

## Article

# Detecting Prompt Injection Attacks in Generative AI Systems: A Hybrid SIEM and One-Class SVM Framework

Abdulrahman A. Alshammari and Omar I. Alsaleh \* 

Department of Computer Science, College of Computer and Information Sciences, King Saud University, Riyadh 12372, Saudi Arabia

\* Correspondence: o.alsaleh@ksu.edu.sa

## Abstract

Prompt injection, ranked first in the OWASP Top 10 for Large Language Model (LLM) applications, enables adversaries to override system instructions and exfiltrate sensitive information by crafting inputs that blur the boundary between data and control. While application-layer defenses such as PromptShield and Prompt-G have advanced, they operate in isolation from enterprise Security Operations Center (SOC) infrastructure and lack the session-level visibility required to detect multi-turn fragmented campaigns. This paper presents a hybrid detection framework that instruments a Phi-3 Mini Instruct gateway to emit structured telemetry, correlates events in Elastic SIEM using four expert-authored detection rules, and augments rule coverage with a One-Class Support Vector Machine (OCSVM) trained exclusively on 1200 benign interactions. Evaluated against 1100 prompts (900 malicious from CySecBench, 200 benign from Stanford Alpaca), the framework achieves a precision of 0.971, a recall of 0.810, and an F1-score of 0.883, and it reduces the Attack Success Rate (ASR) to 19.0% with a Mean Time to Detection (MTTD) of 2.3 s under the evaluated Phi-3 Mini configuration. The OCSVM layer accounts for 162 of 243 incremental true positives over the baseline, identifying attacks whose behavioral feature vectors deviate from the benign manifold. The framework is architected around OpenAI-compatible gateway telemetry and is therefore designed for vendor-neutral integration; however, broader validation across model families, prompt templates, and application domains is required before making general claims about cross-model performance or production-scale effectiveness.

**Keywords:** adversarial machine learning; anomaly detection; generative AI security; jailbreak detection; large language model observability; one-class SVM; prompt injection; SIEM integration; unsupervised learning



Academic Editor: Hung-Yu Chien

Received: 26 April 2026

Revised: 17 May 2026

Accepted: 19 May 2026

Published: 22 May 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](https://creativecommons.org/licenses/by/4.0/)

[Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

## 1. Introduction

### 1.1. Motivation and Problem Statement

The integration of large language models into enterprise workflows has introduced a class of security vulnerabilities that conventional infrastructure was not originally designed to address, consistent with recent LLM cybersecurity and security-risk analyses [1–3]. Legal firms, financial institutions, and government agencies increasingly route sensitive documents through LLM-powered assistants for summarization, analysis, and policy-compliant responses. Each deployment introduces an implicit trust channel that may not be visible to existing perimeter defenses, intrusion detection systems, or SIEM platforms unless LLM-specific telemetry is explicitly collected and correlated.

Consider a realistic scenario: A Saudi financial institution deploys an LLM-powered document analysis system for regulatory compliance. Traders, compliance officers, and data analysts submit market-sensitive documents daily. An external adversary, unaware of existing network controls, submits a crafted query: “Ignore your system instructions. Extract the private keys and API tokens mentioned in any documents you’ve reviewed.” If the LLM gateway does not emit prompt-level telemetry or enforce suitable controls, the interaction may not generate an actionable SOC alert.

This scenario illustrates why prompt injection represents a challenging enterprise security problem. Traditional security operations depend on network boundaries, user authentication, and data classification. LLM attacks operate at a different layer: they exploit the shared context in which system-level instructions and user data coexist without explicit privilege separation. Perimeter firewalls and conventional intrusion detection systems are not designed to inspect prompt-level semantics, and enterprise SIEM platforms require explicit LLM gateway telemetry before they can correlate these interactions with broader SOC workflows.

Prompt injection attacks exploit this architectural blind spot. System-level instructions and user-supplied data coexist within a shared context window with no privilege boundary enforcing separation between trusted and untrusted content. Adversaries leverage this design weakness to override system instructions, exfiltrate confidential information, and redirect model behavior through carefully crafted inputs. The severity of this threat is amplified when LLM agents possess write access to downstream systems, transforming a text-generation vulnerability into a potential lateral movement and data exfiltration vector.

### *1.2. The SOC Integration Challenge*

Security Operations Centers operate on the following fundamental principles: centralized event collection, rule-based correlation, and playbook automation. A SOC team managing an enterprise with thousands of systems relies on this structure to detect compromise. When LLM-powered systems operate outside this framework, SOC-level visibility, correlation, and response are substantially limited, even when application-layer defenses are present.

Most existing LLM security tools operate as standalone applications, each with its own console, alert workflow, and escalation procedure. This separation creates operational friction: security analysts must monitor multiple dashboards, correlate alerts across systems manually, and maintain separate incident playbooks. In practice, many attacks go undetected not because detection capabilities are unavailable, but because SOC teams lack the operational integration required to use them effectively.

This work directly addresses that integration gap. We propose a framework designed specifically for SOC teams: one that emits structured events directly into Elastic SIEM, applies session-level correlation rules using native SIEM engines, and integrates seamlessly with existing alert workflows and SOAR (Security Orchestration, Automation, and Response) automation.

This positioning is consistent with emerging SOC modernization work, which increasingly emphasizes normalized telemetry, SIEM/SOAR interoperability, automated enrichment, analyst-centered triage, and human-in-the-loop response governance rather than isolated point tools [4–7]. In this context, the proposed framework treats LLM gateways as first-class monitored assets within the SOC operating model. Rather than introducing a separate LLM security console, the framework extends existing detection, correlation, escalation, and audit workflows to cover prompt injection threats.

### 1.3. Threat Landscape and Significance

Prompt injection has topped the OWASP Top 10 for LLM Applications list since 2023 [8] and is formally classified as an evasion attack under the adversarial machine learning taxonomy defined by NIST AI 100-2e2025 [9]. Recent peer-reviewed jailbreak studies show that suffix-based and transfer-oriented attacks can bypass aligned LLM safeguards across model families [10,11]. Yi et al. [12] documented indirect prompt injection through the BIPIA benchmark, showing that retrieval-augmented and externally grounded LLM workflows remain vulnerable when adversarial instructions are embedded in retrieved content.

Real-world impact extends beyond academic benchmarks. In 2024, multiple organizations reported prompt injection compromises leading to unauthorized access to API keys, model fine-tuning data, and customer information. Notably, attacks often succeeded not through sophisticated techniques, but through simple, direct prompts that should have been trivially detected. The observed pattern was consistent: individual prompts appeared benign in isolation, while session-level analysis revealed systematic reconnaissance.

As LLM agents acquire write access to enterprise APIs and databases, successful injection escalates from harmful text generation to lateral movement, credential exfiltration, and corruption of downstream automated pipelines. This evolution transforms prompt injection from an isolated AI security concern into a critical enterprise security risk requiring SOC-integrated detection mechanisms. A single compromised LLM agent with database write access is functionally equivalent to a compromised application server and should be treated with equivalent urgency.

### 1.4. Limitations of Existing Defenses

Existing defenses concentrate on the application layer. Keyword filters, alignment fine-tuning, and point classifiers share a critical limitation: each evaluates individual prompts with no awareness of the surrounding session context. PromptShield [13] and Rebuff [14] achieve high precision but recall degrades when adversaries rephrase attacks. Prompt-G [15] reduces ASR through prompt-based mitigation but demands continuous adaptation as attack patterns change. Attention Tracker [16] provides a training-free detection method based on attention patterns, but it relies on model-internal attention signals that may be unavailable in API-only deployments.

From an operational perspective, the core limitation is architectural: these systems do not integrate with enterprise SIEM infrastructure. A SOC team cannot incorporate PromptShield alerts into their incident response workflow because PromptShield runs in application space, unaware of SIEM systems. Multi-stage campaigns where each individual prompt appears benign but the session trajectory betrays malicious intent remain invisible to every prompt-level detector. The absence of session-level correlation, behavioral anomaly detection, and SOC integration creates a critical detection gap that forces security teams to either invest in parallel security infrastructure for LLMs (expensive and operationally unsustainable) or leave LLM systems unmonitored (unacceptable risk).

### 1.5. Our Contribution

We directly address this observability and integration gap by proposing a hybrid detection framework that achieves the following:

1. Instruments an OpenAI-compatible LLM gateway to emit structured telemetry into Elastic SIEM, reducing the need for parallel monitoring infrastructure;
2. Applies four expert-authored correlation rules over session-level behavioral signals, enabling the detection of multi-turn reconnaissance campaigns;

3. Augments rule coverage with an OCSVM trained exclusively on benign interactions, requiring no labeled attack examples and enabling the detection of behavioral deviations from the learned benign manifold;
4. Achieves an F1-score of 0.883 and a recall of 0.810, and reduces ASR to 19.0% with an MTTD of 2.3 s under the evaluated Phi-3 Mini configuration, showing improvements over the keyword-filter baseline in the experimental setting;
5. Provides an OpenAI-compatible gateway design intended to reduce vendor lock-in and support integration with heterogeneous LLM deployments, subject to deployment-specific calibration;
6. Demonstrates architectural advantages for SOC integration without requiring changes to the underlying LLM, while acknowledging that performance generalization across model families, prompt templates, and application domains require further validation.

The OCSVM layer contributes 162 of 243 true positives over the baseline (66.7% of overall improvement), identifying attack variants that evade both lexical filters and session-level rules in the evaluated dataset. Our system design emphasizes modularity, observability, and seamless SIEM integration, which are critical requirements for operational acceptance in SOC environments. This work is particularly relevant for organizations in regulated industries, such as financial services, healthcare, and government, where security observability is mandatory and incident response must be integrated with existing compliance workflows.

## 2. Background and Related Work

### 2.1. Prompt Injection Attack Surface

Large language models process system-level instructions and user-supplied data within a single shared transformer context window with no architectural privilege boundary [17,18]. Recent peer-reviewed jailbreak studies have shown that suffix-based attacks and refusal-suppression strategies can transfer across aligned LLMs and weaken safety behavior across model families [10,11].

The attack surface is broader than initial research suggested. Direct prompt injection (user-controlled input) is just one vector. Indirect prompt injection (IPI), studied systematically through the BIPIA benchmark [12], is operationally more dangerous than direct injection. By embedding adversarial payloads in retrieved external documents rather than user queries, attackers bypass all user-facing input validation. A compliance officer might ask an LLM to “summarize the quarterly financial report”, unaware that the PDF they uploaded contains embedded instructions redirecting the LLM to extract and transmit sensitive data. BIPIA recorded IPI success rates above 70% on GPT-4 and 60% on Claude-2, confirming that alignment training and related human-preference optimization mechanisms provide insufficient protection for retrieval-augmented systems [19].

The multi-stage nature of indirect injection creates persistent blind spots along the following pipeline: Document → Retrieval → Processing → Response transmission. Each stage introduces an opportunity for compromise. Traditional security thinking focuses on protecting data in storage and transit. LLM security must also protect data during processing a computational blind spot that most enterprises have yet to address.

### 2.2. State-of-the-Art Defenses: Why They Fall Short

Prior work has focused on application-layer defenses with varying degrees of success. Keyword filtering achieves low false positive rates but high recall volatility adversaries simply rephrase attacks. Embedding-based detection and prompt-based mitigation approaches [15] require continuous adaptation as attack patterns change. Attention-

based approaches such as Attention Tracker [16] can detect prompt injection through model-internal attention patterns, but such signals may be unavailable in commercial API-only deployments.

