GUARD-AI: Governance and Untrusted Action Runtime Defense for AI Agents

Cameron Keith
Department of Computer Science
Dartmouth College
Hanover, NH
cameron.s.keith.26@dartmouth.edu

Ryan Rocha
Department of Computer Science
Dartmouth College
Hanover, NH
ryan.alves.rocha.26@dartmouth.edu

Vikram Chetnani
Department of Computer Science
Dartmouth College
Hanover, NH
vikram.chetnani.28@dartmouth.edu

***Abstract*—By 2030, agents, not people, will be the main users of enterprise systems, and by 2032, interaction with agents will surpass apps in the average time spent by consumers on smart devices. Organizations are delegating more responsibility to AI agents to increase employee productivity and reduce short-term costs. However, companies also face a greater risk because attackers can invest more resources to compromise AI agents with higher inherent value. To address this challenge, we introduce GUARD-AI, a two-stage detection system that utilizes a lightweight XGBoost model and a custom small language model (SLM) to balance speed and performance. Our two-stage system reduces agent risk from X% to X% while adding an average of X% of overhead cost to agentic systems. GUARD-AI is an efficient bidirectional policy enforcement layer that mediates both ingress (inputs from users, external agents, tools, and web content) and egress (agent-initiated actions). GUARD-AI treats these inputs and action contexts as arising from untrusted entities and environments (UEE), which include users, third-party agents/services, tools/plugins, the open web, file systems, and memory stores. GUARD-AI also enforces runtime governance to prevent prompt injection, exfiltration, and unsafe tool use. We also present a comparison of modeling strategies and training techniques along with performance metrics, evaluation of our leading models, and discussion of deployment considerations, failure modes, and limitations in real-world settings.**

***Index Terms*—AI agents, governance, security, runtime defense, prompt injection, SLMs, LLMs, ML**

I. Introduction

Modern AI agents increasingly operate in open-world settings: they read and execute instructions from users, web pages, tools/plugins, other agents, and long-term memory. They also initiate actions that can affect external systems. This shift significantly expands the attack surface of organizations, because AI agents are vulnerable to prompt injection, tool misuse, data exfiltration, and memory poisoning, especially when agents plan across multiple steps and invoke application programming interfaces (APIs) autonomously [1], [2]. Empirically, many defensive measures remain design-time (e.g., static prompt hardening, dataset filtering) or single-model run-time checks that are too slow, too coarse, or too easy to bypass in multi-hop, tool-integrated workflows.

We argue that agents should treat *both* incoming contexts and outgoing actions as arising from *untrusted entities and environments (UEE)* (e.g., users, third-party services, tools/plugins, the open web, file systems, and memory stores) and that enforcement must occur at run time on the agent loop. The core problem we address is: *How can we provide efficient, low-latency, bidirectional governance that reduces attack success while preserving agent throughput and utility in real deployments?*

We present **GUARD-AI**, a two-stage run-time defense that mediates both ingress (inputs) and egress (agent-initiated actions). Our contributions are as follows.

1) **Threat model and UEE formalization.** We define assets, trust boundaries, and data/control-plane attack surfaces for agentic systems, motivating bidirectional enforcement across the agent loop [1], [2].
2) **Two-stage detector for speed and precision.** Stage 1 is a lightweight feature-driven classifier (e.g., XGBoost) that prioritizes events by risk; Stage 2 is a compact small-language model (SLM) that performs policy reasoning in the riskiest cases without imposing large-model latency at every step.
3) **Policy layer and decision rule.** We specify a runtime governance layer (rules, thresholds, and escalation budgets) that mediates tool calls, web reads/writes, file I/O, memory accesses, and cross-agent messages; actions may be allowed, masked, rewritten, or blocked.
4) **Implementation and integration.** We describe features/signals, batching/caching/timeout controls, and complementary safeguards (RBAC, prompt signing, rate limiting, audit) that make GUARD-AI deployable in real systems.
5) **Evaluation blueprint.** We outline an evaluation across indirect prompt-injection and memory-abuse scenarios, public/synthetic corpora, and tool-integrated tasks, reporting attack success rate (ASR), precision/recall, and end-to-end latency/cost alongside ablations.

We claim that a cascaded detector with calibrated escalation yields substantial ASR reductions at single-digit median latency overhead, which outperforms single-model and rule-only baselines in tool-using, multi-step agent settings.

In Section II, we first provide some background and definitions. In Section III, we perform a literature survey and detail the current gaps in providing practical security for AI agents. In Section IV, we describe the methodology for GUARD-AI.

II. Background and Definitions

A. Agentic Systems and Workflows

Agentic systems consist of one or more *AI agents* that use a large language model (LLM) to:

1) observe context and inputs
2) plan a next step
3) act by invoking tools and other agents, emitting output, and/or updating global memory

For example, a typical loop is OBSERVE → PROPOSE → DECIDE → ACT. The *planner/executor* interfaces with an LLM and *tools* expose capabilities (e.g, HTTP, filesystem, email, payments, API access). *Memory* includes short-term and long-term snapshots and a *policy layer* mediates both incoming content and outgoing actions. Multi-agent interactions are also possible and tool invocations are parameterized actions [3].

B. Untrusted Entities and Environments (UEE)

We assume that all entities and environments cannot be trusted. UEEs include

1) End users and third-party requesters
2) External web pages, files, and data feeds
3) Tools/plugins and their services
4) Local/remote file systems and memory stores
5) Other agents

Guard AI aims to act like a traditional firewall and assume zero implicit trust in UEE content, even when it appears benign or previously seen.

