluate the use of small-scale open-source LLM models, such as LLaMA 3 (8B) or Mistral, hosted locally (on-premise) within the institution's server to guarantee absolute data sovereignty.

ACKNOWLEDGMENT

The author gratefully acknowledges the financial support provided by the Ministry of Higher Education, Science, and Technology of the Republic of Indonesia through the Beginner Lecturer Research (PDP) grant program, which made this research possible.

REFERENCES

Aboaoja, F. A., Zainal, A., Ghaleb, F. A., Al-rimy, B. A. S., Eisa, T. A. E., & Elnour, A. A. H. (2022). Malware detection issues, challenges, and future directions: A survey. *Applied Sciences, 12*(17), 8482. https://doi.org/10.3390/app12178482

Ang, S., Ho, M., Huy, S., & Janarthanan, M. (2025). Utilizing IDS and IPS to improve cybersecurity monitoring process. *Journal of Cyber Security and Risk Auditing, 2025*(3), 77–88. https://doi.org/10.63180/jcsra.thestap.2025.3.6

Bankole, F., Taiwo, A., & Claims, I. (2022). An extended digital forensic readiness and maturity model. *Forensic Science International: Digital Investigation, 40*, 301348. https://doi.org/10.1016/j.fsidi.2022.301348

Chamkar, S. A., Zaydi, M., Maleh, Y., & Gherabi, N. (2025). Improving threat detection in Wazuh using machine learning techniques. *Journal of Cybersecurity and Privacy, 5*(2). https://doi.org/10.3390/jcp5020034

González-Granadillo, G., González-Zarzosa, S., & Diaz, R. (2021). Security information and event management (SIEM): Analysis, trends, and usage in critical infrastructures. *Sensors, 21*(14). https://doi.org/10.3390/s21144759

ISACA. (2021). *Cybersecurity fundamentals study guide*. ISACA.

Isma, C. N., Rahmi, R., & Jamin, H. (2022). The urgency of digitalization of school education. *At-Ta'dib: Scientific Journal of Islamic Religious Education Study Program, 14*(2), 129–141.

Karlsen, E., Luo, X., Zincir-Heywood, N., & Heywood, M. (2024). Benchmarking large language models for log analysis, security, and interpretation. *Journal of Network and Systems Management, 32*(3), 59. https://doi.org/10.1007/s10922-024-09831-x

Nguyễn, T. (2022). Cybersecurity logging & monitoring security program. *School of Computer Science and Engineering*, 1–8.

Ogunsanya, V. A., Adesokan, A., Eleweke, I., Obu, A. U., Afolabi, R., & Abbas, R. (2025). Cybersecurity incidents on digital infrastructure and industrial networks. *Journal of Computational Analysis and Applications, 34*(3), 85–106.

Pratama, N. F. (2023). Perancangan sistem deteksi dini keamanan informasi DISKOMINFO Kabupaten Bandung. *Jurnal Teknik Informatika dan Sistem Informasi, 10*(1), 808–820.

Robbani, F. D., Haryatmi, E., Riyadi, T. A., Supono, R. A., Kurniawan, A. B., & Rosdiana. (2025). Implementation of an intrusion detection system using Snort and log visualization using ELK Stack. *International Journal of Engineering, Science and Information Technology, 5*(3), 220–228. https://doi.org/10.52088/ijesty.v5i3.901

Rustianto, A., Murobbie, F., & Rusmanto. (2025). Predictive maintenance on Fortinet firewall devices using artificial intelligence. *Brilliance: Research of Artificial Intelligence, 5*(2), 1215–1223. https://doi.org/10.47709/brilliance.v5i2.7448

Scarfone, K., & Souppaya, M. (2023). *Cybersecurity log management planning guide* (Vol. 1, No. 1). National Institute of Standards and Technology.

Tedyyana, A., Ghazali, O., & Purbo, O. W. (2024). Machine learning for network defense: Automated DDoS detection with Telegram notification integration. *Indonesian Journal of Electrical Engineering and Computer Science, 34*(2), 1102–1109. https://doi.org/10.11591/ijeecs.v34.i2.pp1102-1109

This is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.