International Journal of Informatics and Computation (IJICOM)
Vol. 8, No. 2, 2026
ISSN: 2685-8711, E-ISSN: 2714-5263 DOI: 10.35842/ijicom

Hermes-Based Agentic Security Response System for Log Anomaly Detection and Limited Mitigation

Surya Tri Atmaja Ramadhani[1], Fiyas Mahananing Puri[2], Vikky Aprelia Windarni[3], Dewi Anisa Istiqomah[4], Efrat Tegris[5]

Abstract
Ubuntu Server security depends on timely interpretation of authentication, firewall, and system logs, particularly in the Generative AI era, where security operations increasingly need contextual analysis, triage support, and controlled response recommendations. The problem addressed in this study is that manual log analysis and pattern-based tools can identify explicit indicators, but they provide limited incident reasoning, severity assessment, and playbook-validatable mitigation recommendation. This study proposes and evaluates a Hermes-based agentic security response system as an analysis and response-recommendation support layer, not as a replacement for SIEM, IDS, Fail2Ban, or rule-based detection. The proposed method consists of log collection, preprocessing into structured event groups, Hermes-based classification and contextual reasoning, threat-decision mapping, and playbook-based mitigation validation. The evaluation used a controlled proof-of-concept dataset generated from local simulations, consisting of 404 log records and eight scenario groups. Under this controlled setting, both the rule-based baseline and Hermes achieved accuracy, precision, recall, and F1-score of 1.00, while Hermes also achieved attack type accuracy, severity accuracy, mitigation action accuracy, and reasoning validity rate of 1.00. These findings show that the workflow behaved consistently when it was applied to explicit, well-bounded simulated scenarios. The value of the study therefore lies not in claiming better detection accuracy, but in showing how automated severity triage, attack-type interpretation, contextual reasoning, auditability, and safe limited response recommendation can complement deterministic rule-based approaches.

Keywords:
Hermes, Log Anomaly Detection, Ubuntu Server, Security, Mitigation

1. Introduction

Ubuntu Server is often chosen to host web services, databases, internal tools, teaching labs, and other organizational systems. Many small and medium-sized cloud services also rely on it. Since Ubuntu Server usually acts as a main point for network services, it faces risks like SSH brute-force attacks, unauthorized logins, port scans, suspicious root access, and privilege escalation attempts [1], [2]. This makes it important to pay attention to logs such as auth.log, syslog, ufw.log, and journalctl. These logs provide technical evidence by recording authentication events, firewall activity, system processes, and user actions.

Reading these logs manually becomes difficult when the volume is high, service formats vary, and administrators must respond quickly. Fail2Ban and deterministic rules can flag or block events that match known patterns, yet their behaviour depends on predefined filters and usually stops at detection rather than producing incident-oriented reasoning [3]–[5]. In the current Generative AI context, this situation presents a practical opportunity to

Corresponding Author: Surya Tri Atmaja Ramadhani (surya@amikom.ac.id)
1 Surya Tri Atmaja Ramadhani, Universitas Amikom Yogyakarta, surya@amikom.ac.id
2 Fiyas Mahananing Puri, Universitas Amikom Yogyakarta, fiyas@amikom.ac.id
3 Vikky Aprelia Windarni, Universitas Amikom Yogyakarta, vikkyaprelia@amikom.ac.id
4 Dewi Anisa Istiqomah, Universitas Amikom Yogyakarta, dewianisaist@amikom.ac.id
5 Efrat Tegris, Universitas Amikom Yogyakarta, tigris78@amikom.ac.id

extend log handling beyond pattern matching toward structured interpretation, severity triage, and response recommendations, while still enforcing safety boundaries and keeping the output auditable [6]–[18]. Hermes was chosen for this study since the problem goes beyond just deciding if something is normal or an anomaly. It also involves understanding groups of events, explaining what kind of attack is happening, deciding how serious it is, and suggesting only a few actions from a set playbook. Using AI Agent technology can help administrators with these tasks by reading event groups, judging the severity of threats, figuring out attack types, and recommending a limited set of responses. In this research, Hermes is meant to support analysis and response recommendations, not to replace tools like SIEM, IDS, Fail2Ban, or rule-based detection.

The research problem is how to design a Hermes-based system that can support Ubuntu Server log analysis, interpret security anomalies, perform severity triage, and provide safe limited mitigation recommendations. The objectives are to design the system architecture, build a structured log dataset, evaluate Hermes against a rule-based baseline and Fail2Ban, and assess the safety of recommended actions using a mitigation validator [10]–[24]. The main contributions are a lightweight agentic security response architecture that positions Hermes as an analysis and response-recommendation support layer; integration of classification, attack_type, severity, reason, and recommended_action outputs; a playbook-based mitigation validator; and an auditability-oriented evaluation using reasoning validity, severity accuracy, mitigation action accuracy, unsafe action rate, and rejected action rate.

This article does not frame its contribution as a claim that Hermes improves detection accuracy by itself, because the explicit anomaly patterns in the controlled dataset are also captured by the rule-based baseline [11], [12], [21], [25]–[27]. The stronger contribution is the placement of Hermes after log structuring as a triage-oriented layer that reads evidence, assigns severity, interprets attack_type, and produces recommendations that can be checked against a playbook. Hermes is used here to help with automated severity triage, contextual reasoning, attack-type interpretation, and making mitigation recommendations that follow playbooks. It also supports auditability and gives safe, limited-response suggestions, working together with deterministic rules [6]–[10], [13], [15]–[24]..

This article makes five primary contributions within a controlled Generative AI-assisted cybersecurity framework. First, it applies Hermes as a constrained analysis and response-recommendation layer for Ubuntu Server log events rather than granting autonomous execution capabilities. Second, it integrates event classification, attack interpretation, severity assessment, reasoning, mitigation recommendations, and human-review requirements into a unified and auditable decision output. Third, it combines agentic reasoning with a playbook-based mitigation validator that permits only predefined safe response categories while rejecting unsafe or destructive actions. Fourth, it evaluates the framework beyond detection accuracy by measuring reasoning validity, severity classification accuracy, mitigation recommendation accuracy, unsafe action rate, and rejected action rate. Finally, it validates the proposed approach through a controlled proof-of-concept against a deterministic rule-based baseline and Fail2Ban, demonstrating that Hermes complements existing security tools by enhancing analysis and decision support rather than replacing established defensive mechanisms.

