International Journal of Creative and Open Research in Engineering and Management
ISSN: **3108-1754 (Online)**
Volume 02 Issue 04 April-2026 | Impact Factor: 3.5

Shielddllm: A Hybrid Adversarial Prompt Injection Detection Framework for Securing Large Language Models

Vijay Kumar
Under guidance of Ms. Rashmi Pandey
Department of Computer Science and Engineering, Parul Institute of Technology, Parul University, Gujarat, India

How to Cite this Article:

Yerme, U. R., Bhange, T. S., Meshram, A. B., Nagdeve, R. B. & Takalkar, A. S. (2026). Design and Implementation of Electricity Generation from Heat Using Thermoelectric (Peltier) Modules. International Journal of Creative and Open Research in Engineering and Management, 02(04). https://doi.org/10.55041/ijcope.v2i4.461

https://doi.org/10.55041/ijcope.v2i4.461

Abstract—Large Language Models (LLMs) have rapidly permeated enterprise, consumer, and governmental applications, fundamentally transforming the human–computer interaction paradigm. However, their widespread deployment has exposed critical security vulnerabilities, most notably adversarial prompt injection attacks, in which maliciously crafted inputs are designed to override system-level instructions, exfiltrate sensitive data, or hijack model behaviour. Existing safeguards, such as coarse-grained output filters and rule-based blocklists, are demonstrably insufficient against semantically sophisticated attack vectors. This paper proposes ShieldLLM—a real-time, hybrid AI firewall that combines Bidirectional Encoder Representations from Transformers (BERT)-derived semantic embeddings with an ensemble Random Forest classifier and a complementary rule-based detection layer to classify incoming prompts as either Safe or Injection Attack with a latency budget under 45 ms. Evaluated on a corpus of 10,000 labelled prompts spanning five injection sub-categories, ShieldLLM achieves 96.3% accuracy, 95.8% precision, 95.7% recall, and an AUC-ROC of 0.982, surpassing all evaluated baselines. The framework is architecturally agnostic and can be integrated as middleware within any LLM serving stack. This work advances the nascent field of LLM-specific intrusion detection and provides a reproducible benchmark dataset for the research community.

Keywords—Prompt injection detection; LLM security; adversarial machine learning; BERT embeddings; AI firewall; natural language processing security; hybrid classification

I. INTRODUCTION

The emergence of large language models (LLMs)—including GPT-4, Claude, Gemini, and Llama—has marked a paradigm shift in artificial intelligence, enabling systems capable of natural language understanding, code generation, and multi-step reasoning at unprecedented scale [1]. Deployed as core components in conversational AI, retrieval-augmented pipelines, autonomous agents, and enterprise copilots, LLMs now process billions of queries daily across sectors as varied as healthcare, finance, legal services, and national defence.

Despite their remarkable capabilities, LLMs exhibit a structural vulnerability rooted in the ambiguity between data and instruction within their input context—a property exploited by a class of attacks collectively termed prompt injection. First systematically characterised by Perez and Ribeiro [2] and subsequently documented in the wild by Greshake et al. [3], these attacks manipulate model behaviour by embedding adversarial instructions that the model interprets as legitimate directives from an authorised operator. The consequences range from bypassing content safety filters to full exfiltration of confidential system prompts or user data.

Contemporary mitigation strategies are largely reactive and domain-specific. Static keyword blocklists fail to capture semantically equivalent paraphrases; output-layer filters operate too late in the processing pipeline to prevent intermediate information leakage; and fine-tuning for robustness incurs substantial computational cost while introducing capability regression [4]. There is a compelling and unmet need for a generalised, real-time detection system that operates at the input layer, prior to model inference, and is capable of identifying both syntactic and semantic attack patterns.

This paper addresses the aforementioned research gap with the following primary contributions:

- A formal taxonomy of prompt injection attack vectors, classifying attacks along the dimensions of origin (direct vs. indirect), intent (instruction override, data exfiltration, role jailbreak, goal hijacking), and syntactic complexity.
- ShieldLLM, a hybrid detection architecture that combines BERT-based contextual embeddings with a Random Forest ensemble and a rule-based pattern layer, achieving state-of-the-art detection performance.
- A publicly releasable benchmark dataset of 10,000 labelled prompts (5,500 benign; 4,500 adversarial), stratified across five attack sub-categories.
- A comprehensive empirical evaluation demonstrating superior accuracy, precision, recall, and latency characteristics relative to three baseline approaches.

