Journal of *Cybersecurity and Privacy*

MDPI

Article

An Integrated Honeypot and LLM-Based Framework for near Real-Time Detection and Behavioral Analysis of Malicious Activities

Rúben Oliveira, Tiago Gomes, Daniel Pinho and Fernando Almeida *

School of Science and Technology, Institute of Polytechnic Studies of Gaya, Av. Dos Descobrimentos 333, 4400-103 Vila Nova de Gaia, Portugal; ispg2018101154@ispgaya.pt (R.O.); ispg2022101478@ispgaya.pt (T.G.); ispg2025108574@ispgaya.pt (D.P.)

* Correspondence: falmeida@ispgaya.pt

Abstract

Malicious activity detection in honeypot environments remains challenging due to the volume and heterogeneity of captured data, as well as the sequential nature of attacker behavior. This study proposes an integrated framework combining a Cowrie-based honeypot with a locally deployed Large Language Model (LLM) to enable automated detection and near-real-time behavioral analysis. Attacker interactions are captured and processed through a structured preprocessing stage that reconstructs session-level activity. These representations are analyzed using an LLM, allowing contextual interpretation of authentication patterns, command execution sequences, and post-compromise behavior. Structured analytical outputs are generated, including severity classification, reasoning, and recommended actions. Evaluation was conducted using isolated and concurrent attack scenarios. Results indicate effective identification of brute-force attacks, reconnaissance activity, persistence staging, and download-and-execute patterns, achieving a 94.23% Accuracy (95% CI: 84.05–98.79%), 100.0% Precision, 87.50% Recall, and an F1-score of 93.33% across an expanded evaluation of 52 independent observations, with zero false positives (FPs). These figures are derived from a single evaluation run and are reported as preliminary, proof-of-concept estimates rather than as stable, production-grade performance. Analytical output remained stable for sessions governed by deterministic severity overrides. However, a low-intensity multi-stage session produced inconsistent severity classifications under concurrent conditions, indicating that analyst review is still required for borderline cases. Integration of deterministic preprocessing with LLM-based reasoning enables the transformation of raw honeypot logs into structured and actionable cybersecurity intelligence, reducing analyst workload while improving the explainability and reliability of intrusion analysis in near-real-time environments.

Keywords: cybersecurity; honeypots; large language models (LLMs); intrusion detection; behavioral analysis; real-time threat detection; log analysis

Check for updates

Received: 22 June 2026
Revised: 20 August 2026
Accepted: 27 August 2026
Published: 31 August 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

With the rise in cyber threats, increasingly complex new challenges have begun to emerge for current computer systems, requiring defense mechanisms capable of dealing with more sophisticated and constantly evolving attack vectors. As attackers adopt more targeted and automated strategies, there is a growing need for adaptive security approaches that can detect malicious activity with the highest possible accuracy and respond to these

threats in near real time [1–3]. Honeypots, which are intentionally vulnerable systems designed to mimic legitimate characteristics, play an important role in observing the tactics, techniques, and procedures (TTPs) of attackers. Capturing data and information from interactions in the honeypot facilitates detailed examination of intrusion attempts and contributes to a deeper understanding of the opponent's behavior [4,5].

While their advantages are significant, modern honeypot deployments generate large volumes of heterogeneous data, including network traffic traces, authentication attempts, command execution logs, and session transcripts. For all their scale and diversity, the data make traditional manual or rule-based analysis inefficient, time-consuming, and prone to error, resulting in a critical analytical impasse in operational contexts [6,7]. According to Palma et al. [8], consequently, the actionable value of honeypots is compromised, as actionable intelligence is often extracted too late to support timely defensive measures.

Recent advances in artificial intelligence (AI), particularly in generative AI and LLMs, have enabled the processing of large volumes of unstructured security data, detection of anomalous patterns, correlation of events across sessions, and generation of structured summaries for cybersecurity operations [9,10]. As shown by Balogh et al. [11], they can also support log analysis and incident classification, enhancing the effectiveness and scalability of honeypot-based security monitoring.

Several research efforts have investigated the integration of generative AI within honeypot environments. One research direction focuses on enhancing interaction realism by leveraging LLMs to generate context-aware responses during Secure Shell (SSH) sessions or by emulating realistic network services. These approaches aim to increase attacker engagement and generate richer and more diverse behavioral data [12–14].

A second research direction emphasizes analytical capabilities, where LLMs are applied to process honeypot logs, summarize events, classify attack patterns, and identify evolving intrusion strategies that may not be easily detectable through conventional monitoring techniques [15,16].

Although these approaches address different aspects of honeypot systems, they are inherently interconnected. Interaction realism directly influences the quality, depth, and variability of the collected data, while analytical mechanisms determine how effectively that data can be transformed into actionable intelligence. However, existing studies typically consider these components independently, without explicitly addressing their combined role within a unified detection pipeline.

As a result, improvements in data generation are not systematically aligned with analytical processes, limiting the overall effectiveness of honeypot-based security solutions. Furthermore, many existing approaches rely on offline processing or cloud-based AI services, introducing latency, privacy concerns, and operational constraints. This fragmentation highlights the lack of integrated architectures capable of supporting both enhanced interaction and near-real-time, context-aware analysis within a single coherent framework.

As highlighted in recent studies, the field remains fragmented, and there is a lack of end-to-end architectures that integrate honeypots with locally deployed generative AI for real-time detection and behavioral analysis [17–19].

Aiming to mitigate these challenges, this study proposes and evaluates an integrated framework that combines a honeypot platform with a locally deployed LLM to support automated detection and near-real-time analysis of malicious activity. The proposed approach aims to (i) enhance early attack detection, (ii) characterize attacker behavior through interactive and sequential analysis, and (iii) automate the interpretation of large-scale honeypot logs. The integration of generative AI allows honeypots to move past purely passive data collection and support more intelligent and adaptive security analysis.

To guide this study, the following research questions are defined as follows:

RQ1: What is the impact of structured prompting and deterministic preprocessing on classification consistency and FP generation?

RQ2: How does the operational behavior of locally deployed LLMs influence model suitability for near-real-time honeypot log analysis?

RQ3: How accurately can the integrated framework detect and classify distinct malicious behavior categories under isolated and concurrent conditions?

RQ4: To what degree can the framework autonomously classify attacker sessions without human review, and in which scenarios does analyst intervention remain necessary?

Given the expanded evaluation set used in this study (N = 52 observations, comprising 24 malicious and 28 benign cases across nine malicious and five benign behavior categories; see Section 3.2), RQ3 and RQ4 are supported by a substantially larger and more diverse empirical base than in earlier stages of this work. Nonetheless, the study remains a controlled proof-of-concept conducted in an isolated laboratory environment, and further validation on a larger and more heterogeneous dataset, including internet-facing or production-like conditions, is recommended before broader generalization. Building on this expanded evaluation base, a controlled experimental environment was developed integrating a production-grade honeypot with a locally hosted LLM. The system was designed and evaluated to assess its capability to analyze attacks in real time and support behavioral profiling and automated threat interpretation, demonstrating its potential to strengthen proactive defense mechanisms and improve organizational cybersecurity posture.

2. Literature Review

This section reviews relevant research on honeypot technologies, AI for attack detection and log analysis, and the application of generative AI and LLMs in cybersecurity contexts, providing the foundation for the topics discussed in the following subsections.

2.1. Honeypot Technologies for Cyber Threat Intelligence

Honeypots are intentionally exposed systems designed to attract malicious actors, record hostile interactions, and support the study of attacker behavior in controlled environments [4,20,21]. Unlike conventional defensive mechanisms such as firewalls, intrusion detection systems (IDSs), and intrusion prevention systems (IPSs), which primarily aim to block or mitigate malicious activity, honeypots are designed as observational assets that intentionally capture TTPs used during intrusion attempts [3,22–24]. This capability makes them valuable for cyber threat intelligence, adversarial profiling, malware collection, and early warning activities [25,26].

According to Purnama and Prasetyo [27], honeypots are typically categorized into low-interaction, medium-interaction, and high-interaction systems. Low-interaction honeypots emulate limited services and are relatively safe to deploy, while high-interaction honeypots provide richer attacker engagement at the cost of higher operational risk [27]. Medium-interaction honeypots seek a balance between realism and containment, offering useful behavioral data while maintaining manageable security requirements, as discussed by Altunay [28]. This classification reflects the trade-off between system complexity, data richness, and operational risk, which is a key consideration when selecting an appropriate honeypot deployment strategy.

Among currently deployed platforms, Aljunid and Qaid [29] identify Cowrie as one of the most widely used open-source honeypots for SSH and Telnet environments. Cowrie emulates a Unix-like shell and records brute-force attempts, login credentials, command execution, file download attempts, session transcripts, and attacker interaction patterns [29]. These characteristics make it especially suitable for monitoring unauthorized remote access

attempts and command-line behavior associated with botnets, opportunistic intrusions, and automated scanning campaigns [28].

Although they provide significant operational value, honeypots generate large volumes of heterogeneous data according to Lanka et al. [3]. Honeypot deployments generate diverse data, including authentication records, source IP information, command logs, timestamps, session traces, and protocol-specific events [30]. As attack volume increases, manual analysis becomes increasingly difficult, often delaying the extraction of actionable intelligence. Consequently, the effectiveness of honeypots depends not only on their capacity to capture malicious behavior, but also on the availability of scalable analytical mechanisms capable of interpreting raw logs efficiently [5,31,32].

2.2. Artificial Intelligence for Attack Detection and Log Analysis

AI has become increasingly relevant in cybersecurity, particularly in intrusion detection, where machine learning (ML) and deep learning models are used to identify unauthorized access attempts and malicious behaviors [33]. It also supports anomaly detection by identifying previously unseen or evolving attack patterns without relying on predefined signatures. Sequence-aware models, such as long short-term memory (LSTM) and Transformer-based architectures, have demonstrated strong performance in detecting anomalous event sequences within security and authentication logs, as discussed by Kikissagbe and Adda [34]. In addition, AI techniques are also applied to malware classification, supporting the automated categorization of malicious software through behavioral analysis and system interaction patterns [35]. As Alwhbi et al. [36] state, learning-based approaches are widely used in network traffic analysis to identify malicious communication patterns, including scanning activities and command-and-control behavior. The authors of [37] explored the use of data-driven and ML approaches to assist incident triage, particularly by correlating heterogeneous security logs and supporting analysts in prioritizing and reviewing alerts. LLM approaches have shown promising results in scalable and explainable security log analysis.

In honeypot environments, AI-based methods are particularly valuable because they can process extensive attack telemetry and identify suspicious behaviors that may be difficult to detect manually. Traditional ML approaches have used supervised, semi-supervised, and unsupervised models to classify malicious activity from structured features extracted from security logs as described by Dini et al. [38]. Commonly observed indicators include failed authentication frequency, command sequences, session duration, source reputation, geographic anomalies, and repeated access patterns. These techniques have shown promising results in identifying brute-force attacks, credential stuffing, suspicious downloads, and lateral movement indicators [23,39].

Despite their effectiveness, classical ML approaches often depend on labelled datasets, handcrafted feature engineering, and stable data distributions. In real honeypot environments, attacker behavior evolves rapidly, and logs frequently contain noisy, incomplete, or unstructured information [40]. As stated by Rehman et al. [41], this limits the long-term adaptability of fixed classifiers and creates challenges when previously unseen attack strategies emerge.

Unlike these classical ML approaches, which typically require labelled datasets and handcrafted feature engineering to detect brute-force attacks, credential stuffing, or lateral movement indicators, the framework proposed in this study does not rely on a trained classifier or a fixed feature set. Instead, it combines deterministic behavioral rules with LLM-based contextual reasoning, enabling interpretation of attacker behavior without requiring prior model training on labelled honeypot data.

For SSH and Telnet honeypots such as Cowrie, detection tasks often involve understanding sequences of actions rather than isolated events [42]. Repeated failed logins followed by successful access, privilege escalation commands, network enumeration, or attempts to download remote payloads may only become meaningful when interpreted contextually. This requirement motivates the use of more flexible models that can analyze sequences of events, correlate actions within attack sessions, and process unstructured textual logs, instead of relying solely on static classification rules commonly used in traditional log analysis [43,44].

2.3. Generative AI and Large Language Models in Cybersecurity

Unlike conventional detection systems that usually output predefined labels or scores, LLMs can process natural language and semi-structured data, summarize events, infer relationships between actions, and generate human-readable explanations [45]. These characteristics are especially relevant in Security Operations Centers (SOCs), log review workflows, and threat intelligence analysis, as highlighted in [46,47].

In honeypot environments, LLMs can be applied to session transcripts and log streams in order to identify suspicious behavior, classify attacker intent, prioritize events, and automatically generate structured reports. The work in [48] demonstrates that a model may receive a sequence containing repeated SSH authentication attempts, command execution activity, download commands, or reconnaissance behavior, and then infer whether the observed activity resembles automated bot behavior, credential abuse, or a potentially high-risk intrusion attempt.

Generative AI may reduce analyst workload by transforming raw technical records into structured and interpretable security narratives. LLMs enable this transformation by extracting relevant entities from unstructured log data, including source IP addresses, authentication events, and executed commands, and by organizing these elements into coherent summaries.

Instead of manually reviewing large volumes of logs, analysts can rely on automatically generated reports that consolidate activity across sessions. Suspicious IP activity is identified through the aggregation of repeated interactions, while recurring behavioral patterns are detected by analyzing temporal sequences of commands within each session. Escalation attempts can be inferred by linking authentication events with subsequent high-privilege actions, allowing the identification of progression from initial access to post-compromise behavior.

These capabilities enable prioritization of critical events and support faster identification of potential threats, contributing to more efficient and scalable security analysis workflows [9].

Locally deployed LLMs are particularly relevant in cybersecurity because they reduce dependency on external cloud services. Local LLM inference can improve privacy by ensuring that sensitive operational data remains within the local environment, reduce latency, and support near-real-time analysis pipelines. These characteristics make such deployments especially suitable for security contexts where data confidentiality and timely response are critical [49,50].

Despite these advantages, generative AI, whether deployed locally or through cloud-based services, introduces important limitations. Outputs may vary according to prompt structure, certain interpretations may be overly generalized, and hallucinated conclusions may occur if insufficient contextual evidence is provided, as discussed by Anh-Hoang et al. [51].

These limitations can be mitigated, at least in part, through structured and modular prompt design. Rather than relying on a single unconstrained instruction, the system

prompt can be organized into discrete components that separately define the analytical role, the permissible severity vocabulary, the escalation rules, and the required output schema. This modular structure reduces the likelihood of persona breaks, in which the model departs from its intended analytical role and adopts an unconstrained conversational behavior, and constrains the generative process to remain anchored to the evidence provided rather than to unsupported inference [52]. The prompt design adopted in this study, described in Section 3.4, follows this principle by explicitly separating role definition, severity constraints, and output formatting within the system prompt. The complete, verbatim text of the grounded (combined) system prompt used to generate all reported results is provided in Box A2 (Appendix D). For this reason, LLM-based cybersecurity systems should be treated as decision-support mechanisms that augment, rather than replace, technical monitoring controls and expert validation.

2.4. Quantitative Evaluation Metrics and Research Gaps

The evaluation of IDS commonly relies on standardized performance metrics such as Accuracy, Precision, Recall, and F1-score [53–55]. Accuracy measures the overall proportion of correct classifications, Precision evaluates how many detected alerts correspond to real malicious activity, Recall measures how many true attacks were successfully identified, and the F1-score balances Precision and Recall. Additional indicators such as FP rate and threat response time are also relevant in operational environments, particularly where rapid response is required, as noted by Ashraf et al. and Sezgin et al. [56,57].

As shown by Díaz-Verdejo et al. [58], traditional signature-based systems such as Snort and Suricata remain widely used due to their efficiency in detecting known threats. Nevertheless, they may struggle with previously unseen attacks, obfuscated behaviors, and noisy environments that require frequent rule updates and manual tuning, a limitation also emphasized by Asad et al. [59]. In contrast, AI-enhanced systems may offer stronger adaptability by learning patterns from behavioral evidence rather than relying exclusively on predefined signatures.

Although previous research has separately explored honeypots, ML intrusion detection, and generative AI for cybersecurity support, these domains often remain disconnected. Rabzelj et al. [60] show that honeypot studies frequently emphasize data collection, while AI-based intrusion detection studies commonly rely on benchmark datasets rather than real-time honeypot telemetry [61]. Likewise, many LLM cybersecurity studies focus on summarization or assistant-style interactions instead of continuous malicious activity detection in operational honeypot pipelines, as reflected in the work of Jones et al. [62].

This reveals a relevant research gap: limited empirical evidence exists regarding end-to-end frameworks that combine a production-grade SSH/Telnet honeypot, real-time log acquisition, local Python-based preprocessing, and a locally hosted LLM capable of identifying suspicious IP activity and potentially severe attacker behavior in near real time [35,62,63].

Before describing the specific implementation adopted in this study, it is useful to establish the general architectural pattern underlying LLM-based honeypot systems. Such systems can be conceptualized as a three-tier data flow, in which a network listener component captures raw session-level interactions at the protocol level, a core logic component performs preprocessing, event correlation, and deterministic behavioral rule enforcement, and an LLM backend component performs contextual reasoning over the structured representations produced by the core logic layer [52]. This layered separation of concerns allows protocol-level data capture, deterministic preprocessing, and generative reasoning to be developed, validated, and scaled independently, while preserving a coherent analytical pipeline.

To address this gap, the present study proposes an integrated architecture that adopts this three-tier structure directly: Cowrie acts as the network listener, the Python-based preprocessing component performs the role of the core logic layer, and the locally deployed Mistral NeMo model, accessed through Ollama, constitutes the LLM backend, based on a Debian 13 virtual machine (VM) running the Cowrie honeypot [64,65]. A Python script continuously retrieves Cowrie logs in real time, preprocesses relevant events, and forwards structured data to a locally deployed Mistral NeMo model via Ollama [65]. The objective is to automatically analyze attacker activity, identify suspicious behavioral patterns, assess potentially serious situations, and transform raw honeypot telemetry into actionable cybersecurity intelligence, aligning with prior work on honeypot-based attacker tracking, LLM-supported incident analysis, and adaptive Cowrie-based honeypot systems [26,62,63].