2. Related Works

Log-based anomaly detection is still a major focus in research since logs record what actually happens during system operation and can help identify failures or signs of attacks [3]–[5], [11], [21]. However, there are still a lot of challenges in this area. Some of the main issues are the huge amount of data, the fact that logs come in many different formats, and the need to rely on parsers. Other problems include how to represent features, imbalanced datasets, and the difficulty of getting good results across different environments. Many approaches that use parsers argue that it is important to first turn raw logs into structured data, like timestamps, source IPs, usernames, services, event types, or event windows, before trying to detect anomalies or make sense of the data [5].

Traditional machine learning methods like Support Vector Machine (SVM), Naive Bayes, and decision trees are still common in intrusion and anomaly detection. They are popular because they use clear steps for classification and can show measurable results when the features are structured [11], [12], [25]–[27]. SVM works well if you can turn normal and abnormal behaviour into labelled feature vectors that can be separated by a boundary. On the other hand, Naive Bayes is a simpler probabilistic method. It estimates how likely a class is based on the features and is often effective with text-based or categorical security data. Even though these methods are still useful, they usually need well-labelled data, separate sets for training and testing, careful feature selection, and large benchmark datasets. In this study, the Hermes workflow is used differently. It comes into play after the logs have been structured, acting as a focused layer for reasoning, triage, and suggesting mitigation steps. This means it adds to what SVM or Bayesian classifiers do, instead of repeating their functions.

Rule-based approaches such as Fail2Ban remain relevant because they are simple, deterministic, and auditable. However, they are limited to explicit patterns and provide limited contextual reasoning across events. In contrast, LLMs and AI Agents are increasingly explored in cybersecurity for log summarization, alert triage, threat intelligence, incident response, and security automation [6]–[10], [13], [15]–[18], [19], [20], [23]. These opportunities must be balanced with risks such as prompt injection, unsafe command execution, and over-permissioned agents, which justify the use of allowed actions, forbidden actions, mitigation playbooks, and audit logging [17], [18], [19], [20], [22], [24].

The research gap addressed here is the limited evaluation of Hermes as an AI Agent for Ubuntu Server log analysis and limited mitigation recommendation. This study does not introduce a new SVM, Naive Bayes, deep learning, or ensemble classifier. Instead, Hermes is treated as a reasoning and triage module that works on structured log data and is constrained by a mitigation validator with whitelist, blacklist, human review, and audit logging mechanisms. This position is important since the study is not just about improving binary detection accuracy. Instead, it looks at whether an agentic layer can actually use clear log evidence to assess how severe an incident is, figure out the type of attack, and suggest safe ways to respond.

3. Proposed Method

The proposed method follows a design-and-evaluation experimental approach. The system includes the Ubuntu Server Environment, Log Collector, Log Preprocessing Module, Hermes Agent Analysis Module, Threat Decision Module, Mitigation Validator and Playbook Module, Limited Mitigation Execution, and Audit and Reporting Module. Hermes receives a scenario-level event summary and returns classification, attack_type, severity, reason, recommended_action, and requires_human_review.

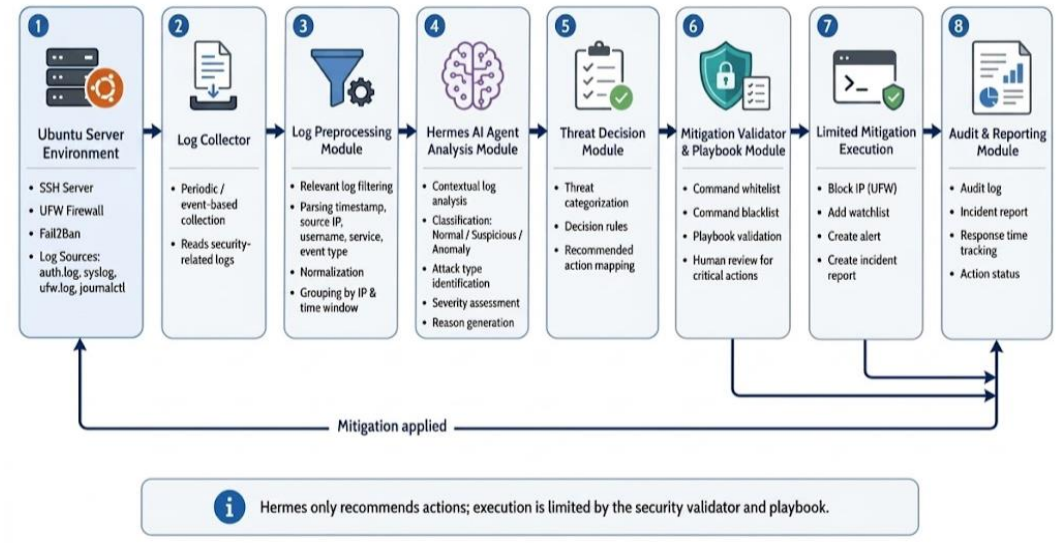

Fig. 1. Architecture of the Hermes-based agentic security response system

Preprocessing converts raw logs into a structured dataset. The main fields are timestamp, source_ip, destination_ip, username, service, event_type, raw_log, failed_attempts, time_window, ground_truth_label, attack_type, severity, and mitigation_expected. Ground truth is taken from the controlled scenario executed by the researcher, not from Hermes or from the baselines. In this way, ground truth serves as the evaluation key for comparing predictions across methods.

The mitigation playbook limits Hermes to the actions no_action, add_watchlist, block_ip, create_alert, create_incident_report, and human_review. Actions such as delete_user, stop_ssh, reboot_server, flush_iptables, and modify_system_permission are treated as unsafe and are not permitted. In this experiment, mitigation validation was carried out at the action-category level and did not involve direct shell execution on the target server.

3.1 Prompt Engineering & LLM Configuration

Hermes was tested as an agentic log-analysis module on a separate analysis machine, not directly on the Ubuntu Server target. The input was prepared as a scenario-level event-group summary rather than full raw logs, so the model could focus on the relevant security evidence. Ground-truth labels were intentionally omitted from the prompt to avoid evaluation leakage.

The prompt was designed to keep Hermes within an analysis and response-recommendation role, rather than allowing it to behave as an unrestricted system-command executor. It contained four parts: a system role, task instruction, decision policy, and output schema. The system role described Hermes as a cybersecurity log analysis agent; the task instruction asked it to classify events as Normal, Suspicious, or Anomaly, identify attack_type, assign severity, and choose exactly one action from the allowed list; the decision policy described signals such as brute-force behavior, invalid-user evidence, root-login attempts, port-scan evidence, and sudo authentication failures; and the output schema required JSON so that the mitigation validator could process the result. The input unit was a scenario-level event-group summary with relevant sample raw logs. Ground-truth labels, original severity, and expected mitigation were excluded from the prompt to prevent leakage. In this experiment, Hermes Agent v0.14.0 (2026.5.16) was used with

GPT-5.5 (gpt-5.5) through the remote openai-codex provider. Because inference was performed through a remote LLM provider, temperature, maximum tokens, quantization, and serving details were not fully controlled by the researchers and are reported as configuration limitations. This experiment does not claim local inference; Hermes was operated as an analysis and response-recommendation support layer on a separate analysis machine from the Ubuntu Server target.

