Automated Cyberattack Response System: A Combination of AI and Human Control with Recommendations for Measurable Actions

Febrian Sulistyo Budi[1*], Bondan Wahyu Pamekas[2], Arif Setiawan[3]

[1,2,3] *Universitas Duta Bangsa Surakarta, Jl. Bhayangkara No. 55, Tipes, Serengan, Surakarta, Central Java 57154, Indonesia*

Keywords
SIEM; Wazuh; LLM; Incident Response; MTTR; n8n

Correspondence Email:
220103091@mhs.udb.ac.id

Abstract
The growing frequency and complexity of cyberattacks have placed significant pressure on the security of digital infrastructure, particularly because manual incident handling tends to be slow and prone to delayed responses. This study aims to design and build an automated security incident response system that combines artificial intelligence with human oversight to improve both the speed and quality of response. The system was developed using the Waterfall method, integrating Wazuh as the network security monitoring platform, n8n as the workflow orchestration engine, and the Llama 3.1 8B language model accessed through the Groq API to generate mitigation recommendations in plain, easy-to-understand language. A human-in-the-loop approval mechanism was implemented through a Telegram bot, requiring analyst confirmation before any mitigation action is executed. The system was evaluated across three attack scenarios, SSH brute force, multiple failed login attempts, and suspicious file modification, each tested 15 times in a controlled virtual environment. The results show an average response time of 21.2, 34.1, and 36.3 seconds for each scenario respectively, far outperforming manual handling, which typically requires between 600 and 1,800 seconds. This translates into a response speed improvement of up to 98%, while the AI-generated recommendations remained fully consistent with the rule-based engine across all 45 trials conducted. These findings suggest that combining open-source SIEM, workflow automation, and LLM-based reasoning with human supervision offers a practical, low-cost, and reliable approach for strengthening incident response capability in resource-constrained environments.

1. Introduction

Cyberattacks are growing in frequency and complexity, putting real pressure on how organizations protect their digital infrastructure. Security Information and Event Management (SIEM) systems have become central to detecting threats and correlating logs across a network [1], but the threat landscape keeps evolving just as fast [2].

The real challenge, though, is rarely detection it is reacting fast enough once a threat is found. Nguyen et al. [3] found that organizations typically take 63 days to discover an incident and another 20 days to fix it, long enough for an attack to spread well beyond its original scope. This is pushing SIEM platforms to evolve from simple monitoring tools into systems built to save analysts' time rather than generate more alerts [4].

Wazuh, an open-source SIEM and XDR platform, is a practical answer because its affordable, scalable [5], and effective at catching real attacks in production environments [6]. Folding AI into a SOC's workflow takes this further, with real potential to reshape how analysts respond to threats [7]. Tools like n8n add another piece, connecting services through a no-code interface and enriching alerts with context before they reach an analyst, speeding up decision-making [8], [9]. Large Language Models (LLMs) bring yet another layer, generating natural language coherently enough to open new possibilities in cybersecurity [10], [11]. Here, an LLM accessed via the Groq API (Llama 3.1 8B Instant) turns raw alerts into plain-language recommendations, an approach shown elsewhere to reach an F1-score of 87.2% [12] and cut MTTR by 77.8% [3].

Even so, a rule-based engine is fast, consistent, auditable [13] and a human-in-the-loop approval step still matter, since LLM recommendations need a person's sign-off before any action affecting live infrastructure [12], [14].

A few recent studies offer useful comparison points. Ismail et al. [12] combined Wazuh with an LLM via Retrieval-Augmented Generation but without human approval. Nguyen et al. [3] achieved a large MTTR reduction on a proprietary platform. Saputra et al. [6] validated Wazuh operationally, but without AI or automation. These differences are summarized in Table 1.

Table 1. Comparison with Previous Studies

Researcher	Method	Difference from This Study
Ismail et al. [12] (2025)	Wazuh + RAG + LLM (GPT-4o)	No rule-based engine or human approval via Telegram; relies on RAG and a vector database rather than direct prompt engineering
Nguyen et al. [3] (2024)	AI4SOAR + enterprise SOAR platform	Proprietary, not open-source; no LLM component for natural-language analysis or hybrid human-approval mechanism
Saputra et al. [6] (2024)	Wazuh SIEM monitoring only	Monitoring only, without any AI, workflow automation, or automated response component

These gaps point to a clear opportunity: a system combining an open-source SIEM, workflow orchestration, LLM-based reasoning, and real-time human approval in one measurable architecture. That is what this study builds, a hybrid system integrating Wazuh, n8n, and Llama 3.1 8B via the Groq API, developed using the Waterfall method and tested across three attack scenarios (SSH brute force, multiple failed login, file modification), with approval delivered through a Telegram bot. The system responded in seconds rather than minutes or hours, keeping its AI-generated recommendations fully aligned with the rule-based engine. The main contribution lies in bringing these four elements together: open-source SIEM, no-code orchestration, prompt-engineering-driven LLM analysis, and real-time analyst approval into one measurable system not previously implemented this way.