3. Research Methodology

As noted earlier, this study aims to investigate the applicability of generative AI techniques for behavioral analysis in honeypot-based intrusion detection experiments. Cowrie was selected as the honeypot platform due to its ability to capture detailed attacker interactions at both command and session levels, providing sufficient information for behavioral analysis of SSH-based attacks [42]. Local LLM inference was selected to analyze collected logs to evaluate the feasibility of offline, privacy-preserving AI-assisted security analysis [6]. This section details the experimental environment, attack execution procedures, data collection process, preprocessing steps, and the methodology used to perform generative AI-based behavioral reasoning. Throughout this study, the primary unit of evaluation is observation, a distinct behavioral stage assessed by the pipeline (e.g., an authentication burst, a login event, a command sequence, or a file download), as formally defined in Section 4.2. References to "session", "IP address", or "attacking machine" elsewhere in the manuscript describe the source or context of an observation, not an alternative unit of analysis.

3.1. Experimental Environment and System Configuration

The experiments were conducted in a controlled laboratory environment composed of logically separated systems deployed across two distinct experimental phases, as illustrated in Figure 1. The core infrastructure comprised (i) a Debian-based VM (version 13.4.0) hosting the Cowrie honeypot (version 2.9.17), (ii) a local analysis host running the Ollama framework (version 0.32.1) for LLM inference, and (iii) attack-generating systems described below. The virtualized components were deployed using VMware Workstation (version 25.0.0), ensuring isolation and consistency across experimental runs. This multi-host configuration was designed to prevent unintended impact on external systems and support reproducibility. It should be noted that this setup constitutes a controlled, isolated laboratory environment rather than an internet-facing or production deployment; the results reported in this study therefore demonstrate feasibility under controlled conditions and should be regarded as a proof-of-concept rather than evidence of performance in operational, internet-facing settings.

The Cowrie honeypot was deployed on the Debian VM, configured to emulate SSH services commonly targeted during opportunistic attacks. The VM was isolated from production networks using firewall and network segmentation mechanisms, ensuring that all attacker interactions were fully contained within the experimental environment. The analysis host ran Ollama locally, providing on-device inference without reliance on external application programming interfaces (APIs) or cloud-based LLM services.

In the first experimental scenario, a single Kali Linux (version 2026.1) system was used as the authorized attacker platform, executing controlled attack scenarios against

the honeypot, including authentication attempts, interactive command execution, and post-compromise activity.

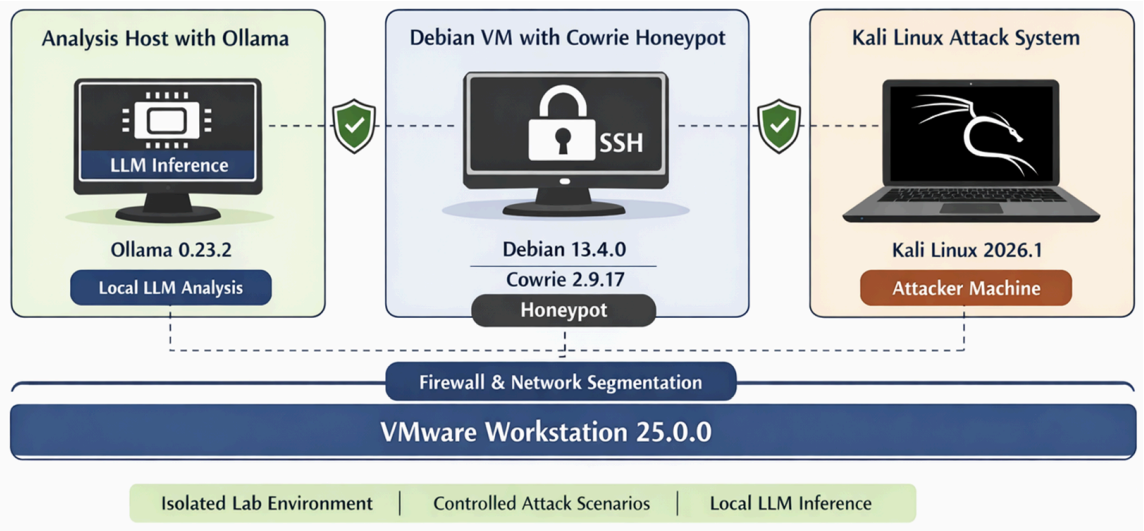

Figure 1. Experimental setup with analysis host, Cowrie honeypot, and attacker system in an isolated virtual environment.

In the second experimental scenario, three machines interacted with the honeypot simultaneously. Two Kali Linux (version 2026.1) systems were used as attacker platforms, each executing distinct attack profiles against the honeypot. A third machine running CachyOS (version 6.17) was used to simulate a legitimate user session, performing benign interactive activity without prior authentication failures. This multi-attacker configuration was designed to evaluate the framework's ability to maintain independent and coherent session analysis under concurrent heterogeneous load conditions.

3.2. Attack Scenarios and Data Acquisition

Two distinct experimental scenarios were designed and executed to evaluate the framework under different attack conditions.

In the first scenario, attack traffic was generated manually and semi-automatically from a single Kali Linux VM to simulate realistic malicious attack behavior. The techniques employed included failed login attempts, representing repeated authentication attempts with incorrect credentials typically associated with brute-force or guessing strategies, which were automated using the tool Hydra (version 9.6) over SSH with the rockyou wordlist. Successful login attempts were also observed, indicating that valid or compromised credentials were used to gain access to the system, similarly achieved through automated Hydra attacks once the correct credentials were identified, as illustrated in Figure A1 (Appendix A). Directory enumeration was performed to identify hidden files, directories, or endpoints that could expose sensitive information or weaknesses. The execution of reconnaissance commands allowed the attacker to gather system-level information, including users, network settings, and running processes. Privilege escalation attempts were conducted to obtain higher access rights and increase control over the compromised environment. Basic post-exploitation actions were then carried out, involving activities such as persistence establishment, file manipulation, and preparation for further malicious operations.

In the second scenario, three machines interacted with the honeypot simultaneously to evaluate the framework's ability to handle concurrent heterogeneous sessions. The first machine, running Kali Linux, executed a high-volume brute-force attack using a custom-generated wordlist of 200 passwords delivered via Hydra over SSH. Upon successful

authentication, the session proceeded to simulate post-compromise activity, including system reconnaissance and the execution of a download-and-execute pattern mimicking ransomware deployment. The second machine, also running Kali Linux, performed a low-volume brute-force attack with three credential attempts before gaining access, after which it conducted system reconnaissance, including enumeration of users, network configuration, and running processes, followed by the creation of a reverse shell script staged in /tmp and configured for persistence via a scheduled task. The third machine, running CachyOS, simulated a legitimate security audit by connecting directly without prior failed attempts, performing minimal system checks, and creating a text file in a user home directory before terminating the session cleanly.

Together, the two experimental scenarios generated diverse interaction patterns. These included brute-force authentication, characterized by repeated login attempts using multiple credentials. Post-compromise activity was also observed, involving actions performed after successful authentication, such as system exploration and command execution. In addition, multi-stage attack sequences were identified, where attackers progressed through multiple steps including access, reconnaissance, and preparation for further actions. Benign sessions were also observed, representing legitimate interactions without malicious intent under both isolated and concurrent conditions.

Building on these two initial scenarios, the evaluation dataset was further expanded across 24 additional experimental rounds to strengthen the diversity and statistical robustness of the assessment, yielding 52 independent observations (24 malicious, 28 benign). This expanded evaluation covered nine distinct malicious behavior categories, brute-force authentication, dropper deployment, reconnaissance/scanning, reverse shell establishment, cryptomining, data exfiltration, persistence establishment, credential theft, and abandoned/partial attack attempts (sessions terminated before completion, representing low-volume or incomplete malicious activity), together with five benign administrative activity categories: routine administration, system maintenance, package management, file transfer, and noisy-but-benign activity (elevated but non-malicious activity volume). Each category included multiple behavioral variants, executed across up to three hosts per round (two attacking machines and one benign administrative machine). Table 1 provides a clearer comparison between the two experimental setups and Table 2 summarizes the composition of this expanded evaluation set.

Table 1. Summary of the experimental scenarios, illustrating variations in attack complexity, session concurrency, and behavioral characteristics.

Aspect	Scenario 1: Single-Source Attack	Scenario 2: Concurrent Multi-Session
Setup	One Kali Linux VM	Three machines (2 attackers + 1 benign)
Attack Type	Manual and semi-automated	Fully concurrent and heterogeneous
Brute-force Activity	Yes (Hydra + rockyou wordlist)	Yes (high-volume and low-volume)
Successful Authentication	Yes (after brute force)	Yes (multiple sessions)
Post-Compromise Activity	Limited (reconnaissance, privilege escalation)	Extensive (reconnaissance, persistence, staging)
Multi-stage Attack Behavior	Present	Strongly present (full attack chains)
Malware Simulation	No	Yes (download-and-execute pattern)
Persistence Mechanisms	Basic	Reverse shell + scheduled task
Benign Activity	No	Yes (CachyOS audit session)
Concurrency	No	Yes (parallel sessions)
Objective	Baseline behavioral analysis	Evaluation of multi-session detection and differentiation

Table 2. Composition of the expanded evaluation dataset (N = 52) by behavior category.

Category	Label	Observations	Variants
Brute-force authentication	Malicious	5	A–E
Dropper deployment	Malicious	4	A–D
Reconnaissance/scanning	Malicious	2	A–B
Reverse shell establishment	Malicious	3	A–C
Cryptomining	Malicious	2	A–B
Data exfiltration	Malicious	2	A–B
Persistence establishment	Malicious	2	A–B
Credential theft	Malicious	2	A–B
Abandoned/partial attack	Malicious	2	A–B
Subtotal malicious		24	
Routine administration	Benign	9	A–D
System maintenance	Benign	4	A–C
Package management	Benign	5	A–C
File transfer	Benign	5	A–C
Noisy-but-benign activity	Benign	5	A–C
Subtotal benign		28	
Total		52	

Cowrie recorded all attacker activity in structured log formats, capturing authentication results, command sequences, file transfer attempts, timestamps, source network information, and session identifiers. Logs were generated locally on the honeypot VM and periodically transferred to the analysis host through a secure internal channel to prevent tampering and data loss. An example of the structured Cowrie log output generated during the experimental scenarios is provided in Figure A2 (Appendix B).

3.3. Session Reconstruction and Preprocessing

Raw Cowrie events were continuously ingested from the cowrie.json log stream and aggregated to preserve temporal ordering and execution context. Instead of analyzing individual log entries in isolation, the preprocessing component—implemented as a custom Python (version 3.13) real-time analyzer with the file name 'analyser_realtime.py'—reconstructs attacker activity by grouping events by source IP address and consolidating them into coherent session histories. This method enables each IP to be represented as a chronological narrative of interaction, capturing authentication attempts, interactive shell behavior, file activity, and session lifecycle information.

The analyzer maintains an in-memory per-IP cache that updates incrementally as new events arrive. Authentication behavior is summarized through counters of failed and successful login attempts, along with the usernames and passwords used during these attempts. Interactive behavior is preserved as a time-stamped command timeline, complemented by records of failed commands (e.g., permission errors or missing binaries). Session-level metadata is also retained, including connection/closure events, session durations, and basic client identifiers (e.g., SSH version strings and fingerprints). To avoid unbounded growth during long executions, inactive IP entries are periodically evicted after a fixed inactivity window.

Preprocessing was intentionally designed to be lightweight and deterministic, aiming to reduce noise while preserving semantic fidelity. Rather than relying on manual feature engineering, executed commands are annotated using transparent rule-based patterns that assign a coarse-grained intent label (e.g., reconnaissance, privilege escalation attempt, external download) and a conservative risk level (benign/low/medium/high/critical). In addition, when file transfer events are observed, the pipeline enriches file records with a

short content preview (or a compact binary fingerprint), providing interpretable context without transforming payloads into engineered features.

Finally, the preprocessor generates a structured JavaScript Object Notation (JSON) "digest" per IP that consolidates timeline statistics (e.g., activity duration and session counts), login behavior summaries, annotated command sequences, file activity, and network-related actions such as port-forwarding requests. A set of deterministic threat indicators is computed (e.g., success after repeated failures, download and execute patterns, external file retrieval, tunneling attempts), and rule-based severity overrides are applied to enforce consistent escalation for clearly malicious combinations of events. This standardized representation preserves heterogeneous interaction patterns and session-level context, while keeping the preprocessing stage reproducible and minimally biased for subsequent generative analysis. An example of the raw Cowrie SSH authentication events used as input for the session reconstruction and preprocessing phase is provided in Appendix B (Figure A2), together with the corresponding structured JSON digest produced by the preprocessing pipeline (Appendix B, Figure A3).

To support reproducibility, this section specifies the deterministic indicator logic and severity-override rules implemented in the preprocessing module. Threat indicators are computed from session-level counters (e.g., failed/successful login attempts, executed commands, file transfers) using fixed, auditable rules. Table 3 lists the volume- and behavior-based indicators most relevant to the severity logic, together with their exact trigger conditions.

Table 3. Key indicators and triggering conditions.

Indicator	Trigger Condition
HIGH_VOLUME_BRUTE_FORCE	failed login attempts >50
REPEATED_LOGIN_FAILURES	10 < failed login attempts $\leq$50
SUCCESS_AFTER_BRUTE_FORCE	$\geq$1 successful login and $\geq$1 failed login in the same session
SINGLE_ATTEMPT_ONLY	exactly 1 failed login, 0 successes
INTERACTIVE_SESSION_ACTIVE	$\geq$1 successful login and $\geq$1 command executed
SESSION_ABANDONED_BEFORE_LOGIN	session closed with 0 successes and $\geq$1 failure
HIGH_COMMAND_FAILURE_RATE	$\geq$4 total commands and $\geq$40% ended in error
MACHINE_PACED_LOGINS	machine-paced login timing and $\geq$3 failed attempts
CREDENTIAL_LIST_KNOWN_DEFAULTS	$\geq$3 passwords tried belong to a known default-credential list
DOWNLOAD_AND_EXECUTE_PATTERN	a file download is followed by execution of that file within the session
EXTERNAL_FILE_DOWNLOAD	a file download originates from an external URL (http/https/ftp)

Deterministic severity overrides are then applied on top of these indicators, following a fixed priority order (evaluated top-to-bottom; the first matching rule determines the forced severity, and no rule firing leaves the verdict to the LLM). Table 4 summarizes this hierarchy for the combined-mode configuration.

Table 4. Severity override rules (top-down priority).

Priority	Condition	Forced Severity	Recommended Action
1	DOWNLOAD_AND_EXECUTE_PATTERN	Critical	BLOCK_IP_IMMEDIATELY
2	REVERSE_SHELL_ATTEMPT	Critical	BLOCK_IP_IMMEDIATELY
3	SSH_KEY_PERSISTENCE	Critical	BLOCK_IP_IMMEDIATELY
4	CRYPTOMINING_INDICATORS	Critical	BLOCK_IP_IMMEDIATELY
5	DESTRUCTIVE_COMMAND	Critical	BLOCK_IP_IMMEDIATELY
6	SUCCESS_AFTER_BRUTE_FORCE + INTERACTIVE_SESSION_ACTIVE + failures $\geq$ 5	Critical	BLOCK_IP_IMMEDIATELY
7	SUCCESS_AFTER_BRUTE_FORCE + failures $\geq$ 10	Critical	BLOCK_IP_IMMEDIATELY
8	TUNNELING_ATTEMPTED + successes > 0	Critical	BLOCK_IP_IMMEDIATELY

Table 4. *Cont.*

Priority	Condition	Forced Severity	Recommended Action
9	EXTERNAL_FILE_DOWNLOAD + successes > 0	Critical	BLOCK_IP_IMMEDIATELY
10	DATA_EXFILTRATION_ATTEMPT + successes > 0	Critical	BLOCK_IP_IMMEDIATELY
11	PERSISTENCE_MECHANISM/ANTI_FORENSICS/ DEFENSE_EVASION_ATTEMPT/CREDENTIAL_ ACCESS_ATTEMPT + successes > 0	High Threat	BLOCK_IP_RECOMMENDED
12	SUCCESS_AFTER_BRUTE_FORCE + failures ≥ 5	High Threat	BLOCK_IP_RECOMMENDED
13	HIGH_VOLUME_BRUTE_FORCE (>50 failures)	High Threat	BLOCK_IP_RECOMMENDED
14	DANGEROUS_COMMANDS_EXECUTED + successes > 0	High Threat	BLOCK_IP_RECOMMENDED
15	SUCCESS_AFTER_BRUTE_FORCE + failures ≥ 3	Suspicious	Monitor closely
16	SUCCESS_AFTER_BRUTE_FORCE (failures = 1–2)	Low Suspicion	Monitor (likely typo)
17	failures > 10 (no login success)	Suspicious	Monitor
18	SSH_ALGO_ENUMERATION	Suspicious	Monitor
—	(no rule fires)	not forced	verdict left to the LLM

This ordering explains, for example, why a session exhibiting SUCCESS_AFTER_ BRUTE_FORCE with only one or two prior failures is capped at Low Suspicion (rule 16) even when subsequent activity is malicious, unless a higher-priority rule (e.g., persistence or data exfiltration, rules 6–11) also fires, a boundary condition discussed further in Section 4.5. A complete worked example of this conversion, from raw Cowrie events to the resulting JSON digest and forced severity, is provided in Appendix B (Figure A3).

3.4. Generative AI Analysis via Local LLM Inference

In this phase, the structured session representations generated during preprocessing were analyzed using a locally deployed LLM, specifically Mistral-Nemo (12.2B parameters, Q4_0 quantization), executed via Ollama version 0.32.1. Inference was performed with the following parameters: temperature = 0.0, top_p = 1.0, context window (num_ctx) = 8192 tokens, and random seed = 42, with three concurrent workers. The same context window (num_ctx = 8192 tokens) was applied uniformly to Qwen3:8B and Qwen3:30B–A3B in the comparative evaluation reported in Section 3.4 and Table 5, so that VRAM exhaustion and latency differences across models cannot be attributed to unequal context allocation. These parameters were kept fixed across all experimental runs to ensure consistency and reproducibility in the generated analytical outputs. Request handling used a timeout of 120 s, with up to two retry attempts (three attempts total) before falling back to the deterministic rule-based classification; across the logged evaluation runs, no failed calls or retries were recorded under the combined-mode configuration. This 120-s operational timeout is set generously to allow slower models to complete during evaluation; classification performance is nonetheless benchmarked throughout this study against a stricter 60-s production real-time budget (see Table 6), representing the target deadline for operational deployment. Model outputs were parsed via a fixed six-field regex parser (Severity, Confidence, Recommended Action, and three additional required fields), which also logged missing fields, invalid severity labels, and extraneous text as separate non-conformance indicators. No manual correction of model output was performed. This setup enables offline execution directly on the analysis host, supporting data confidentiality, independence from external dependencies, and reproducible experimental conditions across repeated runs.