Table 1. Prompt and LLM configuration used in Hermes evaluation

Parameter	Configuration
Evaluation mode	Offline scenario-level event-group analysis as a controlled proof of concept.
Input unit	Scenario-level event-group summary with relevant sample raw logs; ground truth was not provided in the prompt.
Prompt structure	System role, task instruction, decision policy, and output schema.
Output schema	JSON-only: classification, attack_type, severity, reason, recommended_action, requires_human_review.
Allowed actions	no_action, add_watchlist, block_ip, create_alert, create_incident_report, human_review.
Forbidden actions	Direct shell-command generation and destructive actions such as delete_user, stop_ssh, reboot_server, flush_iptables, and modify_system_permission.
Model and provider	Hermes Agent v0.14.0 (2026.5.16) using GPT-5.5 (gpt-5.5) through the remote openai-codex provider.
Deployment mode	Hermes was executed through CLI/Gateway on a Linux analysis VM/machine separate from the Ubuntu Server target; no direct shell execution on the target server was performed in this evaluation.
Parameter control	Temperature and maximum tokens were not explicitly configured and therefore followed provider/model defaults; context window followed Hermes fallback metadata of approximately 272,000 tokens.
Quantization and serving	Quantization and serving details were not controlled or exposed because inference used a remote LLM provider; this experiment does not claim local inference through Ollama, LM Studio, vLLM, or llama.cpp.

3.2 Mathematical Formulation of the Proposed System

To clarify the proposed mechanism, the Hermes-based response workflow is written as a constrained inference and validation process. The notation below keeps the agentic component inside a bounded decision space, so the model is not treated as an unconstrained command generator.

Let E denote the set of collected log events. The event collection is defined in (M1).

$$E = \{e_1, e_2, \dots, e_n\} \tag{1}$$

where:

- E denotes the set of log events,
- e_i represents the i-th log event, and
- n is the total number of events collected from the Ubuntu server logs.

Each event is represented by a structured vector containing the timestamp, source IP, destination IP, username, service, event type, and original log message, as shown in (2).

$$x_i = [t_i, \mathrm{src}_i, \mathrm{dst}_i, u_i, \mathrm{svc}_i, \mathrm{evt}_i, \mathrm{raw}_i], i = 1,2,\dots,n \quad (2)$$

where:

- x_i denotes the feature vector of the i-th log event,
- t_i is the event timestamp,
- src_i is the source IP address,
- dst_i is the destination IP address,
- u_i is the associated user account,
- svc_i is the service or process generating the event,
- evt_i is the parsed event type, and
- raw_i is the original raw log message.

For each simulated scenario s, the preprocessing module groups the relevant events according to the execution window and scenario context. The resulting scenario-level event group is expressed in (3).

$$G_s = \{e_i \in \mathrm{E} | \tau_{s,start} \le t_i \le \tau_{s,end}\} \quad (3)$$

where:

- G_s denotes the set of log events belonging to session or time window s,
- E is the complete set of collected log events,
- e_i is the i-th log event,
- t_i is the timestamp of event e_i,
- $\tau_{s,start}$ is the start time of session (or window) s, and
- $\tau_{s,end}$ is the end time of session (or window) s.

The Hermes analysis module is formalized as an inference function H that receives the event group G_s and the prompt policy P. Its output is a structured tuple that contains the predicted class, interpreted attack type, severity, reasoning, recommended action, and human-review flag, as described in (4).

$$\hat{\mathbf{o}}_s = \mathcal{H}(G_s, P) = \left(\hat{c}_s, \hat{a}_s, \hat{v}_s, \hat{r}_s, \hat{m}_s, \hat{h}_s\right) \quad (4)$$

where:

- $\hat{\mathbf{o}}_s$ is the output generated for session s,
- $\mathcal{H}(\cdot)$ denotes the Hermes reasoning function,
- G_s is the set of log events within session s,
- P represents the predefined security playbook and operational policies,
- $\hat{c}_s$ is the predicted attack category,
- $\hat{a}_s$ is the attack interpretation,
- $\hat{v}_s$ is the predicted severity level,
- $\hat{r}_s$ is the reasoning generated by the model,
- $\hat{m}_s$ is the recommended mitigation action, and
- $\hat{h}_s$ indicates whether human review is required.

The mitigation validator V then checks the recommended action against the allowed action set A and the forbidden action set F. The validation rule is given in (5).

$$V(\hat{m}_s, A, F) = \begin{cases} \text{accepted}, & \text{if } \hat{m}_s \in A \ \wedge \ \hat{m}_s \notin F, \\ \text{rejected}, & \text{if } \hat{m}_s \in F, \\ \text{human\_review}, & \text{otherwise.} \end{cases} \quad (5)$$

where:

- $V(\cdot)$ denotes the mitigation validation function,
- $\hat{m}_s$ is the mitigation action recommended by Hermes for session s,
- A is the set of permitted (approved) mitigation actions,
- F is the set of forbidden or unsafe actions,
- accepted indicates that the recommendation complies with the security playbook,
- rejected indicates that the recommendation violates security policies, and
- human_review indicates that the recommendation requires manual verification before execution.

For methodological comparison, conventional SVM and Naive Bayes can be written as classifier functions over a feature vector x. A simplified SVM decision function is presented in (6).

$$f_{\text{SVM}}(\mathbf{x}) = \text{sign}\left(\mathbf{w}^T \phi(\mathbf{x}) + b\right) \quad (6)$$

Meanwhile, the Naive Bayes classifier estimates the most likely class by maximizing the posterior probability, as shown in (7).

$$\hat{c} = \arg\max_{c \in \mathcal{C}} P(c) \prod_{j=1}^{d} P(x_j \mid c) \quad (7)$$

These conventional classifiers are valuable for anomaly detection on structured and sufficiently representative labeled datasets. However, the objective of this study is different: the Hermes workflow evaluates scenario-level triage, reasoning, and playbook-validatable recommendation rather than optimizing a newly trained statistical classifier on a large benchmark dataset.