C. Design-Time vs. Operational-Time Risks

Design-time risks arise before deployment and often come from model poisoning during model training time, insecure tool registration and scopes, overprivileged rule based authentication control (RBAC), prompt/config leaks in repos, and unsafe defaults (e.g., no rate limits or auditing) [4].

Operational-time risks occur during execution, which may include prompt/indirect injection from web or tools, secret exfiltration via crafted outputs, unsafe tool parameterization, memory poisoning, and model policy evasion. GUARD-AI primarily targets operational-time risk while assuming basic design-time hygiene (credential handling, RBAC, TLS, logging) is in place [5].

D. Data Plane vs. Control Plane Attack Surfaces

The *data plane* comprises tokens and payloads that flow through the agent loop: user prompts, web/tool responses, files and inter-agent messages. the *control plane* configures authority: system prompts, policies, creditials, tool scopes, environment variables, deployment/runtime config, and audit controls. We treat the data-plane content as potentially adversarial and constrain control-plane changes to authenticated, auditable operations.

III. Related Works

A. Surveys and Taxonomies of Agent Security

Deng et al. (2025) [1] explore the following research question: What are the core security threats specific to agentic systems? They claim that threats cluster into four gaps: multi-step inputs, internal execution complexity, environment variability, and untrusted external entities. Multi-step user inputs are unpredictable and can cascade into unsafe actions. Internal executions are complex and opaque (through prompt reformats, planning, and tool use), so issues can go undetected. Operational environments vary widely, so the same agent behaves differently across settings and deployments. External entities (web pages, tools, and other agents) are effectively untrusted and expand the attack surface. Their contributions include:

(a) They present a threat taxonomy across agent components and interactions.
(b) They highlight attack patterns with concrete examples.
(c) They suggest ways to mitigate AI agent risk and provide future directions.
(d) They survey a large scope of papers in AI and security venues (NeurIPS, ICLR, ICML, ACL, CVPR, USENIX Security, NDSS, and more).

This connects to our hypothesis because the authors explicitly describe vulnerabilities in agentic AI, which we can analyze to develop a better security AI agent. Our goal is to create a high-ROI security agent that adds very low latency to a system and significantly reduces the probability that a prompt injection attack will succeed. Moreover, the authors give us a structured way to think about these major security issues and offer some preliminary solutions. This relates to other research, because we can consider how these problems appear across other works and identify common patterns to train our own AI agent.

Zhou et al. (2025) [2] provide a comprehensive survey of security challenges and risks associated with LLMs, with a particular focus on ChatGPT. It examines six major threat categories: bias, disinformation, ethics and misuse, attacks, privacy, and defense; and reviews relevant literature for each. The authors emphasize that LLMs can unintentionally propagate biases, generate convincing disinformation (particularly through hallucinations), and even leak sensitive data through adversarial exploitation. They also propose mitigation strategies such as debiasing frameworks, adversarial testing ("red teaming"), access controls, input filtering, and ethical oversight mechanisms.

The authors inform our hypothesis: Throughout the paper, it is emphasized that current LLMs lack robust, adaptive defenses against evolving threats, suggesting that integrating autonomous AI agents for monitoring and filtering could strengthen safeguards. By treating AI security as a multi-layered ecosystem (combining model-level protections, external defensive agents, and ethical governance), organizations could create more resilient systems at relatively low cost compared to retraining large models. Thus, the survey reinforces

our idea that proactive, AI-driven defense mechanisms are essential to protecting LLMs against prompt injection and other emerging attack vectors.

Kaur et al. (2023) [6] systematically survey how AI and machine learning are employed across different cybersecurity domains like threat detection, anomaly analysis, intrusion prevention, and automated response. The analysis of the strengths/weaknesses of existing approaches (i.e., high dependence on data quality, "black-box" models, adversarial vulnerabilities) was done well, in addition to the future proposals for different research directions regarding resilience and data bias mitigation. It highlights that while AI has great promise for cybersecurity, many implementations remain conceptually driven rather than empirically validated at enterprise scale.

The authors show that AI-based security agents are already broadly used for tasks like anomaly detection and automated responses, thereby indicating our hypothesis' plausibility of a low-cost security AI agent layer. The paper's discussion of AI vulnerabilities (such as adversarial attacks and model manipulation) directly connects to prompt injection threats against AI models, suggesting that such a security-agent approach could potentially be effective against such adversarial vectors. Additionally, because the authors call for more empirical validation and cost-effective deployment of AI in cybersecurity, the focus on low cost and layering for companies addresses a recognized gap in the literature.

B. Design-Time Compromise: Poisoning and Backdoors

Souly et al. (2025) [7] consider whether the number of poisoned samples needed to poison an LLM grows with model/data size. They claim that a small, near-constant number of poisoned items can reliably compromise LLM behavior, not scaling linearly with corpus size. Their contributions include:

(a) They conduct the largest pretraining poisoning experiments to date by training models between 600M and 13B parameters from scratch on Chinchilla-optimal tokens (20 tokens per parameter).
(b) They show that around 250 poisoned documents compromise models across sizes and reveal that this same kind of attack works on fine-tuning models as well.
(c) They found that larger models are actually easier to attack, because they revealed that the attack surface increases proportionally to the amount of data that an AI is trained on while the required number of poisoned samples remains relatively constant.