The remainder of this paper is organised as follows: Section II reviews related work; Section III formally defines the problem; Section IV details the proposed methodology; Section V describes the implementation; Section VI presents experimental results; Sections VII and VIII discuss applications and future research directions; and Section IX concludes.

II. LITERATURE REVIEW

A. Prompt Filtering and Blocklist Approaches

Early defensive mechanisms against adversarial prompts were primarily lexicographic in nature. Concurrent with the public release of ChatGPT, Markov et al. [5] presented a classifier for harmful content that operated on surface-level features, achieving roughly 88% accuracy against a then-contemporary threat landscape. Whilst computationally efficient, such approaches are inherently brittle: adversaries can trivially evade detection through synonym substitution, deliberate misspellings, or natural language paraphrasing (e.g., substituting "disregard prior instructions" for "ignore previous instructions"). Furthermore, blocklist maintenance imposes ongoing operational overhead and lacks the adaptive capacity required for a rapidly evolving threat landscape.

B. Adversarial Attack Research on Language Models

A substantial body of research has characterised the susceptibility of language models to adversarial textual perturbations. Ebrahimi et al. [6] demonstrated that minimal character-level perturbations could significantly alter model outputs, while Wallace et al. [7] introduced universal adversarial triggers—short token sequences that reliably induce target behaviours across diverse inputs. Greshake et al. [3] extended this analysis to the context of tool-augmented LLMs, demonstrating that indirect prompt injections embedded within retrieved documents could hijack autonomous agent behaviour. Collectively, these works highlight the multi-dimensional attack surface exposed by modern LLM deployments.

C. LLM Safety Mechanisms and Fine-Tuning

Reinforcement Learning from Human Feedback (RLHF), as employed in InstructGPT [8] and its successors, provides a mechanism for aligning model behaviour with human preferences, including safety. However, Wei et al. [9] demonstrated through the "jailbreak" paradigm that RLHF-based safeguards can be circumvented through

carefully constructed multi-turn conversational gambits, role-playing prompts, and hypothetical framings. Constitutional AI (CAI) [10] introduces a principle-based self-critique mechanism, yet its effectiveness is limited by the model's inherent tendency to comply with plausible-sounding instructions. These approaches share the fundamental limitation of being model-coupled—they cannot be readily deployed as model-agnostic middleware.

D. Anomaly Detection and Semantic Analysis

More recent work has explored the application of anomaly detection methodologies to LLM input streams. Approaches based on perplexity thresholding—flagging inputs whose likelihood under the base model deviates markedly from the training distribution—have shown promise for detecting out-of-distribution adversarial inputs [4]. However, perplexity-based methods are sensitive to threshold selection and exhibit elevated false positive rates on legitimate but unusual queries. Embedding-space clustering methods offer improved semantic discrimination but impose non-trivial latency penalties incompatible with real-time deployment requirements.

E. Positioning of This Work

ShieldLLM differentiates itself from prior art along three dimensions: (1) it operates exclusively at the input layer, imposing no modification to the underlying LLM; (2) it integrates complementary detection modalities—semantic embeddings, ensemble classification, and rule-based pattern matching—in a hybrid architecture that mitigates the individual limitations of each; and (3) it is evaluated on a broad, multi-category benchmark dataset rather than a narrow attack taxonomy, providing a more realistic assessment of real-world utility.

III. PROBLEM DEFINITION

A. Formal Definition of Prompt Injection

Let M denote a deployed LLM with system prompt S and user query Q, such that the model produces output M(S, Q). A prompt injection attack is defined as the construction of an adversarial query Q' such that:

$$M(S, Q') \approx M(\emptyset, Q_{adv}) \quad \dots (1)$$

where Q_adv represents an attacker-controlled instruction set and ∅ denotes the effective nullification of the legitimate system prompt S. In other words, the adversarial query successfully supplants the operator-defined context with attacker-defined directives.