4. Experimental Setup

The experiment was carried out in a controlled local virtual laboratory. Ubuntu Server served as the log-generating target, while a client machine produced normal and anomalous activities. Data were collected on June 8, 2026 from journalctl, auth.log, and ufw.log according to each executed scenario. Dataset generation was separated from the Fail2Ban baseline test: Fail2Ban was disabled during brute-force data collection so that enough failed-authentication events could be recorded, and then enabled again for the baseline measurement.

The eight scenario groups were selected as a proof of concept to represent common operational Ubuntu/SSH log categories observed in small to medium server environments: normal activity, light authentication failure, SSH brute force, invalid user login, root login attempt, port scanning, and sudo attempts by an unauthorized user. S03B was separated

as a dedicated path for the Fail2Ban baseline. Therefore, this evaluation is not intended as a large-scale statistical test of an LLM, but as an initial validation that the Hermes pipeline can handle distinct security-log categories in a structured manner.

The record distribution across scenarios was not manually balanced. S01 and S07 produced more entries because normal login activity and sudo authentication failure generated verbose PAM, session, and sudo-auth event sequences. S05 root login attempt, on the other hand, produced fewer lines because the scenario itself was short. The imbalance was retained to reflect the different verbosity levels that appeared across event types in the local simulation.

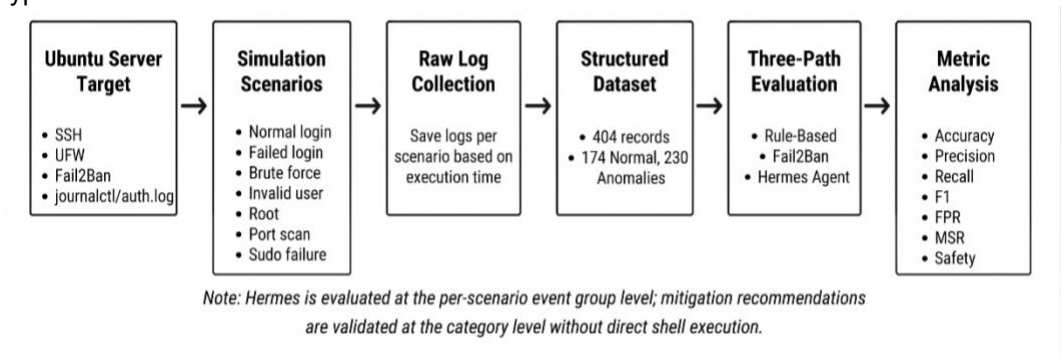

Fig. 2. Experimental workflow and log evaluation process

Table 2. Dataset summary by scenario

Scenario	Scenario Name	Records	Label	Attack Type
S01	Normal SSH Login	167	Normal	none
S02	Light Failed Login	7	Normal	none
S03	SSH Brute Force	21	Anomaly	ssh_brute_force
S03B	Fail2Ban Baseline	14	Anomaly	ssh_brute_force
S04	Invalid User Login	12	Anomaly	invalid_user_login
S05	Root Login Attempt	5	Anomaly	root_login_attempt
S06	Port Scanning	10	Anomaly	port_scan
S07	Sudo Failure / Privilege Attempt	168	Anomaly	privilege_escalation_attempt

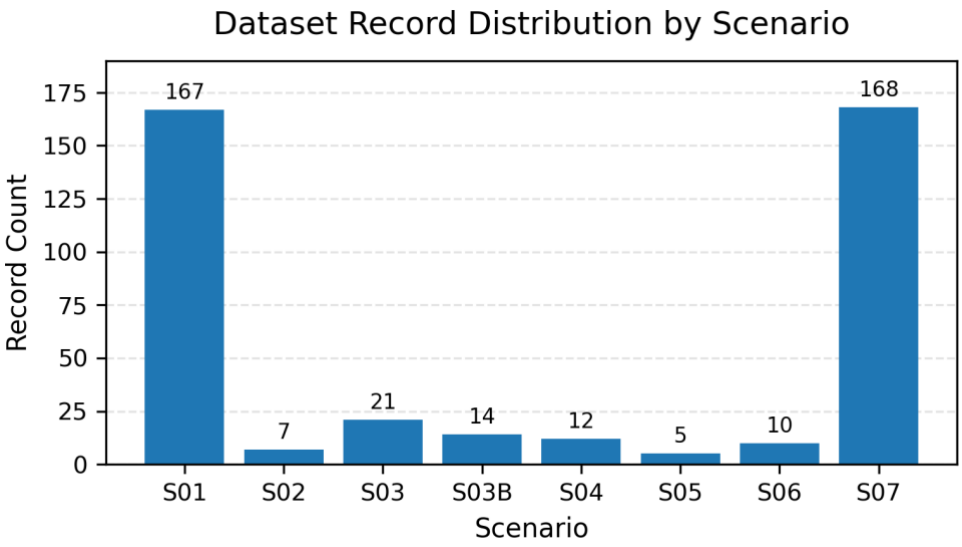

Fig. 3. Distribution of dataset records by scenario

Table 3. Evaluation paths and detector scope

Path	Method	Input	Output	Scope
A	Rule-Based Script	dataset_log_hermes.csv	rule_based_prediction.csv	Rule-based detection on structured dataset
B	Fail2Ban	Raw SSH/auth log	fail2ban_result.csv	Practical baseline for SSH brute force
C	Hermes Agent	Scenario-level event summary	hermes_prediction.csv	Agentic analysis and mitigation recommendation
D	Mitigation Validator	Hermes recommendation	mitigation_validation_result.csv	Playbook validation and action safety

4.1 Data Analysis

Data analysis compared each method output with the ground truth. For the binary setting, Normal was used as the negative class and Anomaly as the positive class. The evaluation metrics were accuracy, precision, recall, F1-score, false positive rate, attack type accuracy, severity accuracy, mitigation action accuracy, unsafe action rate, and rejected action rate.

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN}) \quad (8)$$

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP}) \quad (9)$$

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN}) \quad (10)$$

$$\text{F1} = 2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall}) \quad (11)$$

Because Hermes is evaluated mainly for triage, reasoning, and mitigation recommendation, the analysis was not limited to binary classification metrics. The reasoning outputs were checked with a qualitative rubric consisting of four criteria: (8) evidence alignment, meaning that the rationale must refer to signals present in the input, such as failed password, invalid user, root login attempt, port scan, or sudo failure; (9)

triage consistency, meaning that severity must follow the playbook and the scenario characteristics; (10) action compliance, meaning that the recommended action must come from the allowed action set; and (11) hallucination check, meaning that the rationale must not introduce IP addresses, usernames, services, commands, or incident facts that were not included in the input. A reasoning output was counted as valid only if all criteria were met.