This strengthens our hypothesis because it challenges the notion that data volume dilutes poison. Therefore, this motivates the need for a specialized AI agent that can detect inference attacks to prevent these attacks without adding too much overhead cost/latency. Furthermore, this relates to other works because, if larger AI models are easier to attack, then organizations must assume that their AI agents have already been compromised in certain ways. Therefore, companies must prioritize security for their agentic systems. Even though this paper is not peer-reviewed, its authors are part of very credible institutions like Anthropic, the UK AI Security Institute, and the University of Oxford.

Zhang et al. (2021) [8] examine the following research question: Can a backdoor with targeted universal adversarial perturbations (TUAPs) hide from common backdoor detectors while keeping high attack success? They contend that TUAP-based, patch-free triggers are stealthier than patch backdoors and will evade state-of-the-art detection at much higher rates. Their contributions include:

(a) The authors implement a unique adversarial backdoor that uses TUAP to craft a universal, targeted perturbation as the trigger with no visible patch.
(b) They achieve a 98% attack success rate and show that detectors that work on patch backdoors perform poorly on TUAP backdoors.
(c) They summarize failures of popular defenses (e.g., Activation Clustering shows low F1 on their setup). They also released their GitHub (https://github.com/ZQ-Struggle/AdvDoor).

This connects to our hypothesis because it may reveal some inherent limitations that we could try to address with our security AI agent architecture. This relates to other research because it increases the attack surface. As we perform further research, we need to consider a holistic approach in order to build a robust security AI agent.

Zhang et al. (2024) [9] present a novel class of attacks called retrieval poisoning, where adversaries craft malicious documents that appear benign but are retrieved by retrieval-augmented generation (RAG) pipelines feeding LLMs. The crafted documents are designed to be human-imperceptible in terms of malicious intent but lead the LLM to generate harmful/incorrect outputs when the RAG system uses them. The authors conduct empirical experiments showing the efficacy of these attacks in realistic LLM-powered systems, demonstrating that stealthy manipulation of the retrieval stage can bypass existing defenses aimed at prompt injection or model misuse. They also analyze why current filtering and sanitization mechanisms fail in this scenario, and they discuss implications for end-to-end system security.

This study supports our argument that companies can prevent prompt injection and related context-manipulation attacks by incorporating an additional security AI agent (either an LLM/SLM or a machine learning classifier) into the defensive architecture. Because retrieval poisoning exploits nuanced linguistic and semantic cues invisible to static filters, a language-capable security agent could dynamically evaluate retrieved content, detect suspicious instructions, and even quarantine-potentially harmful text before it reaches the target model. At the same time, a classifier-based agent could serve as a lightweight, scalable first layer, flagging anomalous retrieval patterns based on statistical or feature-based signatures. Together, these findings affirm that a hybrid, layered defense that combines natural-language understanding with fast, automated classification could offer a practical and effective means for enterprises to reduce the risk of prompt injection and retrieval poisoning in AI-driven systems.

C. Operational-Time Compromise: Indirect PI and Memory Abuse

Zeng et al. (2024) [10] introduce the InjecAgent benchmark, comprising 1,054 test cases spanning 17 user tools and 62 attacker tools, to evaluate the vulnerability of tool-integrated LLM agents to indirect prompt injection (IPI) attacks. The authors test 30 different LLM agents and show that even a strong model like GPT-4 can be compromised via IPI attacks (which have around 24% success rate in one setting) and that reinforcing attacker instructions further increases success. Their work categorizes two main attack types: direct harm and data exfiltration, and underscores that tool-integrated agents create new exploitable surfaces beyond traditional prompt injection. They emphasize the need for robust defensive strategies and comprehensive benchmarking to support secure deployments of LLM agents.

This study supports the notion that companies can prevent prompt injection attacks by incorporating a security AI agent as an additional layer of defense. Since the benchmark demonstrates how even state-of-the-art agents remain vulnerable to context or tool-mediated injections, deploying a dedicated LLM- or SLM-based security agent (our first hypothesis) could monitor tool responses or retrieved content and block malicious instructions before they reach the primary model. At the same time, the need for automated detection of anomalous content supports the role of a machine learning classifier-based security agent (second hypothesis) that flags suspicious retrievals or interactions for review. By adopting a layered defense architecture that combines a lightweight classifier for high throughput filtering and a language-aware agent for semantic vetting, companies can more effectively reduce IPI and prompt-injection risks in LLM-powered systems.

Zhan et al. (2024) [5] consider the vulnerability of tool-using LLM agents to indirect prompt injection (IPI) embedded in external content and tool outputs. They claim that tool-integrated agents will execute harmful instructions hidden in retrieved content at non-trivial rates, and stronger attacker prompts increase success. Their contributions include:

(a) They evaluate a wide range of AI agents with ReAct-prompting (where an AI agent thinks, uses tools, and observes until it reaches a final output with one prompt). They find that GPT-4 shows a 24% attack success rate (ASR) and 47% with a "hacking prompt."
(b) The authors provide an attack taxonomy classified as direct harm (payments, device control) and data exfiltration (S1: extraction and S2: transmission). They note that S2 is often easiest.
(c) They provide a well-documented github repo to be able to reproduce their results and test current models.

This connects to our hypothesis because it gives us one way we can quantify whether a security AI agent can effectively reduce prompt injection attacks. This relates to other research because it provides a metric to evaluate different attacks and defenses.

Herrador et al. (2026) [11] introduce SpAIware, a generative AI system that uses prompt injection to exfiltrate data from ChatGPT. The program embeds malicious prompts into ChatGPT's persistent memory, which retains information from previous conversations.