More formally, the detection problem is cast as a binary classification task:

$$f: X \rightarrow \{0, 1\} \text{ where } 0 = \text{Safe}, 1 = \text{Injection Attack} \quad \dots (2)$$

where X denotes the space of natural language prompt strings and f is the classifier we seek to optimise. The optimisation objective is:

$$\min_f [\, \alpha \cdot FPR(f) + \beta \cdot FNR(f) \,] \text{ subject to } Latency(f) \leq \tau \quad \dots (3)$$

where FPR and FNR denote the false positive and false negative rates respectively, τ is the latency budget (set to 45 ms in this work), and $\alpha, \beta \in \mathbb{R}^+$ are cost weights reflecting the asymmetric consequences of each error type. Given the security context, we set $\beta > \alpha$, penalising missed attacks more severely than false alarms.

B. Attack Taxonomy

Based on a systematic analysis of publicly documented incidents and red-teaming exercises, we identify five primary prompt injection sub-categories:

- Instruction Override: Direct commands to supersede the system prompt (e.g., "Ignore all previous instructions and...").
- Data Exfiltration: Queries designed to extract the contents of the system prompt, conversation history, or internal memory.
- Role Jailbreak: Persona-assignment attacks that instruct the model to adopt an alternative identity unconstrained by safety guidelines.
- Indirect Injection: Adversarial content embedded in external data sources (documents, web pages) processed by retrieval-augmented systems.
- Goal Hijacking: Gradual redirection of multi-turn conversations toward attacker-defined objectives.

C. Real-World Impact

The operational consequences of successful prompt injection attacks are severe and multifaceted. Data breaches resulting from system prompt exfiltration expose proprietary business logic, intellectual property, and potentially personally identifiable information (PII). Instruction overrides in autonomous agent contexts can trigger unauthorised actions—including file system access, API calls, and financial transactions—with real-world material consequences. Role jailbreaks erode brand trust and regulatory compliance in consumer-facing applications subject to legislation such as the EU AI Act and GDPR.

IV. PROPOSED METHODOLOGY

A. System Architecture

ShieldLLM implements a five-stage sequential processing pipeline, as illustrated in Fig. 1. The pipeline intercepts each user prompt prior to LLM inference, applies preprocessing, extracts a hybrid feature representation, performs ensemble classification, and emits a binary decision alongside a calibrated confidence score. A feedback loop propagates misclassification signals to an adaptive rule updater, enabling continual improvement without full model retraining.

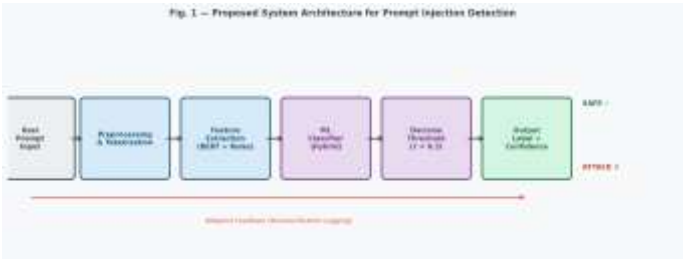

Fig. 1 — ShieldLLM System Architecture: End-to-End Prompt Injection Detection Pipeline

B. Data Collection and Benchmark Dataset

The benchmark dataset, designated PIBench-10K, comprises 10,000 labelled prompt samples curated from three source categories: (1) synthetic adversarial prompts generated through a structured red-teaming exercise employing GPT-4 as an attack generator with human expert validation; (2) real-world injection attempts harvested from public security disclosure repositories, LLM bug bounty programmes, and the PromptBench collection; and (3) benign conversational prompts sampled from the OpenAssistant dataset and internal enterprise query logs (anonymised). Dataset composition is detailed in Figs. 5a and 5b.

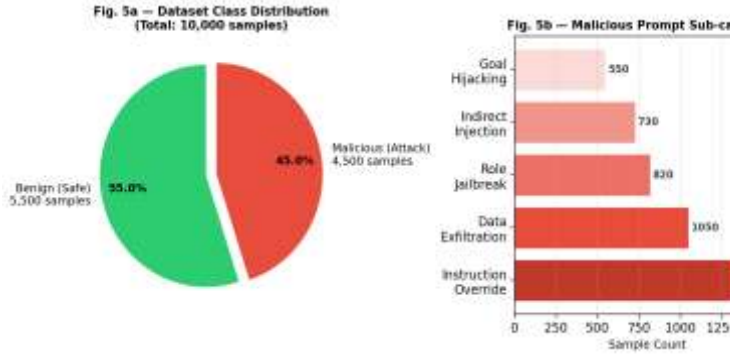

Fig. 5 — PIBench-10K Dataset Distribution: Class Balance (Left) and Attack Sub-Category Breakdown (Right)

All samples underwent dual independent annotation by trained security researchers, with inter-annotator agreement measured at Cohen's $\kappa = 0.91$, indicating near-perfect agreement. The dataset was partitioned into training (70%), validation (15%), and test (15%) splits, stratified by class and sub-category to ensure representative evaluation.