The researcher acted as the domain evaluator for reasoning validity and applied the same rubric to all scenario groups. Each Hermes output was checked against the scenario ground truth, log evidence excerpts, severity rules, and mitigation playbook. An output was accepted when the reasoning matched the log evidence, the severity was consistent with the attack type, the recommendation belonged to the allowed playbook, and no risky action or command hallucination appeared. For this reason, reasoning validity is treated as a controlled rubric-based assessment, not as an absolute claim of objectivity in production environments.

$$\text{Reasoning Validity Rate} = \text{Valid reasoning outputs} / \text{Total scenario-group outputs} \quad (12)$$

$$\text{Severity Accuracy} = \text{Correct severity assignments} / \text{Total scenario-group outputs} \quad (13)$$

$$\text{Mitigation Action Accuracy} = \text{Correct mitigation recommendations} / \text{Total scenario-group outputs} \quad (14)$$

This rubric validates reasoning against the playbook and log evidence rather than against the length or style of the generated explanation. Therefore, a long rationale that is unsupported by evidence or introduces non-existent facts is counted as invalid.

5. Result and Analysis

The dataset reported in this section follows the event representation described in (M1) and (M2), where raw Ubuntu Server logs are treated as structured event records. For Hermes evaluation, these records were grouped into scenario-level event sets according to (M3), which explains why Hermes was assessed at the event-group level while aggregate detection metrics were later propagated to record level. The final dataset contained 404 records, consisting of 174 Normal records and 230 Anomaly records distributed across eight scenario groups. The rule-based baseline produced TP=230, TN=174, FP=0, and FN=0. Based on Equations (8) to (11), the rule-based baseline achieved accuracy, precision, recall, and F1-score of 1.00 on the controlled proof-of-concept dataset. This result is reasonable because the simulated anomalies used explicit indicators such as repeated failed authentication, invalid user attempts, root login attempts, port-scan evidence, and sudo-authentication failures. In other words, the rule-based baseline was able to classify all records correctly because the controlled dataset contained clear attack signatures that could be mapped to deterministic rules. Therefore, the result demonstrates that deterministic rules can fully capture clean explicit patterns in this dataset, but it should not be interpreted as evidence of robustness against noisy, imbalanced, and dynamic production logs.

Hermes Agent was assessed at the event-group level for each scenario, and the resulting scenario-level predictions were propagated to record level for aggregate metric calculation. Hermes produced TP=230, TN=174, FP=0, and FN=0. Based on Equations (8) to (11), accuracy, precision, recall, and F1-score were therefore 1.00 on this controlled dataset.

Attack type accuracy, severity accuracy, mitigation action accuracy, and reasoning validity rate were also 1.00 according to Equations (12) to (14) and the reasoning rubric. The comparison shows that Hermes matched the rule-based baseline on explicit simulated patterns, not that it outperformed the baseline as a detector. What matters more is that Hermes produced structured outputs that binary accuracy cannot show, namely reason, severity, recommended_action, and requires_human_review. These outputs are used to support incident explanation and playbook validation, which are central to the proposed agentic response-support workflow.

Using the reasoning rubric, all eight scenario-group outputs met the criteria for evidence alignment, triage consistency, action compliance, and hallucination checking. The Reasoning Validity Rate in Equation (12) was therefore 1.00 in this evaluation. This value indicates that Hermes generated rationales that were consistent with the log evidence in the explicit scenarios tested, but it should not be read as a guarantee of hallucination-free behavior on real-world logs that are more diverse, incomplete, adversarial, or outside the predefined playbook.

To clarify the form of Hermes output, Table 4 presents representative outputs for several scenario groups. The examples show that the contribution of Hermes is not limited to classification labels, but also includes attack-type interpretation, severity assignment, evidence-based reasoning, and playbook-validatable action recommendation. This table was added to make the agentic output observable for readers and to show how Hermes transforms log evidence into a structured response-recommendation format. The Hermes output in the results follows the structured inference tuple defined in (M4). Thus, each scenario-level output is not limited to a predicted class, but also includes attack_type, severity, reason, recommended_action, and requires_human_review. The recommended_action field is then checked using the mitigation validator in (M5), so that only playbook-compliant actions are accepted.

Table 4. Example of Hermes output at the event-group level

Scenario	classification	attack_type	severity	recommended_action	reasoning summary
S01 Normal SSH Login	Normal	none	Low	no_action	Normal login.
S03 SSH Brute Force	Anomaly	ssh_brute_force	High	block_ip	SSH brute force.
S07 Sudo Failure	Anomaly	privilege_escalation_attempt	Medium	create_incident_report	Sudo failure.

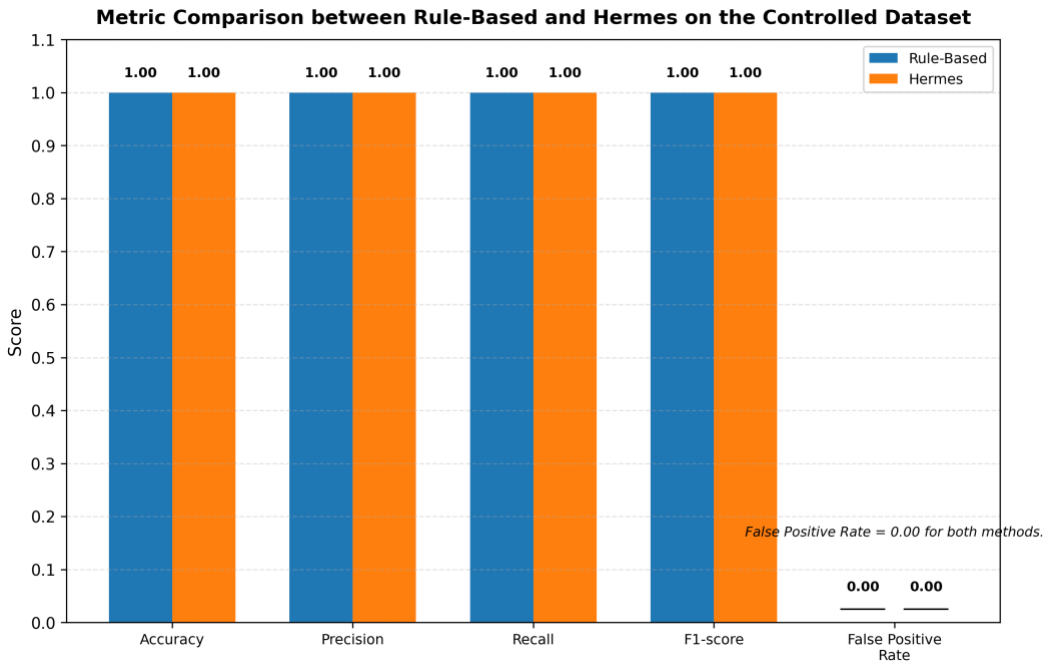

Fig. 4. Metric comparison between Rule-Based and Hermes on the controlled dataset