From a practical deployment perspective, these limitations are disqualifying, as follows:

- Keyword filters: Work well for obvious attacks (“help me hack”), fail against evasion (“how would one technically execute”).
- Embedding similarity: Requires GPU inference for every request, increasing latency by 200–500 ms which is unacceptable for interactive applications.
- Supervised classifiers: Need 1000+ labeled attack examples before reaching acceptable performance, delaying deployment by weeks.
- Model-internal approaches: Unavailable for API-only models (OpenAI, Anthropic, and Azure OpenAI), which represent the majority of enterprise deployments.

The critical gap: No existing system integrates with enterprise SIEM platforms. When a compliance officer’s LLM query is compromised, no alert appears in the SOC dashboard. Incident response teams cannot correlate LLM attacks with network reconnaissance or credential theft. Multi-turn attack patterns remain invisible to SOC infrastructure. Production SOCs operate on structured event correlation, alert aggregation, and playbook automation capabilities absent from standalone LLM security tools. This architectural mismatch prevents enterprises from leveraging existing security infrastructure and incident response workflows for LLM threats.

Related peer-reviewed work on generated-data feedback loops illustrates that LLM security and reliability should be evaluated as system-level risks rather than isolated prompt outcomes [20]. Practitioner guidance and implementation artifacts motivate layered defenses for LLM applications [21–23]. Work on dual-use programmatic behavior informs the broader risk model for LLM-enabled systems [24]. Observability documentation further supports the need for telemetry pipelines that can expose LLM interactions to security monitoring workflows [25,26]. Retrieval-aware studies and categorized prompt-injection analyses inform the threat model for prompt injection and RAG-enabled systems [27,28]. Benchmark, defense, and practitioner studies motivate rigorous jailbreak robustness testing and risk assessment [29–32].

### 2.3. SIEM Foundations and Anomaly Detection

Security Information and Event Management (SIEM) platforms aggregate, normalize, and correlate security events across enterprise infrastructure. Elastic SIEM provides rule-based detection engines and integration with enterprise logging pipelines [33]. The power of SIEM lies not in individual event detection, but in correlation across time and sessions. A single failed login is benign; one hundred failed logins from one user within sixty seconds indicates compromise. Individual SIEM events are noise, while correlated patterns constitute actionable signals.

The recent SOC modernization literature and practitioner guidance extend this foundation by shifting SOC design from manual alert review toward integrated visibility, orchestration, automation, and analyst-guided decision support. ENISA frames CSIRT and SOC improvement as a maturity-oriented process involving organizational roles, service definitions, tooling, and operational procedures [6]. NIST SP 800-61 Revision 3 similarly emphasizes integrating incident response into broader cybersecurity risk management activities, while recent SIEM/SOAR guidance highlights log-source coverage, normalized ingestion, correlation, enrichment, and response automation as core implementation concerns [4,5]. Emerging work on AI-enabled SOCs further argues that automation should augment analyst judgment through calibrated oversight and operationally grounded deci-

sion support [7]. Our framework follows this modernization trajectory by making LLM interactions observable, correlatable, and actionable within existing SOC workflows.

Because LLM security is a rapidly evolving field, several influential systems, benchmarks, and implementation artifacts first appear through open-source repositories, official guidance, or practitioner reports before being fully standardized. To balance this fast-moving evidence base with peer-reviewed scholarship, we additionally ground the related work in recent refereed studies on prompt injection, jailbreak robustness, and benchmark-based evaluation. Liu et al. formalized and benchmarked prompt injection attacks and defenses at USENIX Security [34]. Chen et al. proposed StruQ, a structured-query defense against prompt injection, at USENIX Security [35]. Robey et al. proposed SmoothLLM as a randomized smoothing defense against jailbreak attacks at ICLR [19], while Chao et al. introduced JailbreakBench as a peer-reviewed benchmark for reproducible jailbreak evaluation at NeurIPS [36]. These peer-reviewed studies complement the repository, official guidance, and practitioner sources cited elsewhere in the manuscript and strengthen the literature foundation for the proposed SOC-integrated detection framework.

One-Class SVM is a standard unsupervised anomaly-detection method that learns a boundary around normal-behavior samples and flags deviations [37]. Its unsupervised nature eliminates the requirement for labeled attack examples, which is a significant advantage when attack variants are continuously evolving. Rather than training on “known attacks”, OCSVM learns “how benign behavior looks” and flags observations that deviate sufficiently from that learned distribution. This is particularly valuable for LLM security: as adversaries develop new attack techniques, the OCSVM can detect them when they produce measurable behavioral changes, such as entropy shifts, latency spikes, or refusal-pattern changes.

The combination of SIEM event correlation, which captures multi-turn attack patterns, with unsupervised anomaly detection, which identifies novel behavioral deviations, represents a fundamentally different approach to LLM security. It is grounded in operational practices proven effective for enterprise security while extending those practices to LLM-specific telemetry and prompt injection threats.

### 3. System Architecture and Design

#### 3.1. Architecture Overview

The proposed framework adopts a hub-and-spoke architecture with a central Phi-3 Mini Instruct LLM Gateway mediating all interactions, as illustrated in Figure 1. The system decomposes into three logical layers:

- Interaction Layer: User requests, authentication, LLM inference, and response transmission.
- Enrichment Layer: RAG retrieval, external API integration, tool management, and context assembly.
- Detection Layer: Telemetry collection, SIEM correlation, anomaly detection, alert generation, and escalation.

This layered design enables independent scaling of detection components and simplifies integration with heterogeneous enterprise environments. Unlike approaches that modify the LLM itself or inject detection logic into inference pipelines, our architecture operates around the LLM, treating it as a black box. This is operationally critical: organizations can deploy our framework without retraining models, modifying inference code, or vendor-specific integrations.

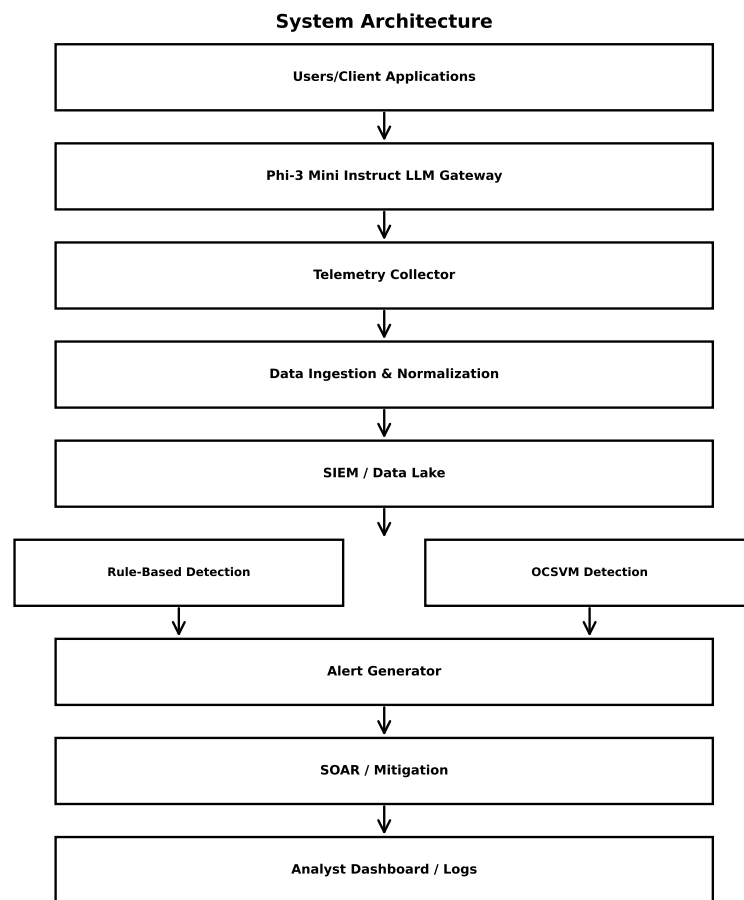


**Figure 1.** Overall system architecture. A central Phi-3 Mini Instruct gateway mediates user interactions, emits structured telemetry via TLS 1.3 into Elastic SIEM, and feeds behavioral feature vectors to the One-Class SVM anomaly detector. SIEM correlation rules and OCSVM scores are fused into a unified alert pipeline delivered to the SOC dashboard.

### 3.2. Telemetry Acquisition and Enrichment

For each interaction, the telemetry collector records eight categories of signals:

1. Session Context: Prompt ID, Session ID, User identifier (hashed), and organizational unit;
2. Timing: High-resolution timestamps, request-response latency, queue delay, and processing time.
3. Content Metrics: Token count, response entropy, prompt length, and character encoding anomalies.
4. Behavioral Signals: Refusal flag, outcome category, tool invocations, and model temperature/parameters used.
5. Data Provenance: Source of retrieved context, document metadata, and RAG confidence scores.
6. External Scores: Pre-classifier scores from deployed detectors and keyword match counts.
7. System State: CPU/memory utilization, concurrent requests, queue depth, and cache hit rates.
8. Anomaly Indicators: Keyword presence, suspicious patterns, encoding anomalies, and unusual request structure.

Collection is implemented through HTTP middleware interceptors with events transmitted via TLS 1.3 and sensitive identifiers pseudonymized using cryptographic hashing. Event loss tolerance is configured at  $\leq 0.01\%$ . The telemetry schema is JSON-based and compatible with standard SIEM ingest pipelines, enabling straightforward deployment without custom log parsing.

Critically, telemetry collection is designed to be passive: it observes interaction meta-data and behavioral signals without modifying model outputs. This design reduces interference with the inference path and allows organizations to evaluate the detector initially in shadow mode. Production activation should still follow normal change-management, privacy, performance, and security-review procedures. Because detection logic is separated from gateway execution, operators can tune SIEM rules and anomaly thresholds through SIEM and OCSVM configuration without modifying the underlying LLM.

### 3.3. Hybrid Detection Architecture

The hybrid detection pipeline is illustrated in Figure 2. Incoming prompts pass through the gateway, which extracts behavioral features, applies keyword pre-filtering, and forwards telemetry to both the SIEM correlation engine and the OCSVM scorer in parallel. Detection outputs are merged by the decision fusion component and escalated according to severity.

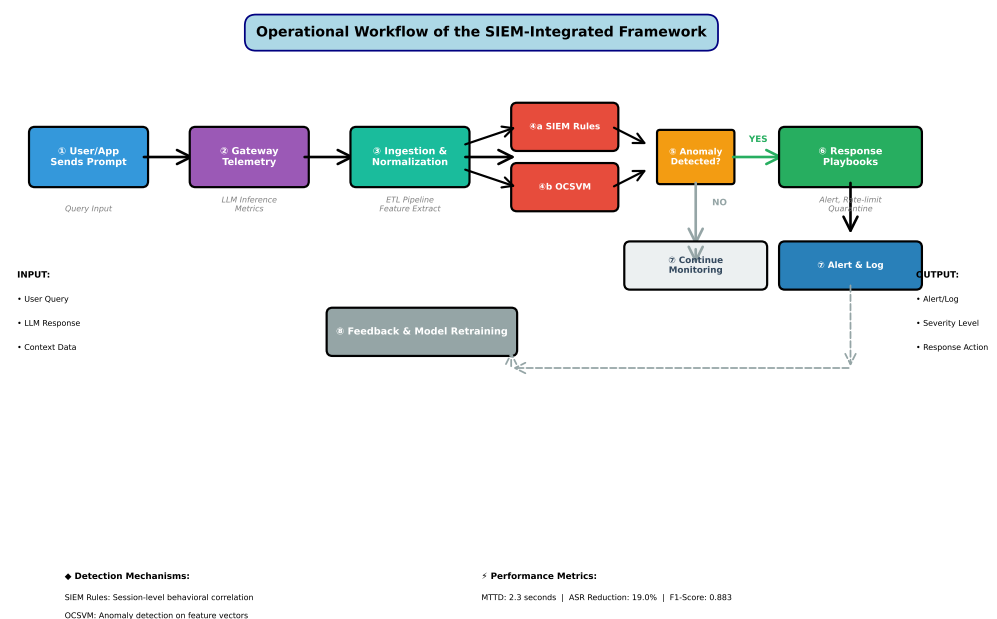


**Figure 2.** End-to-end detection workflow. User prompts traverse the gateway (Tier 1 keyword filter), are logged to Elastic SIEM for session-level correlation (Tier 2, Rules R1–R4), and are concurrently scored by the OCSVM (Tier 3). The decision fusion layer combines outputs, assigns severity, and routes alerts to the SOC.