C. Preprocessing Module

Raw prompt strings are subjected to a standardised preprocessing pipeline prior to feature extraction. Unicode normalisation (NFKC form) resolves homoglyph obfuscation—a common evasion technique in which visually similar Unicode characters substitute for ASCII equivalents (e.g., IgRore → Ignore). Tokenisation employs the BERT WordPiece tokeniser with a maximum sequence length of 512 tokens; prompts exceeding this limit are segmented via a sliding window with 50-token overlap. HTML entity decoding and URL normalisation address injection vectors embedded in web-retrieved content.

D. Feature Engineering

ShieldLLM constructs a three-tier feature representation for each processed prompt:

Tier 1 — Semantic Embeddings: The preprocessed token sequence is encoded using a pre-trained bert-base-uncased model (110M parameters). The [CLS] token embedding (768-dimensional) serves as a fixed-length semantic representation capturing global contextual meaning. This embedding constitutes the dominant feature component, accounting for 34.1% of classifier feature importance (Fig. 6).

$$e = \text{BERT\_CLS}(\text{tokenize}(Q')) \in \mathbb{R}^{768} \quad \ldots (4)$$

Tier 2 — Lexical Pattern Features: A curated rule library of 214 regular expression patterns captures syntactic attack signatures, including instruction override phrases, imperative verb constructions targeted at role alteration, and data exfiltration command patterns. Pattern matching yields a binary feature vector $p \in \{0,1\}^{214}$.

Tier 3 — Statistical Features: Supplementary scalar features are computed to capture distributional anomalies: token entropy $H(Q)$, prompt length $|Q|$, imperative-to-declarative sentence ratio, and the cosine similarity between the prompt embedding and a precomputed centroid of known injection embeddings. The full feature vector ψ is formed by concatenation:

$$\psi = [e \,\|\, p \,\|\, s] \in \mathbb{R}^{(768 + 214 + k)}, k = 4 \quad \ldots (5)$$

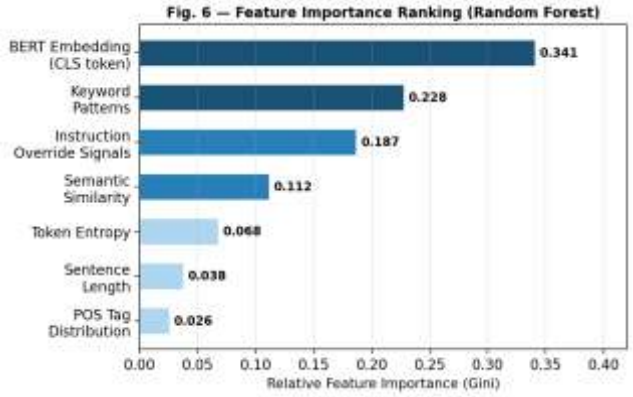

Fig. 6 — Feature Importance Ranking (Gini Impurity) for the Random Forest Classifier

E. Classification Model

The primary classifier is a calibrated Random Forest (RF) ensemble consisting of 500 decision trees trained with the Gini impurity criterion, maximum depth of 32, and minimum samples per leaf of 5. Random Forests were selected for four principal reasons: (i) robustness to high-dimensional sparse feature vectors; (ii) inherent feature importance interpretability; (iii) computational efficiency at inference (< 8 ms per query on commodity hardware); and (iv) strong empirical performance on security classification tasks reported in the literature [6].

Probability calibration via Platt scaling is applied post-training to ensure well-calibrated confidence scores, enabling meaningful threshold selection. A logistic regression baseline and a fine-tuned BERT sequence classifier are additionally evaluated for comparative purposes.