2. Research Methods

This study was developed using the Waterfall method, structured into six sequential stages: system analysis and design, system design, environment setup and implementation, workflow development and integration, evaluation, and report writing. Waterfall was chosen because the system requirements were already well defined from the start, and each stage needed a clear, measurable output before moving to the next, an approach that tends to produce implementations that are easier to audit and rely on later [13].

Fig. 1. System development stages

The system was built entirely in a virtual environment using VirtualBox, with three virtual machines running on a single physical host. VM1 ran Wazuh Manager (version 4.14.3) together with n8n, forming the core of the system; VM2 ran Ubuntu Server 24.04 with a Wazuh Agent installed as the test target; and VM3 ran Kali Linux as the attacking machine. All three were connected through a Bridged Adapter network configuration, keeping them on the same network segment.

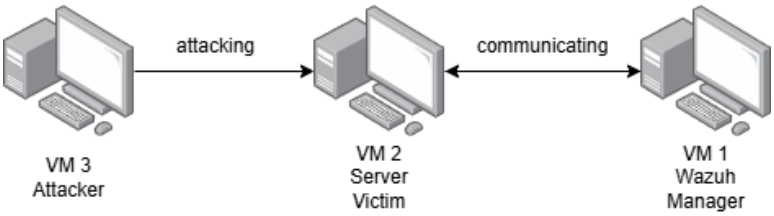

Fig. 2. Lab network architecture/topology

The workflow begins the moment the Wazuh Agent on VM2 detects suspicious activity and sends a log to the Wazuh Manager. The Manager processes that log through its rule engine and generates an alert in JSON format. A custom integration script installed on the Wazuh Manager then automatically forwards this alert to an n8n webhook via HTTP POST. Once inside n8n, the alert passes through a Switch node acting as a rule-based engine to classify the type of incident, then gets sent to the Groq API for analysis by the LLM. The resulting recommendation is forwarded to a Telegram bot, along with Approve and Reject buttons for the analyst to act on. If the analyst selects Approve, the system automatically executes the mitigation action, blocking the attacker's IP via iptables on VM2. This kind of open-source, low-code orchestration setup mirrors a broader trend in recent cybersecurity automation work, where platforms like n8n are being used to handle attack mitigation directly rather than just passing alerts along [15], and it reflects how an architecture built entirely on open-source components can still offer real integration flexibility without a heavy implementation cost [5].

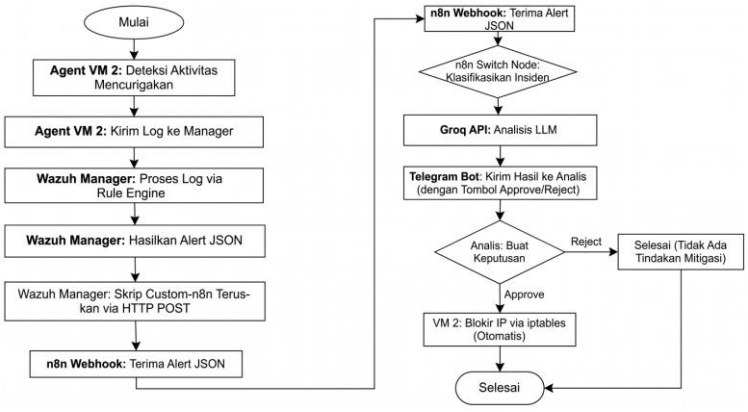

Fig. 3. System workflow flowchart, from detection to mitigation execution

Table 2 summarizes the technical specifications of each component used in the system.

Table 2. Technical Specifications of System Components

Component	Technology	Version
SIEM	Wazuh Manager & Wazuh Agent	4.14.3
Workflow Orchestration	n8n	2.8.4
LLM	Llama 3.1 8B Instant via Groq API	-
Notification and Approval	Telegram Bot	-
Target Server	Ubuntu Server	24.04 LTS
Attacker Machine	Kali Linux	2024.x
Hypervisor	VirtualBox	-

As Table 2 shows, every core component in this system is either open-source or free to use, which means the entire setup can be replicated without any licensing cost, a growing pattern in SOC implementations that increasingly rival commercial solutions while staying far more flexible to adapt [7].