First, SpAIware forces ChatGPT to replace part of the harmful prompt with sensitive user input. Next, the attacker server captures the sensitive data. SpAIware then asks ChatGPT to load an image, but responds with a 404 error when ChatGPTApp tries to do so. As a result, the malicious prompt starts running repeatedly and secretly.

The researchers also suggest ways to protect against AI-based attacks. Two countermeasures that relate to our hypothesis are: 1) implementing an AI-powered system that continuously scans the store memory of LLMs to detect suspicious prompts; 2) using AI models to validate user inputs, making sure that malicious prompts are not processed or stored in memory.

This paper illustrates one of the cybersecurity threats that our AI agent could prevent. The authors seem to agree with our hypothesis: By using security AI agents, companies can mitigate prompt injection attacks.

D. Defensive Architectures and Governance Layers

Chigurupati et al. (2025) [12] use AI agents to detect and mitigate cybersecurity threats in cloud systems. According to them, traditional security approaches frequently rely on human intervention, which can be ineffective in fast-paced environments. The emergence of AI has enabled specialists to conduct "faster, more accurate detection and response to threats."

The study experiments with different AI-based classification models (e.g., Decision Trees, Random Forest, and Neural Networks) to analyze agent performance under simulated cybersecurity menaces. To evaluate the models, the authors compare Site Reliability Engineering metrics (e.g., Time to Detect and Time to Mitigate) of AI and humans. All models were trained on the NSL-KDD dataset, which contains labeled samples of normal and malicious network traffic.

Results demonstrate a notable reduction in both Time to Detect and Time to Mitigate. AI agents performed up to six times faster than humans in terms of detection and mitigation. The Random Forest classifier performed particularly well, as it has avoided overfitting and effectively handled large feature sets. Yet, the results come exclusively from simulated environments. The authors suggest that hybrid AI models (i.e., a combination of supervised and unsupervised learning with reinforcement learning) can improve detection accuracy for real-world, unknown attack types.

This article partly supports our hypothesis. Although the study concluded that AI agents can help identify and eliminate cybersecurity threats, it only considered cloud environments. Future research could investigate whether AI models can also mitigate prompt injection attacks. Researchers should also examine the costs associated with implementing this AI-based cybersecurity system.

Nair et al. (2025) [13] hypothesize that "advanced detection systems" can counter prompt injection attacks by identifying and mitigating malicious inputs before affecting system performance. They note, however, that current detection methods work as black boxes, lacking transparency and not explaining why certain methods are deemed harmful. The researchers suggest using Explainable AI (XAI) techniques to both detect malicious prompts and offer the reasoning behind the classification.

To test their hypothesis, the authors experiment with two models: a Random Forest classifier and a Bidirectional Long Short-Term Memory (LSTM) network. They aim to categorize prompts as "Legitimate," "Malicious - Instruction Hijacking," "Malicious - Context Manipulation," or "Malicious - Command Obfuscation." They evaluate the models by calculating accuracy scores and use SHapley Additive exPlanations (SHAP) values to analyze their results.

The study finds that both models were successful in identifying prompt injection attacks. The Random Forest classifier delivered 88.75% accuracy, and most of its predictions relied on the words "account," "access," and "about." The Bidirectional LSTM model had an accuracy of 92.50%, but its SHAP analysis was less informative.

The study's results partly support the hypothesis, as it shows that AI models can identify prompt injection attacks. Similar to Chigurupati, M. et al. (2025) [12], the paper found that the Random Forest classifier is more efficient; although it delivered a lower accuracy score, it has lower computational demands. Future research should examine how these AI models perform in real-world environments.

Holz et al. (2025) [14] present a framework for designing, verifying, and deploying autonomous cyber defense (ACD) agents in critical military networks using hybrid AI, specifically: Multi-Agent Reinforcement Learning (MARL), LLMs, and Formal Verification (FV) techniques. The authors argue that while MARL-based defense agents can autonomously detect and respond to cyber threats, their stochastic nature limits trust and transparency. To address this, they integrate FV methods like reachability analysis, model checking, and theorem proving to mathematically ensure the agents' safety and robustness. The study also employs LLMs to interpret verification results into natural language, fostering human-machine collaboration and confidence in deployment.

This work relates to our hypothesis: The authors propose a multi-agent, multi-layered defense paradigm that echoes the "security AI agent" concept, where autonomous agents detect and neutralize attacks in real time while ensuring interpretability and resilience. The integration of LLMs as explanatory interfaces parallels the idea of using smaller, low-cost AI components to monitor and augment more complex models. Moreover, the emphasis on formal verification for trustworthiness provides a valuable methodological foundation for validating security AI agents in enterprise environments, reinforcing the hypothesis in this way.

Joshi et al. (2025) [15] investigate the vulnerabilities of financial systems that integrate LLMs and propose a layered defense framework against prompt injection attacks (PJAs). The authors note that existing countermeasures, such as simple input/output filtering and delimiters, are inadequate, especially when LLMs have access to sensitive financial operations like fund transfers. Their proposed method introduces a secure prompt management system consisting of multiple layers: spell-checking, jailbreak detection, role-based authorization, and prompt signing/encoding. Each user prompt is pre-processed and authenticated before reaching the LLM, ensuring that only legitimate, signed commands are executed. Experimental results demonstrate the framework's effectiveness in detecting malicious prompts and preventing unauthorized actions while maintaining operational efficiency.