Table 5. Detection performance comparison

Method	Accuracy	Precision	Recall	F1-score	FPR
Rule-Based Script	1.00	1.00	1.00	1.00	0.00
Hermes Agent	1.00	1.00	1.00	1.00	0.00

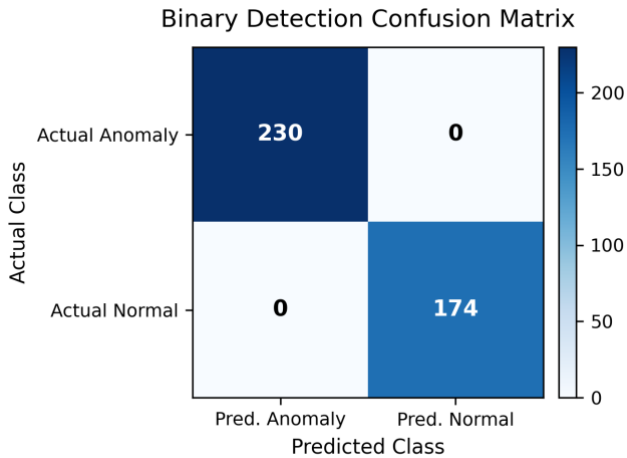

Fig. 5. Confusion matrix of Hermes evaluation

Table 5 and Fig. 4 show that the Rule-Based Script and Hermes Agent reached the same binary classification performance on the controlled dataset. Accuracy, precision, recall, and F1-score were 1.00 for both methods, and the false positive rate was 0.00, which means

that no Normal record was incorrectly classified as Anomaly. This comparison is useful because it prevents an overclaim: Hermes is not presented as a more accurate detector than the deterministic rule-based baseline under clean simulated conditions. Rather, the result shows that the Hermes pipeline maintained classification consistency while adding interpretable outputs for severity triage, attack-type interpretation, reasoning, and safe mitigation recommendation. Thus, this section should be read as an evaluation of both detection consistency and response-support quality, not as a pure accuracy contest.

Although SVM and Naive Bayes are discussed in the related works and mathematical formulation as common supervised anomaly detection approaches, they were not implemented as experimental baselines in this study. This decision is consistent with the scope of the controlled proof-of-concept evaluation, where the dataset consists of 404 records derived from eight scenario groups and the main objective is to evaluate the feasibility of Hermes as a triage and response-recommendation support layer. SVM and Naive Bayes generally require a larger and more representative labeled feature matrix, training and testing partitions, and broader feature engineering to produce a meaningful supervised-learning comparison. Therefore, the experimental comparison in this study is limited to the rule-based baseline, Fail2Ban, and Hermes, while SVM and Naive Bayes are positioned as conventional machine-learning references and future comparative baselines for larger-scale evaluations. In relation to the mathematical formulation, SVM and Naive Bayes are represented conceptually in (M6) and (M7). However, they are not executed as experimental baselines in this study because the present evaluation focuses on the constrained Hermes inference-and-validation workflow in (M4) and (M5), not on training a supervised classifier over a large labeled feature matrix.

Fail2Ban was evaluated as a practical baseline for the SSH brute-force scenario S03B. Fail2Ban reached detected=yes and banned=yes with a response time of 14 seconds. The measurement was performed proactively through the auth.log trace to track the precise second at which authentication-failure evidence exceeded the maxretry threshold within the findtime window and the ban condition was confirmed. This approach was used as an operational response-time proxy to reduce bias caused by potential write delay in a separate Fail2Ban log. The Fail2Ban result confirms that established rule-based mitigation remains effective for its intended scope, especially SSH brute-force blocking, whereas Hermes is evaluated as a complementary analysis layer that explains event context and recommends safe limited actions.

5.1 Execution-Time Comparison and Output Quality

In addition to detection performance, this study reports execution-time analysis to distinguish deterministic rule execution from agentic reasoning. The rule-based baseline was executed locally on 404 records and required an average of approximately 0.0078 seconds for the core rule inference process. Fail2Ban reached the banned condition in S03B with an estimated response time of 14 seconds based on the authentication log pattern. Hermes was re-measured on 8 scenario groups through the Hermes CLI runtime with the remote openai-codex provider using the gpt-5.5 model. Latency was calculated as wall-clock time from request to final response, resulting in a mean of 11.61 seconds, a median of 10.82 seconds, a minimum of 9.17 seconds, and a maximum of 17.48 seconds per event group. This value includes Hermes routing, provider/network communication, model inference, and Hermes CLI invocation overhead. Therefore, Hermes is not positioned as a faster or more accurate real-time detector than the rule-based baseline, but as an analysis-support layer that trades longer computation time for additional outputs, including decision rationale, severity triage, attack_type interpretation, and playbook-based action recommendation.

Table 6. Execution-time comparison and output characteristics

Method	Evaluation unit	Time	Measurement status	Output characteristics
Rule-Based	404 records	≈ 0.0078 seconds	Measured from local rule execution	Rule-derived label, attack_type, severity, action
Fail2Ban	S03B SSH brute force	14 seconds	Auth-log-derived operational response proxy; ban condition confirmed in S03B	Detected and banned
Hermes Agent	8 scenario groups	Mean 11.61 seconds; median 10.82 seconds; min–max 9.17–17.48 seconds per event group	Measured as wall-clock request-to-final-response through Hermes CLI with the remote openai-codex provider, n=8; includes Hermes routing, network/provider, model inference, and CLI overhead	Label, attack_type, severity, reasoning, recommended_action

Table 7. Fail2Ban baseline result

Scenario	Source IP	Detected	Banned	Response Time	Notes
S03B	10.1.1.9	Yes	Yes	14.0 seconds	Response time derived proactively from auth.log trace at the maxretry threshold and ban-confirmation condition.

Mitigation validation showed that every Hermes recommendation was within the set of actions allowed by the playbook. No unsafe action was identified, and no recommendation was rejected by the validator. This result, however, reflects action-category validation rather than direct shell execution on the target server. The finding is important because the design keeps Hermes inside a bounded response space and prevents destructive actions such as deleting users, stopping SSH services, rebooting the server, flushing firewall rules, or modifying system permission.