The prompt design used to query the LLM was kept deliberately simple, structured to take in the raw alert details. Rule ID, severity level, and a short description of the triggering event and return a mitigation recommendation written in plain language rather than technical shorthand. This mirrors a broader shift in how prompt engineering is being approached specifically for cyber defense use cases, where the goal is less about demonstrating model capability and more about producing guidance an analyst can act on immediately [16]. The decision to add a human approval step before any action executes was guided by a similar line of reasoning found across recent human-in-the-loop research: even highly capable automated systems benefit from a deliberate checkpoint where a person retains the final call, particularly when an action has a direct, real-world effect on infrastructure [17].

Testing was carried out across three attack scenarios chosen to represent common threats in a Linux-based server environment, a deliberate choice, since picking scenarios that genuinely reflect real-world conditions matters for evaluating a detection system's effectiveness in any meaningful way [2]. Each scenario was run 15 times, spread across four different time windows: Saturday night, Sunday early morning, Monday afternoon, and Tuesday midday to simulate varying network traffic conditions. Table 3 details each scenario.

Table 3. Testing Scenario Details

Scenario	Method	Tool	Rule ID	Severity Level
SSH Brute Force	Repeated login attempts using a wordlist	Hydra	5710	10
Multiple Failed Login	Repeated failed logins via automated script	Python with Paramiko library	2502	10
File Modification	Manual modification of configuration files in /etc/*	Manual	550	7

As shown in Table 3, the three scenarios cover two distinct categories of threat: authentication-based attacks (Scenarios 1 and 2) and system integrity threats (Scenario 3). For every trial, two metrics were recorded MTTR, measured in seconds from the moment the attack was executed to the moment the notification reached Telegram, and the consistency of the AI-generated recommendation against the rule-based engine's classification.

3. Result and Discussion

3.1 System Implementation

The system came together smoothly, with every component running stably throughout testing in the virtual environment. Wazuh Manager picked up and processed alerts from the Wazuh Agent without issue, n8n

executed its workflows automatically as designed, and the Groq API consistently returned mitigation recommendations written in plain, easy-to-follow language. The human-in-the-loop step worked as intended too, analysts could approve or reject a recommended action in real time directly from Telegram. None of this happened by accident; the reliability of an incident response automation system ultimately comes down to how well its individual components integrate with one another [18], and across every testing session in this study, that integration held up without a single failure.

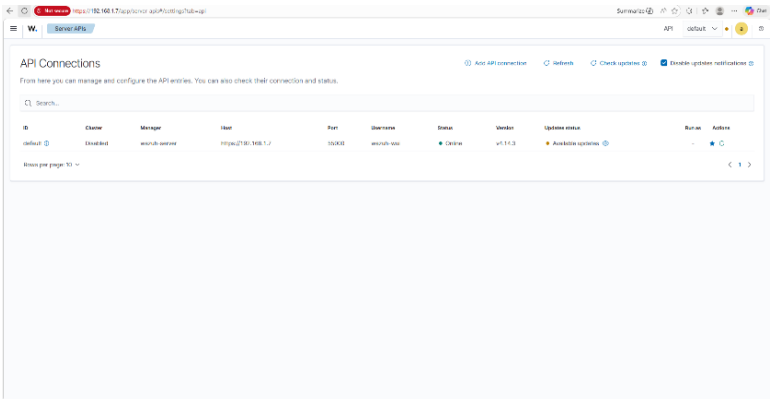

Fig. 4. Wazuh Dashboard interface, showing the API connection status.

Fig. 4 shows the Wazuh Dashboard during testing, confirming an active connection between the dashboard and the Wazuh API.

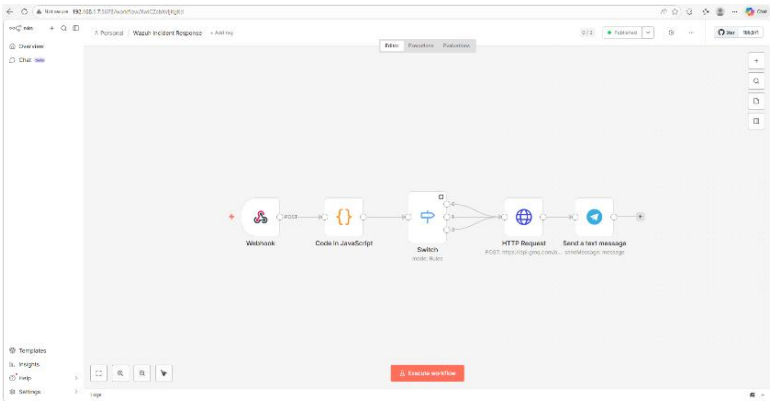

Fig. 5. n8n incident response workflow

Fig. 5 illustrates the incident response flow inside n8n, starting from the webhook trigger and ending with the notification sent to the Telegram bot.