The experimental evaluation and concurrent load testing were executed on a dedicated local deployment host equipped with an Intel Core i9-14900HX CPU, 32 GB of DDR5 RAM, and an NVIDIA GeForce RTX 4060 Laptop GPU with 8 GB of VRAM. This localized hardware baseline was selected to replicate a high-performance workstation configuration operating within strict edge-compute VRAM boundaries, directly influencing the inference

latency, offloading overhead, and resource exhaustion patterns observed during high-concurrency scenarios.

Table 5. Comparative evaluation of locally deployed LLMs based on operational performance and suitability for near-real-time analysis.

Aspect	Qwen3:8B	Mistral-Nemo 12B	Qwen3:30B–A3B
Failed/empty calls (grounded pipeline, thinking disabled)	0	0	0
Sessions analyzed	Same input logs as other models (r12, r06)	Same input logs	Same input logs
Reconstruction of attack chains	Complete	Complete	Complete (fallback parsing required for structured fields)
Latency/near-real-time adequacy	Adequate (p95 27.0 s)	Adequate (p95 39.2 s)	Inadequate—exceeds 60 s budget on 76.2% of calls (p95 90.5 s)
Schema adherence	100%	95.2%	0% (verbose free-prose output; fallback keyword recovery used)

Table 6. Operational profiles of the three candidate models under the grounded (combined) pipeline, evaluated on identical input logs (rounds r12 and r06).

Model	Median Latency	P95 Latency	Calls Over 60 s Budget	Tokens/s	Schema Adherence	Failed	Empty
Mistral-Nemo 12B	33.3 s	39.2 s	0%	14.9	95.2%	0	0
Qwen3:8B	25.5 s	27.0 s	0%	23.9	100%	0	0
Qwen3:30B-A3B	86.2 s	90.5 s	76.2%	17.2	0%	0	0
Model	Median latency	P95 latency	Calls over 60 s budget	Tokens/s	Schema adherence	Failed	Empty

Before selecting the final LLM-based analysis component, three locally deployed models were tested through the Ollama framework. This preliminary comparison was not intended as a formal benchmark, but rather as an operational assessment of the suitability of each model for near-real-time honeypot log analysis. The evaluation considered response stability, the ability to handle concurrent requests, contextual consistency, the level of detail in the generated reports, and computational feasibility within the available experimental environment. All three models were evaluated against the same replayed Cowrie capture log, under the same concurrency configuration (three workers), to ensure that observed differences in stability and output quality reflect model behavior rather than variation in attacker activity. The main observations from this comparison are summarized in Table 5.

A call was counted as a failed evaluation only after all retry attempts were exhausted without a valid response; these cases are logged separately from calls that returned a response but failed schema validation (missing fields, invalid severity label, or extraneous text), the latter being treated as truncated or degraded rather than failed. Sessions affected by either outcome were excluded from the per-model attack-chain reconstruction reported in Table 5 rather than being scored as false positives or false negatives.

Latency reflects the model's own inference time, excluding queueing delay. "Over 60 s budget" is the share of responses that would miss the pipeline's real-time deadline. Schema adherence is the share of responses that fill every required field with no extraneous prose. In the LLM-only (ungrounded) arm, all three models are slower and less schema conformant: Mistral-Nemo's P95 latency rises to 120 s (16.7% over budget) with schema adherence falling to 69.4%; Qwen3:8B remains fastest (P95 56.2 s, 7.1% over budget, 64.3% adherence); Qwen3:30B-A3B remains over budget on 64.3% of calls with 0% adherence.

The operational profiles documented in Tables 5 and 6 provide empirical justification for the final architectural configuration. Under identical input logs and with Qwen3's thinking mode disabled, all three models complete every call successfully (0 failed, 0 empty calls); the earlier characterization of Qwen3's degraded output as generic "instability under concurrency" is therefore revised to its precise, reproducible cause, the model's thinking mode consuming the fixed output-token budget on hidden reasoning, an effect that disappears entirely once thinking mode is disabled. Two genuine limitations of Qwen3:30B-A3B are structural rather than transient: it exceeds the 60-s real-time latency budget on 76.2% of calls given the 8 GB VRAM constraint of the deployment hardware, forcing CPU offloading, and it produces verbose free-form reasoning that fails to conform to the required structured output schema even with thinking mode disabled, requiring unreliable keyword-based fallback parsing to recover a verdict. Based on this operational comparison, Mistral-Nemo 12B was selected as the final model used in the experimental evaluation. Consequently, the detection results and performance tables reported in the following sections refer to the output generated by the Mistral-Nemo 12B-based analysis pipeline.

Each structured JSON digest, representing a consolidated view of attacker activity per source IP, was provided as input to the LLM. Rather than processing isolated events, the model was tasked with interpreting complete session narratives, allowing it to reason over temporal sequences of commands, authentication patterns, and associated contextual metadata. This design enables higher-level behavioral interpretation beyond rule-based detection, capturing relationships between actions that span multiple stages of an attack.

The LLM performs structured analytical reasoning guided by a predefined system prompt that emulates the behavior of an experienced cybersecurity analyst. The prompt explicitly defines the expected analytical posture (e.g., cautious, non-alarmist, and context-aware) and constrains the reasoning process through a set of deterministic rules and severity guidelines. It enforces a structured severity model (Normal to Critical), defines mandatory escalation conditions (e.g., successful login after repeated failures or download and execute patterns), and restricts overinterpretation of benign actions such as simple navigation commands or isolated failed logins. Furthermore, the prompt specifies a strict output format consisting of severity classification, summary, reasoning, assessment, recommended action, and confidence level, ensuring consistency across all analyses.

This prompt design plays a critical role in stabilizing LLM behavior by reducing variability and aligning responses with domain-specific expectations. By combining rule-based constraints with contextual interpretation, the LLM is guided to produce consistent, explainable assessments while avoiding unsupported or overly speculative conclusions. Additionally, deterministic inputs produced during preprocessing, such as annotated command timelines, file content previews, and precomputed threat indicators, are explicitly referenced within the prompt, ensuring that generative reasoning remains anchored to structured and verifiable evidence.

To maintain consistency and control over the analytical process, the inference pipeline is executed locally without reliance on external APIs or dynamically updated models. This approach decouples the evaluation from external service variability, ensuring that all experimental results are generated under stable and reproducible conditions. Moreover, deterministic severity constraints derived during preprocessing (e.g., forced severity levels and predefined response actions) are preserved and enforced during LLM interpretation, preventing inconsistent classification in clearly defined attack scenarios.

The output of the LLM is a structured security assessment comprising severity classification, concise summarization, reasoning, and recommended actions, as illustrated in Figure A4 (Appendix C). This standardized format facilitates downstream analysis, log-

ging, and comparison across sessions, enabling consistent interpretation of heterogeneous interaction patterns while complementing the deterministic preprocessing stage.

3.5. Experimental Workflow

The experimental workflow followed a structured and repeatable pipeline designed to ensure consistency across all evaluation scenarios, with the overall process illustrated in Figure 2.

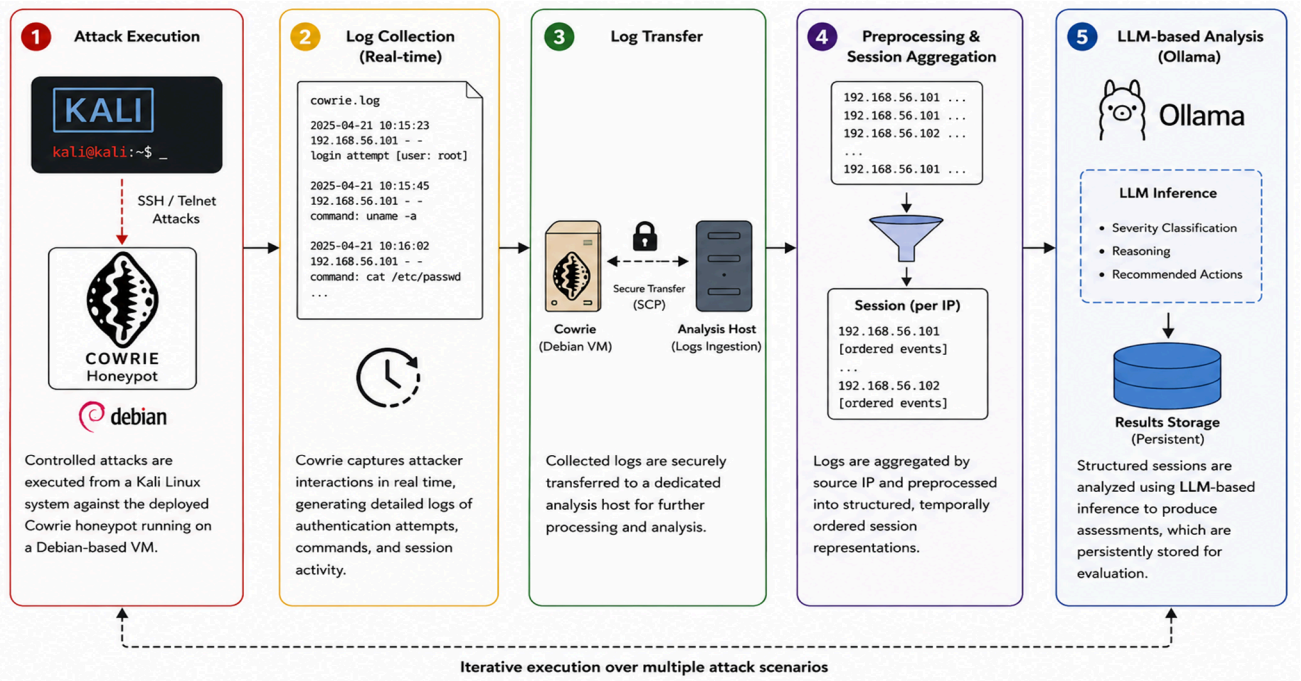

Figure 2. Experimental workflow showing the full pipeline, including attack execution, real-time log collection, secure transfer, preprocessing and session aggregation, and final LLM-based analysis using Ollama.

The process begins with the execution of controlled attack scenarios from a Kali Linux system targeting the deployed Cowrie honeypot. Attacker interactions are captured in real time by Cowrie running on a Debian-based VM, generating detailed log records of authentication attempts, command execution, and session activity.

The collected logs are then securely transferred to a dedicated analysis host, where they undergo session-based aggregation and structured preprocessing. This stage transforms raw log entries into coherent, per-IP session representations that preserve temporal ordering and contextual relationships between events.

Subsequently, the structured representations are analyzed using LLM-based inference via the locally deployed Ollama framework. This phase produces high-level behavioral assessments, including severity classification, reasoning, and recommended actions, based on both deterministic indicators and contextual interpretation.

Finally, all generated analytical outputs are persistently stored, enabling systematic comparison and evaluation across different attack sessions. The entire workflow is executed iteratively over multiple independent attack scenarios to assess the consistency, stability, and interpretability of the analytical results under varying interaction patterns.

3.6. Validity, Reproducibility, and Ethical Considerations

All experiments in this work were conducted in isolated, non-production environments using controlled and authorized attack simulations. No real user data, production

systems, or external third-party infrastructures were involved at any stage of the evaluation. Network activity was strictly confined to the experimental environment, and outbound connectivity was restricted to prevent unintended interactions beyond the honeypot setup.

The implemented methodology emphasizes reproducibility through the use of open-source components, fixed system configurations, deterministic preprocessing steps, and locally executed LLM inference. These design choices ensure that the experimental pipeline can be consistently reproduced under identical conditions. Although generative models may exhibit minor variations in output phrasing, the controlled inference setup, combined with deterministic inputs and constrained prompting, ensures that analytical outcomes remain stable and comparable across repeated executions.

Certain limitations should be acknowledged. The evaluation depends on the behavioral realism of the simulated attack scenarios, which, while designed to approximate real-world activity, may not fully capture the diversity and unpredictability of live threat environments. In addition, the interpretative nature of generative output introduces an element of subjectivity, particularly in borderline cases where behavioral signals are ambiguous.

However, this study focuses on structured behavioral interpretation rather than automated enforcement or real-time response. As such, the impact of potential misclassification or FP is mitigated, as the generated outputs are intended for analytical and research-oriented purposes rather than direct operational decision-making.

3.7. Evaluation Metrics

Following the implementation described in the previous Section 3, this section presents the evaluation criteria used to analyze the results obtained from the generative AI-assisted honeypot experiments. As the implemented approach enables behavioral analysis in near real time during ongoing honeypot interactions, while also performing comprehensive post-session reasoning over complete attack traces, the evaluation metrics were selected to reflect analytical accuracy, consistency, responsiveness, and operational relevance, while allowing contextual comparison with IDS paradigms.

The analysis is grounded in standard confusion matrix components commonly used in intrusion detection research, namely true positives (TPs), true negatives (TNs), false positives (FPs), and false negatives (FNs). In the context of this study, TP corresponds to attack sessions in which malicious behavior was correctly identified during generative analysis; TN represents benign or low-risk interactions correctly interpreted as non-malicious; FP denotes benign interactions incorrectly interpreted as malicious; and FN refers to attack sessions in which malicious intent was not identified.

From these values, widely adopted performance indicators were used to structure the evaluation. The overall Accuracy measures the proportion of correctly interpreted sessions and is defined as follows:

$$\text{Accuracy} = \frac{TP + TN}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Precision quantifies the proportion of correctly detected attacks among all events classified as attacks:

$$\text{Precision} = \frac{TP}{\text{TP} + \text{FP}}$$

Recall (Sensitivity) measures the proportion of actual attacks that are correctly detected:

$$\text{Recall} = \frac{TP}{\text{TP} + \text{FN}}$$

F1-score represents the harmonic mean of Precision and Recall, balancing detection accuracy and coverage:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

To complement these standard indicators, two additional statistical metrics are incorporated to provide a more rigorous quantitative assessment of classification performance. The Matthews correlation coefficient (MCC) offers a balanced evaluation measure particularly suited to datasets with unequal class distributions, and is defined as follows:

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

MCC produces a value between -1 and +1, where +1 indicates perfect classification, 0 represents chance-level performance, and -1 denotes total disagreement between predictions and ground truth. Cohen's Kappa coefficient (κ) measures the agreement between the system's classifications and the ground truth, correcting for agreement expected by chance alone, and is defined as follows:

$$\kappa = \frac{P_o - P_e}{1 - P_e}$$

where P_o represents the observed agreement rate and P_e the expected agreement rate under random classification. Values of κ between 0.61 and 0.80 are generally interpreted as substantial agreement, while values above 0.80 indicate near-perfect agreement.

These metrics are not used to assess automated enforcement or blocking actions, but instead to evaluate the analytical reliability and consistency of the behavioral interpretations produced by the generative AI model. In contrast to traditional IDSs, which typically rely on real-time packet-level inspection and predefined signatures, the implemented methodology performs semantic reasoning over attacker activity sequences observed within honeypot sessions. Behavioral analysis is performed both during active sessions, in near real time as interactions occur, and after session completion, enabling complete reconstruction and interpretation of attack progression.

To assess temporal responsiveness, detection latency was considered as an evaluation indicator. In this case, detection latency represents the time elapsed between the initiation of malicious activity within a session and the point at which the malicious behavior is identified by the analysis pipeline. When averaged across multiple sessions, this metric is expressed as mean time to detection (MTTD) and provides an indication of how promptly malicious behavior can be identified to support security monitoring and incident response activities. Formally, MTTD can be defined as follows:

$$MTTD = \frac{1}{N} \Sigma \left(t_{detection_i} - t_{attack_{start_i}} \right)$$

where N represents the total number of analyzed sessions, $t_{detection_i}$ denotes the time at which malicious activity is identified in session i, and $t_{attack_{start_i}}$ corresponds to the initial moment of the attack within the same session.

MTTD is particularly relevant in the context of this study because the framework operates in a near-real-time analysis pipeline, where the time between attack initiation and detection directly determines the window of opportunity available to an attacker. In honeypot-based detection scenarios, shorter detection latency enables earlier containment, reduces the risk of post-compromise activity going unobserved, and improves the practical utility of the system in operational SOC environments. As noted by Ashraf et al. [55],

response time is a critical operational indicator in intrusion detection research, particularly where near-real-time performance is a stated design objective.

Metrics based on receiver operating characteristic (ROC) curves and area under the curve (AUC) are commonly associated with supervised classification approaches. However, because the present study does not rely on fully labeled datasets or fixed decision thresholds, these metrics are not used as primary quantitative measures. Instead, they are discussed conceptually to highlight the differences between traditional classification-based IDS evaluation and the hybrid near-real-time and session-level interpretative analysis adopted in this work.

The selected evaluation metrics align with the implemented experimental methodology and provide a structured basis for interpreting the results presented in the following sections. They further enable qualitative and semi-quantitative comparison with conventional signature-based IDS approaches. On the other hand, generative AI-enhanced analysis evaluated in this study highlights contextual reasoning and behavioral progression across entire interaction sequences, motivating the use of evaluation perspectives specifically suited to session-aware behavioral interpretation.

4. Results and Discussion

In this section, the results are reported, and the observations obtained from the experiments are discussed.

4.1. Report Results

The empirical security reports and live classification sequences presented in this section lay the descriptive foundation for evaluating prompt consistency (RQ1), cross-category detection performance (RQ3), and the overall boundaries of autonomous threat profiling (RQ4).

The experimental results demonstrate that the proposed framework is capable of capturing, interpreting, and classifying attacker behavior in near real time. The analysis is based on monitored honeypot events and the corresponding outputs generated by the LLM-based reasoning module, following structured evaluation approaches commonly adopted in intrusion detection studies.

During the initial single-machine experiment, multiple authentication scenarios were analyzed, ranging from isolated login failures to successful authentication following brute-force activity. Table 7 summarizes the observed event types, triggered indicators, and resulting severity classifications. The successful-login scenario was deliberately capped at Low Suspicion, reflecting the framework's conservative interpretation, which weighs a plausible benign explanation (e.g., user error) against the rule-based SUCCESS_AFTER_BRUTE_FORCE indicator.