Table 8. Hermes mitigation recommendation validation

Scenario	Attack Type (attack_type)	Recommended Action (recommended_action)	Validator Status	Unsafe Action
S01	none	no_action	accepted	False
S02	none	no_action	accepted	False
S03	ssh_brute_force	block_ip	accepted	False
S03B	ssh_brute_force	block_ip	accepted	False
S04	invalid_user_login	add_watchlist	accepted	False
S05	root_login_attempt	create_alert	accepted	False
S06	port_scan	block_ip	accepted	False

S07	privilege_escalation_attempt	create_incident_report	accepted	False

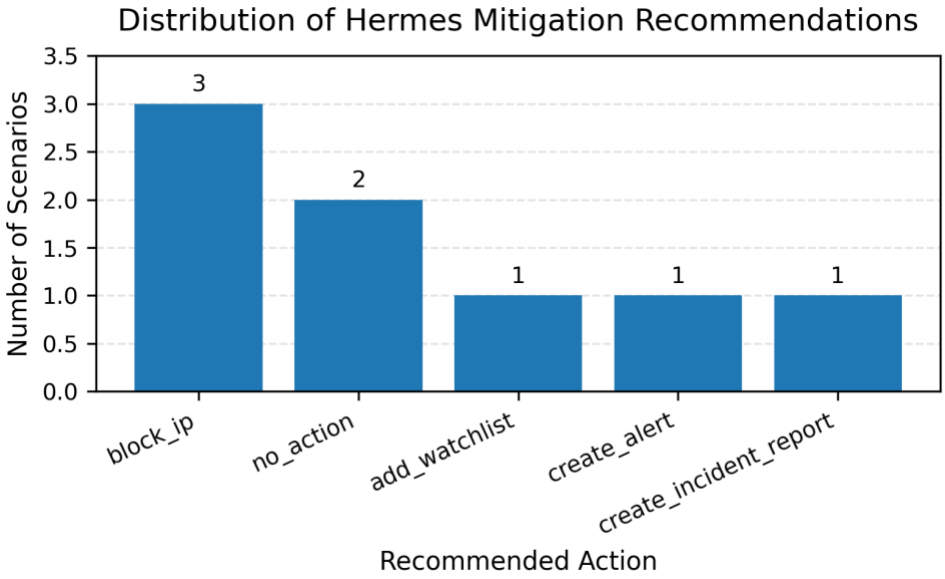

Fig. 6. Distribution of Hermes mitigation recommendations

Taken together, the results show that rule-based detection remains effective for explicit attack patterns in a controlled dataset. Hermes therefore should not be read as a more accurate detector or as a replacement for rule-based detection. Its added value appears after initial classification, where it provides automated severity triage, contextual reasoning, attack-type interpretation, playbook-compliant mitigation recommendation, and auditability. Through classification, attack_type, severity, reason, recommended_action, and requires_human_review, Hermes turns event classification into a more informative, structured, and auditable decision-support workflow. The tables and figures in this section support two findings: first, both methods classified the controlled records consistently; second, Hermes added response-oriented information that can help administrators understand why an event is suspicious and which limited mitigation category is appropriate. This discussion also explains why the study does not directly compare Hermes with SVM or Naive Bayes as trained classifiers: the goal is to evaluate a constrained agentic response-recommendation layer, while SVM and Bayesian approaches are conventional detection models that require larger labeled feature matrices and separate training procedures. All interpretations remain limited to an initial controlled evaluation and should not be treated as claims of production performance.

6. Conclusion

This study designed and evaluated a Hermes-based agentic security response system to support Ubuntu Server log analysis, severity triage, attack-type interpretation, contextual reasoning, and limited playbook-based mitigation recommendation. The study was motivated by the problem that manual log inspection is time-consuming and that pattern-based tools, although effective for explicit signatures, do not provide rich incident reasoning or structured mitigation recommendations. The proposed method combines log collection, preprocessing into structured event groups, Hermes-based inference, threat-decision mapping, playbook validation, and audit-oriented reporting. The most valuable finding is that Hermes can be positioned as a structured analysis and response-recommendation

support layer rather than as an autonomous security executor.

On the controlled proof-of-concept dataset of 404 records and eight scenario groups, both the rule-based baseline and Hermes achieved accuracy, precision, recall, and F1-score of 1.00. Hermes also achieved attack type accuracy, severity accuracy, mitigation action accuracy, and reasoning validity rate of 1.00. These results show that the workflow operated consistently in the local simulation and that Hermes outputs were aligned with the ground truth and mitigation playbook. Because the rule-based baseline also reached perfect performance, the contribution of Hermes should not be interpreted as superior accuracy. Instead, the study validates Hermes as an analysis and response-recommendation support layer that adds severity triage, contextual reasoning, attack-type interpretation, playbook-compliant mitigation recommendation, and auditability on top of deterministic detection. The mathematical formulation also shows that the system operates as a constrained inference-and-validation pipeline: event groups are transformed into structured outputs, and recommended actions are accepted only when they belong to the allowed playbook and do not violate forbidden-action constraints.

These findings are still limited to local simulation and eight scenario groups. They do not demonstrate generalization to production environments with noisy, imbalanced, and dynamic logs, nor do they show that Hermes is ready to replace SIEM, IDS, Fail2Ban, or other production security mechanisms. Future work should evaluate the system on larger and real-world log datasets, increase the number and diversity of scenario groups, compare Hermes with other LLM providers and conventional ML baselines such as SVM and Naive Bayes on larger labeled datasets, assess Hermes in near real-time monitoring mode, implement sandboxed command-whitelist execution, compare latency across remote API and local inference deployments, and test resilience against prompt injection and risky action recommendations. Later evaluations should also involve multiple independent human experts to assess reasoning quality and include more novel or obfuscated log variations to reduce possible bias from LLM prior knowledge of standard Ubuntu log formats.

References

[1] S. T. A. Ramadhani, F. M. Puri, and A. K. Huda, “Enhanced security of Linux server-based servers with a combination of iptables and knocking ports,” J. Sist. Telekomun. Elektron. Sist. Kontrol Power Sist. dan Komput., vol. 4, no. 2, pp. 113–124, 2024, doi: 10.32503/jtecs.v4i2.5541.

[2] S. T. A. Ramadhani, F. M. Puri, B. Gozali, M. Alfarozi, and A. K. Huda, “Implementasi Cluster Proxmox dengan Fitur High Availability pada Laboratorium Sysadmin Universitas Amikom Yogyakarta,” J. TECNOSCIENZA, vol. 10, no. 1, pp. 139–153, 2025, doi: 10.51158/hzd0r562.