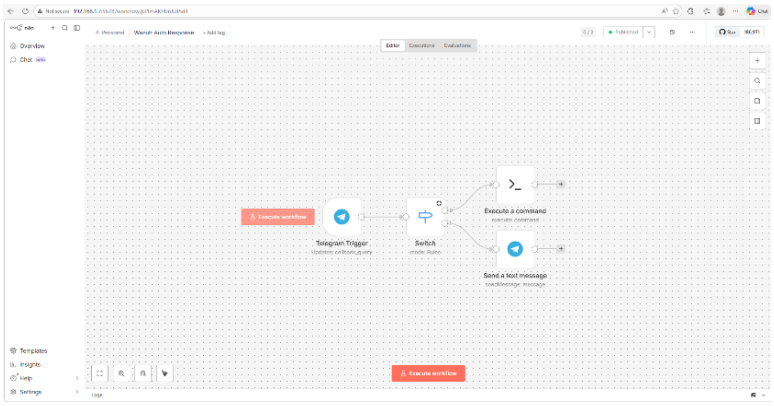

Fig. 6. n8n auto-response workflow

Fig. 6 shows the auto-response workflow that picks up once the incident response node finishes — this is the stage where the Approve and Reject buttons are generated and sent out.

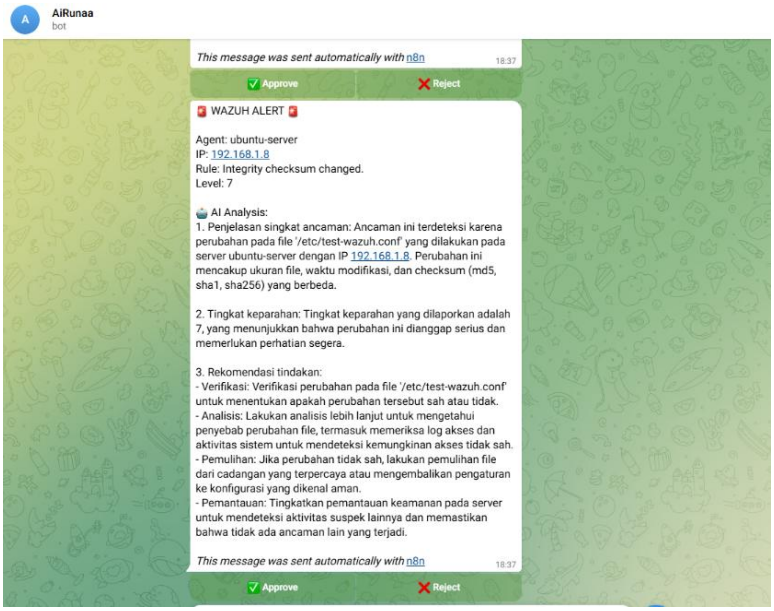

Fig. 7. Example Telegram notification with Approve/Reject buttons

Fig. 7 shows a sample notification as it appears to the analyst, complete with the AI-generated analysis, a severity rating, and a recommended action, everything needed to make an informed approve-or-reject decision in seconds rather than minutes.

3.2 Testing Results

3.2.1 Scenario 1: SSH Brute Force

The SSH brute force scenario used Hydra with a 10-password wordlist targeting the root account on VM2. Wazuh caught every single attempt across all 15 trials through rule ID 5710 at severity level 10. Full results are shown in Table 4.

Table 4. Scenario 1 Testing Results SSH Brute Force

No	Attack Time	Alert Time	MTTR (sec)	Condition
1	23:02:40	23:03:09	29	Saturday Night

2	23:03:50	23:04:07	17	Saturday Night
3	23:04:30	23:04:45	15	Saturday Night
4	23:05:30	23:05:55	25	Saturday Night
5	23:07:10	23:07:29	19	Saturday Night
6	14:30:10	14:30:32	22	Monday Afternoon
7	14:33:00	14:33:23	23	Monday Afternoon
8	14:40:05	14:40:29	24	Monday Afternoon
9	14:45:10	14:45:31	21	Monday Afternoon
10	14:48:15	14:48:30	15	Monday Afternoon
11	13:50:05	13:50:25	20	Tuesday Midday
12	13:52:00	13:52:22	22	Tuesday Midday
13	13:54:10	13:54:36	26	Tuesday Midday
14	13:57:00	13:57:24	24	Tuesday Midday
15	13:58:00	13:58:19	16	Tuesday Midday
Rata-rata			21,2	

As shown in Table 4, the system detected and notified within an average of 21.2 seconds, ranging from 15 to 29 seconds, indicating a fairly tight spread. This variation comes down more to how long Hydra itself takes to run through its password attempts than to anything happening on the network side, since Hydra needs roughly 10–15 seconds to work through 10 login attempts before Wazuh accumulates enough log activity to trigger rule 5710.