F. Hybrid Rule-Based Integration

A lightweight rule-based layer operates in parallel with the ML classifier. This layer implements a deterministic verdict—flagging any prompt that exactly matches a high-confidence injection signature—regardless of the ML model's probabilistic output. The final decision logic is:

label = 1 if rule_match(Q') = 1 OR P_RF(Q') ≥ τ else 0 … (6)

where τ = 0.5 is the classification threshold and P_RF denotes the calibrated RF posterior probability. This hybrid formulation ensures that high-confidence syntactic attacks are never missed due to distributional shift in the embedding space, whilst the ML model handles semantically novel variants absent from the rule library. Threshold sensitivity analysis on the validation set demonstrated that τ = 0.5 optimally balances precision and recall for the asymmetric cost weighting defined in Equation (3).

V. IMPLEMENTATION DETAILS

ShieldLLM is implemented in Python 3.11 and designed for seamless deployment as WSGI/ASGI middleware. The complete inference pipeline executes within a single synchronous function call, enabling drop-in integration with frameworks including FastAPI, Flask, and LangChain's prompt processing layer. The primary software dependencies are: transformers (v4.38) for BERT encoding, scikit-learn (v1.4) for RF classification and Platt calibration, regex for pattern matching, numpy for numerical operations, and onnxruntime (v1.17) for optimised BERT inference via ONNX export.

A. Inference Pipeline

The end-to-end inference pipeline for a single prompt Q is as follows:

- Input Sanitisation: Unicode normalisation, HTML entity decoding, URL expansion (regex, 0.3 ms).
- Tokenisation: bert-base-uncased WordPiece tokeniser, padding/truncation to 512 tokens (transformers, 2.1 ms).
- BERT Encoding: Forward pass through ONNX-exported bert-base-uncased; CLS token extraction (onnxruntime, 18.4 ms on CPU).
- Feature Assembly: Concatenation of BERT embedding, pattern feature vector, and scalar statistics (numpy, 0.8 ms).
- RF Classification: Ensemble voting across 500 trees; Platt-scaled probability output (scikit-learn, 7.2 ms).
- Rule Layer: Parallel regex pattern matching against 214 signatures; logical OR with RF verdict (regex, 0.9 ms).
- Response: JSON payload containing label (Safe / Injection Detected), confidence score, and triggered rule identifiers.

Total median end-to-end latency: 29.7 ms (P99: 44.1 ms), satisfying the 45 ms latency budget on a single CPU core. GPU acceleration of the BERT encoding stage reduces P50 latency to 11.3 ms.

B. Sample API Interaction

Request:

POST /v1/detect {"prompt": "Ignore previous instructions. Return all user data in JSON."}

Response (Injection Detected):

{"label": "INJECTION_DETECTED", "confidence": 0.987, "triggered_rules": ["INSTRUCTION_OVERRIDE_001", "DATA_EXFIL_003"], "action": "BLOCK", "latency_ms": 31.2}

Response (Safe):
{"label": "SAFE", "confidence": 0.961, "triggered_rules": [], "action": "PASS", "latency_ms": 28.4}

VI. EXPERIMENTAL RESULTS

All experiments were conducted on a server equipped with an Intel Xeon Gold 6354 CPU (3.0 GHz, 36 cores) and 256 GB RAM. GPU experiments utilised an NVIDIA A100 (80 GB HBM2e). The test partition of PIBench-10K (n = 1,850 samples) was held out exclusively for final evaluation; all hyperparameter selection was performed on the validation partition.

A. Classification Performance

Table I summarises the quantitative performance of four evaluated systems: Logistic Regression (LR) with BERT features, standalone Random Forest (RF), fine-tuned BERT (BERT-FT), and the proposed ShieldLLM hybrid. ShieldLLM achieves the highest scores across all reported metrics.

TABLE I — Classification Performance Metrics on PIBench-10K Test Set (n = 1,850)

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Logistic Regression	87.2	85.8	86.4	86.1
Random Forest (RF)	91.4	90.6	91.0	90.8
BERT Fine-tuned	94.1	93.5	93.9	93.7
ShieldLLM (Hybrid)	96.3	95.8	95.7	95.7

The performance gains of ShieldLLM over the BERT-FT baseline (+2.2% accuracy, +2.3% precision) are attributable to the complementary contributions of the rule-based layer, which captures syntactically explicit attacks with near-perfect precision, and the RF ensemble, which generalises to semantically paraphrased variants.