[3] M. Landauer, S. Onder, F. Skopik, and M. Wurzenberger, “Deep learning for anomaly detection in log data: A survey,” Machine Learning with Applications, vol. 12, p. 100470, Jun. 2023, doi: 10.1016/j.mlwa.2023.100470.

[4] A. Aziz and K. Munir, “Anomaly detection in logs using deep learning,” IEEE Access, vol. 12, pp. 176124–176135, 2024, doi: 10.1109/ACCESS.2024.3506332.

[5] S. Lupton, H. Washizaki, N. Yoshioka, and Y. Fukazawa, “Landscape and taxonomy of online parser-supported log anomaly detection methods,” IEEE Access, vol. 12, pp. 78193–78218, 2024, doi: 10.1109/ACCESS.2024.3387287.

[6] S. Srinivas et al., “AI-augmented SOC: A survey of LLMs and agents for security automation,” Journal of Cybersecurity and Privacy, vol. 5, no. 4, p. 95, 2025, doi: 10.3390/jcp5040095.

[7] M. Ismail et al., “Toward robust security orchestration and automated response in security operations centers with a hyper-automation approach using agentic artificial intelligence,” Information, vol. 16, no. 5, p. 365, 2025, doi: 10.3390/info16050365.

[8] J. Zhang et al., “When LLMs meet cybersecurity: A systematic literature review,” Cybersecurity, vol. 8, p. 55, 2025, doi: 10.1186/s42400-025-00361-w.
[9] H. F. Atlam, “LLMs in cyber security: Bridging practice and education,” Big Data and Cognitive Computing, vol. 9, no. 7, p. 184, 2025, doi: 10.3390/bdcc9070184.
[10] I. Hasanov, S. Virtanen, A. Hakkala, and J. Isoaho, “Application of Large Language Models in Cybersecurity: A Systematic Literature Review,” IEEE Access, vol. 12, pp. 176751–176778, 2024, doi: 10.1109/ACCESS.2024.3505983.
[11] S. Ali, C. Boufaied, D. Bianculli, P. Branco, and L. Briand, “A Comprehensive Study of Machine Learning Techniques for Log-Based Anomaly Detection,” Empirical Software Engineering, vol. 30, no. 5, p. 129, 2025, doi: 10.1007/s10664-025-10669-3.
[12] A. M. Abdallah, A. Alkaabi, G. Alameri, S. H. Rafique, N. S. Musa, and T. Murugan, “Cloud Network Anomaly Detection Using Machine and Deep Learning Techniques—Recent Research Advancements,” IEEE Access, vol. 12, pp. 56749–56773, 2024, doi: 10.1109/ACCESS.2024.3390844.
[13] M. Gupta, C. Akiri, K. Aryal, E. Parker, and L. Praharaj, “From ChatGPT to ThreatGPT: Impact of Generative AI in Cybersecurity and Privacy,” IEEE Access, vol. 11, pp. 80218–80245, 2023, doi: 10.1109/ACCESS.2023.3300381.
[14] J. Yu, A. V. Shvetsov, and S. H. Alsamhi, “Leveraging Machine Learning for Cybersecurity Resilience in Industry 4.0: Challenges and Future Directions,” IEEE Access, vol. 12, pp. 159579–159596, 2024, doi: 10.1109/ACCESS.2024.3482987.
[15] M. A. Ferrag et al., “Generative AI in Cybersecurity: A Comprehensive Review of LLM Applications and Vulnerabilities,” Internet of Things and Cyber-Physical Systems, vol. 5, pp. 1–46, 2025, doi: 10.1016/j.iotcps.2025.01.001.
[16] H. Xu et al., “Large Language Models for Cyber Security: A Systematic Literature Review,” ACM Transactions on Software Engineering and Methodology, 2025, doi: 10.1145/3769676.
[17] F. He, T. Zhu, D. Ye, B. Liu, W. Zhou, and P. S. Yu, “The Emerged Security and Privacy of LLM Agent: A Survey with Case Studies,” ACM Computing Surveys, vol. 58, no. 6, pp. 1–36, 2025, doi: 10.1145/3773080.
[18] O. Afolalu and M. S. Tsoeu, “Artificial Intelligence as the Next Frontier in Cyber Defense: Opportunities and Risks,” Electronics, vol. 14, no. 24, p. 4853, 2025, doi: 10.3390/electronics14244853.
[19] N. O. Jaffal, M. Alkhanafseh, and D. Mohaisen, “Large language models in cybersecurity: A survey of applications, vulnerabilities, and defense techniques,” AI, vol. 6, no. 9, p. 216, 2025, doi: 10.3390/ai6090216.
[20] W. Kasri et al., “From vulnerability to defense: The role of large language models in enhancing cybersecurity,” Computation, vol. 13, no. 2, p. 30, 2025, doi: 10.3390/computation13020030.
[21] Y. Duan et al., “LogEDL: Log anomaly detection via evidential deep learning,” Applied Sciences, vol. 14, no. 16, p. 7055, 2024, doi: 10.3390/app14167055.
[22] S. Shrestha, “Investigation of cybersecurity bottlenecks of AI agents in industrial automation,” Computers, vol. 14, no. 11, p. 456, 2025, doi: 10.3390/computers14110456.
[23] N. Kshetri, “Transforming cybersecurity with agentic AI to combat emerging cyber threats,” Telecommunications Policy, vol. 49, no. 6, p. 102976, Jul. 2025, doi: 10.1016/j.telpol.2025.102976.
[24] D. Tuscano and J. P. Disso, “A Practical Incident-Response Framework for Generative AI Systems,” Journal of Cybersecurity and Privacy, vol. 6, no. 1, p. 20, 2026, doi: 10.3390/jcp6010020.
[25] A. Hozouri, A. Mirzaei, and M. Effatparvar, “A comprehensive survey on intrusion detection systems with advances in machine learning, deep learning and emerging cybersecurity challenges,” Discover Artificial Intelligence, vol. 5, no. 1, p. 314, 2025, doi: 10.1007/s44163-025-00578-1.
[26] R. Chinnasamy, M. Subramanian, S. V. Easwaramoorthy, and J. Cho, “Deep learning-driven methods for network-based intrusion detection systems: A systematic review,” ICT Express, vol. 11, no. 1, pp. 181–215, Feb. 2025, doi: 10.1016/j.icte.2025.01.005.
[27] H. M. R. U. Rehman et al., “A systematic literature study of machine learning techniques based intrusion detection: datasets, models, challenges, and future directions,” Journal of Big Data, vol. 12, p. 264, 2025, doi: 10.1186/s40537-025-01323-2.