3.2.2 Scenario 2: Multiple Failed Login

This scenario used a Python script built with the Paramiko library to simulate five consecutive failed logins, each spaced three seconds apart. Wazuh picked up this pattern through rule ID 2502 at severity level 10. Results are shown in Table 5.

Table 5. Scenario 2 Testing Results Multiple Failed Login

No	Attack Time	Alert Time	MTTR (sec)	Condition
1	21:02:10	21:02:41	31	Saturday Night
2	21:04:30	21:05:08	38	Saturday Night
3	21:06:20	21:06:55	35	Saturday Night
4	21:08:00	21:08:36	36	Saturday Night
5	21:10:10	21:10:40	30	Saturday Night
6	15:01:05	15:01:40	35	Monday Afternoon
7	15:05:00	15:05:32	32	Monday Afternoon
8	15:08:00	15:08:34	34	Monday Afternoon
9	15:15:00	15:15:30	30	Monday Afternoon
10	15:25:00	15:25:36	36	Monday Afternoon
11	13:30:30	13:31:07	37	Tuesday Midday
12	13:35:10	13:35:42	32	Tuesday Midday
13	13:40:15	13:40:46	31	Tuesday Midday
14	13:44:30	13:45:08	38	Tuesday Midday
15	13:47:00	13:47:36	36	Tuesday Midday
Rata-rata			34,1	

As Table 5 shows, the average MTTR here came out to 34.07 seconds noticeably higher than Scenario 1. That gap makes sense given how the script was designed: the three-second pause built into each login attempt was meant to mimic a more realistic attack pattern, which pushed the script's total runtime to 15–20 seconds before Wazuh triggered an alert. What stands out, though, is how tight the range is just 30 to 38 seconds across all 15 trials, which says a lot about how consistently the system behaved.

3.2.3 Scenario 3: File Modification

This scenario involved manually modifying a configuration file inside the /etc/ directory monitored by Wazuh's File Integrity Monitoring (FIM). Unlike the first two scenarios, detection here depends entirely on FIM's scan cycle, which runs every 60 seconds. Wazuh flagged the change through rule ID 550 at severity level 7. Results are shown in Table 6.

Table 6. Scenario 3 Testing Results File Modification

No	Attack Time	Alert Time	MTTR (sec)	Condition
1	00:40:10	00:40:18	8	Sunday Early Morning
2	00:44:30	00:45:30	60	Sunday Early Morning
3	00:51:50	00:52:45	55	Sunday Early Morning
4	01:03:25	01:04:03	38	Sunday Early Morning
5	01:05:20	01:06:10	50	Sunday Early Morning
6	15:46:36	15:46:46	10	Monday Afternoon
7	15:47:05	15:47:45	40	Monday Afternoon
8	15:50:00	15:50:53	53	Monday Afternoon
9	15:51:30	15:51:55	25	Monday Afternoon
10	15:55:00	15:55:50	50	Monday Afternoon
11	13:00:00	13:00:10	10	Tuesday Midday
12	13:05:20	13:06:10	50	Tuesday Midday
13	13:10:25	13:11:04	39	Tuesday Midday
14	13:11:50	13:12:45	55	Tuesday Midday
15	13:15:10	13:15:18	8	Tuesday Midday
Rata-rata			36,73	

This scenario shows by far the widest spread of the three, anywhere from 8 to 60 seconds. That is not noise; it is simply how a periodic FIM cycle behaves. A file modified right as a new 60-second scan window opens gets caught almost instantly, as seen in trial 1 at just 8 seconds, while one modified right before the window closes ends up waiting nearly the full 60 seconds. This periodic nature of FIM is a known limitation worth factoring into the design of any Wazuh-based system [19].

3.3 MTTR Recap and Comparison

Table 7 brings together the hybrid system's MTTR across all three scenarios alongside an estimated MTTR for manual handling.

Table 7. MTTR Comparison Recap

Scenario	Avg Hybrid MTTR (sec)	Manual MTTR (sec)	MTTR Reduction (%)
Brute Force SSH	21.2	600	96.5
Multiple Failed Login	34.07	600	94.3
File Modification	36.73	1800	98

As Table 7 shows, the hybrid system cut MTTR significantly across every scenario tested. The 600-second manual estimate for Scenarios 1 and 2 assumes a mid-level system administrator checking authentication logs roughly every 10 minutes, in line with figures reported elsewhere [20]. The 1,800-second estimate for Scenario 3 assumes periodic file audits without any automated monitoring in place [12]. A reduction in the range of 94.3% to 98.0% holds up well against comparable work. Nguyen et al. [3] reported a 77.8% MTTR reduction through their AI4SOAR platform, and the results here land noticeably higher, even if the comparison has to be read carefully given the difference in testing conditions (discussed further below).