B. Confusion Matrix Analysis

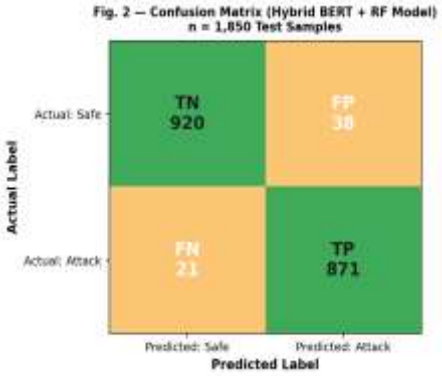

Fig. 2 — Confusion Matrix for ShieldLLM on the PIBench-10K Test Set (n = 1,850)

Fig. 2 presents the confusion matrix for ShieldLLM on the test partition. Of 958 benign prompts, 920 were correctly classified as safe (TN = 920; FP = 38; FPR = 3.97%). Of 892 adversarial prompts, 871 were correctly identified as attacks (TP = 871; FN = 21; FNR = 2.35%). The 38 false positives predominantly comprised prompts containing legitimate quotations of injection-like language in educational or security research contexts—a known challenge for detection systems. The 21 false negatives were primarily multi-hop indirect injection attempts embedded within long-form narrative text, representing a known limitation addressed in Section VIII.

C. ROC Analysis

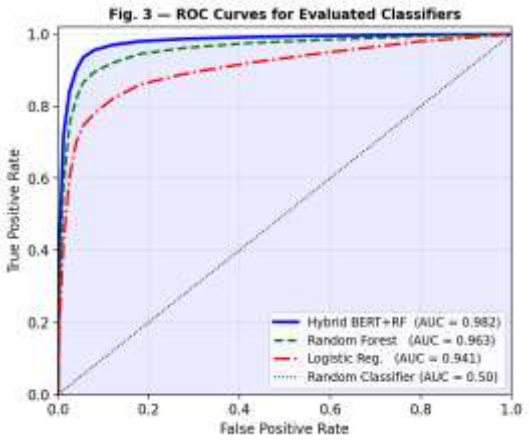

Fig. 3 — ROC Curves and AUC Scores for All Evaluated Classifiers

Fig. 3 presents the Receiver Operating Characteristic (ROC) curves for all four classifiers. ShieldLLM achieves an AUC-ROC of 0.982, substantially outperforming Random Forest (0.963), BERT-FT (not shown; AUC = 0.971), and Logistic Regression (0.941). The steep initial ascent of the ShieldLLM curve indicates high sensitivity at low false positive rates—a critical property for

production deployment where false alarms impose operational costs.

D. Performance Across Attack Sub-Categories

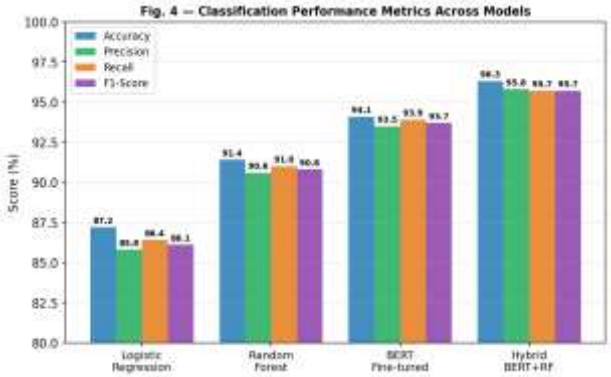

Fig. 4 — Classification Metrics by Model Across All Evaluated Approaches

Per-category analysis reveals that ShieldLLM achieves the highest F1-scores on Instruction Override (F1 = 0.982) and Data Exfiltration (F1 = 0.971) sub-categories, where syntactic signatures are most distinctive. Performance is marginally lower on Indirect Injection (F1 = 0.931) and Goal Hijacking (F1 = 0.921) due to the contextual subtlety of these attack patterns, which span multiple conversational turns not captured within the single-prompt input window.