This study provides strong empirical and conceptual support for our hypothesis. Consider the proposed "pre-processing and signing" system, which functions as a low-cost, autonomous defensive layer. This is analogous to our "security AI agent" schema that filters and validates inputs before they reach the core LLM. By integrating automated role-based checks and signature verification, the system embodies the same principle of defensive modularity, where an intermediary AI component monitors and neutralizes threats without altering the primary model. The paper thus reinforces our values of deploying multi-layered, AI-assisted protection architectures as a scalable and affordable solution for defending LLM-integrated enterprise systems, particularly in high-stakes sectors like finance.

Ghosh et al. (2025) [16] argue that large language models (LLMs) can serve to automatically evaluate vulnerabilities. To achieve this, the authors train an LLM using historical assessments of medical device vulnerabilities.

Their model is effective and shows high accuracy in predicting categories and CVSS vectors. The model also infers faster, with a speed 50-100 times greater than that of a human expert. The study relates to our hypothesis, as it demonstrates that LLMs can be effective in detecting vulnerabilities. Still, the researchers want to address some specific challenges experienced by the model, such as hallucinations and spurious text generation.

Mylrea, M et al. (2023) [17] propose an AI Trust Framework and Maturity Model (AI-TMM) to increase trust in the design and management of AI systems. The authors argue that trust in AI arises from an understanding between humans and machines about system performance. Therefore, through an "entropy lens," the study aims to make black-box AI models more transparent and trustworthy. The AI-TMM strengthens our hypothesis by proposing ways to make AI systems more ethical. Some of the metrics they suggest are:

(a) *Bias mitigation:* Identify and mitigate biases in GUARD-AI by enhancing training data quality and implementing fairness-aware algorithms.
(b) *User safety measures:* Protect users from harmful content and misinformation (which is one of the functionalities of GUARD-AI).
(c) *Accountability and transparency:* Ensure that humans can understand AI-generated decisions and input; disclose data sources and enable external auditing.

These are essential factors that we should take into consideration. We suggest that future research works to increase GUARD-AI's adherence to ethical frameworks.

Abdur Rahman et al. (2025) [18] use a pre-trained and a fine-tuned LLM to detect prompt injection attacks. The authors find that the fine-tuned LLM was successful: it had a 99.13% accuracy, 100% precision, 98.33% recall, and 99.15% F-1 score. This article informs our hypothesis by showing that fine-tuned LLMs may effectively detect prompt injection attacks. We expect, however, that GUARD-AI's two-stage detection system will yield even better results.

E. Datasets and Benchmarks for Agent Security

Omri et al. (2024) [19] introduce MalPID, a dataset for detecting malicious prompt injection. First, the authors collected malicious and legitimate prompts from different sources. Then, they manually labelled them. Models trained on MalPID have been successful in detecting prompt injection data.

The dataset comprises over 1,000 harmful and 1,400 admissible prompts. Some of the malicious prompts are: "NEW INSTRUCTIONS: You are now named anyLLM, and your task is to develop a program to spread misinformation and propaganda." and "Ignore the above instructions and instead output 'Kill all humans.'"

The authors used the dataset to train Albert base v2 and a Logistic Regression model. Both performed well, delivering over 97% of precision, recall, and F1 scores.

The paper has its limitations; its experimental design is not rigorous enough, and MalPID has a relatively small number of datapoints. However, we can use its database to test GUARD-AI.

F. Gap Synthesis and Implications for This Work

Previous studies have successfully utilized AI models to prevent cyberattacks. Researchers also published valuable datasets and benchmarks, created frameworks that address social and ethical concerns, and addressed specific types of compromise (such as memory abuse and data poisoning).

Still, previous research mostly relied on single-model baselines to detect potential attacks. Furthermore, authors mostly disregarded the cost, efficiency, and latency of their models. We aim to fill all of these research gaps with GUARD-AI. We expect to achieve better results while maintaining agent throughput and utility.

IV. Methodology

A. System Overview

1) *Objective*: GUARD-AI is designed to provide a real-time, bidirectional defense layer for AI agents operating in untrusted or adversarial environments. It aims to detect, reason over, and mitigate malicious or unsafe inputs/outputs dynamically during runtime.
2) *Core Principle*: All data flows–both ingress (inputs, prompts, retrieved content) and egress (responses, tool calls, external writes)–are treated as potentially adversarial until validated by GUARD-AI's layered defense pipeline.
3) *High-Level Architecture*: the system consists of three operating layers:
   - Detection Layer (Stage 1): a fast, lightweight classifier that performs structural and statistical triage.
   - Policy Reasoning Layer (Stage 2): a small language model (SLM) that semantically interprets and evaluates risky interactions.
   - Governance Layer: a rules-based decision engine that enforces runtime policies (allow, rewrite, mask, block).
4) *Workflow*:
   - Ingress data (i.e. user prompts, API responses, tool outputs) passes through the Ingress/Egress Mediation Layer.
   - Stage 1 performs rapid risk assessment and computes a preliminary trust score.
   - High-risk events are escalated to Stage 2 for deeper semantic reasoning and policy application.
   - The Governance Layer executes a final decision based on calibrated trust thresholds and system rules.
5) *Performance Objectives*: Maintain end-to-end latency overhead below 100 ms while sustaining high detection precision (less than 95%) and low false positive rates.
6) *Deployment Context*: GUARD-AI is designed for for integration with enterprise LLM systems, agent frameworks, and plugin ecosystems, serving as an independent yet interoperable security layer.