3.4 Consistency of AI-Generated Recommendations

Across all 45 trials spanning the three scenarios, the AI-generated recommendations matched the rule-based engine's classification 100% of the time. The recommendations were always relevant to the type of attack

detected for authentication-based incidents, the LLM consistently suggested blocking the offending IP, enabling two-factor authentication, and stepping up log monitoring, while for file modification incidents, it consistently recommended investigating the change and verifying whether the modification was legitimate. This kind of contextual, consistent output is generally attributed to LLMs being trained on large security-focused corpora, which lets them infer the nature of an attack from fairly short log descriptions alone [11].

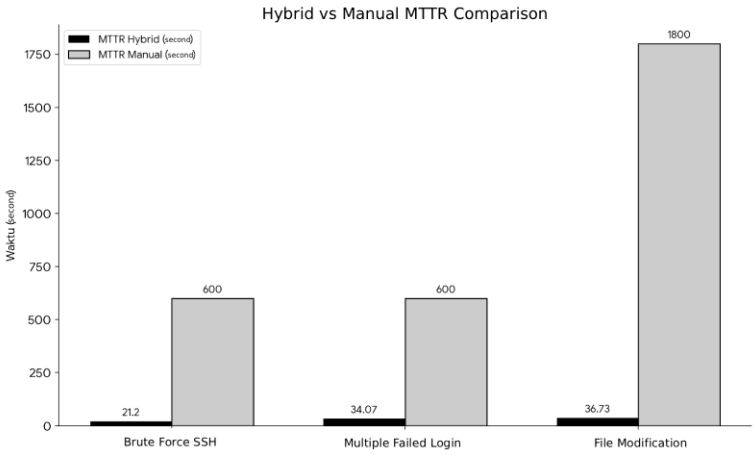

Fig. 8. Hybrid vs. manual MTTR comparison chart across the three scenarios

Fig. 8 lays out the gap between hybrid and manual MTTR side by side, making it clear just how much more efficient the hybrid system is compared to manual handling.

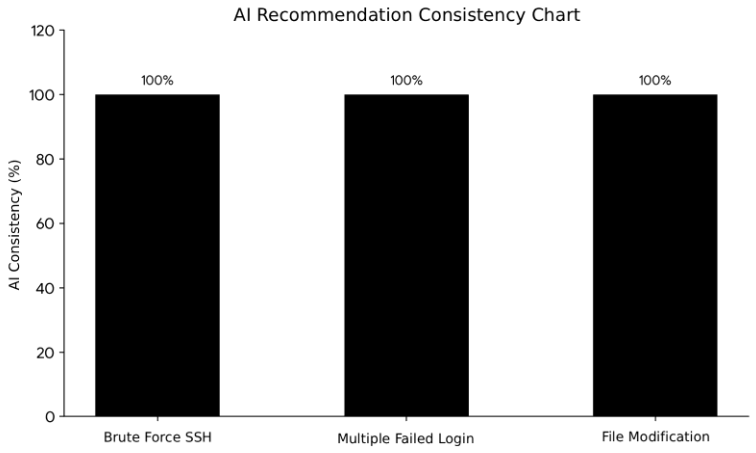

Fig. 9. AI recommendation consistency chart per scenario

Fig. 9 shows the consistency of AI-generated recommendations per scenario, with results staying relevant to the corresponding attack type throughout.

3.5 Discussion

Taken together, these results answer the core question this study set out to ask: can a system built from open-source, low-code, and AI components respond to incidents fast enough to matter, without removing the human from the decision? The answer, at least within this controlled environment, is yes. And the gap between 21–37 seconds of hybrid response time versus 600–1,800 seconds of manual handling is wide enough that it is hard to dismiss as a marginal improvement.

Compared to Ismail et al. [12], who combined Wazuh with an LLM through a Retrieval-Augmented Generation approach, this study takes a noticeably lighter path direct prompt engineering instead of a RAG pipeline with a vector database, paired with a rule-based engine and an explicit human-approval step that their setup does not include. The trade-off is reasonable: RAG can pull in deeper contextual knowledge, but it also adds infrastructure complexity that smaller organizations may not have the budget or staff to maintain. Compared to Nguyen et al.'s [3] AI4SOAR, the MTTR reduction achieved here is actually higher (94.3–98.0% versus 77.8%), though this needs to be read with some caution AI4SOAR was tested on an enterprise SOAR platform with presumably more complex, production-like traffic, while this study ran in a controlled lab setting with a narrower set of attack patterns. The comparison still tells us something useful, though: even a fully open-source, low-cost setup can hold its own against proprietary alternatives, at least at this scale.