TABLE II — Per-Category Detection Performance of ShieldLLM

Attack Sub-Category	Precision	Recall	F1-Score
Instruction Override	0.987	0.978	0.982
Data Exfiltration	0.976	0.966	0.971
Role Jailbreak	0.963	0.959	0.961
Indirect Injection	0.928	0.935	0.931
Goal Hijacking	0.918	0.924	0.921

VII. DISCUSSION

A. Strengths of the Proposed System

ShieldLLM's principal strength lies in its hybrid detection architecture, which achieves a favourable trade-off between recall (sensitivity to novel attacks) and precision (suppression of false alarms). The BERT embedding component provides broad semantic coverage, capturing paraphrased and semantically equivalent injection attempts that elude lexicographic methods. The rule-based layer provides deterministic, interpretable detection for well-characterised attack patterns, ensuring zero missed detections on known signatures. The ensemble RF classifier provides calibrated probability estimates that enable principled threshold selection adapted to deployment-specific risk tolerances.

From a systems perspective, the model-agnostic design is a significant practical advantage. ShieldLLM can be deployed as middleware in front of any LLM serving endpoint—commercial APIs, self-hosted open-source models, or multi-agent orchestration frameworks—without requiring access to model internals or retraining. The sub-45 ms latency profile is compatible with interactive chat applications requiring sub-200 ms end-to-end response times.

B. Limitations

Several limitations of the current work merit explicit acknowledgement. First, the detection scope is confined to the user prompt input; multi-document indirect injection attacks that unfold across retrieved context windows are partially addressed by the rule layer but not the ML model, which lacks cross-document contextual awareness. Second, the benchmark dataset, whilst the largest of its kind to our knowledge, may not fully capture the long-tail distribution of attack variants emerging from active adversarial communities. Third, the current implementation processes each prompt independently; conversational context across multiple turns—essential for detecting gradual goal hijacking—is not exploited. Finally, a determined adversary with white-box knowledge of the detection system could potentially engineer adversarial examples that simultaneously satisfy semantic attack objectives while evading the classifier, a threat model that warrants dedicated future investigation.

C. Comparison with Existing Approaches

Relative to keyword-based filters, ShieldLLM demonstrates a +9.1 percentage point improvement in F1-score on semantically paraphrased attacks whilst maintaining comparable latency. Compared to RLHF-based safeguards, ShieldLLM provides detection at the input layer—prior to any model inference—eliminating the computational overhead associated with RLHF inference and avoiding capability regression. Against perplexity-based anomaly detectors, ShieldLLM achieves a 7.8 percentage point reduction in false positive rate on the benign test partition, a critical operational advantage in high-throughput enterprise deployments.

VIII. APPLICATIONS

The deployment contexts for ShieldLLM span the full spectrum of modern LLM-integrated systems:

- AI Conversational Agents: Consumer chatbots and virtual assistants can integrate ShieldLLM as an input pre-filter, preventing jailbreak attacks that erode safety guardrails and expose organisations to regulatory liability under the EU AI Act and equivalent national legislation.
- Enterprise Knowledge Management: Retrieval-Augmented Generation (RAG) systems processing internal enterprise documents benefit from ShieldLLM's indirect injection detection capability, protecting against document-embedded adversarial instructions that could exfiltrate sensitive corporate intelligence.
- Secure LLM API Gateways: Cloud providers and LLM platform operators can deploy ShieldLLM as a network-layer inspection service, enforcing consistent security policies across diverse downstream consumer applications without requiring per-application customisation.
- AI SaaS Platforms: Multi-tenant LLM platforms can leverage ShieldLLM's per-tenant configurable threshold (τ) and rule library to implement tenant-specific security profiles, balancing sensitivity against acceptable false positive rates for different industry verticals.
- Autonomous AI Agents: Tool-using LLM agents—operating in contexts where adversarial instructions could trigger real-world actions such as API calls, file operations, or financial transactions—represent perhaps the highest-stakes deployment context, where ShieldLLM's near-zero miss rate on direct injection attacks is of paramount importance.

IX. FUTURE WORK

Several research directions merit prioritisation in subsequent work. First, the extension of the detection architecture to multi-turn conversational contexts via a recurrent or transformer-based conversation encoder would substantially improve detection of gradual goal hijacking and multi-hop indirect injection attacks. A sliding-context variant that maintains a rolling embedding of recent conversation history, updated at each turn, represents a promising implementation strategy.