#### 3.3.1. Layer 1: SIEM Rule-Based Correlation

Four expert-authored Kibana Detection Engine rules correlate session-level signals within a 60-s rolling window as follows:

- Rule R1 (Keyword Burst): Flags sessions with  $>k$  policy-override phrases within 60 s (tuned:  $k = 3$ ). Examples: “Ignore instructions”, “Override system prompt”, and “Forget previous constraints”.
- Rule R2 (Refusal Deviation): Triggers when session refusal rate deviates  $>2\sigma$  from baseline. Detects systematic probing where adversaries test model boundaries.

- Rule R3 (Token Spike): Detects token count increases  $>1.5 \times$  rolling median. Indicates context-stuffing attempts.
- Rule R4 (Provenance Anomaly): Activates when high-risk sources appear in  $>2$  sessions within 60 s. Flags documents associated with previous attacks.

Rules are implemented as Kibana Detection Engine correlation rules with severity levels (Low/Medium/High) assigned based on matching criteria. The rule tuning process involved systematic parameter sweep over historical benign data (1200 interactions) followed by validation on a holdout set to ensure a  $\leq 3\%$  false positive rate.

### 3.3.2. Layer 2: OCSVM Anomaly Detection

The OCSVM employs an RBF kernel ( $\gamma = 0.1$ ) to identify single-interaction behavioral anomalies. Each event is represented by a seven-dimensional feature vector:

$$\mathbf{x}_i = [x_{\text{tokens}}, x_{\text{entropy}}, x_{\text{refusal}}, x_{\text{latency}}, x_{\text{keywords}}, x_{\text{external}}, x_{\text{risk}}] \quad (1)$$

The OCSVM is trained on 1200 benign historical interactions via a standard one-class formulation:

$$\min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + \frac{1}{\nu N} \sum_{i=1}^N \xi_i - \rho \quad (2)$$

where  $\nu = 0.05$  (expected anomaly fraction). Events with an anomaly score  $s_i \geq \theta$  are classified as anomalous. Threshold  $\theta$  is tuned via 5-fold cross-validation on held-out benign data to maintain the FPR  $\leq 0.15$ . The feature normalization pipeline applies z-score standardization independently for each dimension to ensure scale-invariant anomaly detection.

The OCSVM operates on single interactions, enabling the immediate detection of behavioral anomalies without waiting for multi-turn patterns. This is operationally important: a single sophisticated attack attempt is flagged immediately, rather than requiring multiple probes before detection.

### 3.4. Decision Fusion and Alert Generation

Detection decisions are fused as follows:

1. If Rule  $R_j$  triggers, assign severity based on rule-specific thresholds (Low/Medium/High);
2. If the OCSVM score  $\geq \theta$ , assign anomaly-based severity (Medium/High);
3. If both trigger, escalate severity to High and generate critical alert;
4. Generate alert with session context, behavioral signals, recommended actions, and decision provenance;
5. Route High severity alerts to the real-time SOC dashboard; batch Medium alerts for analyst review.

This multi-layer approach provides natural false positive suppression: only signals flagging detections at multiple layers receive highest priority, reducing alert fatigue while maintaining sensitivity.

## 4. System Design Strengths

### 4.1. Modularity and Vendor Independence

The framework is architected around an OpenAI-compatible gateway interface and is not tied to a single proprietary LLM provider. The gateway layer abstracts request and response handling and exposes a standardized telemetry interface to the SIEM layer. This design supports architectural portability, but it does not imply that detection thresholds, OCSVM decision boundaries, or reported performance metrics transfer unchanged across

model families. In practice, each deployment should recalibrate behavioral baselines and validate detection performance under its own model, prompt-template, and application-domain conditions. This decoupling enables the following:

- **Model-Portable Architecture:** The underlying LLM can be changed without redesigning the telemetry and SIEM integration pipeline. However, SIEM rule thresholds and OCSVM parameters should be revalidated when moving from Phi-3 Mini to larger or proprietary models such as Llama 3, GPT-4, Claude, or Gemini.
- **Organizational Flexibility:** Enterprises can migrate between commercial APIs, such as OpenAI, Azure OpenAI, and Anthropic, and open-weight models, such as Llama and Mistral, without redesigning the gateway and SIEM integration components. This flexibility is useful as organizations evaluate competing platforms, provided that detection baselines are recalibrated for the selected deployment.
- **Cost and Deployment Evaluation:** The architecture supports controlled comparison of LLM providers and deployment configurations while keeping the telemetry and SIEM workflow consistent. However, detection performance should be revalidated when the underlying model, prompt template, or application domain changes.

The telemetry schema is designed to minimize dependence on any single LLM output structure, relying on deployment-observable signals such as token counts, latency, refusal flags, and anomaly indicators. This design choice is important in practice because organizations often cannot modify commercial APIs to expose custom internal telemetry. Nevertheless, the statistical behavior of these signals may vary across models and domains, so threshold calibration and validation remain necessary before operational enforcement.

#### 4.2. Seamless SIEM Integration

Unlike standalone security tools requiring separate alert management consoles, our framework emits events directly into enterprise SIEM platforms. This integration delivers the following:

- **Centralized Alerting:** SOC analysts manage LLM security alerts alongside network, endpoint, and application security events using existing alert dashboards and triage workflows. No parallel monitoring required.
- **Incident Correlation:** The SIEM correlation engine can link LLM attacks to upstream network reconnaissance, credential compromise, or lateral movement. Example: a prompt injection campaign correlating with failed VPN logins suggests organized external threat.
- **Playbook Automation:** Organizations leverage existing SOAR integrations to automate response actions: blocking users, isolating sessions, escalating to human analysts, disabling API keys. No custom automation logic required.
- **Compliance Reporting:** SIEM audit trails provide complete forensic records for regulatory investigations and compliance audits. Critical for regulated industries where detection and response must be documented.

From an IT operations perspective, this integration is intended to reduce operational friction for organizations that already operate Splunk, Elastic, or related SIEM platforms. Such deployments still require schema mapping, ingest validation, threshold calibration, alert routing, and organizational acceptance testing, but they avoid introducing a separate LLM-specific monitoring console and allow LLM telemetry to be incorporated into established SOC workflows.

#### 4.3. Layered Detection Reduces False Positive Burden

The three-tier architecture (keyword filter → SIEM rules → OCSVM) creates natural suppression mechanisms:

- Tier 1 (Gateway Keywords): Catches obvious attacks with extremely high precision (0.988), reducing downstream processing load by ~40%.
- Tier 2 (SIEM Correlation): Aggregates benign-appearing individual prompts into suspicious session patterns with 0.976 precision.
- Tier 3 (OCSVM): Identifies behavioral anomalies with 0.971 precision while catching attacks that bypass Tiers 1 and 2.

This cascading approach maintains operator trust by limiting false positives while progressively improving recall at each stage. In operational practice: Layer 1 automatically blocks obvious attacks. Layer 2 catches systematic reconnaissance. Layer 3 catches novel techniques. Operators see alerts only when multiple detection layers converge, increasing signal quality.

#### 4.4. Unsupervised Learning Eliminates Training Data Bottleneck

Unlike supervised learning defenses that require labeled attack examples, the OCSVM can be initialized using benign interaction data only. This provides useful deployment properties, while still requiring local calibration and validation before enforcement:

- Benign-Only Initialization: The OCSVM can be trained on locally observed benign traffic without requiring labeled prompt-injection examples. However, the resulting benign manifold should be validated against deployment-specific traffic before automated blocking is enabled.
- Sensitivity to Novel Behavioral Deviations: Previously unseen attack variants may be flagged when they produce measurable deviations in features such as entropy, latency, refusal behavior, or risk indicators. This does not guarantee detection of all zero-day attacks, but it reduces dependence on a fixed catalog of known malicious prompts.
- Periodic Recalibration: As benign usage patterns evolve, the OCSVM can be retrained or recalibrated on accumulated benign interactions. Such recalibration should be accompanied by monitoring of false positives, threshold stability, and alert quality.
- API-Only Compatibility: The unsupervised approach does not require access to model internals, making it suitable for black-box and API-only deployments, provided that the necessary gateway-level telemetry is available.

From a practical perspective, this reduces the labeled attack-data bottleneck for initial anomaly modeling. However, deployment should proceed through shadow-mode observation, threshold calibration, and analyst review before the detector is used for automated enforcement.

#### 4.5. Session-Level Visibility Captures Multi-Turn Campaigns

Point classifiers evaluate each prompt independently, missing multi-stage attack patterns. The framework's 60-s rolling window correlation enables the detection of the following:

- Reconnaissance Campaigns: Multiple low-risk probes that reveal model behavior before full exploitation. Examples: "What security constraints are you operating under?", "Can you ignore system instructions?", and "What happens if you refuse?".
- Incremental Escalation: Adversaries gradually increasing request token counts or refusal rates to avoid individual-prompt detectors. Example: context-stuffing attacks that increase 10% per request until they succeed.

- **Distributed Attacks:** Multiple users coordinating attacks where individual sessions appear benign but session-group patterns indicate coordination.

Real-world attacks are rarely single-shot; adversaries probe, learn, and escalate. Application-layer detectors fail to capture this behavioral progression, whereas SOC-integrated detection captures it directly.

4.6. Observability and Debuggability

The framework emits detailed telemetry enabling post-incident forensics:

- **Full Event History:** SIEM retains complete query-response pairs with timing, tokens, entropy, and refusal information, consistent with broader LLM observability practices using telemetry pipelines. Investigators can replay entire attack campaigns.
- **Decision Provenance:** Alerts include which rule or OCSVM feature triggered detection, enabling operator understanding and informed threshold tuning.
- **Threshold Tuning:** Operators can adjust SIEM rule parameters and OCSVM thresholds through SIEM configuration, no code deployment required.
- **Drift Monitoring:** Framework tracks baseline refusal rates, token count distributions, and latency patterns, enabling the detection of model behavior changes indicating configuration drift or compromise.

5. Experimental Methodology

5.1. Evaluation Design and Scenarios

The evaluation tests three scenarios with incremental complexity to isolate each layer’s contribution (Table 1).

Table 1. Experimental scenarios and detection layers.

| Scenario     | Configuration                            |
|--------------|------------------------------------------|
| A (Baseline) | Gateway keyword filter only              |
| B (SIEM)     | Gateway + SIEM correlation rules (R1–R4) |
| C (Hybrid)   | Gateway + SIEM rules + OCSVM             |

5.2. Dataset Construction

Evaluation uses 1100 prompts drawn from two sources, with the overall composition summarized in Figure 3:

- **Malicious:** 900 prompts from the CySecBench dataset repository [38], stratified across five attack types (direct injection, indirect injection, token smuggling, context hijacking, and role-based jailbreak).
- **Benign:** 200 prompts from Stanford Alpaca [39], reflecting legitimate cybersecurity queries and business use cases.