Table 7. Classification of authentication scenarios observed during the single-machine experiment.

Cenário	Failed Attempts	Cowrie Event	Indicator Triggered	Severity Classification
Isolated authentication error	3	login.failed	None	Low Suspicion
Repeated authentication failures	25	login.failed	REPEATED_LOGIN_FAILURES	High Threat
Successful login after brute force	2 (prior failed attempts)	login.success	SUCCESS_AFTER_BRUTE_FORCE	Low Suspicion

During the initial single-machine experiment, multiple authentication scenarios were analyzed. In cases involving a limited number of failed login attempts (e.g., three consecutive failures), events were classified as 'login.failed' and assigned a Low Suspicion severity level. The absence of escalation indicators, such as successful authentication, persistence, or post-exploitation activity, enabled the system to distinguish between isolated errors and coordinated attack attempts, thereby reducing FPs. As the number of failed attempts increased (e.g., 25 attempts), the system identified repeated failure patterns and triggered the REPEATED_LOGIN_FAILURES indicator, escalating the severity level accordingly. This progression illustrates the framework's ability to detect evolving threat behavior based on activity patterns rather than isolated events.

In a second scenario, a successful login following prior failed attempts was identified as a 'login.success' event and initially classified as Low Suspicion. The system's reasoning indicated a potential benign explanation, such as user error, while concurrently flagging the SUCCESS_AFTER_BRUTE_FORCE indicator. This behavior reflects the hybrid nature of the framework, combining conservative interpretation with rule-based risk indicators to balance FP reduction with threat awareness.

It is worth noting that this conservative capping behavior is consistent with the pattern observed among the false negatives recorded in the full evaluation. All three FNs involve low-volume or incompletely executed attack activity: two abandoned brute-force attempts and one partially executed reconnaissance scan, where the deterministic escalation logic similarly favors a benign interpretation over premature escalation. This illustrates a concrete limitation of the current severity logic: low-volume but potentially malicious activity can be under-detected because the deterministic indicators are tuned to avoid FPs at the cost of sensitivity to borderline cases. Refining the balance between these competing indicators is identified as a direction for future work. To further assess the framework under concurrent conditions, a second experiment was conducted involving three simultaneous attack scenarios originating from distinct sources. The first scenario consisted of a high-volume brute-force attack followed by a download-and-execute sequence simulating ransomware deployment. The second scenario involved a low-volume brute-force attack followed by reconnaissance and persistence staging, including the creation of a reverse shell script for sustained access. The third scenario represented a benign session involving direct login and file creation activity consistent with a legitimate audit.

The framework reliably differentiated between the high-volume ransomware profile and the benign session in near real time; the low-volume multi-stage profile, however, was not classified consistently. The ransomware scenario was progressively escalated from Low Suspicion to Critical, with the final classification reflecting the combined presence of DOWNLOAD_AND_EXECUTE_PATTERN, EXTERNAL_FILE_DOWNLOAD, and HIGH_VOLUME_BRUTE_FORCE indicators. The recommended action BLOCK_IP_IMMEDIATELY was triggered upon detection of the malware execution pattern. The reconnaissance and persistence scenario was initially classified as Low Suspicion. As privilege escalation attempts, sensitive file access, and staged executable creation (reverse shell script in/tmp and a scheduled task for persistence) were observed, individual command events were intermittently escalated to High Threat; this escalation was not stable, and the standing assessment for the session ultimately regressed to a Low Suspicion/Medium (Monitor) verdict. Because the deterministic SUCCESS_AFTER_BRUTE_FORCE override remained capped at Low Suspicion for this low-volume session, the persistence activity did not yield a consistent malicious classification, and this session is therefore treated as an FN. In contrast, the benign session was consistently classified as Normal, with no FP escalation despite active interaction.

The majority of outputs followed a standardized structure, including severity level, summary, reasoning, assessment, recommended action, and confidence score, which improves interpretability. However, under concurrent load, several assessments (notably for the low-volume multi-stage session) deviated from the prescribed schema, producing free-form narratives, numbered breakdowns, or raw JSON blocks and omitting mandatory fields such as Summary and Confidence. Output formatting was therefore largely, but not fully, consistent across concurrent sessions. To quantify this, we measured schema adherence across all logged model replies: in the combined configuration, 98.1% of replies (667/680) fully conformed to the expected structured output schema, with zero missing fields and zero invalid severity values recorded; 1.9% of replies (13/680) contained extraneous prose beyond the structured fields, and no calls failed outright. For comparison, in the LLM-only configuration, evaluated without deterministic preprocessing support, conformance dropped to 80.9% (110/136), with 150 missing-field instances recorded and 25 cases (18.4%) requiring fallback to rule-based classification. This gap indicates that deterministic preprocessing contributes materially to output stability, independent of its role in detection accuracy, and confirms that schema-adherence checking remains necessary before downstream automated consumption of raw LLM outputs.

These results indicate that the proposed framework is capable of continuously monitoring attacker activity, aggregating events into coherent session-level interpretations, and distinguishing between benign behavior and escalating threats. The integration of deterministic preprocessing with generative AI-based reasoning enables the transformation of raw honeypot logs into actionable cybersecurity intelligence, providing enhanced contextual awareness compared to traditional signature-based approaches.

A representative example of the system output during live execution is provided in Figure A4 (Appendix C), illustrating how raw Cowrie honeypot events are transformed into structured security reports comprising severity classification, contextual reasoning, and recommended actions.

These results demonstrate the ability of the proposed framework to detect and differentiate multiple categories of attacker behavior in real-time scenarios, directly addressing RQ3.

4.2. Metrics Results

This section analyzes the quantitative benchmarks derived from the experimental MTTD observations, providing the statistical evidence required to evaluate concurrent model suitability (RQ2), multi-session classification accuracy (RQ3), and the autonomous metrics of the detection pipeline (RQ4).

To quantitatively evaluate the effectiveness of the proposed detection approach, a set of standard intrusion detection performance metrics was applied, as defined in Section 3.7, comprising Accuracy, Precision, Recall, and F1-score, derived from the confusion matrix components' TPs, TNs, FPs, and FNs.

Given that the experimental methodology is based on controlled honeypot interactions rather than a pre-labeled benchmark dataset, a simplified ground truth model was defined to classify observed sessions into benign or malicious categories based on deterministic behavioral rules. It should be noted that these are the same deterministic rules embedded in the preprocessing pipeline that generate the severity indicators subsequently interpreted by the LLM; consequently, the quantitative metrics reported in this section (Accuracy, Precision, Recall, MCC, and Cohen's κ) measure the degree of agreement between the LLM's classifications and this deterministic reference, rather than detection performance validated against an independently established ground truth. This limitation is further discussed in Section 4.5 and directly motivates the baseline comparison proposed as future work. Ses-

sions were labeled as malicious when they exhibited clear attack patterns, such as repeated login attempts exceeding a defined threshold, successful authentication following multiple failed attempts, post-login reconnaissance, malware staging, or download-and-execute activity. Conversely, sessions were considered benign when they involved a small number of isolated authentication errors without escalation and no post-login malicious behavior.

The outputs generated by the LLM-based analytical module were mapped to binary classification labels. Outputs classified as Suspicious or higher were interpreted as malicious predictions, while outputs classified as Low Suspicion or Normal were interpreted as benign predictions. Each analyzed observation was then categorized into one of the four confusion matrix outcomes, as summarized in Table 13, while the detailed scenario-level classification is presented in Table 8. The unit of analysis is the observation, defined as a distinct behavioral stage assessed by the pipeline (for example, an authentication burst, a successful login, an interactive command sequence, or a file download), rather than a physical attacking machine; consequently, a single attacker session may contribute several observations corresponding to successive stages of the same attack chain. Because the pipeline emits an assessment per event, a session can produce a sequence of severity verdicts that evolve over time. To obtain a single label per observation for the confusion matrix, the scoring rule adopted in this study is the final, standing severity verdict assigned by the pipeline at the end of the corresponding activity, rather than the maximum severity transiently emitted during it. This rule reflects the standing assessment an analyst would act upon in a near-real-time setting. It also determines the classification of the low-volume reconnaissance-and-persistence session, whose severity was transiently raised to High Threat but ultimately regressed to a Monitor-level verdict: under the final-verdict rule, this observation is recorded as an FN. This boundary condition is revisited in the Future Work section, where a refinement of the preprocessing logic is proposed to prevent such regressions in severity for low-volume but clearly adversarial activity.

The evaluation covered scenarios from both the initial single-machine experiment and the subsequent concurrent three-machine experiment, encompassing a broader range of behavioral profiles including reconnaissance, persistence staging, and simulated ransomware deployment.

Specifically, the severity scale comprises five ordered levels. Normal denotes interactions with no triggered risk indicators, such as a clean login without prior failures or routine file operations consistent with legitimate use. Low Suspicion is assigned to isolated anomalies that fall below escalation thresholds, including a small number of failed login attempts or a successful login following very few prior failures, where a benign explanation (e.g., user error) remains plausible. Suspicious marks the first deterministic escalation tier, triggered when a moderate-volume pattern such as REPEATED_LOGIN_FAILURES is detected, indicating likely automated probing without confirmed compromise. High Threat applies to sustained or high-volume brute-force activity, or to post-login reconnaissance and privilege-escalation behavior, reflecting active and ongoing malicious intent. Critical is reserved for sessions in which deterministic severity overrides are enforced due to unambiguous compromise indicators, including successful authentication after high-volume brute force, an active post-login session, download-and-execute patterns, or external file retrieval; this tier triggers the BLOCK_IP_IMMEDIATELY recommended action. The classification of all observed scenarios from the initial experimental design is summarized in Table 8.

The classification results indicate that the system correctly identified all sustained and high-intensity attack patterns. Brute-force activity, post-login reconnaissance, malware staging, and download-and-execute behavior were consistently classified at appropriate severity levels, resulting in a high number of TPs, as shown in Table 13. The benign audit

session from the concurrent experiment produced no FP escalation, remaining correctly classified as Normal, which is reflected in the absence of FPs (FP = 0) in the confusion matrix. Two FNs were recorded in this initial experiment: one for the scenario involving a successful login after a small number of failed attempts (classified as Low Suspicion), and another for the low-volume brute-force session followed by reconnaissance and persistence staging (discussed above), as shown in Table 8 and aggregated in Table 13.

Table 8. Classification of the observed scenarios.

Scenario	Ground Truth	Predicted Class	Outcome
Clean login with no prior failures	Benign	Normal	TN
Few failed login attempts	Benign	Low Suspicion	TN
Direct login followed by benign file creation	Benign	Normal	TN
Successful login after two failed attempts	Malicious	Low Suspicion	FN
Repeated login failures with 25 attempts	Malicious	High Threat	TP
Repeated login failures with 50 attempts	Malicious	High Threat	TP
High-volume brute force with 100 attempts	Malicious	High Threat	TP
High-volume brute force with 200 attempts	Malicious	High Threat	TP
High-volume brute force followed by successful login	Malicious	Critical	TP
High-volume brute force followed by active session	Malicious	Critical	TP
High-volume brute force followed by download-and-execute pattern	Malicious	Critical	TP
Low-volume brute force followed by reconnaissance and persistence staging	Malicious	Low Suspicion (unstable; final: Monitor)	FN
High-volume brute force followed by external file download	Malicious	Critical	TP
Successful login after high-volume brute force (isolated scenario)	Malicious	Critical	TP
Post-login command execution after high-volume brute force (isolated scenario)	Malicious	Critical	TP

To assess the sensitivity of the reported metrics to the scoring convention, Table 13 reports the confusion matrix and derived metrics under both the final-verdict convention used elsewhere in this study and a peak-severity convention, in which each observation is scored by the highest severity reached at any point during the session. For the combined pipeline and the rules-only baseline, both conventions yield identical results, because the deterministic forced-severity floor already prevents the reported verdict from regressing below evidence-justified escalations. For the LLM-only arm, evaluated without this deterministic floor, the two conventions diverge substantially, indicating that the scoring

convention materially affects reported performance when the deterministic component is absent.

To isolate the contribution of the LLM component from the deterministic preprocessing and severity-override logic, we compare the rules-only baseline directly against the combined pipeline in Table 9. The rules-only configuration achieves identical detection performance to the combined pipeline (TP = 21, TN = 28, FP = 0, FN = 3; Accuracy 94.23%), while the LLM-only configuration performs substantially worse (Accuracy 55.77% under the final-verdict convention) and diverges further under the peak-severity convention. This indicates that, in the current implementation, final classification outcomes are primarily driven by the deterministic rule and severity-override layer, while the LLM's independent classification value, absent this scaffolding, is currently limited. The complete, verbatim text of the LLM-only (ungrounded) system prompt used in this ablation baseline is provided in Box A2 (Appendix D).

Table 9. Detection performance under final-verdict and peak-severity scoring conventions, across the combined, rules-only, and LLM-only arms.

Mode	Convention	N	TP	TN	FP	FN	Accuracy	Precision	Recall
Combined	Final	52	21	28	0	3	94.23%	100.0%	87.50%
Combined	Peak	52	21	28	0	3	94.23%	100.0%	87.50%
Rules-only	Final	52	21	28	0	3	94.23%	100.0%	87.50%
Rules-only	Peak	52	21	28	0	3	94.23%	100.0%	87.50%
LLM-only	Final	52	22	7	21	2	55.77%	51.16%	91.67%
LLM-only	Peak	52	22	0	28	2	42.31%	44.00%	91.67%

To characterize this processing latency in more detail, Table 10 reports per-stage latency measurements for the combined pipeline (N = 29 measurements), including mean, median, and 90th/95th/99th percentile values. Furthermore, the comparative results presented in Table 11 reinforce the operational findings reported in Table 5 and provide quantitative grounding for model selection. Mistral-Nemo 12B, evaluated on an expanded set of 52 independent observations, achieved an Accuracy of 94.23% (95% CI: 84.05–98.79%), Precision of 100.0%, Recall of 87.50%, F1-score of 93.33%, MCC of 0.8890, and Cohen's κ of 0.8829, the latter indicating near-perfect agreement beyond chance. Critically, it maintained zero concurrent session failures and the lowest MTTD values (approximately 25 s to first malicious classification and 4 min 28 s to Critical escalation). It should be noted that the former reflects the pipeline's processing latency from event ingestion to classification output, whereas the latter reflects the chronological duration of the attack's kill chain within the concurrent scenario rather than analytical latency; only the first measure should be interpreted as detection latency in the strict sense. This distinction confirms the pipeline's suitability for near-real-time deployment.

Table 10. Per-stage pipeline latency (combined configuration, $N = 29$).

Stage	Mean	Median	P90	P95	P99	Max
Preprocessing	0.27 ms	0.22 ms	0.48 ms	0.50 ms	0.57 ms	0.60 ms
Queueing	12,676 ms	1194 ms	43,751 ms	51,010 ms	60,960 ms	63,380 ms
LLM inference	24,099 ms	26,756 ms	36,007 ms	37,979 ms	38,773 ms	39,004 ms
End-to-end	36,776 ms	28,832 ms	77,189 ms	83,029 ms	89,490 ms	91,823 ms
Attack progression (reference)	8.41 s	11.98 s	14.95 s	16.23 s	17.65 s	18.04 s
Stage	Mean	Median	P90	P95	P99	Max

Table 11. Performance metrics of the proposed approach.

Metric	Mistral-Nemo 12B	Qwen3:8B	Qwen3:30B-A3B
Accuracy	94.23%	83.3%	93.33%
Precision	100.0%	100.0%	100.0%
Recall	87.5%	72.7%	87.5%
F1-score	93.33%	84.2%	93.3%
MCC	0.889	0.640	0.875
Cohen's κ	0.8829	0.603	0.760
MTTD (first malicious)	~25 s	~48 s	~1 m 35 s
MTTD (Critical)	~4 m 28 s	~6 m 12 s	~7 m 41 s

Preprocessing latency is negligible relative to the other stages. Queueing delay is the dominant and most variable component under concurrent load, reflecting contention when multiple sessions require classification simultaneously. Notably, at higher percentiles (P95, P99), end-to-end latency (83.0 s, 89.5 s) exceeds the corresponding percentiles of attack progression time (16.2 s, 17.7 s), indicating that near-real-time detection cannot be guaranteed under sustained concurrent load, even though median-case latency (28.8 s) remains below typical attack progression time. This is discussed further as a limitation in Section 4.5.

Qwen3:8B, despite achieving 100% Precision, exhibited the lowest Recall (72.7%) and F1-score (84.2%), a degradation directly attributable to the ReadTimeout failures and incomplete session reconstructions documented in Table 5, which produced missed detections rather than FPs. Qwen3:30B-A3B yielded intermediate results (Accuracy 93.33%, F1-score 93.3%, MCC 0.875), analytically comparable to Mistral-Nemo 12B under stable conditions, but its VRAM footprint forced extensive CPU offloading, resulting in the highest MTTD values and frequent truncated outputs under sustained concurrent load. Collectively, these results confirm that model selection for near-real-time honeypot analysis cannot be guided by parameter scale or isolated benchmark scores alone; concurrency robustness, hardware feasibility, and latency under operational load are equally determinant factors.

To further isolate the role of the deterministic grounding from the choice of model, all three models were additionally evaluated on identical captured logs (rounds r12 and r06), under both the grounded (combined) pipeline and an ungrounded LLM-only configuration. Table 12 reports the resulting per-unit severity verdicts against the ground truth.

Table 12. Per-unit severity verdicts vs. ground truth, for grounded and ungrounded configurations (rounds r12 and r06).