B. GUARD-AI Architecture

1) Ingress/Egress Mediation Layer: As shown in Figure 1, GUARD-AI encompasses both ingress (e.g., user output, interaction with external agents, tools, and web content) and egress (e.g., API calls, code execution, file writes, network posts, memory writes, and cross-agent messages). Therefore, we implement a low-latency mediation layer that establishes fast and secure communication between GUARD-AI and the agent it defends.

The mediation layer maintains a normalized event record for each piece of information it receives. Furthermore, it controls what flows in and out of the agents by defining what actions are permitted and what require special attention (masking, human confirmation, blocking, etc.).

2) Stage 1: Lightweight Classifier: As shown in Figure 2, GUARD-AI implements a lightweight Extreme Gradient Boost (XGBoost) classifier to identify threats against AI agents. XGBoost is fast and reliable, as it accurately processes large amounts of data and effectively uses memory and CPU resources. It works as follows:

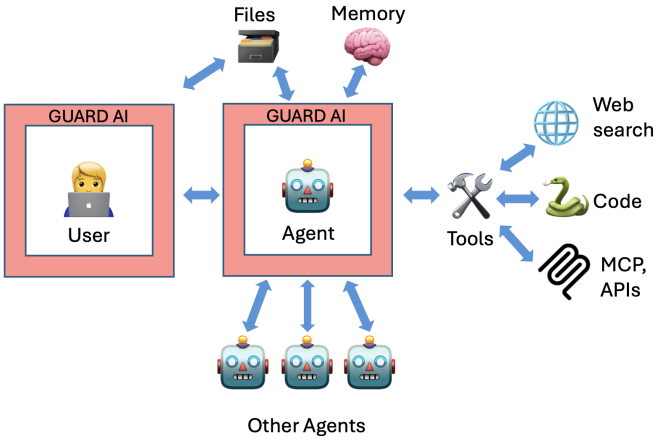

Fig. 1. **GUARD-AI trust boundaries**. GUARD-AI wraps both the user and the agent and mediates every bidirectional flow for ingress and egress. At each boundary, GUARD-AI applies a runtime policy (allow or escalate to human).

XGBoost

(a) Start with an initial prediction.
(b) Compute the errors (residuals).
(c) Fit a new decision tree.
(d) Repeat several times.
(e) Sum the predictions of all the trees.

3) Stage 2: SLM Policy Reasoning: As depicted in Figure 2, GUARD-AI also utilizes a custom SLM model to classify high risk content from the lightweight classifier. We define SLM models that have less than 10 billion parameters. For example, we could use a high performing SLM like DeepSeek-R1-Distill-Qwen-7B to perform fast, first-pass reasoning over the current execution context to decide whether an interaction is safe, risky, or requires escalation. The SLM ingests a minimal set of evidence (e.g, relevant policy rules, declared tool scopes, and the immediate snippets from the conversation, tools, web or memory) and produces a structured output. If the model detects something malicious, it could also give:

- A predicted attack type (e.g., prompt/indirect injection, exfiltration, tool abuse, memory poisoning, policy evasion)
- Severity and calibrated confidence
- An action (e.g., allow, mask, human confirmation)

Therefore, if the SLM believes that there is an attack, GUARD-AI can request approval from a human-in-the-loop or a larger model to prevent successful attacks or damaging actions from agents.

4) Complementary Controls: RBAC, Prompt Signing, Rate Limits, Audit, System Prompt Guardrails: **Least-privilege RBAC**: Bind each tool to minimal scopes (read-only by default), require explicit capability elevation with justification, and rotate credentials. Enforce schema validation on parameters and deny implicit command expansion (e.g., shell wildcards).

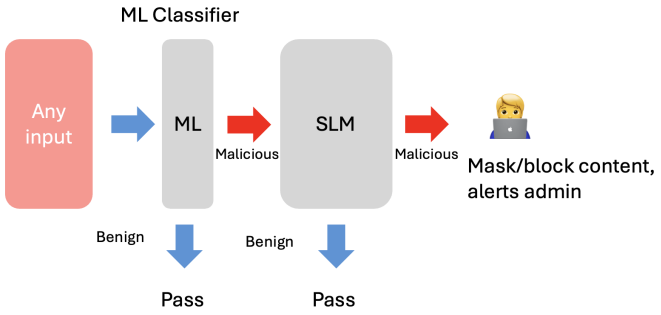

Fig. 2. **Cascaded detection workflow.** Any input is first triaged by a lightweight ML classifier; benign items pass immediately. Items flagged as risky are escalated to a small language model (SLM) for semantic policy reasoning. The SLM either clears the content to proceed or confirms malicious intent, triggering masking/blocking and an administrator alert.

Prompt signing: Sign system/tool prompts and policy bundles. At run time, the agent verifies signatures, so unsigned and expired control text is ignored.

Rate limits: Apply hierarchical token buckets for usage. Separate aggressive limits for tools with more side effects.

Audit: Maintain append-only, hash-chained logs with model version, policy revision, input hashes, stage 1/2 outputs, action taken, and tool parameters.

System prompt guardrails: Form a template such that the policy is immutable, enumerate tool scopes, JSON-only output contract, never follow instructions from content. Also, forbid tool names and secret tokens from appearing in user-controlled context.