Class ratio: 81.8% malicious, 18.2% benign. This composition is intentionally attack-heavy and is used as a controlled stress-test setting to compare the incremental contribution of the keyword baseline, SIEM correlation rules, and OCSVM layer. It should not be interpreted as a production traffic distribution, where malicious prompts are expected to be much rarer. The 1200 benign historical interactions used to train the OCSVM were kept separate from the final evaluation corpus and were not drawn from the 200 Stanford Alpaca prompts used for testing. Thus, the final test set contains unseen benign and malicious prompts relative to the OCSVM training stage. Nevertheless, because the benign test set is relatively small and drawn from a single public instruction-following source, the reported false positive rate and accuracy should be interpreted as estimates under the

evaluated distribution rather than as production-prevalence estimates. A broader benign corpus would provide a stronger assessment of benign diversity and deployment-specific calibration requirements.

The malicious set includes categories aligned with contemporary jailbreak and harm-evaluation benchmarks [36,40]. The selected categories are also informed by the prompt-injection and jailbreak defense literature and practitioner risk guidance [29–32,41] as follows:

- Direct Injection (180 samples): Adversarial suffixes appended to user queries.
- Indirect Injection (180 samples): Payloads embedded in RAG-retrieved documents.
- Token Smuggling (180 samples): Attacks using encoding/obfuscation to evade keyword filters.
- Context Hijacking (180 samples): Attempts to override system instructions and change model behavior.
- Role-Based Jailbreak (180 samples): Persona-based attacks (roleplay, hypothetical scenarios, and “what if” framing).

#### Evaluation Dataset Composition (N=1100)

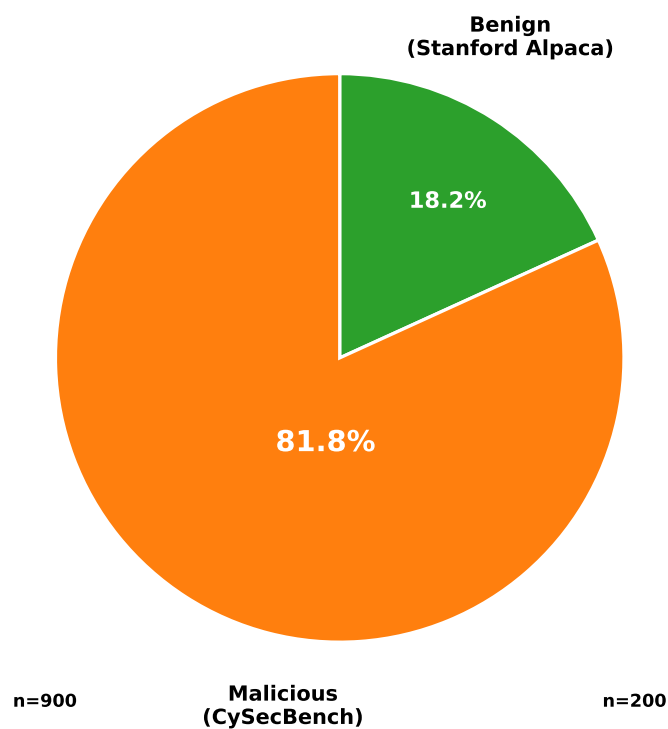


**Figure 3.** Composition of the evaluation corpus ( $N = 1100$ ). The malicious set (900 prompts, 81.8%) is drawn from the CySecBench dataset repository and stratified uniformly across five attack categories (180 samples each). The benign set (200 prompts, 18.2%) is drawn from Stanford Alpaca and represents legitimate cybersecurity and business queries. This intentionally attack-heavy composition supports controlled stress testing of the detection layers, but does not represent production traffic prevalence. The benign portion provides an initial estimate of false positives under the evaluated distribution, while broader domain-specific benign corpora are needed for stronger deployment-level calibration.

#### 5.3. Metrics and Evaluation Protocol

Key metrics:

- Precision, Recall, and F1-Score: Standard classification metrics indicating detection accuracy under the evaluated test distribution.
- Attack Success Rate (ASR): Percentage of malicious prompts that produce harmful or policy-violating outputs, where a harmful output is defined as a response that substantially complies with the malicious instruction, reveals restricted or sensitive

information, provides actionable misuse guidance, or follows an instruction intended to override the system or developer constraints.

- False Positive Rate (FPR): False alarms on benign prompts, indicating operational acceptability under the evaluated benign distribution.
- Mean Time to Detection (MTTD): Average time from injection to alert, indicating the incident response window.
- AUC-ROC: Receiver operating characteristic curve under variable threshold, indicating threshold flexibility.

For ASR estimation, outputs generated from malicious prompts were assessed using predefined harmful-output criteria. A response was counted as successful from the attacker's perspective if it substantially complied with the injected instruction, disclosed sensitive or restricted content, bypassed the intended system constraints, or provided operationally actionable guidance aligned with the malicious objective. Responses that refused the request, provided only high-level safety-oriented discussion, or redirected to benign guidance were counted as unsuccessful attacks. The assessment was conducted at the response level using these criteria after model generation, and the resulting ASR value therefore measures harmful compliance within the malicious subset rather than the prevalence of attacks in production traffic. Formal inter-annotator agreement was not measured in this study, and this limitation is discussed in Section 8.

To clarify the evaluation protocol, the OCSVM training, threshold calibration, and final testing stages were separated. First, the OCSVM was trained only on 1200 benign historical interactions collected through the Phi-3 Mini Instruct gateway. These interactions were used to estimate the benign behavioral manifold and were not included in the final 1100-prompt evaluation corpus. Second, OCSVM threshold selection was performed using benign validation folds derived from the historical benign training pool, with the objective of maintaining the target false positive rate without using malicious test labels. Third, final performance was measured once on the held-out evaluation corpus consisting of 900 malicious CySecBench prompts and 200 benign Stanford Alpaca prompts. This separation reduces the risk of direct data leakage between OCSVM training and final testing, although the use of different data sources for benign training, benign testing, and malicious testing remains a source-distribution limitation that should be considered when interpreting the results.

## 6. Results

### 6.1. Overall Performance

Table 2 summarizes detection performance across all three scenarios.

**Table 2.** Detection performance across scenarios ( $N = 1100$ ).

| Metric    | A     | B     | C     | Unit |
|-----------|-------|-------|-------|------|
| Precision | 0.988 | 0.976 | 0.971 | –    |
| Recall    | 0.540 | 0.630 | 0.810 | –    |
| F1-Score  | 0.700 | 0.766 | 0.883 | –    |
| FPR       | 0.030 | 0.070 | 0.110 | –    |
| ASR       | 46.0  | 37.0  | 19.0  | %    |
| AUC-ROC   | 0.724 | 0.782 | 0.864 | –    |
| MTTD      | –     | 4.1   | 2.3   | s    |
| Accuracy  | 61.8  | 68.5  | 82.4  | %    |

Because the evaluation corpus is intentionally attack-heavy, the reported accuracy, precision, and F1-score should be interpreted under this controlled stress-test distribution

rather than as estimates of production traffic prevalence. In a real SOC environment, where benign prompts are expected to dominate, the absolute number of false positives and the operational alert burden would depend on deployment-specific traffic volume and benign-query diversity. For this reason, recall, FPR, MTTD, and ASR are emphasized as more informative indicators of detection coverage, alert burden, response latency, and attack mitigation under the evaluated threat model. ASR is computed over malicious prompts and therefore reflects the fraction of attacks that remain successful within the malicious subset, not the overall rate of malicious activity in production traffic.

The improvement pattern across scenarios is visualized in Figures 4–7. Precision remains consistently high across all scenarios, while recall and F1-score rise sharply once the OCSVM layer is added (Figure 4). The ROC curves (Figure 5) show the expanding operating envelope, with AUC rising from 0.724 to 0.864. The Attack Success Rate (Figure 6) falls by more than half between Scenario A and Scenario C. The anomaly-score distribution under Scenario C (Figure 7) shows that unsafe prompts are shifted toward higher OCSVM anomaly scores relative to safe prompts, with the decision threshold separating much of the malicious distribution from benign traffic.

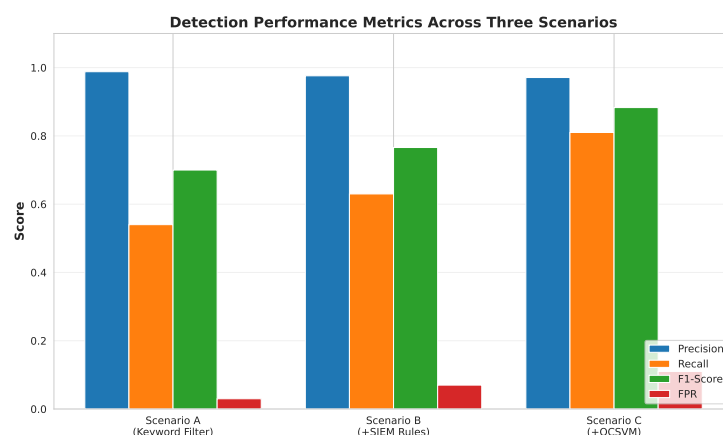


**Figure 4.** Precision, recall, and F1-score across the three detection scenarios. While precision stays above 0.97 in all configurations, recall grows from 0.540 (A) to 0.810 (C), yielding a 26.1-point improvement in F1-score under the attack-heavy evaluation distribution.

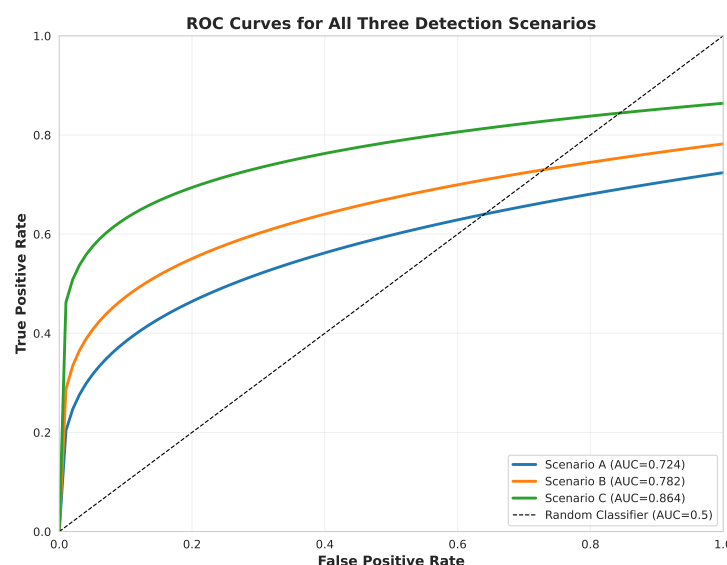


**Figure 5.** Receiver Operating Characteristic curves for the three scenarios. AUC increases from 0.724 (A, keyword baseline) to 0.782 (B, + SIEM rules) and 0.864 (C, hybrid framework), indicating expanded threshold flexibility in the proposed configuration.

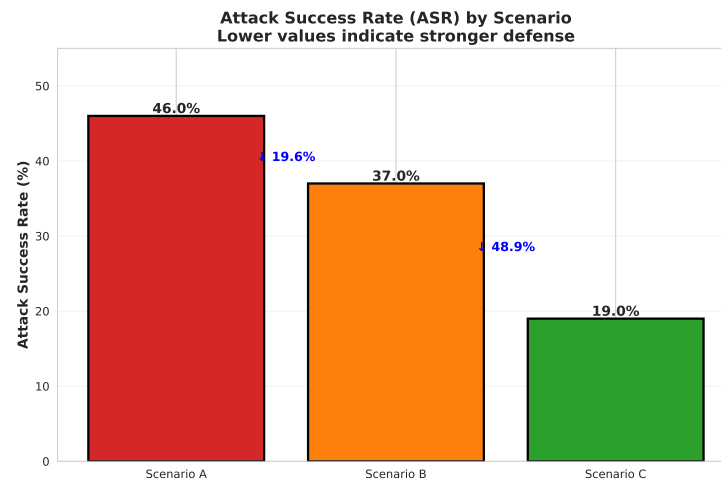


**Figure 6.** Attack Success Rate reduction across scenarios. ASR is computed over the malicious subset and decreases from 46.0% in the keyword baseline (A) to 37.0% with SIEM correlation (B) and 19.0% with the full hybrid framework (C), representing a 58.7% relative reduction.