A more conceptually similar effort is Özgözler et al.'s [21] Event-Driven Agentic SOC, which also proposes a supervisor-based LLM framework for dynamic incident response. The two systems share a similar philosophy using an LLM as a reasoning layer with a supervisory check before action, but differ in execution: this study leans on a no-code orchestration tool and a lightweight Telegram-based approval channel rather than a more heavyweight agentic framework, which arguably makes it easier for smaller teams to deploy without specialized engineering support.

The near-perfect consistency between the AI's recommendations and the rule-based engine is worth sitting with for a moment, too. It is tempting to read 100% as simply "the system always works," but a more useful reading is that the rule-based engine and the LLM were, in effect, checking each other's work and across every trial run here, neither contradicted the other. That agreement matters more than the raw percentage itself, because it suggests the two layers of judgment are reinforcing rather than working against one another, which is precisely the kind of redundancy a security system benefits from.

The widest variation observed in this study, the 8-to-60-second spread in Scenario 3 is not a flaw in the system itself but a structural property of how FIM operates on a periodic scan cycle. This is a fair point to flag honestly: a real-world deployment would need scanning intervals tuned to the sensitivity of what is being monitored, and this is a limitation this study does not resolve. Even so, this does not undercut the broader finding. The objective set out at the start of this research, building a system that responds meaningfully faster than manual handling while keeping a human in control of the final decision holds up clearly across all three scenarios tested.

4. Conclusions

This study set out to answer a fairly practical question: can a system built from open-source tools and a lightweight AI layer respond to security incidents fast enough to matter, without taking the human out of the loop entirely? Across all three attack scenarios tested, the answer holds up. The hybrid system, built around Wazuh, n8n, and Llama 3.1 8B accessed via the Groq API, with a Telegram-based approval step standing between detection and action, consistently detected and responded within seconds rather than minutes, cutting MTTR by 96.5%, 94.3%, and 98.0% across the SSH brute force, multiple failed login, and file modification scenarios respectively, compared to manual handling. Just as importantly, the AI-generated recommendations stayed fully aligned with the rule-based engine's classification throughout all 45 trials, which suggests the two layers of judgment were reinforcing each other rather than working at odds, a meaningful signal that this kind of hybrid reasoning can be trusted to support, rather than replace, an analyst's decision.

The scientific value of this work lies less in any single component and more in how the four pieces come together: an open-source SIEM, a no-code orchestration engine, an LLM-driven analysis layer, and a real-time human approval mechanism, combined into one coherent and measurable pipeline. That combination addresses a gap that earlier studies, taken individually, left open whether through missing a human-approval step, relying on proprietary infrastructure, or stopping at monitoring without any automated response layer. In practical terms, this points to a genuinely usable path for organizations that cannot afford enterprise-grade SOAR platforms but still need faster, more reliable incident response. Because every component here is either

open-source or free to use, the architecture is realistic to replicate, even for smaller teams or institutions working with limited budgets without forcing them to give up human oversight over actions that affect live infrastructure.

That said, this work was carried out in a controlled lab environment with a narrow set of attack scenarios, and that boundary is worth being upfront about rather than glossing over. Future research could extend this system with a broader range of automated response actions, such as network isolation or automatic file restoration, to handle a wider variety of incidents. Replacing the externally hosted LLM with a locally deployed model would also be worth exploring, both to reduce dependency on third-party APIs and to address potential data privacy concerns in more sensitive deployments. Testing the system in a real production environment, with the kind of complex, unpredictable traffic that a lab setup cannot fully replicate, would be a natural next step toward validating these findings at scale. Finally, the variation observed in the file modification scenario suggests that adapting the FIM scan frequency based on system activity levels rather than keeping it fixed could help smooth out response time further, and is a direction worth pursuing in its own right.

5. References

[1] T. T. Bukhari, O. Oladimeji, E. D. Etim, and J. O. Ajayi, "Systematic Review of SIEM Integration for Threat Detection and Log Correlation in AWS-Based Infrastructure," 2023. doi: 10.32628/SHISRRJ.

[2] C. Zhang, D. Jia, L. Wang, W. Wang, F. Liu, and A. Yang, "Comparative research on network intrusion detection methods based on machine learning," *Comput. Secur.*, 2022, doi: 10.1016/j.cose.2022.102861.

[3] M.-D. Nguyen, W. Mallouli, A. R. Cavalli, and E. M. de Oca, "AI4SOAR: A Security Intelligence Tool for Automated Incident Response," 2024. doi: 10.1145/3664476.3670450.

[4] J. M. López Velásquez, S. M. Martínez Monterrubio, L. E. Sánchez Crespo, and D. Garcia Rosado, "Systematic review of SIEM technology: SIEM-SC birth," *Int. J. Inf. Secur.*, 2023, doi: 10.1007/s10207-022-00657-9.