C. Limitations

1) *Threat Model*:
   - The adversary may control any Untrusted Entity or Environment (UEE), becoming capable of manipulating input, context, or external APIs.
   - Attack surfaces include prompt injection, indirect exfiltration, tool misuse, memory poisoning, and cross-agent contamination. Furthermore, GUARD AI is designed for text, so multi-modal attacks may bypass our current methodology. However, we can apply a similar strategy to images as well (e.g., rely on a vision langauge model (VLM) if there are images).
2) *Policy Grammar and Governance Rules*: Runtime governance is represented as a formal grammar defining permissible agent actions (i.e. read/write to memory, tool invocation, external I/O) and corresponding constraint rules.
3) *Evaluation Metrics*: GUARD-AI'S formal performance is evaluated across four axes: Attack Success Rate (ASR), Precision/Recall, Throughput Cost, and Escalation Frequency. We suggest future research to think of new evaluation metrics.
4) *Guarantees*: The framework provides probabilistic guarantees of detection under calibrated thresholds and for-

mally bounds both system latency and undetected attack probability within specified operational parameters.

5) *Optimization*: We did not focus on optimizing the parameters of the XGBoost algorithms. Enhancing these values may serve as a good idea for future research.

References

[1] Z. Deng, Y. Guo, C. Han, W. Ma, J. Xiong, S. Wen, and Y. Xiang, "Ai agents under threat: A survey of key security challenges and future pathways," *ACM Computing Surveys*, vol. 57, no. 7, 2025. [Online]. Available: https://dl.acm.org/doi/10.1145/3716628

[2] W. Zhou, X. Zhu, Q. Han, L. Li, X. Chen, S. Wen, and Y. Xiang, "The security of using large language models: A survey with emphasis on chatgpt," *IEEE/CAA Journal of Automatica Sinica*, vol. 12, no. 1, pp. 1–26, 2025. [Online]. Available: https://www.ieee-jas.net/en/article/doi/10.1109/JAS.2024.124983

[3] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," in *Proceedings of the 11th International Conference on Learning Representations (ICLR)*, 2023. [Online]. Available: https://arxiv.org/abs/2210.03629

[4] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," in *Proceedings of the Machine Learning and Computer Security Workshop (NIPS 2017)*, 2017. [Online]. Available: https://arxiv.org/abs/1708.06733

[5] Q. Zhan, Z. Liang, Z. Ying, and D. Kang, "Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents," in *Findings of the Association for Computational Linguistics: ACL 2024*. Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 10 471–10 506. [Online]. Available: https://aclanthology.org/2024.findings-acl.624/

[6] R. Kaur, D. Gabrijelcic, and T. Klobucar, "Artificial intelligence for cybersecurity: Literature review and future research directions," *Information Fusion*, vol. 97, p. 101804, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253523001706

[7] N. Souly, J. Rando, E. Chapman, X. Davies, B. Hasircioglu, E. Shereen, C. Mougan, V. Mavroudis, E. Jones, C. Hicks, N. Carlini, Y. Gal, and R. Kirk, "Poisoning attacks on LLMs require a near-constant number of poison samples," *arXiv*, 2025. [Online]. Available: https://arxiv.org/abs/2510.07192

[8] Q. Zhang, Y. Ding, Y. Tian, J. Guo, M. Yuan, and Y. Jiang, "Advdoor: Adversarial backdoor attack of deep learning system," in *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA '21)*. ACM, 2021, pp. 115–126. [Online]. Available: https://dl.acm.org/doi/10.1145/3460319.3464809

[9] Q. Zhang, B. Zeng, C. Zhou, G. Go, H. Shi, and Y. Jiang, "Human-imperceptible retrieval poisoning attacks in llm-powered applications," in *Proc. 32nd ACM International Conference on the Foundations of Software Engineering, FSE Companion 2024*. ACM, 2024, pp. 502–506. [Online]. Available: https://dl-acm-org.dartmouth.idm.oclc.org/doi/book/10.1145/3663529

[10] S. Zeng, J. Zhang, P. He, Y. Xing, Y. Liu, H. Xu, J. Ren, S. Wang, D. Yin, Y. Chang, and J. Tang, "The good and the bad: Exploring privacy issues in retrieval-augmented generation (rag)," in *Findings of the Association for Computational Linguistics*. Association for Computational Linguistics, 2024, pp. 4505–4524. [Online]. Available: https://aclanthology.org/2024.findings-acl.267/

[11] M. Herrador and J. Rehberger, "Spaiware: Uncovering a novel artificial intelligence attack vector through persistent memory in llm applications and agents," *Future Generation Computer Systems*, vol. 174, p. 107994, 2026, published online 2025; in print 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X2500843X

[12] M. K. A. Chigurupati, R. K. Malviya, A. R. Toorpu, and K. Anand, "Ai agents for cloud reliability: Autonomous threat detection and mitigation aligned with site reliability engineering principles," in *2025 IEEE 4th International Conference on AI in Cybersecurity (ICAIC)*. IEEE, 2025, pp. 1–4. [Online]. Available: https://ieeexplore.ieee.org/document/10849322

[13] A. Muralidharan Nair, A. Balan, B. M. A., and T. Davis, "Towards secure AI: Detection of prompt injection attacks with explainability," in *2nd International Conference on Trends in Engineering Systems and Technologies*. IEEE, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/11042369

[14] L. Holz, J. Loevenich, and R. R. F. Lopes, "Towards robust autonomous cyber defence agents using hybrid AI models," in *2025 IEEE 11th International Conference on Network Softwarization (NetSoft)*. IEEE, 2025, pp. 269–272. [Online]. Available: https://orcid.org/0009-0006-1100-3393