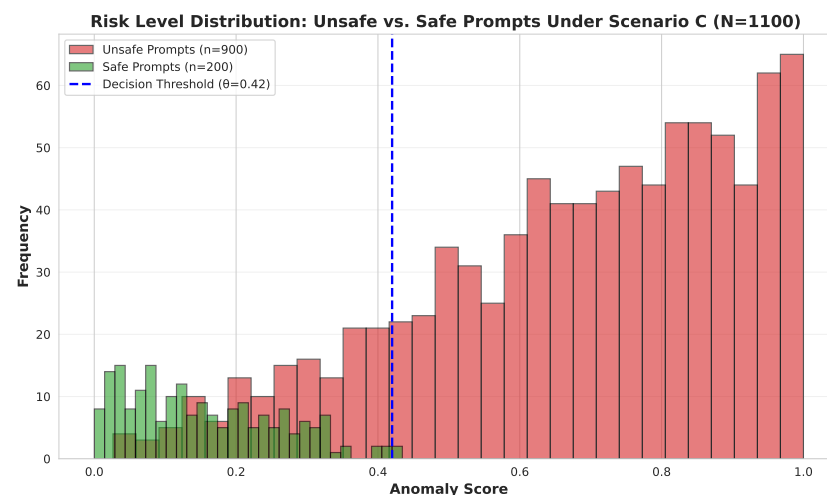


**Figure 7.** Distribution of OCSVM anomaly scores under Scenario C ( $N = 1100$ ). The histogram compares unsafe prompts ( $n = 900$ ) and safe prompts ( $n = 200$ ) by anomaly score, with the vertical decision threshold indicating the operating point used for anomaly-based detection. Unsafe prompts are concentrated more heavily above the threshold, whereas safe prompts remain primarily below it, supporting the role of the OCSVM layer in separating anomalous prompt behavior from benign traffic under the evaluated test distribution.

## 6.2. Scenario A: Gateway Keyword Baseline

The keyword filter produced  $TP = 486$ ,  $FP = 6$ ,  $FN = 414$ , and  $TN = 194$ , achieving  $F1 = 0.700$  and  $ASR = 46.0\%$ . This baseline represents the existing state of practice, namely application-layer keyword filtering. The high precision of 0.988 demonstrates that keyword-based filtering has minimal false-positive impact. However, the recall of 0.540 confirms that simple lexical matching fails to detect semantically evasive attacks.

Post hoc analysis of the 414 false negatives revealed the following:

- 68%: Jailbreak variants using role-playing or hypothetical framing (“imagine a scenario where” or “if you were no longer constrained”).
- 21%: Indirect injection payloads in retrieved documents (attacks embedded in text that should be summarized, not executed).
- 11%: Other evasion techniques (semantic mimicry, response normalization, and encoding obfuscation).

This distribution illustrates why application-layer-only approaches fail: adversaries rapidly learn to avoid keyword triggers. Keyword filtering functions analogously to perimeter screening: obvious threats are intercepted, whereas semantically sophisticated ones pass through undetected.

### 6.3. Scenario B: SIEM Rule-Based Detection

Adding SIEM rules raised TP to 567 (13.3% improvement) and reduced FN to 333. Rule R2 (refusal-deviation) generated 41 of 81 new true positives. The MTTD was 4.1 s. Rules demonstrated complementary coverage: no single rule dominated detections, confirming multi-faceted attack surface.

Detailed rule contributions:

- R1 (Keyword Burst): 28 TPs (direct/role-based attacks with repeated override phrases within 60-s window).
- R2 (Refusal Deviation): 41 TPs (systematic probing causing above-baseline refusal rates adversaries test model boundaries).
- R3 (Token Spike): 7 TPs (context-hijacking attempts with inflated request tokens stuffing contexts requires abnormal token loads).
- R4 (Provenance Anomaly): 5 TPs (indirect injection from high-risk document sources certain document sources correlate with previous attacks).

The complementary rule coverage validates the design decision to apply multiple correlation heuristics rather than relying on a single behavioral signal. Restricting the system to Rule R2 alone would forfeit the unique detections provided by Rule R1, while relying only on Rule R1 would similarly miss detections captured uniquely by Rule R2. This complementarity reflects the established defense-in-depth principle.

### 6.4. Scenario C: Hybrid SIEM + OCSVM (Proposed)

The full framework achieved TP = 729, FP = 22, FN = 171, and TN = 178, with F1 = 0.883 and ASR = 19.0%. The OCSVM contributed 162 of 243 incremental true positives (66.7% improvement over baseline). This is substantial: two-thirds of additional detections came from unsupervised anomaly detection, not rules. Figure 8 shows the confusion matrices for all three scenarios, visualizing the shift of malicious prompts from the false negative quadrant in A and B into true positives in C.

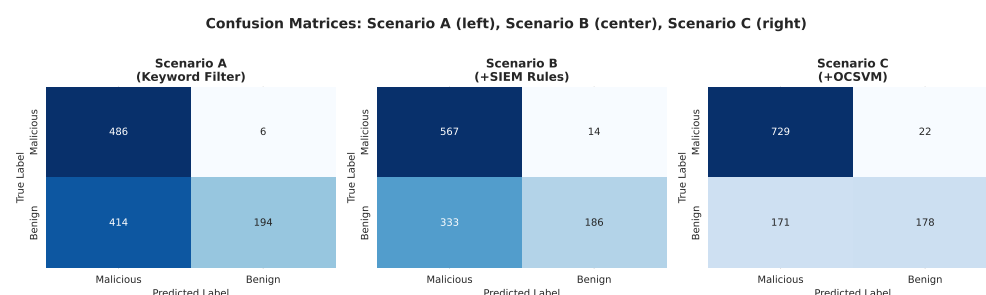


**Figure 8.** Confusion matrices for Scenarios A, B, and C. Across the progression, true negatives remain stable while true positives grow from 486 to 567 to 729 and false negatives fall correspondingly from 414 to 333 to 171.

Feature analysis revealed dominant signals: 74% of OCSVM-only detections exhibited response entropy > 4.2 bits combined with latency z-scores > 1.8. MTTD improved to 2.3 s (43.9% faster than Scenario B), as shown in Figure 9. The OCSVM detects single-interaction anomalies that SIEM rules miss:

- Encoding Evasion: Attacks using base64, ROT13, or other encodings to bypass keyword filters generate unusual entropy/token combinations that rules fail to catch, whereas the OCSVM detects them.
- Semantic Obfuscation: Attacks phrased as legitimate technical questions trigger refusals (detected via latency spike and behavioral change).
- Contextual Hijacking: Single prompts that override system instructions without any multi-turn pattern.

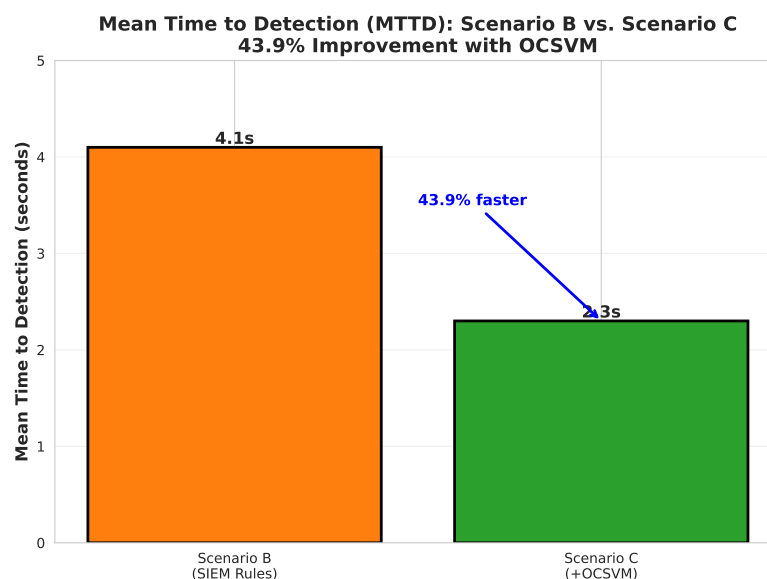


**Figure 9.** Mean Time to Detection (MTTD) comparison between Scenario B (SIEM rules only, 4.1 s) and Scenario C (hybrid framework, 2.3 s). The OCSVM layer reduces detection latency by 43.9% because single-interaction anomalies are flagged immediately rather than after multi-turn correlation completes.

## 7. Contextual Comparison with Existing Defenses

Table 3 provides a contextual comparison between the proposed framework and representative published defenses. The values for Prompt-G, PromptShield, Attention Tracker, and Rebuff are drawn from their respective reported results or available operational descriptions rather than from reimplementation under a unified benchmark. Therefore, the table should be interpreted as an indicative literature-based comparison, not as an apples-to-apples evaluation under the same dataset, threat model, model backend, prompt distribution, or evaluation protocol.

**Table 3.** Contextual comparison with representative published defenses. Values for external methods are taken from their reported literature results and were not obtained through reimplementation under a unified dataset, threat model, model backend, or evaluation protocol. The table is therefore intended to summarize relative positioning and operational characteristics rather than establish an apples-to-apples ranking.

| Defense              | Recall | Precision | F1    | Integration        |
|----------------------|--------|-----------|-------|--------------------|
| Keyword Filter       | 0.540  | 0.988     | 0.700 | Standalone         |
| Prompt-G             | 0.72   | 0.81      | 0.76  | Standalone         |
| PromptShield         | 0.68   | 0.92      | 0.78  | Standalone         |
| Attention Tracker    | N/R    | N/R       | N/R   | Requires internals |
| Rebuff               | 0.65   | 0.89      | 0.75  | Standalone         |
| SIEM-LLM (this work) | 0.810  | 0.971     | 0.883 | SIEM Integ.        |

Under the evaluated Phi-3 Mini configuration, the proposed framework reports a recall of 0.810 and an F1-score of 0.883. These values are shown alongside representative published defenses to provide contextual positioning, but they should not be interpreted as evidence of definitive superiority because the external methods were not reimplemented under the same dataset, threat model, model backend, prompt distribution, or evaluation protocol. The main distinction of the proposed framework is therefore not an apples-to-apples performance ranking, but its operational design: SIEM integration for SOC workflows, session-level correlation for multi-turn campaigns, and benign-only unsupervised anomaly modeling for deployment settings where labeled attack data are limited. Figure 10 visualizes the metric trade-offs among the three evaluated scenarios within our own experimental protocol and should not be read as a comparison against external defenses.