[5] J. Manzoor, A. Waleed, A. F. Jamali, and A. Masood, "Cybersecurity on a budget: Evaluating security and performance of open-source SIEM solutions for SMEs," *PLoS One*, 2024, doi: 10.1371/journal.pone.0301183.

[6] F. A. Saputra, T. R. Dharmawan, and A. Rustianto, "Implementasi Wazuh SIEM untuk Manajemen Log Event di Pesantren Teknologi Informasi dan Komunikasi Jombang," *Jurnal Informasi dan Teknologi*, 2024, doi: 10.54914/jit.v10i2.1435.

[7] M. Khayat, E. Barka, M. A. Serhani, F. Sallabi, K. Shuaib, and H. M. Khater, "Empowering Security Operation Center With Artificial Intelligence and Machine Learning-A Systematic Literature Review," *Applied Sciences*, 2025, doi: 10.3390/app13116610.

[8] S. Dwivedi, B. Rajendran, P. V Akshay, A. Acha, P. Ampatt, and S. D. Sudarsan, "IntelliSOAR: Intelligent Alert Enrichment Using Security Orchestration Automation and Response (SOAR)," 2024. doi: 10.1007/978-3-031-80020-7_27.

[9] H. Casaclang, B. Ionescu, Y. Kim, and J.-Y. Jo, "Self-Hosted Workflow Automation for AI-Based Cybersecurity Operation," *Journal of The Colloquium for Information Systems Security Education (JCISSE)*, vol. 13, no. 1, p. 21, 2026, doi: 10.53735/cisse.v13i1.241.

[10] J. Zhang *et al.*, "When LLMs meet cybersecurity: a systematic literature review," *Cybersecurity*, vol. 8, no. 1, p. 55, 2025, doi: 10.1186/s42400-025-00361-w.

[11] 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.

[12] Ismail *et al.*, "Enhancing Security Operations Center: Wazuh Security Event Response with Retrieval-Augmented-Generation-Driven Copilot," *Sensors*, vol. 25, no. 3, p. 870, 2025, doi: 10.3390/s25030870.

[13] U. Ahmed *et al.*, "Signature-based intrusion detection using machine learning and deep learning approaches empowered with fuzzy clustering," *Sci. Rep.*, vol. 15, p. 1726, 2025, doi: 10.1038/s41598-025-85866-7.

[14] S. Kumar, S. Datta, V. Singh, D. Datta, S. K. Singh, and R. Sharma, "Applications, Challenges, and Future Directions of Human-in-the-Loop Learning," *IEEE Access*, vol. 12, pp. 75735–75760, 2024, doi: 10.1109/ACCESS.2024.3401547.

[15] A. B. Pratomo, "Automation of Cyber Attack Mitigation on RouterOS Using n8n Orchestration Platform," *BUFNETS*, vol. 4, no. 1, pp. 1–5, 2026, doi: 10.59688/fq83hq56.

[16] N. Shenoy and A. V Mbaziira, "An Extended Review: LLM Prompt Engineering in Cyber Defense," in *2024 International Conference on Electrical, Computer and Energy Technologies (ICECET)*, Sydney, Australia, 2024, pp. 1–6. doi: 10.1109/ICECET61485.2024.10698605.

[17] K. Lazaros, A. G. Vrahatis, and S. Kotsiantis, "Human-in-the-Loop Artificial Intelligence: A Systematic Review of Concepts, Methods, and Applications," *Entropy*, vol. 28, no. 4, p. 377, 2026, doi: 10.3390/e28040377.

[18] S. Vethachalam, "Cybersecurity automation: Enhancing incident response and threat mitigation," *World Journal of Advanced Engineering Technology and Sciences*, 2025, doi: 10.30574/wjaets.2025.15.3.0972.

[19] F. Nova, M. D. Pratama, and D. Prayama, "Wazuh sebagai Log Event Management dan Deteksi Celah Keamanan pada Server dari Serangan DoS," *Jurnal Ilmu Teknik dan Sistem Informasi*, 2022, doi: 10.62527/jitsi.3.1.59.

[20] G. Ali, S. Shah, and M. ElAffendi, "Enhancing cybersecurity incident response: AI-driven optimization for strengthened advanced persistent threat detection," *Results in Engineering*, 2025, doi: 10.1016/j.rineng.2025.104078.

[21] E. Özgözler, A. Varol, and \.Ihsan Tanrverdi, "Event-Driven Agentic SOC (ED-ASOC): Supervisor-Based LLM Framework for Dynamic Incident Response and SOAR Orchestration," in *2026 14th International Symposium on Digital Forensics and Security (ISDFS)*, Boston, MA, USA, 2026, pp. 1–5. doi: 10.1109/ISDFS69419.2026.11458922.