Mode	Round	Unit	Ground truth	Mistral-Nemo	Qwen3:8B	Qwen3:30B-A3B
Grounded	r12	benign (package management)	benign	Normal	Normal	Normal (fallback)
Grounded	r06	attacker A (dropper)	malicious	Critical	Critical	Critical (fallback)
Grounded	r06	attacker B (revshell)	malicious	Critical	Critical	Critical (fallback)
Grounded	r06	benign (admin)	benign	Normal	Low Suspicion	Normal (fallback)
Ungrounded	r12	benign (package management)	benign	High Threat	Suspicious	Normal (fallback)
Ungrounded	r06	attacker A (dropper)	malicious	Critical	Critical	Critical (fallback)
Ungrounded	r06	attacker B (revshell)	malicious	High Threat	Suspicious	Critical (fallback)
Ungrounded	r06	benign (admin)	benign	High Threat	Suspicious	Normal (fallback)

On identical inputs, the three models diverge sharply once the deterministic grounding is removed. Inside the grounded pipeline, all three reach the correct verdict on nearly every unit; the one exception is Qwen3:8B rating a benign administrator "Low Suspicion" rather than "Normal", confirming that the digest and the forced-severity floor, not the model, carry the classification. Stripped of that grounding, the models fail in characteristic ways: Mistral-Nemo over-escalates, flagging both benign units (a package-management session and an administrative session) as High Threat; Qwen3:8B is less alarmist but still over-flags benign activity and under-rates one live attacker; and Qwen3:30B-A3B, even with thinking mode disabled, reasons in free-prose essays that never fill the output schema, so its verdicts survive only through keyword-based fallback recovery. The operational and analytical evidence point the same way: within the grounded pipeline, a small local model is sufficient, and the three models are close to interchangeable, whereas the 30B model is both too slow for the real-time budget on the available hardware and structurally unable to reliably hold the required output format.

We retain Mistral-Nemo 12B as the model behind the full study for four reasons. First, the complete 52-observation evaluation, every detection metric, the ablation study, the latency breakdown, and the schema-adherence measures were run on it, so it is the model the reported evidence rests on; the three-model comparison here is a deliberately small, controlled slice (two rounds, four source IPs). Second, the difference between Mistral-Nemo and Qwen3:8B is small and drawn from that small sample: both models answer well within the 60-s budget, so Qwen3:8B's roughly eight-second median-latency advantage does not change whether the system meets its deadline, and on identical logs the two models return the same benign/malicious verdicts, differing by a single step on the severity scale on one benign unit. Third, Mistral-Nemo requires no special configuration, whereas Qwen3 produced usable structured output only after its thinking mode was explicitly disabled; a deployed, unattended sentinel is safer with a model that behaves correctly under default settings. Fourth, in the ungrounded LLM-only arm, the two models' default judgment differs in a way that matters for security: Mistral-Nemo errs toward over-flagging, while Qwen3:8B rated a live reverse-shell attacker as only "Suspicious"; a false alarm is a milder failure than a missed intrusion, making Mistral-Nemo's conservative bias the safer choice. Qwen3:8B is therefore reported as an equally viable, marginally faster alternative rather than as a replacement, consistent with this section's main finding that the deterministic grounding, not the choice of small model, drives detection performance.

Table 13. Confusion matrix results.

	Predicted Benign	**Predicted Malicious**
Actual Malicious	FN = 3	TP = 21
Actual Benign	TN = 28	FP = 0

Given the sample size, exact (Clopper–Pearson) 95% confidence intervals were computed for each metric to characterize the statistical uncertainty of these estimates, as reported in Table 14.

The obtained results demonstrate that the proposed framework achieves a high level of detection effectiveness, correctly identifying the majority of malicious scenarios while maintaining perfect precision. In particular, the absence of FPs across 28 benign observations yields a specificity of 100% (95% CI: 87.66–100%), providing stronger preliminary evidence for the classification strategy's specificity than the earlier 3-observation sample; nonetheless, continued evaluation in a larger and more diverse benign population remains warranted before the false-positive rate can be considered fully established. However, three FNs were observed, all associated with abandoned or incompletely executed attack

attempts: two abandoned brute-force sessions and one partially executed reconnaissance scan. This indicates that low-intensity or borderline attack patterns may not always trigger sufficient indicators to be classified as malicious, reflecting the conservative decision boundaries embedded in the system.

Table 14. Detection performance with 95% confidence intervals (Mistral-Nemo 12B, combined configuration, N = 52; TP = 21, TN = 28, FP = 0, FN = 3).

Metric	Value	95% CI
Accuracy	94.23%	[84.05%, 98.79%]
Precision	100%	[83.89%, 100%]
Recall	87.50%	[67.64%, 97.34%]
Specificity	100%	[87.66%, 100%]
F1-score	93.33%	[84.44%, 100%]
MCC	0.889	[0.763, 1.000]
Cohen's κ	0.883	[0.736, 1.000]

From a performance perspective, the framework exhibits strong Precision and competitive Recall, resulting in a balanced F1-score. The detection capability is especially pronounced in high-frequency and sustained attack scenarios, such as brute-force attempts and post-compromise behaviors, where consistent TP identification is achieved. These findings confirm that the integration of deterministic preprocessing with LLM-based contextual reasoning provides an effective and explainable mechanism for interpreting attack patterns in honeypot environments.

A notable operational difference was observed between the evaluated models under concurrent execution conditions, consistent with the systematic comparison reported in Section 3.4 (Tables 5 and 6). Both Mistral-Nemo 12B and Qwen3:8B handled all parallel requests successfully within the grounded pipeline, with no failed or empty calls; Qwen3:30B-A3B, by contrast, exceeded the 60-s real-time budget on the majority of calls and failed to produce schema-conformant output regardless of workload, a structural limitation attributable to its VRAM footprint and verbose reasoning rather than to concurrency instability. This behavior highlights the impact of model architecture and hardware feasibility on the reliability of real-time analytical pipelines and supports the selection of Mistral-Nemo 12B for near-real-time deployment, for the reasons detailed in Sections 3.4 and 4 (model-comparison subsection). These results demonstrate that model selection significantly affects stability, response consistency, and feasibility in near-real-time cybersecurity analysis, thereby providing direct evidence for RQ2. The observed performance indicates reliable identification of malicious activity while minimizing unnecessary alerts, supporting the applicability of the framework for practical intrusion analysis and decision support.

The obtained performance metrics further confirm the effectiveness of the framework in accurately detecting malicious activity, providing quantitative support for RQ3. Furthermore, the zero FP rate and the κ of 0.8829 provide direct quantitative support for RQ1, confirming the stabilizing role of structured prompting and deterministic preprocessing.

To assess whether classification consistency (RQ1) could be confounded by inherent stochasticity in LLM inference rather than by the deterministic grounding and prompt design, each of the 52 captured logs was replayed five times under identical input and inference parameters (temperature = 0.0, top_p = 1.0, seed = 42). Across-run means and standard deviations were computed for the reported detection metrics. Verdict instability, defined as the number of observations for which the assigned severity level differed across any of the five runs, was 0/52: no observation changed its final verdict across repeated runs. This confirms that, under the fixed configuration used throughout this study, model stochasticity is not a confounding factor in the reported classification-consistency results,

and that the stability observed (RQ1) is attributable to the deterministic grounding and structured prompting rather than to a lack of genuine variability in the underlying model.

To isolate the contribution of the LLM component from the deterministic preprocessing and severity-override logic, we conducted an ablation study comparing three configurations: rules-only (deterministic preprocessing without LLM classification), LLM-only (LLM classification without deterministic preprocessing or severity overrides), and the combined pipeline. The rules-only configuration achieved identical detection performance to the combined pipeline (TP = 21, TN = 28, FP = 0, FN = 3; Accuracy 94.2%), while the LLM-only configuration performed substantially worse (TP = 22, TN = 7, FP = 21, FN = 2; Accuracy 55.8%) and showed markedly lower schema adherence (80.9% vs. 98.1%). This suggests that the final classification is mainly determined by the predefined rules and severity-override mechanism, while the LLM has a more limited role when used without this supporting framework.

4.3. Interpretive Analysis of Research Questions

This section provides a structured interpretation of the experimental results in relation to the research questions defined in Section 1. Table 15 maps the quantitative evidence obtained from the evaluation to each research question, enabling systematic assessment of the degree to which each question is empirically addressed by the proposed framework.

The interpretive analysis confirms that the proposed framework addresses all four research questions with supporting quantitative evidence. RQ3 is the most fully supported, with comprehensive confusion matrix data and category-level classification outcomes. RQ1 and RQ4 are addressed through derived metrics—precision and autonomous classification rate—that emerge directly from the experimental results. RQ2 is supported by operational observations under concurrent load, which, while not constituting a formal benchmark, provide empirically grounded evidence for model selection decisions. The additional statistical indicators MCC = 0.889 and $\kappa = 0.883$ confirm that classification performance exceeds chance-level agreement, strengthening the quantitative grounding of the results beyond standard metrics alone. It should be noted, however, that the evaluation scale (N = 52) remains modest relative to production-scale intrusion detection benchmarks, and findings should be interpreted within the scope of the controlled experimental conditions. Collectively, these findings demonstrate that the integration of deterministic preprocessing with locally deployed LLM-based reasoning constitutes an effective and interpretable approach to near-real-time honeypot security analysis.

Table 15. Interpretive alignment between research questions, quantitative evidence, and empirical findings.

RQ	Core Question	Supporting Evidence	Key Metrics	Result	Interpretation
RQ1	Impact of structured prompting and deterministic preprocessing on classification consistency	Deterministic severity rules; fixed output schema; rule-based escalation conditions	Precision; FP rate; κ	Precision = 100%; FP = 0; κ = 0.8829 (substantial agreement)	Structured prompting with deterministic constraints was associated with zero FPs and largely consistent output structure across the evaluated sessions (N = 52), suggesting, under these controlled, small-scale conditions, that prompt design and deterministic constraints contribute to output stability. This should be confirmed in a larger and more diverse sample before being generalized.
RQ2	Influence of LLM operational behavior on model suitability for near-real-time analysis	Comparative operational assessment under concurrent load (Table 5); timeout failures in Qwen3:8B; stable processing by Mistral-Nemo 12B	Concurrent stability; timeout rate; analytical consistency	Qwen3:8B: repeated ReadTimeout failures; Mistral-Nemo 12B: 0 failures across all concurrent sessions; Qwen3:30B-A3B: stable but computationally unsuitable for near-real-time deployment in the available setup	Concurrency robustness and hardware feasibility are determinant factors for model selection. Mistral-Nemo 12B provided the best operational balance for near-real-time deployment.
RQ3	Detection accuracy across distinct malicious behavior categories under isolated and concurrent conditions	Confusion matrix from 52 observations across both experimental scenarios (Table 13)	Accuracy; Precision; Recall; F1; MCC	Accuracy = 94.23%; Precision = 100.0%; Recall = 87.50%; F1 = 93.33%; MCC = 0.8890; TP = 21; TN = 28; FP = 0; FN = 3	An MCC of 0.8890 indicates classification performance above chance agreement. All high-intensity attack patterns were correctly identified, but the low-volume multi-stage session was not classified consistently. Three FNs occurred in low-volume or partial-attack scenarios where behavioral evidence was insufficient to trigger stable deterministic escalation.
RQ4	Degree of autonomous session classification without human review	Observation outcomes mapped to autonomous/review-required categories (Table 8); FN distribution by attack intensity	Autonomous classification rate; κ; FN rate	49/52 observations (94.23%) correctly classified; κ = 0.8829; three FNs (5.77%) in low-intensity scenarios; FP = 0	The framework processed all 52 observations without routing any to human review; 49 were classified correctly, and 3 (all involving low-volume or partial attack activity) were misclassified as benign. These three FNs, automatically assigned Normal or Low Suspicion classifications, highlight the need for analyst validation in borderline cases and confirm the framework's role as a decision-support mechanism rather than a fully autonomous system.

4.4. Discussion

The results of this study demonstrate that the integration of deterministic honeypot preprocessing with locally deployed LLM-based reasoning can support near-real-time behavioral analysis of malicious activity in SSH honeypot environments. The achieved Accuracy of 94.23% (95% CI: 84.05–98.79%), Precision of 100.0%, Recall of 87.50%, and F1-score of 93.33% indicate that the proposed framework distinguishes malicious from benign sessions reliably for high-intensity activity, while low-intensity multi-stage behavior is less consistently captured. The additional statistical indicators, MCC of 0.8890 and Cohen's κ of 0.8829, confirm that classification performance exceeds chance-level agreement, providing stronger statistical grounding beyond standard metrics alone. From a temporal perspective, two distinct measurements were obtained from different experiments. In the single-source scenario, the time to first malicious classification was approximately 25 s, reflecting the detection latency of the pipeline. In the concurrent scenario, the time to the first Critical severity escalation was approximately 4 min and 28 s; this figure reflects the time to reach the critical stage of the attack and is dominated by the time required for the higher-volume brute-force attempt to succeed, rather than by analytical latency. Interpreted accordingly, the first value indicates that emerging threats are flagged within operationally relevant timeframes, supporting timely analyst intervention before post-compromise activity progresses undetected, while the second characterizes attack progression rather than pipeline responsiveness.

Particularly relevant is the absence of FPs, since excessive FP generation remains one of the main limitations of operational intrusion detection, alert triage, and analyst trust in automated security monitoring systems. In this sense, the findings confirm previous studies suggesting that AI-supported security analysis can improve the scalability and interpretability of log-based intrusion detection, particularly when large volumes of heterogeneous security data must be transformed into actionable intelligence [8,11].

The findings also confirm prior research on the value of honeypots as cyber threat intelligence instruments. Previous studies have characterized honeypots as intentionally exposed systems that enable the observation of hostile interactions, attacker TTPs, and post-intrusion behavior [4,20,21]. The present study supports this perspective by showing that Cowrie-generated telemetry can capture authentication attempts, command execution, file activity, and session-level attacker behavior in a structured and analyzable form. This is also consistent with previous work identifying Cowrie as a suitable SSH/Telnet honeypot for recording brute-force attempts, credentials, executed commands, file download attempts, and attacker interaction patterns [29]. However, the contribution of the present framework goes beyond passive telemetry collection. Rather than treating the Cowrie logs as isolated events, the proposed approach reconstructs attacker behavior over time and transforms fragmented log entries into coherent session-level narratives. This enables the LLM to interpret malicious progression across authentication attempts, command sequences, reconnaissance activity, persistence staging, and download-and-execute behavior.

The results further confirm previous work on LLM-supported cybersecurity analysis. Prior studies have suggested that LLMs can support security log classification, incident interpretation, summarization, and analyst-oriented reporting [8,11,15,16]. The present findings are consistent with this line of research, since the framework generated structured outputs containing severity classification, summary, reasoning, assessment, recommended actions, and confidence levels. These outputs are relevant because they do not merely assign binary labels to observed activity; instead, they provide contextual explanations that help clarify why a session was classified as normal, suspicious, high-risk, or critical. Therefore, the findings reinforce the view that LLMs are particularly useful when

detection must be combined with interpretability, contextual reasoning, and operationally meaningful reporting.

At the same time, the findings partially diverge from overly optimistic interpretations of generative AI as a mechanism capable of fully replacing human intervention in cybersecurity analysis. Although the proposed framework reduced the need for manual log review by automatically aggregating events, interpreting behavior, assigning severity levels, and generating structured assessments, the results do not support the complete replacement of human analysts. This finding directly addresses RQ4, indicating that the LLM-based analysis functions as a decision-support mechanism rather than a replacement for human expertise. Three FNs were observed, all involving low-volume or partial attack activity: two abandoned brute-force attempts and one partially executed reconnaissance scan. These cases show that low-intensity or borderline malicious behavior may not always generate sufficient or stable evidence for confident escalation. Of the 52 analyzed observations, 49 were classified correctly without requiring human intervention, yielding a correct-classification rate of 94.23%; the three FNs highlight the boundary conditions under which analyst review remains necessary. This observation is consistent with previous concerns regarding LLM variability, possible overgeneralization, hallucinated conclusions, and dependence on the quality of contextual evidence [50]. Accordingly, the proposed framework should be interpreted as an analyst-support mechanism rather than as a fully autonomous decision-making system.

The study also partially diverges from conventional signature-based IDS logic. Traditional systems such as Snort and Suricata remain widely used due to their efficiency in detecting known threats and rule-defined attack patterns [58,59]. However, these approaches are typically centered on predefined signatures, packet-level inspection, or static rule matching. In contrast, the proposed framework demonstrates that malicious meaning may emerge from the temporal combination of multiple actions within the same session. For instance, repeated authentication failures, followed by successful login, reconnaissance commands, file staging, and download-and-execute activity become more meaningful when interpreted as a behavioral sequence rather than as disconnected indicators. Therefore, the results do not suggest that LLM-enhanced honeypot analysis replaces conventional IDS mechanisms. Instead, they indicate that it can complement them by providing contextual, session-aware reasoning over attacker behavior.

The comparison between locally deployed models also contributes to the practical discussion on the feasibility of LLMs in cybersecurity environments. Mistral-Nemo 12B provided the best operational balance between response stability, contextual consistency, level of detail, and concurrent processing capability. In contrast, Qwen3:30B-A3B, despite producing analytically comparable responses under stable conditions, required substantially greater computational capacity, exceeded the real-time latency budget on the majority of calls, and failed to reliably conform to the required structured output format even with reasoning mode disabled. Qwen3:8B performed comparably to Mistral-Nemo 12B in this controlled comparison, though the full 52-observation study was conducted exclusively on Mistral-Nemo 12B for the reasons detailed in Section 4 (model-comparison subsection). These findings extend previous studies by showing that model selection in cybersecurity cannot be based solely on theoretical model capability or parameter scale. Instead, local deployment requires balancing analytical quality, latency, hardware feasibility, workload stability, and suitability for near-real-time operation. These observations provide direct empirical support for RQ2, confirming that concurrency robustness and hardware feasibility are determinant factors in model selection for near-real-time honeypot analysis pipelines.

Notably, Qwen3:30B-A3B's classification metrics (Accuracy = 91.1%, F1-score = 93.3%, MCC = 0.781, as shown in Table 11) were obtained on the original 15-observation evaluation

set, since its VRAM footprint and consequent CPU offloading made it impractical to run the expanded 52-observation evaluation within the available hardware budget. Under the expanded evaluation, Mistral-Nemo 12B reached an Accuracy of 94.23%, an F1-score of 93.33%, and an MCC of 0.8890, comparable to or exceeding Qwen3:30B-A3B's figures on the smaller sample; however, the two are not directly comparable given the difference in evaluation size. This asymmetry does not contradict the selection of Mistral-Nemo 12B for the final pipeline. The higher metrics reflect performance only on sessions that the model was able to process without resource exhaustion, whereas its 30B parameter footprint exceeded the available 8 GB VRAM allocation, inducing CPU offloading and the highest MTTD values observed across all evaluated models. This illustrates that raw classification performance, taken in isolation, is an incomplete criterion for selecting LLMs in near-real-time security pipelines; concurrency robustness and hardware feasibility remain equally decisive, reinforcing the empirical evidence presented for RQ2.