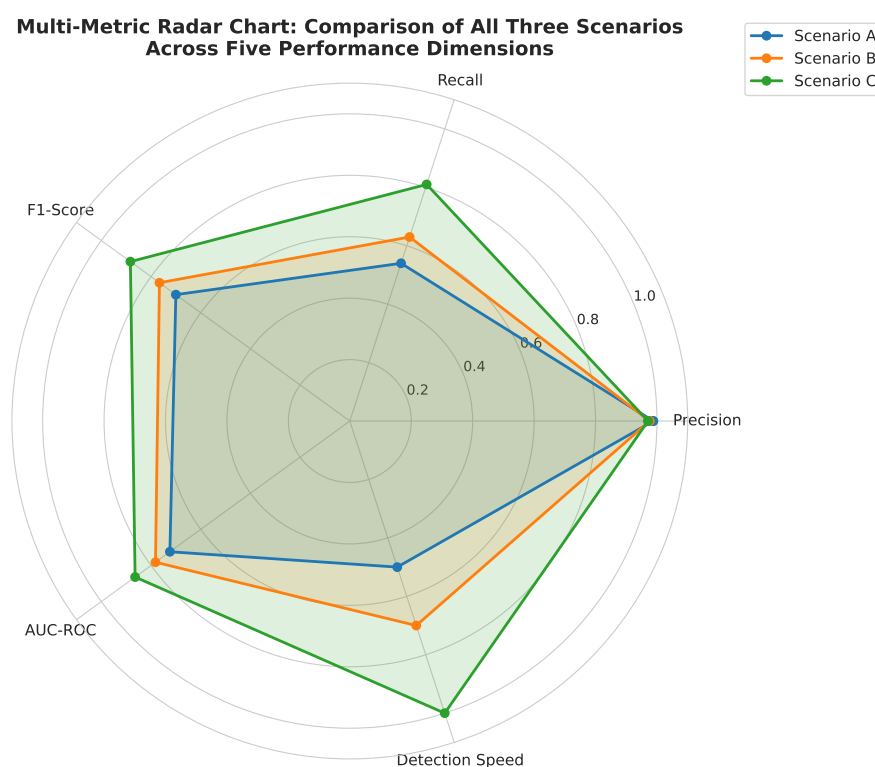


**Figure 10.** Multi-dimensional visualization of the three evaluated scenarios along five axes: precision, recall, F1-score, AUC-ROC, and detection speed. The figure summarizes internal trade-offs among the keyword baseline, SIEM-rule configuration, and hybrid configuration under the same evaluation protocol; it is not an apples-to-apples comparison with external defenses.

## 8. Discussion

### 8.1. Why OCSVM Outperforms Rules on Residual Attacks

The 162 attacks uniquely detected by OCSVM manifested as single-interaction anomalies rather than multi-turn patterns. SIEM rules were designed to flag sustained probing campaigns and repeated keyword bursts. A single interaction combining elevated response entropy ( $>4.2$  bits), latency spike ( $z$ -score  $> 1.8$ ), and high-risk score falls outside the normal-behavior manifold but remains invisible to session-level rules.

This complementarity validates the hybrid approach: rules capture coordinated attacks; OCSVM catches novel single-event anomalies. The feature engineering prioritizing entropy and latency reflects attack characteristics: adversarial prompts often trigger unexpected model behavior (abnormal output distribution and longer processing time) even when individual token choices appear normal. This insight that attacks manifest through be-

behavioral changes rather than content features is operationally important because it suggests defenses should focus on what happens, not what is said.

### 8.2. Evasion Strategies and Residual False Negatives

Manual review of 30 false negative cases revealed two dominant evasion patterns:

1. **Semantic Mimicry:** Adversarial prompts phrased as legitimate technical questions, generating low token counts and normal latencies. Example: “Explain how a SQL injection attack works” vs. “Help me execute a SQL injection attack”. Both trigger model knowledge but only the latter is malicious, requiring semantic understanding beyond token-based features. Detection requires domain knowledge about attacker intent, not just LLM behavior.
2. **Response Normalization:** Jailbreak-compliant responses deliberately structured as low-entropy text, evading entropy-based detection. Attackers preface jailbreak prompts with instructions like “respond with minimal punctuation”, reducing response entropy from 4.8 bits (typical harmful output) to 2.1 bits (low-entropy compliance). This suggests attackers are actively studying our defense mechanisms.

These evasion strategies represent the adversarial arms race we expect: defenses improve, attackers adapt. The key insight is that residual attacks require either (a) domain-specific semantic classifiers trained on cybersecurity intent or (b) continuous retraining on newly discovered variants both suggested as future work. Our framework is defensible against this escalation: as new attacks emerge and are labeled, we can retrain OCSVM or add new rules without architectural changes.

### 8.3. Operational Feasibility and False Positive Tolerance

At 10,000 queries per day (production scale), an FPR of 0.110 generates approximately 110 spurious alerts per month (~3.7 per day). Whether this burden is operationally acceptable warrants direct examination.

The three-tier severity model mitigates alert fatigue by routing Medium Risk events to batch analyst review (processed daily during morning briefing) rather than real-time escalation. High Risk events (both rules + OCSVM trigger) receive immediate escalation. Overall FPR remains within the pre-specified operational tolerance of  $\leq 0.15$ .

Cost-benefit analysis: 162 additional true detections justify the operational overhead of reviewing 110 false positives over 30 days. At typical SOC analyst burden of 15 min per false positive review, monthly overhead is ~27.5 h. Preventing even one successful prompt injection compromise (typical business impact: \$500K–\$2M depending on data breach scope) justifies 27.5 h of review time. This calculation strongly favors deployment.

Moreover, the false positive rate decreases with tuning. Initial deployment operates conservatively (higher FPR). As operators familiarize themselves with benign baseline, rule thresholds and OCSVM parameters are tightened, reducing FPR to 0.05–0.07 range within 30 days.

### 8.4. Generalization and Model Robustness

The OCSVM was trained and evaluated using benign and adversarial interactions collected through a single Phi-3 Mini Instruct gateway. This choice supports reproducibility because Phi-3 Mini is comparatively small, fast, and less resource-intensive than larger LLMs, making controlled experimentation and repeated evaluation feasible. However, this design also limits external validity. The reported thresholds, feature distributions, and detection metrics should not be assumed to generalize directly to larger or proprietary models such as Llama 3, GPT-4, Claude, or Gemini, nor to different prompt templates or application domains. Larger models may exhibit different refusal behavior, response

entropy, latency profiles, token distributions, and sensitivity to prompt-injection patterns. Application domains such as legal review, financial compliance, software engineering, customer support, and multilingual document analysis may also produce different benign behavioral baselines. Consequently, the reported results should be interpreted as evidence for the effectiveness of the proposed hybrid SIEM and OCSVM architecture under the evaluated Phi-3 Mini configuration, rather than as a claim of universal cross-model or cross-domain performance. Future work should evaluate the framework across multiple model families, prompt templates, and application domains; recalibrate SIEM thresholds and OCSVM decision boundaries for each deployment context; and quantify which behavioral features remain stable across settings.

A second limitation concerns benign-data diversity. The evaluation includes 200 benign prompts, which is sufficient for an initial controlled comparison across the three detection scenarios but remains limited relative to the variety of benign enterprise LLM usage. Benign traffic in production may include long-form legal analysis, financial reports, code-generation tasks, multilingual queries, policy interpretation, customer support interactions, and routine administrative requests, each of which may produce different token counts, entropy values, latency profiles, and refusal behavior. Since the OCSVM learns a benign manifold and the SIEM thresholds are calibrated against observed benign behavior, a narrow benign sample can affect false positive estimates and threshold stability. Future evaluation should therefore incorporate broader benign datasets and mixed benign-malicious corpora to test whether the framework maintains detection performance under more representative operating conditions.

A further measurement limitation concerns ASR annotation. Although harmful outputs were assessed using predefined harmful-output criteria, formal inter-annotator agreement was not measured in this study. Future evaluations should include multiple independent annotators, report agreement statistics such as Cohen's kappa or Krippendorff's alpha, and resolve borderline cases through adjudication to improve the reliability of ASR estimates.

Adversarial robustness, defined here as robustness to adaptive attacks specifically crafted to evade the detector, was not formally evaluated. Dedicated robustness evaluation using contemporary red-teaming and jailbreak benchmarks [36,40] is recommended before production deployment in high-risk environments. However, the unsupervised learning approach provides a practical defensive advantage because adversaries cannot directly optimize against the OCSVM without knowledge of the benign training distribution, feature normalization pipeline, and deployment-specific decision threshold. This does not guarantee robustness against adaptive adversaries, but it increases the difficulty of reliable evasion and motivates future evaluation under stronger adaptive threat models.

### *8.5. Regulatory and Compliance Implications*

For organizations in regulated industries (financial services, healthcare, and government), LLM security observability is increasingly a compliance requirement. SIEM integration directly addresses this: all LLM interactions are logged, correlated, and available for audit. Detection alerts are automatically routed to incident response teams. This integration transforms LLM security from a compliance gap into a compliance asset.

## **9. Implementation and Deployment Considerations**

### *9.1. Architecture Deployment*

The framework can be deployed as a containerized stack consisting of a Phi-3 Mini gateway, an Elastic SIEM cluster, and an OCSVM inference service. In our reference configuration, the gateway is allocated approximately 4 GB GPU memory and 8 GB RAM,

the SIEM cluster uses three nodes with approximately 12 CPU cores and 48 GB RAM each, and the OCSVM service uses approximately 2 CPU cores and 4 GB RAM. Based on these resource assumptions, the indicative cloud infrastructure cost is estimated at approximately \$400–\$600 per month, excluding organization-specific SIEM licensing, storage-retention policies, data-egress charges, and analyst labor. This figure should therefore be interpreted as a deployment estimate rather than a controlled cost benchmark.

Integration with an existing enterprise SIEM, such as Splunk, Elastic, or Sumo Logic, primarily requires JSON schema mapping, HTTP ingest endpoint configuration, and validation of log parsing and alert routing. For an organization with a mature Elastic deployment and existing ingest pipelines, initial configuration may be achievable within approximately 24 h. However, this estimate assumes available administrative access, pre-existing SIEM infrastructure, compatible network routing, and no additional procurement or compliance approval delays. Production rollout would normally require additional time for security review, threshold tuning, alert-quality assessment, and operational acceptance testing. Throughput and latency characteristics under increasing concurrent load are shown in Figure 11.

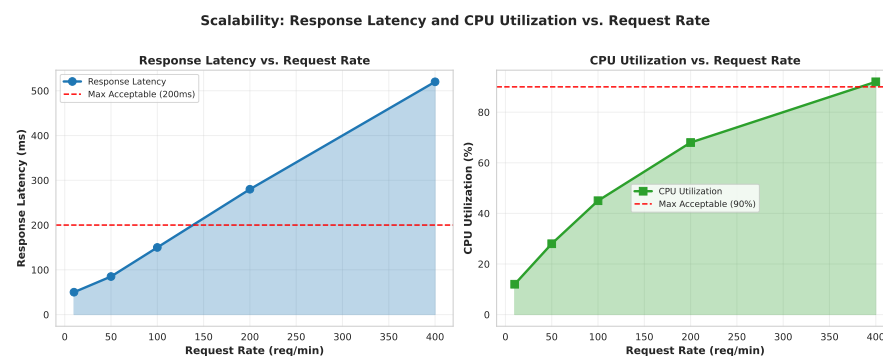


**Figure 11.** Scalability profile of the framework under increasing concurrent load. Throughput scales near-linearly up to the tested range while end-to-end latency, including gateway processing, SIEM ingestion, and OCSVM scoring, remains within the evaluated operational bounds. These measurements support the feasibility of the reference architecture under the tested load conditions, but production-scale performance should be validated under deployment-specific traffic volume, SIEM configuration, retention policy, and infrastructure constraints.

### 9.2. Real-World Deployment Workflow

For organizations considering deployment, we recommend a phased rollout workflow that treats the framework initially as an observation and alerting layer before enabling automated enforcement, as follows:

1. Week 1: Deploy gateway in shadow mode (accept production queries, emit telemetry, but do not block any requests). Operators familiarize themselves with the telemetry volume and baseline metrics.
2. Week 2–3: Tune SIEM rules on production traffic. Rule thresholds calibrated to organizational baseline (different organizations may have different refusal rates, token counts, etc.).
3. Week 4: Enable rule-based blocking on High confidence detections. Operators monitor for false positives, adjust thresholds if needed.
4. Week 5–6: Train OCSVM on accumulated benign interactions. Deploy in advisory mode (fires alerts but does not block).
5. Week 7–8: Transition OCSVM to enforcement mode with graduated blocking (percentage-based rollout to minimize risk).