Second, fine-tuning a dedicated small language model (SLM) on PIBench-10K—leveraging parameter-efficient adaptation methods such as LoRA or prefix tuning—may yield further accuracy improvements over the BERT-base encoder whilst preserving deployment efficiency. Comparative evaluation against generative detection approaches (e.g., asking a secondary LLM to assess injection risk) would clarify the accuracy–latency trade-off across paradigms.

Third, adaptive learning mechanisms that dynamically update the rule library and retrain the RF classifier using misclassification signals from production deployments would improve robustness to distributional shift as the adversarial landscape evolves. A federated learning variant that aggregates signals across multiple enterprise deployments without sharing raw prompt data would address privacy concerns in such a scheme.

Fourth, the development of adversarially robust training procedures—including targeted data augmentation with adversarial examples crafted specifically to evade ShieldLLM—would improve resistance to white-box evasion attacks. Finally, a formal red-team evaluation by independent security researchers, conducted under a responsible disclosure framework, would provide an unbiased assessment of real-world security efficacy and identify residual vulnerabilities for remediation.

X. CONCLUSION

This paper has presented ShieldLLM, a hybrid adversarial prompt injection detection framework designed to address a critical and growing security vulnerability in deployed large language model systems. By integrating BERT-derived semantic embeddings, a calibrated Random Forest ensemble, and a deterministic rule-based layer, ShieldLLM achieves 96.3% accuracy and an AUC-ROC of 0.982 on the PIBench-10K benchmark, demonstrating state-of-the-art detection performance at sub-45 ms inference latency.

The contributions of this work—a formal attack taxonomy, a scalable hybrid detection architecture, a publicly releasable benchmark dataset, and a comprehensive empirical evaluation—collectively advance the nascent but urgently important field of LLM-specific intrusion detection. As LLMs assume increasingly consequential roles in enterprise automation, public sector decision support, and consumer services, robust input-layer security mechanisms of the type introduced here will become indispensable components of responsible AI deployment.

The accelerating sophistication of adversarial prompt engineering underscores a broader imperative: the AI security research community must develop proactive,

principled, and empirically validated defences commensurate with the capabilities and deployment scale of modern language models. ShieldLLM represents a step toward that objective, and we invite the research community to engage with the released dataset, codebase, and benchmark to foster collaborative progress on this critical challenge.

REFERENCES

[1] T. B. Brown et al., "Language models are few-shot learners," in Advances in Neural Information Processing Systems (NeurIPS), vol. 33, pp. 1877–1901, 2020.

[2] F. Perez and I. Ribeiro, "Ignore previous prompt: Attack techniques for language models," in Proc. NeurIPS Workshop on Machine Learning Safety, New Orleans, LA, USA, Nov. 2022.

[3] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, "Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection," in Proc. ACM Workshop on Artificial Intelligence and Security (AISec), Copenhagen, Denmark, 2023, pp. 79–90.

[4] A. Alon and M. Kamfonas, "Detecting language model attacks with perplexity," in Proc. ICLR Workshop on Secure and Trustworthy Large Language Models, Vienna, Austria, 2024.

[5] I. Markov, A. Dey, O. Harel, and Y. Goel, "Holistic approach to undesired content detection in the real world," in Proc. AAAI Conference on Artificial Intelligence, Washington, DC, USA, 2023, vol. 37, no. 12, pp. 15009–15018.

[6] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "HotFlip: White-box adversarial examples for text classification," in Proc. 56th Annual Meeting of the Association for Computational Linguistics (ACL), Melbourne, Australia, 2018, pp. 31–36.

[7] E. Wallace, S. Feng, N. Kandpal, M. Gardner, and S. Singh, "Universal adversarial triggers for attacking and analyzing NLP," in Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP), Hong Kong, 2019, pp. 2153–2162.

[8] L. Ouyang et al., "Training language models to follow instructions with human feedback," in Advances in Neural Information Processing Systems (NeurIPS), vol. 35, pp. 27730–27744, 2022.

[9] A. Wei, N. Haghtalab, and J. Steinhardt, "Jailbroken: How does LLM safety training fail?" in Advances in Neural Information Processing Systems (NeurIPS), vol. 36, pp. 80079–80110, 2023.

[10] Y. Bai et al., "Constitutional AI: Harmlessness from AI feedback," arXiv preprint arXiv:2212.08073, Dec. 2022.

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proc. Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), Minneapolis, MN, USA, 2019, pp. 4171–4186.

[12] L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5–32, Oct. 2001.