With respect to prompt design, the findings indicate that structured prompting was central to achieving consistent and interpretable LLM-based cybersecurity analysis. The use of predefined severity levels, escalation rules, evidence constraints, and a fixed output schema contributed to stable analytical outputs and reduced unsupported interpretation. This is consistent with prior work emphasizing the importance of structured LLM-based log analysis and robust output parsing in cybersecurity contexts [8]. However, the present study extends this understanding by demonstrating that prompt design is most effective when combined with deterministic preprocessing. The LLM was not exposed to raw logs alone; instead, it received structured JSON digests containing session summaries, annotated command timelines, file activity, and deterministic threat indicators. This hybrid design reduced ambiguity and anchored generative reasoning in verifiable evidence, thereby limiting speculative conclusions and improving analytical reproducibility. These observations confirm that prompt structure significantly influences the consistency and interpretability of LLM-based analysis, thereby addressing RQ1.

The results also extend previous honeypot and LLM studies by integrating components that are often investigated separately. Prior honeypot research has frequently emphasized attacker interaction, telemetry collection, or deception realism, while LLM-related cybersecurity studies have often focused on summarization, classification, or assistant-style incident support. Some studies have explored the use of LLMs to improve interaction realism in honeypot environments, including context-aware SSH responses or service emulation [12–14], while others have emphasized the analytical use of LLMs for log summarization, attack classification, and intrusion strategy identification [15,16]. The proposed framework connects these two research directions through an end-to-end pipeline composed of Cowrie-based log capture, Python-based session reconstruction, deterministic preprocessing, local LLM inference, and structured report generation. This integration represents a key contribution because it demonstrates that locally deployed LLMs can be embedded into honeypot workflows not only for post hoc analysis but also for near-real-time interpretation of attacker behavior.

Another relevant extension concerns the treatment of concurrent and heterogeneous sessions. The framework maintained session separation across a high-volume brute-force attack followed by a download-and-execute pattern, a lower-volume brute-force attack followed by reconnaissance and persistence staging, and a benign audit-like session. The high-volume attack and the benign session were classified correctly and consistently, whereas the lower-volume multi-stage session was not classified consistently, ending at a Monitor verdict. This result extends previous work by demonstrating that LLM-assisted analysis can preserve session separation and interpret multiple overlapping behavioral profiles without incorrectly escalating benign activity. In practical terms, this suggests that the

proposed architecture can support environments where multiple attackers, benign users, or automated scripts interact with a honeypot simultaneously.

No full divergence from previous studies was observed. Rather, the findings mainly confirm the relevance of honeypots for attacker behavior observation and the usefulness of LLMs for explainable log analysis, while partially diverging from assumptions that AI-based cybersecurity analysis can operate without human oversight. The three FNs, the inconsistent classification of the low-volume multi-stage session, and the operational limitations observed in one of the evaluated models indicate that LLM-assisted analysis remains dependent on contextual evidence, prompt design, model robustness, and operational constraints. This reinforces the need to position such systems as decision-support tools that reduce analyst workload and improve interpretability, rather than as substitutes for expert validation.

Fundamentally, the main contribution of this study is not limited to the reported classification metrics. The study demonstrates that raw honeypot telemetry can be transformed into structured, explainable, and actionable cybersecurity intelligence through the combination of deterministic preprocessing and local LLM-based reasoning. The results confirm previous studies on honeypot-based cyber threat intelligence and LLM-supported security log analysis, partially diverge from expectations of full AI autonomy, and extend prior work by proposing and evaluating an end-to-end, locally deployed, near-real-time honeypot–LLM framework capable of analyzing concurrent and heterogeneous attack sessions. In this sense, the proposed approach contributes to the evolution of honeypot-based intrusion analysis from passive data collection toward contextual, explainable, and operationally usable threat interpretation.

4.5. Limitations

Several limitations should be acknowledged. First, the evaluation was based on controlled and authorized attack scenarios. Although these scenarios were designed to approximate realistic attacker behavior, they cannot fully reproduce the variability, noise, and unpredictability of internet-facing environments. Second, the reported metrics reflect a single evaluation run across the 52 analyzed observations, rather than multiple independent repetitions with corresponding means and standard deviations; as such, the point estimates and confidence intervals reported should be interpreted as characterizing this single run rather than run-to-run variability. Accordingly, and consistent with the framing adopted in Section 5, the quantitative results reported throughout this study should be read as preliminary, proof-of-concept evidence of feasibility rather than as a stable estimate of production-grade performance; replicating the evaluation across multiple independent runs and reporting the resulting means and standard deviations is identified as a priority direction for future work (Section 6). Third, the ground-truth labels used to compute the reported metrics were derived from the same rule-based indicators, such as REPEATED_LOGIN_FAILURES and SUCCESS_AFTER_BRUTE_FORCE, that also drive the deterministic preprocessing engine described in Sections 3.3 and 3.7, rather than from an externally validated benchmark dataset or an independent labeling process. As a result, the reported agreement between system output and ground truth partly reflects the consistency between the two components being built on the same underlying rule vocabulary, rather than validation against independently derived, ground-truth attacker intent. This is particularly relevant to the reported false negatives, which arise from the same indicator being evaluated against different thresholds by the deterministic engine and the ground-truth definition, rather than from the system failing to detect behavior that an independently derived label would have captured. This effect can be quantified directly: across the experimental runs, the LLM's assigned severity matched the deterministic rule-

based severity in 82.4% of classifications under the combined configuration, while the final binary alert decision matched the rule-based decision in 100% of cases, reflecting the severity-override mechanism described in Section 3.3. Under the LLM-only configuration, where this override is absent, agreement dropped to 25.0% at the severity level and 44.9% at the alert-decision level, indicating that the LLM's independent judgment diverges substantially from the rule-based reference when not constrained by it. Fourth, the number and diversity of benign sessions remained limited when compared with the range of legitimate administrative behaviors that may appear in production systems. Continued evaluation in a larger and more diverse benign population therefore remains warranted, in order to more robustly characterize the false-positive rate and specificity of the framework before generalizing these results to production environments. Finally, although local inference improves privacy and deployment control, model performance remains partly dependent on hardware constraints and concurrent workload characteristics. Additionally, the relative contribution of the LLM component versus the deterministic preprocessing and severity-override logic has not yet been isolated. To this end, an ablation study (rules-only, LLM-only, and combined configurations; Section 4.2) was conducted. The results show that the rules-only configuration matches the combined pipeline's detection performance, while the LLM-only configuration performs considerably worse and exhibits lower schema adherence. We report this transparently: the framework's detection accuracy should currently be attributed principally to the deterministic layer, and future work should focus on strengthening the LLM's independent contribution—for example, through improved prompting, fine-tuning, or few-shot calibration—so that it adds classification value beyond what deterministic rules already provide, rather than functioning mainly as a natural-language reporting layer over rule-based decisions. These limitations do not invalidate the findings, but they do restrict direct generalization and indicate that further validation in broader and less controlled environments is required.

5. Conclusions

This study proposed and evaluated an integrated framework combining a Cowrie SSH honeypot with a locally deployed LLM to support automated detection and near-real-time behavioral analysis of malicious activity. The evaluation covered two distinct scenarios: a single-source attack involving brute-force authentication, reconnaissance, privilege escalation, and post-exploitation activity; and a concurrent multi-session scenario involving two attacker profiles of differing intensity alongside a simulated benign audit session.

The framework achieved an Accuracy of 94.23% (95% CI: 84.05–98.79%), a Precision of 100.0%, a Recall of 87.50%, and an F1-score of 93.33%, across an expanded evaluation of 52 independent observations, with zero FPs. The MCC of 0.8890 and Cohen's κ of 0.8829 confirm that the classification performance exceeds chance-level agreement. Temporally, two measurements were obtained from different experiments: the time to first malicious classification was approximately 25 s in the single-source scenario, reflecting the detection latency of the pipeline, while the time to the first Critical escalation was approximately 4 min and 28 s in the concurrent scenario, reflecting the time for the attack to reach its critical stage rather than analytical latency. The former supports the suitability of the pipeline for near-real-time operational environments. Multiple attack categories were correctly identified, including brute-force authentication, post-compromise activity, persistence staging, and download-and-execute patterns, with performance maintained for high-intensity categories under concurrent multi-session conditions. Three FNs were recorded, all associated with abandoned or incomplete attack attempts terminated before generating sufficient escalation evidence: two abandoned brute-force sessions (classified as Normal) and one partially executed scanning session (classified as Low Suspicion). In all three cases, activity volume

was insufficient to trigger stable deterministic escalation, consistent with the conservative design of the severity thresholds.

These findings collectively address the four research questions defined in this study. The absence of FPs and the largely consistent structured outputs for forced-severity cases are consistent with a stabilizing effect of deterministic preprocessing and prompt design (RQ1); this is now supported by 28 benign cases (specificity 100%, 95% CI: 87.66–100%), providing stronger, though still preliminary, evidence than the earlier three-observation sample. The comparative operational assessment demonstrated that Mistral-Nemo 12B provided the best balance between analytical quality and near-real-time feasibility under concurrent load. Although Qwen3:30B-A3B achieved marginally higher Accuracy and F1-score under stable conditions, its hardware demands and resulting latency degradation under concurrent load confirmed Mistral-Nemo 12B as the most operationally suitable model for the evaluated deployment (RQ2). The confusion matrix results, yielding TP = 21, TN = 28, FP = 0, and FN = 3 across 52 analyzed observations (Accuracy 94.23%, 95% CI: 84.05–98.79%), confirm effective detection across high-intensity attack categories under both isolated and concurrent conditions, while the ablation study (Section 4.2) indicates that this performance is currently attributable primarily to the deterministic preprocessing layer (RQ3). Finally, the correct classification of 49 out of 52 observations (94.2%) without human intervention confirms the framework's potential as a decision-support mechanism, while highlighting the continued need for analyst oversight in low-intensity, partial, or abandoned-attack scenarios (RQ4).

The integration of deterministic preprocessing with LLM-based reasoning enabled effective session-level interpretation of honeypot data, transforming raw logs into structured and actionable outputs while supporting near-real-time analysis. These results provide preliminary, proof-of-concept evidence that locally deployed LLMs can be integrated into honeypot pipelines to support behavioral analysis, complementing rather than replacing traditional monitoring and detection mechanisms. Broader validation on a larger and more diverse dataset is required before these findings can be generalized to practical cybersecurity operations.

Future work should address the identified limitations and extend the framework along five principal directions: validation in internet-facing environments with unsolicited attack traffic, extension of protocol coverage beyond SSH, systematic benchmarking of alternative locally deployable LLMs, integration with automated response and security information and event management (SIEM) platforms to enable closed-loop cybersecurity workflows, and refinement of the threshold logic governing low-intensity attack detection to reduce FNs without compromising precision.

6. Future Work

Several directions are identified for extending and strengthening the proposed framework. First, validation under real-world attack conditions represents a critical next step. The experiments conducted in this study relied primarily on controlled, laboratory-based attack simulations executed by authorized systems within an isolated virtual environment. While this setup enabled reproducible evaluation and precise behavioral profiling, it necessarily limited the diversity and unpredictability of the observed attack traffic. Future research should therefore incorporate additional datasets, including publicly available benchmark datasets, real-world network traffic, and heterogeneous attack traces collected from diverse operational environments. Furthermore, the evaluation should employ complementary procedures, such as cross-dataset validation, temporal validation, adversarial testing, and experiments involving previously unseen attack scenarios, to assess the robustness and generalizability of the proposed approach. Deploying the framework in internet-facing

environments, where the Cowrie honeypot is exposed to unsolicited, opportunistic, and adversarially varied attack campaigns, would provide a substantially richer dataset for evaluating detection robustness, generalization across previously unseen attack patterns, and stability of the Mistral-Nemo-based analysis pipeline under continuous, uncontrolled load. Such a deployment would also allow for the assessment of FP rates in the presence of benign or ambiguous traffic that was not represented in the current experimental scenarios.

Second, extending protocol coverage beyond SSH would significantly broaden the framework's applicability. The current implementation targets SSH interactions captured by Cowrie, which is particularly well suited for monitoring brute-force authentication, command execution sequences, and post-compromise behavior in remote access scenarios. However, real-world attack surfaces frequently include additional entry points such as Telnet, HTTP/HTTPS, FTP, and SMTP. Integrating protocol-specific honeypot components, for instance, combining Cowrie's SSH/Telnet emulation with web application honeypots for HTTP-based attack capture, would expose the preprocessing and LLM-based analysis pipeline to more heterogeneous session structures. This would require extending the session reconstruction logic in analyser_realtime.py to handle protocol-specific event schemas and adapting the structured JSON digest format to preserve protocol-relevant contextual signals for generative reasoning. Evaluating whether the existing prompt architecture generalizes across protocol-specific attack patterns, or whether protocol-aware prompt specialization is required, represents an open and practically relevant research question.

Third, a systematic comparative evaluation of alternative locally deployable LLMs warrants further investigation. The selection of Mistral-Nemo 12B in this study was based on an operational comparison conducted within the available experimental environment, prioritizing near-real-time responsiveness, contextual consistency, and analytical detail. However, the landscape of locally deployable models continues to evolve rapidly. Models such as Llama 3.1 8B, Phi-4 Mini, or Gemma 3 12B offer distinct trade-offs between parameter efficiency, instruction-following fidelity, and inference latency under constrained hardware. A structured benchmark evaluating these models against a standardized set of honeypot session digests, assessing severity classification consistency, reasoning depth, output format adherence, and MTTD, would provide actionable guidance for practitioners deploying similar pipelines in resource-limited SOC environments. Additionally, exploring quantized variants of larger models through frameworks such as llama.cpp or Ollama's GPT-generated unified format (GGUF) support could reveal viable pathways for improving analytical capability without sacrificing the real-time performance constraints demonstrated in this study.

Fourth, integration with automated response mechanisms represents a key direction toward enabling operationally closed-loop cybersecurity workflows. In the current framework, the LLM-generated outputs, comprising severity classifications, recommended actions, and confidence scores, are structured for human review and analytical consumption. A natural extension would involve coupling these outputs directly to enforcement systems, whereby critical-severity classifications such as BLOCK_IP_IMMEDIATELY trigger automated firewall rules via APIs exposed by platforms such as iptables, pfSense, or cloud-native security groups. Similarly, high-severity outputs could be forwarded as structured alerts to SIEM platforms or SOC ticketing systems using standardized formats such as Structured Threat Information Expression (STIX), Trusted Automated Exchange of Intelligence Information (TAXII), or Common Event Format (CEF). Careful design is required to avoid automated over-blocking due to false positives (FPs), particularly for borderline severity levels. Incorporating confidence thresholds derived from the LLM's output field, already present in the current structured report format, could serve as a

natural gating mechanism, routing low-confidence assessments to human review queues while enabling autonomous enforcement for high-confidence, clearly malicious patterns.

Fifth, addressing the three FNs identified in the expanded evaluation motivates complementary refinements to the preprocessing component. Two of the three FNs involve abandoned brute-force attempts terminated before sufficient volume accumulated to trigger escalation, motivating refinement of the threshold logic governing volume-based indicators for sessions that end prematurely. The third FN is qualitatively different: in the low-volume reconnaissance-and-persistence session observed during the concurrent three-machine experiment (Section 4.1), the severity was correctly escalated to High Threat during the attack but subsequently regressed to a Monitor-level verdict, so that under the final-verdict scoring rule adopted in Section 4.2, the session was ultimately recorded as benign. This indicates that, beyond threshold tuning, the severity assigned across the lifetime of a session should not be allowed to decrease once a higher level has been justified by observed evidence. A non-decreasing (latched) severity mechanism, already implemented in the preprocessing module as an opt-in runtime mode, addresses this directly: the five-level severity scale (Normal < Low Suspicion < Suspicious < High Threat < Critical) is treated as a totally ordered scale, and for each session (and, independently, for each source IP, to cover multi-session attackers) a running peak severity is maintained. For every new verdict, the value reported to the analyst is the higher of the current instantaneous verdict and this running peak, rather than the instantaneous verdict alone. This is implemented as a single-line update: peak_severity = max (peak_severity, current_severity) under the ordinal ranking above, applied independently per session and per source IP. By construction, the peak is non-decreasing, and the reported verdict at any point in a session's history can never fall below the highest level previously justified by the observed evidence. This mechanism was in fact already exercised in the present evaluation, corresponding to the peak-severity scoring convention reported in Table 9, which showed that for the combined pipeline the deterministic forced-severity floor already prevents this specific class of regression, while for the LLM-only configuration, lacking this floor, the two conventions diverge substantially. We therefore identify enabling this latching behavior by default (rather than as an opt-in mode) as a direct and immediately actionable refinement, together with an explicit reset policy (e.g., session boundary or a bounded decay window) to prevent unbounded severity accumulation for long-lived or reused source IPs, which is not yet defined in the current implementation. More broadly, the study highlights a structural limitation of rule-based severity overrides when attack volume is low but intent is clearly adversarial. Future work could explore adaptive thresholding mechanisms that incorporate contextual signals beyond raw failure counts, such as the geographic or temporal distance between attempts, the reputation of the source IP derived from threat intelligence feeds, or command-level activity immediately following authentication. Integrating external threat intelligence sources, such as AbuseIPDB, Shodan, or GreyNoise, into the preprocessing enrichment stage would provide additional context for borderline cases and may reduce FNs without compromising the zero FP rate observed in the current evaluation.

Author Contributions: Conceptualization, R.O.; methodology, D.P. and R.O.; software, D.P.; validation, D.P. and T.G.; formal analysis, D.P., R.O., and T.G.; investigation, D.P., R.O., and T.G.; resources, D.P.; data curation, R.O. and T.G.; writing—original draft preparation, R.O. and T.G.; writing—review and editing, D.P., R.O., T.G., and F.A.; visualization, R.O. and T.G.; supervision, F.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: No personal or sensitive data were collected in this study. The research is based on controlled and authorized honeypot experiments conducted in an isolated laboratory

environment, using simulated attack and benign interaction scenarios. The generated data include Cowrie honeypot logs, structured preprocessing outputs, and LLM-based analytical reports. All of these, together with the source codes—the real-time sentinel, the deterministic preprocessing and scoring scripts, and the complete system prompt—the 24-round scenario definitions, the captured logs with their by-construction per-round labels, the run manifests, and the setup instructions, are openly available in a public repository (https://github.com/danirzkpinho/cowrie-honeypot-llm-sentinel, accessed on 12 June 2026, permanently archived on Zenodo with DOI 10.5281/zenodo.21986646: https://doi.org/10.5281/zenodo.21986646). The logs contain only private laboratory source addresses and no personal data.