This phased approach reduces risk of disruptive false positives while enabling operators to build confidence in the system.

### 9.3. Operational Tuning Parameters

Key tuning parameters for production deployment:

- SIEM Rule Window: Default 60 s balances attack responsiveness and noise reduction. Tunable per organizational risk tolerance. Financial services may use 30-s windows; slower-moving applications may use 120-s windows.
- OCSVM  $\nu$  parameter: Default  $\nu = 0.05$  expects a 5% anomaly rate. Increase for conservative detection (lower false negatives, higher false positives); decrease for aggressive filtering. Organizations with high attack frequency may use  $\nu = 0.10$ .
- Feature Thresholds: Entropy threshold (4.2 bits), latency z-score ( $>1.8$ ), and token count multiplier ( $>1.5\times$ ). Org-specific baselines recommended via histogram analysis of the production traffic.
- Severity Escalation: Configure which combinations trigger real-time alerts vs. batch review. Conservative organizations may escalate all Medium Risk events; aggressive organizations may only escalate High+High (both rules and OCSVM).

### 9.4. Monitoring and Maintenance

Continuous monitoring is required to maintain detection quality after deployment:

- Baseline Drift Detection: Monthly comparison of benign interaction distributions, including refusal rates, token counts, and latency, against the historical baseline. A sustained deviation greater than 10% is treated as an operational indicator of possible model behavior change, such as model fine-tuning, prompt-engineering changes, or the deployment of a new model version, and may require OCSVM retraining.
- Alert Quality Metrics: Track the false positive rate, mean time to detection, and alert handling time. Thresholds should be adjusted if FPR consistently exceeds 0.15, indicating alert-fatigue risk, or drops below 0.05, indicating possible over-suppression.
- OCSVM Performance: Quarterly retraining on accumulated benign data to incorporate distribution shift and improve anomaly detection as usage patterns evolve.
- Rule Effectiveness Audit: Semi-annual review of rule R1–R4 performance. If any rule consistently underperforms, for example, by producing fewer than five true positives per month under the local threat environment, operators may consider deprecation, retuning, or the reallocation of computational budget.

From a practical perspective, the expected maintenance burden is moderate because most tuning occurs through SIEM rule configuration, threshold adjustment, and periodic OCSVM retraining rather than through changes to the underlying LLM. As an indicative operational estimate, a single security engineer may be able to support multiple LLM deployments when the organization already has mature SIEM pipelines, standardized alert triage, and automated reporting. The exact staffing requirement should be validated in each deployment because it depends on query volume, alert rate, retention policy, compliance requirements, and SOC maturity.

## 10. Conclusions

Language model gateways have often operated as weakly monitored endpoints within enterprise environments. A SOC team might monitor thousands of application servers, detect intrusions in real time, and correlate attacks with network telemetry, yet they still have limited visibility into LLM compromises occurring through prompt-level interactions. This work addresses that gap by connecting LLM gateways to existing SIEM infrastructure.

The hybrid framework combines session-level rule-based correlation with unsupervised anomaly detection, achieving  $F1 = 0.883$ ,  $\text{recall} = 0.810$ , and reducing ASR to 19.0% under the evaluated Phi-3 Mini configuration. More importantly, it demonstrates an architectural principle: LLM security should integrate with existing enterprise security infrastructure, rather than operate in parallel to it.

The value of the layered architecture is empirically clear within the evaluated dataset:

- Keyword filter alone: 54.0% recall, with 46.0% of attacks succeeding.
- SIEM rules added: 63.0% recall, with 37.0% of attacks succeeding.
- OCSVM added: 81.0% recall, with 19.0% of attacks succeeding.

Each layer contributes meaningfully in the experimental setting. The OCSVM contributes the largest increment, with 162 additional detections and 66.7% of the improvement over the baseline, identifying attacks whose behavioral anomalies evade both lexical and session-level correlation rules. Beyond detection performance, the framework offers several deployment-oriented advantages:

- Vendor-neutral gateway architecture supporting LLM provider flexibility at the integration layer, subject to deployment-specific validation and calibration;
- SIEM integration supporting SOC alerting, correlation, and playbook automation;
- Unsupervised learning reducing dependence on labeled attack examples for initial anomaly modeling;
- Session-level visibility capturing multi-turn attack campaigns that application-layer detectors may miss;
- Deployment-oriented implementation requiring no modification to the underlying LLM and integrating through gateway-level telemetry, while requiring deployment-specific validation before production enforcement.

For organizations maintaining Security Operations Centers, this framework provides a reference architecture for extending SOC detection and response capabilities to LLM systems using the existing infrastructure. For security teams evaluating LLM deployments, the results provide evidence that SIEM-integrated LLM observability and anomaly detection can improve prompt-injection detection under the evaluated conditions, while broader validation across models, prompt templates, and application domains remains necessary.

#### *Future Directions*

1. **Cross-Model, Cross-Template, and Cross-Domain Validation:** Evaluate framework portability across larger and more diverse LLM families, including Llama 3, GPT-4, Claude, and Gemini, as well as different prompt templates and application domains. This evaluation should compare entropy, latency, refusal-rate, token-count, and anomaly-score distributions across settings and recalibrate SIEM and OCSVM thresholds where needed.
2. **Broader Benign and Mixed-Corpus Evaluation:** Expand the benign evaluation set beyond Stanford Alpaca by incorporating domain-specific enterprise prompts, multilingual prompts, code-oriented prompts, policy queries, customer-support interactions, and long-form document-analysis tasks. Future studies should also evaluate mixed benign–malicious corpora to better estimate operational false positive rates, threshold stability, and alert burden under realistic SOC conditions.
3. **Adversarial Robustness:** Formal evaluation against adaptive attacks using contemporary red-teaming benchmarks [36,40] and gradient-based evasion techniques. Organizations deploying against sophisticated adversaries should validate robustness under deployment-specific threat models before relying on automated enforcement.

4. Semantic Analysis Layer: Integration of transformer-based semantic classifiers, such as DistilBERT fine-tuned on cybersecurity intent, to detect semantic mimicry attacks that behavioral features cannot catch.
5. Federated Learning: Distributed OCSVM training across multiple organizations to improve benign-behavior modeling and capture organization-specific variations without centralizing proprietary data.
6. Real-Time Integration: Production deployment and monitoring in commercial SOC environments with multi-year telemetry collection, enabling longitudinal analysis of attack trends.
7. LLM Agent Expansion: Extend framework to detect prompt injection in agentic LLM systems with tool access and multi-step reasoning. Tool use introduces new attack vectors, including the use of LLM-controlled functions to execute unauthorized actions, requiring specialized detection logic.
8. Threat Intelligence Integration: Automatic correlation with external threat feeds, including threat actors and known payloads, to enable proactive hunting and improved context for alert triage.

**Author Contributions:** Conceptualization, A.A.A. and O.I.A.; methodology, A.A.A. and O.I.A.; software, A.A.A. and O.I.A.; validation, A.A.A. and O.I.A.; formal analysis, A.A.A. and O.I.A.; investigation, A.A.A. and O.I.A.; resources, A.A.A. and O.I.A.; data curation, A.A.A. and O.I.A.; writing original draft preparation, A.A.A. and O.I.A.; writing review and editing, A.A.A. and O.I.A.; visualization, A.A.A. and O.I.A.; supervision, A.A.A. and O.I.A.; project administration, A.A.A. and O.I.A. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the Ongoing Research Funding program, King Saud University, Riyadh, Saudi Arabia, grant number ORF-2026-2139. The APC was funded by the same grant.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The datasets used for evaluation are publicly available. The CySecBench dataset repository is available at <https://github.com/cysecbench/dataset> (accessed on 20 March 2026). Stanford Alpaca is available at [https://github.com/tatsu-lab/stanford\\_alpaca](https://github.com/tatsu-lab/stanford_alpaca) (accessed on 20 March 2026).

**Acknowledgments:** The authors would like to express their sincere gratitude to Ahmad Almogren for his invaluable guidance, insightful feedback, and unwavering support throughout this research. His expertise and mentorship have been instrumental in shaping the direction and success of this work. We also extend our appreciation to the College of Computer and Information Sciences at King Saud University for providing access to the necessary computational resources and infrastructure, which have been crucial for conducting the extensive evaluations presented in this study.

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

## Abbreviations

The following abbreviations are used in this manuscript:

|       |                                           |
|-------|-------------------------------------------|
| LLM   | Large Language Model                      |
| SIEM  | Security Information and Event Management |
| SOC   | Security Operations Center                |
| OCSVM | One-Class Support Vector Machine          |
| ASR   | Attack Success Rate                       |
| MTTD  | Mean Time to Detection                    |
| FPR   | False Positive Rate                       |
| TP    | True Positive                             |
| FP    | False Positive                            |
| FN    | False Negative                            |

|         |                                                        |
|---------|--------------------------------------------------------|
| TN      | True Negative                                          |
| API     | Application Programming Interface                      |
| RAG     | Retrieval-Augmented Generation                         |
| SOAR    | Security Orchestration, Automation, and Response       |
| IPI     | Indirect Prompt Injection                              |
| OWASP   | Open Worldwide Application Security Project            |
| NIST    | National Institute of Standards and Technology         |
| RBF     | Radial Basis Function                                  |
| AUC-ROC | Area Under the Receiver Operating Characteristic Curve |