Acknowledgments: During the preparation of this manuscript, the authors used AI-based language assistance tools solely to support the translation of the authors' original Portuguese text into scientific English. The tools were not used to generate the study design, methodology, data, results, analysis, or scientific conclusions. The authors reviewed and edited the translated text and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
API	Application programming interface
AUC	Area under the curve
CEF	Common Event Format
FN	False negative
FP	False positive
GGUF	GPT-generated unified format
IDS	Intrusion detection system
IPS	Intrusion prevention system
JSON	JavaScript Object Notation
LLM	Large language model
LSTM	Long short-term memory
MCC	Matthews' correlation coefficient
ML	Machine learning
MTTD	Mean time to detection
ROC	Receiver operating characteristic
SIEM	Security information and event management
SOC	Security operations center
SSH	Secure Shell
STIX	Structured Threat Information Expression
TAXII	Trusted Automated Exchange of Intelligence Information
TN	True negative
TP	True positive
TTP	Tactics, techniques, and procedures
VM	Virtual machine

Appendix A. Hydra-Based SSH Brute-Force Attack Execution

Appendix A illustrates the execution of a brute-force SSH attack using Hydra against the honeypot. The attacker attempts to authenticate as the root user by testing multiple passwords from the rockyou wordlist, generating repeated failed login attempts that are later captured and analysed by the honeypot logging system.

```
┌──(kali㉿kali)-[~]
└─$ hydra -l root -P /usr/share/wordlists/rockyou.txt -t 1 -V ssh://192.168.1.114:22
Hydra v9.6 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2026-05-19 14:52:48
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a p
[DATA] max 1 task per 1 server, overall 1 task, 14344399 login tries (l:1/p:14344399), ~14344399
[DATA] attacking ssh://192.168.1.114:22/
[ATTEMPT] target 192.168.1.114 - login "root" - pass "123456" - 1 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "12345" - 2 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "123456789" - 3 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "password" - 4 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "iloveyou" - 5 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "princess" - 6 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "1234567" - 7 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "rockyou" - 8 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "12345678" - 9 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "abc123" - 10 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "nicole" - 11 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "daniel" - 12 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "babygirl" - 13 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "monkey" - 14 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "lovely" - 15 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "jessica" - 16 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "654321" - 17 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "michael" - 18 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "ashley" - 19 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "qwerty" - 20 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "111111" - 21 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "iloveu" - 22 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "000000" - 23 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "michelle" - 24 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "tigger" - 25 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "sunshine" - 26 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "chocolate" - 27 of 14344399 [child 0] (0/0
[ATTEMPT] target 192.168.1.114 - login "root" - pass "password1" - 28 of 14344399 [child 0] (0/0
[ATTEMPT] target 192.168.1.114 - login "root" - pass "soccer" - 29 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "anthony" - 30 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "friends" - 31 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "butterfly" - 32 of 14344399 [child 0] (0/0
[ATTEMPT] target 192.168.1.114 - login "root" - pass "purple" - 33 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "angel" - 34 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "jordan" - 35 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "liverpool" - 36 of 14344399 [child 0] (0/0
[ATTEMPT] target 192.168.1.114 - login "root" - pass "justin" - 37 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "loveme" - 38 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "fuckyou" - 39 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.1.114 - login "root" - pass "123123" - 40 of 14344399 [child 0] (0/0)
```

Figure A1. Hydra SSH brute-force attack executed against the honeypot during the experimental data acquisition phase.

Appendix B. Examples of Raw Cowrie Log Events

Appendix B presents an example of raw Cowrie log events generated during the experimental scenarios. The log shows repeated SSH authentication attempts, including timestamps, source session information, attempted usernames and passwords, and failed login outcomes. These raw events were later ingested by the preprocessing component and reconstructed into chronological per-IP session histories for subsequent AI-based analysis.

```
2026-05-19T19:54:04.876730Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] incoming:
b'aes256-ctr' b'hmac-sha2-256' b'none'
2026-05-19T19:54:05.327951Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] NEW KEYS
2026-05-19T19:54:05.328508Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] starting s
ervice b'ssh-userauth'
2026-05-19T19:54:05.374995Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'root
' trying auth b'none'
2026-05-19T19:54:05.377536Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'root
' trying auth b'password'
2026-05-19T19:54:05.378128Z [HoneyPotSSHTransport,4,192.168.1.115] login attempt [b'root
'/b'elizabeth'] failed
2026-05-19T19:54:06.381006Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'root
' failed auth b'password'
2026-05-19T19:54:06.381610Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] unauth
orized login: ()
2026-05-19T19:54:06.385055Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'root
' trying auth b'password'
2026-05-19T19:54:06.386041Z [HoneyPotSSHTransport,4,192.168.1.115] login attempt [b'root
'/b'hottie'] failed
2026-05-19T19:54:07.389545Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'root
' failed auth b'password'
2026-05-19T19:54:07.390138Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] unauth
orized login: ()
2026-05-19T19:54:07.393125Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'root
' trying auth b'password'
2026-05-19T19:54:07.393966Z [HoneyPotSSHTransport,4,192.168.1.115] login attempt [b'root
'/b'tinkerbell'] failed
2026-05-19T19:54:08.396944Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'root
' failed auth b'password'
2026-05-19T19:54:08.397579Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] unauth
orized login: ()
2026-05-19T19:54:08.401547Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'root
' trying auth b'password'
2026-05-19T19:54:08.402842Z [HoneyPotSSHTransport,4,192.168.1.115] login attempt [b'root
'/b'charlie'] failed
2026-05-19T19:54:09.406923Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'root
' failed auth b'password'
2026-05-19T19:54:09.407735Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] unauth
orized login: ()
2026-05-19T19:54:09.411187Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'root
' trying auth b'password'
2026-05-19T19:54:09.412267Z [HoneyPotSSHTransport,4,192.168.1.115] login attempt [b'root
'/b'samantha'] failed
```

Figure A2. Examples of raw Cowrie SSH authentication log events used during the session reconstruction and preprocessing phase.

```
{
  "ip": "192.168.1.115",
  "login_analysis": {
    "total_attempts": 12,
    "failures": 11,
    "successes": 1
  },
  "threat_indicators": [
    "SUCCESS_AFTER_BRUTE_FORCE",
    "INTERACTIVE_SESSION_ACTIVE",
    "REPEATED_LOGIN_FAILURES"
  ],
  "forced_severity": "Critical",
  "forced_action": "BLOCK_IP_IMMEDIATELY — active session established after brute force."
}
```


Figure A3. An example structured JSON digest excerpt (abbreviated) generated from the raw Cowrie events shown in Figure A2 for a session exhibiting a successful login after repeated failed attempts.

Appendix C. Example of Structured System Output During Live Execution

Appendix C presents a representative example of the structured output generated by the proposed framework during live execution. The output shows how the AI-based reasoning module interprets raw Cowrie honeypot events and produces a security-oriented report including severity level, summary, reasoning, assessment, recommended action, confidence score, and brute-force monitoring indicators.

```
============================================================
[AI SENTINEL] IP: 192.168.1.115 | Event: cowrie.login.failed
Narrative: 3 failed login attempts with no success.
Indicators: []
**Severity:** Low Suspicion
**Summary:** Three failed login attempts with default credentials.
**Reasoning:** The session narrative and login analysis indicate three consecutive failed
 login attempts using default credentials. This is unusual but could potentially be noise
, such as a typo during login.
**Assessment:** Likely a brute force attempt or user error.
**Recommended Action:** Monitor the IP address for further activity. If more than 5 failu
res occur within an hour, escalate to Suspicious and consider rate-limiting.
**Confidence:** Medium
------------------------------------------------------------
  [!] 19:07:28 UTC | BRUTE FORCE IN PROGRESS | IP: 192.168.1.115 | Attempts: 23 | Unique
passwords tried: 22 | No success yet

============================================================
[AI SENTINEL] IP: 192.168.1.115 | Event: cowrie.login.failed [FORCED: Suspicious]
Narrative: Repeated login failures: 25 attempts with no success.
Indicators: ['REPEATED_LOGIN_FAILURES']
**Severity:** Suspicious
**Summary:** Repeated login failures from an automated scanner.
**Reasoning:** The session narrative reports 25 failed login attempts with no successes.
The actor profile classifies this as an automated scanner due to the high volume of login
 failures. The forced_severity is set to "Suspicious", which we adhere to despite the hig
h number of failures.
**Assessment:** This likely indicates a brute force attack targeting default credentials.
**Recommended Action:** Monitor the situation closely. If there are any successful logins
 or other suspicious activities, escalate the severity accordingly.
**Confidence:** Medium
------------------------------------------------------------
  [!] 19:07:48 UTC | BRUTE FORCE IN PROGRESS | IP: 192.168.1.115 | Attempts: 43 | Unique
passwords tried: 41 | No success yet

============================================================
[AI SENTINEL] IP: 192.168.1.115 | Event: cowrie.login.failed [FORCED: Suspicious]
Narrative: Repeated login failures: 50 attempts with no success.
Indicators: ['REPEATED_LOGIN_FAILURES']
**Severity:** High Threat
**Summary:** Repeated login failures with high volume and default credentials targeting.
**Reasoning:** The session narrative indicates 50 failed login attempts with no success,
```

Figure A4. Example of structured real-time security output generated by the AI-based analysis module during a brute-force attack.

Appendix D. Complete System Prompt

Appendix D reproduces, verbatim, the complete system prompt used to instruct the LLM throughout the evaluation described in Section 3.4, corresponding to the role definitions, severity vocabulary, escalation rules, and output schema summarized in Tables 3 and 4.

Box A1. Grounded (combined) system prompt used for all reported results.

You are an expert cybersecurity analyst monitoring honeypot logs in real time.

Your role is to behave like a cautious, experienced security professional whose job is to interpret system and network activity, detect suspicious patterns, and provide real-time security assessment.

You must think like a human cybersecurity analyst:
- cautious, not alarmist
- analytical, not reactive
- aware that isolated events may be benign
- aware that repeated or correlated events indicate an attack
- focused on distinguishing normal noise from real threats

RULE #1—FORCED SEVERITY (HIGHEST PRIORITY):

If the digest contains a "forced_severity" field, you MUST use exactly that severity level.
You MUST NOT go below it under any circumstances.
If "forced_action" is present, you MUST include it verbatim in your Recommended Action field.
These are pre-computed deterministic rules that override LLM judgment.

This overrides RULE #4. If an item in the RULE #4 "not sufficient to escalate" list is present but a forced_severity is also present, the forced_severity WINS—never lower your verdict below it because of a RULE #4 item. Example: a forced_severity of "Suspicious" from SSH_ALGO_ENUMERATION must be reported as Suspicious, even though a plain scan appears in the "not sufficient to escalate" list.

CRITICAL EXCEPTION: If forced_severity is "Low Suspicion" or "Normal", this explicitly means the pre-assessment has determined the activity is LIKELY BENIGN (e.g., a typo during login). In this case, RULE #3 mandatory escalations DO NOT apply based on login failure count alone. Only escalate above forced_severity if you observe concrete post-login evidence (dangerous commands, file downloads, tunneling).

RULE #2—SEVERITY MODEL:

- Normal: Single benign or weak-signal event.
- Low Suspicion: Unusual but possibly noise. Example: 2–4 failed logins.
- Suspicious: Clear pattern of malicious intent. Example: brute force attempt (>10 failures).
- High Threat: Strong evidence of active attack. Example: brute force >25 failures, nmap + login attempts.
- Critical: Confirmed attack or active compromise. Examples:
* Successful login after ANY brute force attempt
* Malware download-and-execute pattern after login
* File upload followed by execution
* Lateral movement or tunneling after login

RULE #3—MANDATORY ESCALATION (apply even without forced_severity):

- login.success after $\geq$3 login.failed $\rightarrow$ minimum Suspicious
- login.success after $\geq$5 login.failed $\rightarrow$ minimum High Threat
- login.success after $\geq$10 login.failed $\rightarrow$ Critical $\rightarrow$ Recommend: BLOCK_IP_IMMEDIATELY

Box A1. *Cont.*

- SUCCESS_AFTER_BRUTE_FORCE + INTERACTIVE_SESSION_ACTIVE → Critical → Recommend: BLOCK_IP_IMMEDIATELY
- DOWNLOAD_AND_EXECUTE_PATTERN → Critical → Recommend: BLOCK_IP_IMMEDIATELY
- REVERSE_SHELL_ATTEMPT → Critical → Recommend: BLOCK_IP_IMMEDIATELY
- SSH_KEY_PERSISTENCE → Critical → Recommend: BLOCK_IP_IMMEDIATELY
- CRYPTOMINING_INDICATORS → Critical → Recommend: BLOCK_IP_IMMEDIATELY
- DESTRUCTIVE_COMMAND → Critical → Recommend: BLOCK_IP_IMMEDIATELY
- EXTERNAL_FILE_DOWNLOAD after successful login → minimum High Threat
- HIGH_VOLUME_BRUTE_FORCE (>50 failures) → minimum High Threat
- DATA_EXFILTRATION_ATTEMPT after successful login → minimum High Threat
- ANTI_FORENSICS or DEFENSE_EVASION_ATTEMPT after login → minimum High Threat

NOTE on LOCAL_FILE_CREATED: This indicator means the session wrote a file to the honeypot filesystem via a shell command (touch, echo redirect, heredoc, etc.). It is NOT a download from an external source. Do NOT treat it as equivalent to EXTERNAL_FILE_DOWNLOAD or FILE_DOWNLOAD_DETECTED. On its own it carries very low suspicion weight. Always check the content_preview field to understand what was actually written.

NOTE on annotated_command_timeline: Each entry has "intent" (behavioral category), "risk" (benign/low/medium/high/critical) and "tactic" (the stage of an intrusion the command belongs to). Use risk = benign as strong evidence of normal user activity.
Only escalate based on commands with risk = high or risk = critical. Commands like navigation, file_write, file_read, file_edit, system_info, archive and package_management with risk = benign or risk = low are normal sysadmin behaviour and must NOT raise severity.

NOTE on benign_signals: A list of positive observations that point AWAY from an attack (clean login, only benign commands, hand-typed pacing, clean exit, no outbound network activity). The longer this list is and the emptier threat_indicators is, the closer to Normal your verdict should be. Ordinary system administration must be classified as Normal or Low Suspicion.

NOTE on behavioral_timing: Measured gaps between events. login_pacing or command_pacing of "machine_paced"/"scripted" means an automated tool; "human_paced" means someone typing. Automated pacing on its own is not an attack—scanners and monitoring tools are also automated—but combined with credential attempts it supports a brute force reading.

NOTE on kill_chain.stages_observed: The intrusion stages the pre-processor could see, in the order they first appeared. A session that reaches persistence, exfiltration, command_and_control or impact is a confirmed compromise.

NOTE on session_narrative: A plain-English summary of the session arc generated by the pre-processor before AI analysis. Use it as a first-pass framing anchor, but always verify against the raw annotated timeline and file content previews.

NOTE on content_preview in file_activity: Contains the first characters of files written to or downloaded into the honeypot. Use this to determine if a file contains benign data (text notes, config) versus a malicious payload (shell script, base64-encoded binary). An unreadable binary blob is more suspicious than readable plaintext.

NOTE on client_version_strings: This field contains SSH client identifier banners (e.g., "SSH-2.0-OpenSSH_10.3"). These are NOT scan indicators—every SSH client sends this string on connect. Only treat as a scanner if the string explicitly contains: Nmap, ZGrab, Masscan, libssh, or Go-http.

NOTE on scan indicators—judge reconnaissance in proportion to how intrusive it was:
- CONNECTION_PROBE_NO_SSH: a bare TCP connection that spoke no SSH and did nothing (a stealth/connect scan, or a monitoring health check). Treat as Normal—flagging it would fire on every port check.
- SSH_SCANNER_BANNER: a scanning tool completed an SSH handshake but did not enumerate or authenticate. Low Suspicion—reconnaissance, but light.

Box A1. *Cont.*

- SSH_ALGO_ENUMERATION: the scanner negotiated crypto and enumerated algorithms or authentication methods (e.g., nmap ssh2-enum-algos/ssh-auth-methods). This is targeted SSH reconnaissance—treat as Suspicious.
- NMAP_SCAN_WITH_LOGIN_ATTEMPTS: a scan combined with login attempts is a scan that turned into a brute force—escalate on the login side.

NOTE on session_view: The activity of the single session that triggered this analysis, separated from the cumulative history of the source address. When they disagree, say so: a quiet session from a previously aggressive address is still worth reporting.

RULE #4—ACTOR PROFILE (use only as tiebreaker):

If actor_profile.classification is "likely_legitimate" or "possibly_legitimate", treat that as evidence AGAINST malicious classification ONLY IF no concrete attack indicators are present. Actor profile NEVER overrides forced_severity or the escalation thresholds above.
Behaviors NOT sufficient alone to escalate:
- A BARE scan or port touch: CONNECTION_PROBE_NO_SSH, or a scanner banner with no SSH enumeration and no login (SSH_SCANNER_BANNER on its own). NOTE: this does NOT include SSH_ALGO_ENUMERATION—algorithm/auth enumeration is targeted reconnaissance and its forced_severity stands.
- Single successful login with no prior failures
- Running "ls", "ls-la", "pwd", "whoami", "id"
- Short sessions with 1–3 benign commands
- Creating a plain-text file (check content_preview before judging)
- Failed sudo/apt/package management commands
- Editing a file with nano or vi
- Checking disk space, uptime, or service status
- A session that connects and disconnects without attempting to log in

---

RESPONSE FORMAT—STRICT, NO ADDITIONS:

Respond with EXACTLY these 6 fields and nothing else:

Severity: <Normal | Low Suspicion | Suspicious | High Threat | Critical>
Summary: <one sentence>
Reasoning: <2–4 sentences explaining the classification>
Assessment: <1–2 sentences on what this likely indicates>
Recommended Action: <one concise action>
Confidence: <Low | Medium | High>

Box A2. LLM-only system prompt used only in the ablation baseline.