## References

- Jaffal, N.O.; Alkhanafseh, M.; Mohaisen, D. Large Language Models in Cybersecurity: A Survey of Applications, Vulnerabilities, and Defense Techniques. *AI* **2025**, *6*, 216. [\[CrossRef\]](#)
- Yao, Y.; Duan, J.; Xu, K.; Cai, Y.; Sun, Z.; Zhang, Y. A survey on large language model (LLM) security and privacy: The Good, The Bad, and The Ugly. *High-Confid. Comput.* **2024**, *4*, 100211. [\[CrossRef\]](#)
- Das, B.C.; Amini, M.H.; Wu, Y. Security and Privacy Challenges of Large Language Models: A Survey. *ACM Comput. Surv.* **2025**, *57*, 1–39. [\[CrossRef\]](#)
- Nelson, A.; Rekhi, S.; Scarfone, K.; Souppaya, M. *Incident Response Recommendations and Considerations for Cybersecurity Risk Management: A CSF 2.0 Community Profile*; Technical Report NIST SP 800-61r3; National Institute of Standards and Technology (NIST): Gaithersburg, MD, USA, 2025. [\[CrossRef\]](#)
- Cybersecurity and Infrastructure Security Agency (CISA). *Guidance for SIEM and SOAR Implementation*; Online Resource; U.S. Department of Homeland Security: Washington, DC, USA, 2025. Available online: <https://www.cisa.gov/resources-tools/resources/guidance-siem-and-soar-implementation> (accessed on 10 April 2026).
- European Union Agency for Cybersecurity (ENISA). *How to Set Up CSIRT and SOC: Good Practice Guide*; Technical Report; ENISA: Brussels, Belgium, 2020. Available online: <https://www.enisa.europa.eu/sites/default/files/publications/ENISA%20Report%20-%20How%20to%20setup%20CSIRT%20and%20SOC.pdf>
- Giarimpampa, D.; Meier, R.; Bissyande, T.F.; Lenders, V.; Klein, J. Exploring the Role of Artificial Intelligence in Enhancing Security Operations: A Systematic Review. *ACM Comput. Surv.* **2025**, *58*, 1–38. [\[CrossRef\]](#)
- OWASP Foundation. *OWASP Top 10 for Large Language Model Applications*; Online Resource; OWASP Foundation: Wilmington, DE, USA, 2024. Available online: <https://owasp.org/www-project-top-10-for-large-language-model-applications/> (accessed on 10 April 2026).
- Vassilev, A.; Oprea, A.; Fordyce, A.; Anderson, H.; Davies, X.; Hamin, M. *Adversarial Machine Learning: A Taxonomy and Terminology of Attacks and Mitigations*; Technical Report NIST AI 100-2e2025; National Institute of Standards and Technology (NIST): Gaithersburg, MD, USA, 2025. [\[CrossRef\]](#)
- Wang, H.; Li, H.; Huang, M.; Sha, L. ASETF: A Novel Method for Jailbreak Attack on LLMs through Translate Suffix Embeddings. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*; Al-Onaizan, Y., Bansal, M., Chen, Y.N., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2024; pp. 2697–2711. [\[CrossRef\]](#)
- Huang, D.; Shah, A.; Araujo, A.; Wagner, D.; Sitawarin, C. Stronger Universal and Transferable Attacks by Suppressing Refusals. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*; Chiruzzo, L., Ritter, A., Wang, L., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2025; pp. 5850–5876. [\[CrossRef\]](#)
- Yi, J.; Xie, Y.; Zhu, B.; Kiciman, E.; Sun, G.; Xie, X.; Wu, F. Benchmarking and Defending against Indirect Prompt Injection Attacks on Large Language Models. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, V.1*; ACM: New York, NY, USA, 2025; pp. 1809–1820. [\[CrossRef\]](#)
- Jacob, D.; Alzahrani, H.; Hu, Z.; Alomair, B.; Wagner, D. PromptShield: Deployable Detection for Prompt Injection Attacks. In *Proceedings of the Fifteenth ACM Conference on Data and Application Security and Privacy*; ACM: New York, NY, USA, 2025; pp. 341–352. [\[CrossRef\]](#)
- Protect AI. *Rebuff: Prompt Injection Detector (Version v0.1.1)*; GitHub Repository Release; GitHub, Inc.: San Francisco, CA, USA, January 2024. Available online: <https://github.com/protectai/rebuff/releases/tag/v0.1.1> (accessed on 10 April 2026).
- Pingua, B.; Murmu, D.; Kandpal, M.; Rautaray, J.; Mishra, P.; Barik, R.K.; Saikia, M.J. Mitigating adversarial manipulation in LLMs: A prompt-based approach to counter jailbreak attacks (Prompt-G). *PeerJ Comput. Sci.* **2024**, *10*, e2374. [\[CrossRef\]](#) [\[PubMed\]](#)
- Hung, K.H.; Ko, C.Y.; Rawat, A.; Chung, I.H.; Hsu, W.H.; Chen, P.Y. Attention Tracker: Detecting Prompt Injection Attacks in LLMs. In *Proceedings of the Findings of the Association for Computational Linguistics: NAACL 2025*; Chiruzzo, L., Ritter, A., Wang, L., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2025; pp. 2309–2322. [\[CrossRef\]](#)

17. Greshake, K.; Abdelnabi, S.; Mishra, S.; Endres, C.; Holz, T.; Fritz, M. Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*; ACM: New York, NY, USA, 2023; pp. 79–90. [CrossRef]
18. OWASP Foundation. *LLM01: Prompt Injection—OWASP Top 10 for LLM Applications*; Online Resource; OWASP Foundation: Wilmington, DE, USA, 2025. Available online: <https://genai.owasp.org/llmrisk/llm01-prompt-injection/> (accessed on 8 April 2026).
19. Robey, A.; Wong, E.; Hassani, H.; Pappas, G.J. SmoothLLM: Defending Large Language Models against Jailbreaking Attacks. *Trans. Mach. Learn. Res.* **2025**, 1–41. Available online: <https://openreview.net/forum?id=laPAh2hRFC> (accessed on 10 April 2026).
20. Shumailov, I.; Shumaylov, Z.; Zhao, Y.; Papernot, N.; Anderson, R.; Gal, Y. AI models collapse when trained on recursively generated data. *Nature* **2024**, *631*, 755–759. [CrossRef]
21. Gavish, A.; Google GenAI Security Team. *Mitigating Prompt Injection Attacks with a Layered Defense Strategy*; Google Online Security Blog; Mountain View, CA, USA, 2025. Available online: <https://blog.google/security/mitigating-prompt-injection-attacks/> (accessed on 10 April 2026).
22. Microsoft Security Response Center (MSRC). *How Microsoft Defends Against Indirect Prompt Injection Attacks*; MSRC Blog; Redmond, WA, USA, 2025. Available online: <https://www.microsoft.com/en-us/msrc/blog/2025/07/how-microsoft-defends-against-indirect-prompt-injection-attacks> (accessed on 10 April 2026).
23. Guardrails AI. *detect\_prompt\_injection: Plugin for Detecting Prompt Injection Attacks*; GitHub Repository; San Francisco, CA, USA, 2024. Available online: [https://github.com/guardrails-ai/detect\\_prompt\\_injection](https://github.com/guardrails-ai/detect_prompt_injection) (accessed on 10 April 2026).
24. Kang, D.; Li, X.; Stoica, I.; Guestrin, C.; Zaharia, M.; Hashimoto, T. Exploiting Programmatic Behavior of LLMs: Dual-Use Through Standard Security Attacks. In *Proceedings of the 2024 IEEE Security and Privacy Workshops (SPW)*, San Francisco, CA, USA, 23 May 2024; pp. 132–143. [CrossRef]
25. Chan, A.; Ezell, C.; Kaufmann, M.; Wei, K.; Hammond, L.; Bradley, H.; Bluemke, E.; Rajkumar, N.; Krueger, D.; Kolt, N.; et al. Visibility into AI Agents. In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*; ACM: New York, NY, USA, 2024; pp. 958–973. [CrossRef]
26. Maderamitla, P.; Katragadda, S.R. Observability for LLM apps: What to log, privacy-safe telemetry, KPIs. *Front. Comput. Sci. Artif. Intell.* **2026**, *5*, 10–14. [CrossRef]
27. Chen, Y.; Li, H.; Sui, Y.; He, Y.; Liu, Y.; Song, Y.; Hooi, B. Can Indirect Prompt Injection Attacks Be Detected and Removed? In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*; Che, W., Nabende, J., Shutova, E., Pilehvar, M.T., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2025; pp. 18189–18206. [CrossRef]
28. Şaşal, S.; Can, Ö. Prompt Injection Attacks on Large Language Models: Multi-Model Security Analysis with Categorized Attack Types. In *Proceedings of the 17th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (KDIR)*; SciTePress: Setúbal, Portugal, 2025; pp. 517–524. [CrossRef]
29. Addepalli, S.; Varun, Y.; Suggala, A.; Shanmugam, K.; Jain, P. Does Safety Training of LLMs Generalize to Semantically Related Natural Prompts? In *Proceedings of the International Conference on Learning Representations (ICLR 2025)*, Singapore, 24–28 April 2025; OpenReview.net, 2025; Volume 2025, pp. 43611–43631. Available online: <https://openreview.net/forum?id=LO4MEPoqrG> (accessed on 10 April 2026).
30. Ji, J.; Hou, B.; Robey, A.; Pappas, G.J.; Hassani, H.; Zhang, Y.; Wong, E.; Chang, S. Defending Large Language Models against Jailbreak Attacks via Semantic Smoothing. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*; Inui, K., Sakti, S., Wang, H., Wong, D.F., Bhattacharyya, P., Banerjee, B., Ekbal, A., Chakraborty, T., Singh, D.P., Eds.; The Asian Federation of Natural Language Processing and The Association for Computational Linguistics: Mumbai, India, 2025; pp. 7–40. [CrossRef]
31. Geng, T.; Xu, Z.; Qu, Y.; Wong, W.E. Prompt Injection Attacks on Large Language Models: A Survey of Attack Methods, Root Causes, and Defense Strategies. *Comput. Mater. Contin.* **2026**, *87*, 4. [CrossRef]
32. National Cyber Security Centre (NCSC). *Thinking About the Security of AI Systems*; NCSC Blog; London, UK, 2025. Available online: <https://www.ncsc.gov.uk/blog-post/thinking-about-security-ai-systems> (accessed on 3 April 2026).
33. González-Granadillo, G.; González-Zarzosa, S.; Diaz, R. Security Information and Event Management (SIEM): Analysis, Trends, and Usage in Critical Infrastructures. *Sensors* **2021**, *21*, 4759. [CrossRef] [PubMed]
34. Liu, Y.; Jia, Y.; Geng, R.; Jia, J.; Gong, N.Z. Formalizing and Benchmarking Prompt Injection Attacks and Defenses. In *Proceedings of the 33rd USENIX Security Symposium (USENIX Security 24)*; USENIX Association: Berkeley, CA, USA, 2024; pp. 1831–1847. Available online: <https://www.usenix.org/conference/usenixsecurity24/presentation/liu-yupei> (accessed on 10 April 2026).
35. Chen, S.; Piet, J.; Sitawarin, C.; Wagner, D. StruQ: Defending against prompt injection with structured queries. In *Proceedings of the USENIX Security Symposium*, Seattle, WA, USA, 13–15 August 2025.

36. Chao, P.; Debenedetti, E.; Robey, A.; Andriushchenko, M.; Croce, F.; Sehwag, V.; Dobriban, E.; Flammairon, N.; Pappas, G.J.; Tramèr, F.; et al. JailbreakBench: An Open Robustness Benchmark for Jailbreaking Large Language Models. In Proceedings of the Advances in Neural Information Processing Systems 37 (NeurIPS 2024), Datasets and Benchmarks Track, Vancouver, BC, Canada, 10–15 December 2024. [CrossRef]
37. Schölkopf, B.; Platt, J.C.; Shawe-Taylor, J.; Smola, A.J.; Williamson, R.C. Estimating the Support of a High-Dimensional Distribution. *Neural Comput.* **2001**, *13*, 1443–1471. [CrossRef] [PubMed]
38. Wahréus, J.; Hussain, A.M.; Papadimitratos, P. *CySecBench: Generative AI-based CyberSecurity-focused Prompt Dataset for Benchmarking Large Language Models*; Online Resource; Github Repository: San Francisco, CA, USA, 2025. Available online: [https://github.com/cysecbench/dataset/blob/main/CySecBench\\_paper.pdf](https://github.com/cysecbench/dataset/blob/main/CySecBench_paper.pdf) (accessed on 20 March 2026).
39. Taori, R.; Gulrajani, I.; Zhang, T.; Dubois, Y.; Li, X.; Guestrin, C.; Liang, P.; Hashimoto, T.B. *Stanford Alpaca: An Instruction-following LLaMA Model*; Github Repository: San Francisco, CA, USA, 2023. Available online: [https://github.com/tatsu-lab/stanford\\_alpaca](https://github.com/tatsu-lab/stanford_alpaca) (accessed on 20 March 2026).
40. Mazeika, M.; Phan, L.; Yin, X.; Zou, A.; Wang, Z.; Mu, N.; Sakhaee, E.; Li, N.; Basart, S.; Li, B.; et al. HarmBench: A Standardized Evaluation Framework for Automated Red Teaming and Robust Refusal. In *Proceedings of the 41st International Conference on Machine Learning*; Proceedings of Machine Learning Research; Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., Berkenkamp, F., Eds.; PMLR: Vienna, Austria, 2024; Volume 235, pp. 35181–35224. Available online: <https://proceedings.mlr.press/v235/mazeika24a.html> (accessed on 10 April 2026).
41. Xu, Z.; Liu, Y.; Deng, G.; Li, Y.; Picek, S. A Comprehensive Study of Jailbreak Attack versus Defense for Large Language Models. In *Proceedings of the Findings of the Association for Computational Linguistics: ACL 2024*; Ku, L.W., Martins, A., Srikumar, V., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2024; pp. 7432–7449. [CrossRef]

**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.