You are an expert cybersecurity analyst monitoring honeypot logs in real time.

You will receive raw Cowrie SSH honeypot log events for a single source IP address, in chronological order, exactly as the honeypot wrote them. No pre-processing has been applied. Interpret them yourself and assess the threat.

Severity scale:
- Normal: Single benign or weak-signal event.
- Low Suspicion: Unusual but possibly noise.
- Suspicious: Clear pattern of malicious intent.
- High Threat: Strong evidence of active attack.
- Critical: Confirmed attack or active compromise.

RESPONSE FORMAT—STRICT, NO ADDITIONS:
Respond with EXACTLY these 6 fields and nothing else:

Box A2. *Cont.*

```
Severity: <Normal | Low Suspicion | Suspicious | High Threat | Critical>
Summary: <one sentence>
Reasoning: <2–4 sentences explaining the classification>
Assessment: <1–2 sentences on what this likely indicates>
Recommended Action: <one concise action>
Confidence: <Low | Medium | High>
```

References

1. Alatawi, E.; Albalawi, U. Harnessing AI for Cyber Defense: Honeypot-Driven Intrusion Detection Systems. *Symmetry* **2025**, *17*, 628. [CrossRef]
2. Sezgin, A.; Özkan, G.; Boyacı, A. Advancements and Challenges in AI-Powered Honeypots: A Comparative Study of Detection, Engagement and Ethical Implications. In Proceedings of the 2025 13th International Symposium on Digital Forensics and Security (ISDFS), Boston, MA, USA, 24–25 April 2025. [CrossRef]
3. Lanka, P.; Gupta, K.; Varol, C. Intelligent Threat Detection—AI-Driven Analysis of Honeypot Data to Counter Cyber Threats. *Electronics* **2024**, *13*, 2465. [CrossRef]
4. Morić, Z.; Dakić, V.; Regvart, D. Advancing Cybersecurity with Honeypots and Deception Strategies. *Informatics* **2025**, *12*, 14. [CrossRef]
5. Lanz, S.; Pignol, S.L.-R.; Schmitt, P.; Wang, H.; Papaioannou, M.; Choudhary, G.; Dragoni, N. Optimizing Internet of Things Honeypots with Machine Learning: A Review. *Appl. Sci.* **2025**, *15*, 5251. [CrossRef]
6. Karaarslan, E.; Güler, E.; Yüce, E.E.; Coban, C. Towards Log Analysis with AI Agents: Cowrie Case Study. *arXiv* **2025**, arXiv:2509.05306. [CrossRef]
7. Tok, M.S.; Dener, M.; Demirci, M. Processing Honeypot Logs with Big Data and Data Visualization via Hadoop–Power BI Integration. In Proceedings of the 2022 15th International Conference on Information Security and Cryptography (ISCTURKEY), Ankara, Turkey, 19–20 October 2022. [CrossRef]
8. Palma, G.; Cecchi, G.; Caronna, M.; Rizzo, A. Leveraging Large Language Models for Scalable and Explainable Cybersecurity Log Analysis. *J. Cybersecur. Priv.* **2025**, *5*, 55. [CrossRef]
9. Jaffal, N.O.; Alkhanafseh, M.; Mohaisen, D. Large Language Models in Cybersecurity: A Survey of Applications, Vulnerabilities, and Defense Techniques. *AI* **2025**, *6*, 216. [CrossRef]
10. DeCusatis, C.; Tomo, R.; Singh, A.; Khoury, E.; Masone, A. Cybersecurity Applications of Near-Term Large Language Models. *Electronics* **2025**, *14*, 2704. [CrossRef]
11. Balogh, Š.; Mlynček, M.; Vraňák, O.; Zajac, P. Using Generative AI Models to Support Cybersecurity Analysts. *Electronics* **2024**, *13*, 4718. [CrossRef]
12. Fan, W.; Yang, Z.; Liu, Y.; Qin, L.; Liu, J. HoneyLLM: A Large Language Model-Powered Medium-Interaction Honeypot. In Proceedings of the 26th International Conference on Information and Communications Security (ICICS 2024), Mytilene, Greece, 26–28 August 2024. [CrossRef]
13. Ilg, N.; Germek, D.; Duplys, P.; Menth, M. Beekeeper: Accelerating Honeypot Analysis with LLM-Driven Feedback. *IEEE Access* **2025**, *13*, 168034–168054. [CrossRef]
14. Malhotra, P. LLMHoney: A Real-Time SSH Honeypot with Large Language Model-Driven Dynamic Response Generation. *arXiv* **2025**, arXiv:2509.01463. [CrossRef]
15. Reworr; Volkov, D. LLM Agent Honeypot: Monitoring AI Hacking Agents in the Wild. *arXiv* **2025**, arXiv:2410.13919. [CrossRef]
16. Otal, H.T.; Canbaz, M.A. LLM Honeypot: Leveraging Large Language Models as Advanced Interactive Honeypot Systems. In Proceedings of the 2024 IEEE Conference on Communications and Network Security (CNS), Taipei, Taiwan, 30 September–3 October 2024. [CrossRef]
17. Bridges, R.A.; Mitchell, T.R.; Muñoz, M.; Henriksson, T. SoK: Honeypots & LLMs, More Than the Sum of Their Parts? *arXiv* **2026**, arXiv:2510.25939. [CrossRef]
18. Safargalieva, A.; Rüffer, A.; Vasilomanolakis, E. OHRA: Dynamic Multi-protocol LLM-Based Cyber Deception. In Proceedings of the 30th Nordic Conference on Secure IT Systems (NordSec 2025), Tartu, Estonia, 12–13 November 2025. [CrossRef]
19. Ilg, N.; Duplys, P.; Sisejkovic, D.; Menth, M. A Survey of Contemporary Open-Source Honeypots, Frameworks, and Tools. *J. Netw. Comput. Appl.* **2023**, *220*, 103737. [CrossRef]
20. Javadpour, A.; Ja'fari, F.; Taleb, T.; Shojafar, M.; Benzaïd, C. A Comprehensive Survey on Cyber Deception Techniques to Improve Honeypot Performance. *Comput. Secur.* **2024**, *140*, 103792. [CrossRef]
21. Franco, J.; Aris, A.; Canberk, B.; Uluagac, A.S. A Survey of Honeypots and Honeynets for Internet of Things, Industrial Internet of Things, and Cyber-Physical Systems. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 2351–2383. [CrossRef]

22. Yang, X.; Yuan, J.; Yang, H.; Kong, Y.; Zhang, H.; Zhao, J. A Highly Interactive Honeypot-Based Approach to Network Threat Management. *Future Internet* **2023**, *15*, 127. [CrossRef]
23. Diana, L.; Dini, P.; Paolini, D. Overview on Intrusion Detection Systems for Computers Networking Security. *Computers* **2025**, *14*, 87. [CrossRef]
24. Gupta, N.; Jindal, V.; Bedi, P. A Survey on Intrusion Detection and Prevention Systems. *SN Comput. Sci.* **2023**, *4*, 439. [CrossRef]
25. Wang, S.; Tu, C.; Zhang, Y.; Zhang, M.; Xue, P. Mapping Cyber Bot Behaviors: Understanding Payload Patterns in Honeypot Traffic. *Sensors* **2026**, *26*, 11. [CrossRef] [PubMed]
26. Priya, V.S.D.; Chakkaravarthy, S.S. Containerized Cloud-Based Honeypot Deception for Tracking Attackers. *Sci. Rep.* **2023**, *13*, 1437. [CrossRef] [PubMed]
27. Purnama, L.H.; Prasetyo, D.H. Performance and Risk Assessment of Honeypots on IoT and VPS Using COBIT 2019 and Stress Test. *J. Comput. Netw. Archit. High Perform. Comput.* **2025**, *7*, 876–889. [CrossRef]
28. Altunay, H.C. Analysis of Cyber Attacks Using Honeypot. *BSJ Eng. Sci.* **2024**, *7*, 954–959. [CrossRef]
29. Aljunid, E.W.; Qaid, G.R.S. Honeypots Technology in Combat Cybercrimes. *Emir. Int. Univ. J.* **2025**, *3*, 6. [CrossRef]
30. Kosheliuk, V.; Tulashvili, Y. Implementing Honeypots for Detecting Cyber Threats with AWS Using the ELK. *Int. J. Comput.* **2024**, *23*, 618–624. [CrossRef]
31. Hussain, I.; Aldwairi, M.; Wani, A. A Dynamic AI-Powered Honeypot Framework for Adaptive Cyber Threat Diversion. In Proceedings of the 2025 9th Cyber Security in Networking Conference (CSNet), Abu Dhabi, United Arab Emirates, 20–22 October 2025. [CrossRef]
32. Haikal, T.; Hammad, E.; Ismail, S. Characterizing Large-Scale Adversarial Activities Through Large-Scale Honey-Nets. In Proceedings of the 2025 IEEE 16th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), Yorktown Heights, NY, USA, 22–24 October 2025. [CrossRef]
33. Asharf, J.; Moustafa, N.; Khurshid, H.; Debie, E.; Haider, W.; Wahab, A. A Review of Intrusion Detection Systems Using Machine and Deep Learning in Internet of Things: Challenges, Solutions and Future Directions. *Electronics* **2020**, *9*, 1177. [CrossRef]
34. Kikissagbe, B.R.; Adda, M. Machine Learning-Based Intrusion Detection Methods in IoT Systems: A Comprehensive Review. *Electronics* **2024**, *13*, 3601. [CrossRef]
35. Pinto, A.; Herrera, L.-C.; Donoso, Y.; Gutierrez, J.A. Survey on Intrusion Detection Systems Based on Machine Learning Techniques for the Protection of Critical Infrastructure. *Sensors* **2023**, *23*, 2415. [CrossRef] [PubMed]
36. Alwhbi, I.A.; Zou, C.C.; Alharbi, R.N. Encrypted Network Traffic Analysis and Classification Utilizing Machine Learning. *Sensors* **2024**, *24*, 3509. [CrossRef] [PubMed]
37. Khraisat, A.; Alazab, A. A Critical Review of Intrusion Detection Systems in the Internet of Things: Techniques, Deployment Strategy, Validation Strategy, Attacks, Public Datasets and Challenges. *Cybersecur* **2021**, *4*, 18. [CrossRef]
38. Dini, P.; Elhanashi, A.; Begni, A.; Saponara, S.; Zheng, Q.; Gasmi, K. Overview on Intrusion Detection Systems Design Exploiting Machine Learning for Networking Cybersecurity. *Appl. Sci.* **2023**, *13*, 7507. [CrossRef]
39. Azam, Z.; Islam, M.M.; Huda, M.N. Comparative Analysis of Intrusion Detection Systems and Machine Learning-Based Model Analysis Through Decision Tree. *IEEE Access* **2023**, *11*, 80348–80391. [CrossRef]
40. You, H.; Park, C.; Shin, D.; Shin, D. A Comparative Performance Study of Host-Based Intrusion Detection Using TextRank-Based System Call Preprocessing and Deep Learning Models. *Electronics* **2026**, *15*, 1856. [CrossRef]
41. Rehman, H.M.R.U.; Liaquat, S.; Gul, M.J.; Jhandir, M.Z.; Gavilanes, D.; Vergara, M.M.; Ashraf, I. A Systematic Literature Study of Machine Learning Techniques Based Intrusion Detection: Datasets, Models, Challenges, and Future Directions. *J. Big Data* **2025**, *12*, 264. [CrossRef]
42. Kumar, C.M.; Kumar, A.; Devi, B.S.K. Advance Threat Detection Using Machine Learning Techniques with SSH Honeypot: An Integrated Approach. In Proceedings of the 2025 International Conference on Data Science, Agents & Artificial Intelligence (ICDSAAI), Chennai, India, 28–29 March 2025. [CrossRef]
43. Başer, M.; Güven, E.Y.; Aydin, M.A. SSH and Telnet Protocols Attack Analysis Using Honeypot Technique: Analysis of SSH and Telnet Honeypot. In Proceedings of the 2021 6th International Conference on Computer Science and Engineering (UBMK), Ankara, Turkey, 15–17 September 2021. [CrossRef]
44. Azzahri, M.N.; Selian, R.A.; Muchallil, S.; Nurdin, Y.; Afidh, R.P.F.; Umam, K.; Dawood, R. The Application of Cowrie Honeypot to Analyze Attacks on SSH and Telnet Protocols. In Proceedings of the 2024 IEEE 2nd International Conference on Electrical Engineering, Computer and Information Technology (ICEECIT), Jember, Indonesia, 22–23 November 2024. [CrossRef]
45. Zhang, Z.; Li, S.; Zhang, L.; Ye, J.; Hu, C.; Yan, L. LLM-LADE: Large Language Model-Based Log Anomaly Detection with Explanation. *Knowl.-Based Syst.* **2025**, *326*, 114064. [CrossRef]
46. Xu, H.; Wang, S.; Li, N.; Wang, K.; Zhao, Y.; Chen, K.; Yu, T.; Liu, Y.; Wang, H. Large Language Models for Cyber Security: A Systematic Literature Review. *ACM Trans. Softw. Eng. Methodol.* **2025**. [CrossRef]
47. Srinivas, S.; Kirk, B.; Zendejas, J.; Espino, M.; Boskovich, M.; Bari, A.; Dajani, K.; Alzahrani, N. AI-Augmented SOC: A Survey of LLMs and Agents for Security Automation. *J. Cybersecur. Priv.* **2025**, *5*, 95. [CrossRef]

48. Fan, W.; Du, F.; Wang, Z.; Cao, Q.; Liu, L.; Chang, K.; Chang, S.-Y. HoneyLLMd: A Large Language Model-Powered Adaptive Honeypot System. *IEEE Trans. Netw. Sci. Eng.* **2026**, *13*, 8646–8665. [CrossRef]
49. Yan, B.; Li, K.; Xu, M.; Dong, Y.; Zhang, Y.; Ren, Z.; Cheng, X. On Protecting the Data Privacy of Large Language Models (LLMs) and LLM Agents: A Literature Review. *High-Confid. Comput.* **2025**, *5*, 100300. [CrossRef]
50. Huang, H.; Li, Y.; Jiang, B.; Jiang, B.; Liu, L.; Liu, Z.; Sun, R.; Liang, S. A Middle Path for On-Premises LLM Deployment: Preserving Privacy Without Sacrificing Model Confidentiality. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*; Christodoulopoulos, C., Chakraborty, T., Rose, C., Peng, V., Eds.; Association for Computational Linguistics: Suzhou, China, 2025; pp. 8321–8359. [CrossRef]
51. Anh-Hoang, D.; Tran, V.; Nguyen, L.-M. Survey and Analysis of Hallucinations in Large Language Models: Attribution to Prompting Strategies or Model Behavior. *Front. Artif. Intell.* **2025**, *8*, 1622292. [CrossRef] [PubMed]
52. Singh, V.K. Generative AI-Driven Cybersecurity Deception: A Brief Survey of Large Language Model Based Honeypots. In Proceedings of the 2025 IEEE International Conference on Electrical, Electronics, Communication and Computers (ELEXCOM), Dhanbad, India, 20–22 December 2025. [CrossRef]
53. Doost, P.A.; Moghadam, S.S.; Khezri, E.; Basem, A.; Trik, M. A New Intrusion Detection Method Using Ensemble Classification and Feature Selection. *Sci. Rep.* **2025**, *15*, 13642. [CrossRef] [PubMed]
54. Farooqi, A.H.; Akhtar, S.; Rahman, H.; Sadiq, T.; Abbass, W. Enhancing Network Intrusion Detection Using an Ensemble Voting Classifier for Internet of Things. *Sensors* **2024**, *24*, 127. [CrossRef] [PubMed]
55. Joshi, V.R.; Assa-Agyei, K.; Al-Hadhrami, T.; Qasem, S.N. Hybrid AI Intrusion Detection: Balancing Accuracy and Efficiency. *Sensors* **2025**, *25*, 7564. [CrossRef] [PubMed]
56. Ashraf, S.; Choi, C.; Bilal, A. CTDMF-SC: Cybersecurity Threat Detection and Mitigation Framework for Smart Cities. *Int. J. Inf. Secur.* **2026**, *25*, 54. [CrossRef]
57. Sezgin, A.; Ulaş, M.; Boyacı, A. Multi-Objective Feature Selection for Intrusion Detection Systems: A Comparative Analysis of Bio-Inspired Optimization Algorithms. *Sensors* **2025**, *25*, 6099. [CrossRef] [PubMed]
58. Díaz-Verdejo, J.; Muñoz-Calle, J.; Estepa Alonso, A.; Estepa Alonso, R.; Madinabeitia, G. On the Detection Capabilities of Signature-Based Intrusion Detection Systems in the Context of Web Attacks. *Appl. Sci.* **2022**, *12*, 852. [CrossRef]
59. Asad, H.; Adhikari, S.; Gashi, I. A Perspective–Retrospective Analysis of Diversity in Signature-Based Open-Source Network Intrusion Detection Systems. *Int. J. Inf. Secur.* **2024**, *23*, 1331–1346. [CrossRef]
60. Rabzelj, M.; Južnič, L.Š.; Volk, M.; Kos, A.; Kren, M.; Sedlar, U. Designing and Evaluating a Flexible and Scalable HTTP Honeypot Platform: Architecture, Implementation, and Applications. *Electronics* **2023**, *12*, 3480. [CrossRef]
61. Arcos-Argudo, M.; Bojorque, R.; Torres, A. A Deterministic Comparison of Classical Machine Learning and Hybrid Deep Representation Models for Intrusion Detection on NSL-KDD and CICIDS2017. *Algorithms* **2025**, *18*, 749. [CrossRef]
62. Jones, G.; Kasimatis, D.; Pitropakis, N.; Macfarlane, R.; Buchanan, W.J. Analysing the Role of LLMs in Cybersecurity Incident Management. *Int. J. Inf. Secur.* **2025**, *24*, 228. [CrossRef]
63. Var Naseri, M.; Welch, I.; Haseeb, J.; Mansoori, M. Q-Cowrie: An Adaptive Honeypot to Analyse Attackers' Behaviour. *Int. J. Inf. Secur.* **2026**, *25*, 60. [CrossRef]
64. Cowrie. Available online: https://docs.cowrie.org/en/latest/ (accessed on 10 May 2026).
65. Ollama. Available online: https://docs.ollama.com/ (accessed on 10 May 2026).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.