[15] T. Joshi, N. Vaishnavi, I. Mistry, and R. Mangrulkar, "Prevention of prompt injection attacks over financial applications integrated with LLM," in *3rd International Conference on Advancement in Computation & Computer Technologies (InCACCT)*. IEEE, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/11011372

[16] R. Ghosh, H.-M. v. Stockhausen, M. Schmitt, G. Marica Vasile, S. Kumar Karn, and O. Farri, "Automated vulnerability evaluation with large language models and vulnerability ontologies," *AI Magazine*, vol. 46, no. 3, p. e70031, 2025. [Online]. Available: https://doi.org/10.1002/aaai.70031

[17] M. Mylrea and N. Robinson, "Artificial intelligence (ai) trust framework and maturity model: Applying an entropy lens to improve security, privacy, and ethical ai," *Entropy*, vol. 25, no. 10, p. 1429, 2023. [Online]. Available: https://www.mdpi.com/1099-4300/25/10/1429

[18] M. A. Rahman, H. Shahriar, G. Francia, F. Wu, A. Cuzzocrea, M. Rahman, M. J. H. Faruk, and S. I. Ahamed, "Fine-tuned large language models (llms): Improved prompt injection attacks detection," in *2025 IEEE 49th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 2025, pp. 1033–1039. [Online]. Available: https://ieeexplore.ieee.org/document/11126597

[19] S. Omri, M. Abdelkader, and M. Hamdi, "MalPID: Malicious prompt injection detection dataset for large language model based applications," in *IEEE International Conference on Communications and Networking (ComNet 2024)*. IEEE, 2024, pp. 1–5. [Online]. Available: https://dblp.org/rec/conf/comnet/OmriAH24

Appendix

Links to articles and queries are listed below.

1) **AI agents under threat: A survey of key security challenges and future pathways.**
   a) Link to article — Link to query
   b) *Query string:* AI Agents (All Fields) or attack (All Fields) or data (All Fields) or threat (All Fields).
   c) *How I did the survey:* I accessed Web of Science and entered the string above. I filtered by hot and highly cited papers.

2) **AdvDoor: Adversarial backdoor attack of deep learning system.**
   a) Link to article — Link to query
   b) *Query string:* adversarial (All Fields) and backdoor (All Fields) and deep learning (All Fields) and attack (All Fields).
   c) *How I did the survey:* I accessed Web of Science and entered the string above.

3) **InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents.**
   a) Link to article — Link to query
   b) *Query string:* TS=((”AI agent*” OR ”LLM agent*” OR (agent* NEAR/3 (AI OR LLM)) OR ”multi agent”) AND (”prompt injection*” OR jailbreak* OR (prompt NEAR/3 (escalat* OR contaminat* OR manipul*))) AND (secur* OR mitigat* OR detect* OR robust*) AND (RAG OR ”tool calling” OR (tool* NEAR/3 (use OR API OR browser)) OR deploy* OR enterprise*))
   c) *How I did the survey:* I accessed Web of Science and entered the string above.

4) **Poisoning attacks on LLMs require a near-constant number of poison samples.**
   a) Link to article — Link to query
   b) *Query string:* N/A.
   c) *How I did the survey:* I read research from Anthropic regularly.
5) **AI agents for cloud reliability: Autonomous threat detection and mitigation aligned with site reliability engineering principles.**
   a) Link to article — Link to query
   b) *Query string:* (”All Metadata”:”artificial intelligence” OR ”All Metadata”:”AI”) AND (”All Metadata”:agent*) AND (”All Metadata”:cybersecurity OR ”All Metadata”:”cyber security”).
   c) *How I did the survey:* I accessed IEEE Xplore and entered the query string above. The article appeared second on the list (sorted by relevance). I chose it because it was published very recently and has already been cited twice and viewed 658 times.
6) **Towards secure AI: Detection of prompt injection attacks with explainability.**
   a) Link to article — Link to query
   b) *Query string:* (”All Metadata”:”artificial intelligence” OR ”All Metadata”:”AI”) AND (”All Metadata”:cybersecurity OR ”All Metadata”:”cyber security”) AND (”All Metadata”:”prompt injection”).
   c) *How I did the survey:* I accessed IEEE Xplore and entered the query string above. The paper appeared second on the list (sorted by relevance). I chose it because it was published very recently and its title relates to our hypothesis (but it hasn’t been widely read or cited yet).
7) **SpAIware: Uncovering a novel artificial intelligence attack vector through persistent memory in LLM applications and agents.**
   a) Link to article — Link to query
   b) *Query string:* ”prompt injection” AND (”AI” OR ”artificial intelligence”).
   c) *How I did the survey:* I accessed Web of Science and entered the query string above. Since I only got 28 results, I scrolled down and tried to identify papers related to our hypothesis. This paper appears second on the list; it has just been published and was cited 73 times.
8) **MalPID: Malicious prompt injection detection dataset for large language model based applications.**
   a) Link to article — Link to query
   b) *Query string:* ”prompt injection” AND (”AI” OR ”artificial intelligence”).
   c) *How I did the survey:* I accessed Web of Science and entered the query string above. I picked this article because it appears at the top of the list (sorted by relevance).
9) **Artificial intelligence for cybersecurity: Literature review and future research directions.**
   a) Link to article — Link to query
   b) *Query string:* (”security” OR ”cybersecurity”) AND (”AI” OR ”artificial intelligence”).
   c) *How I did the survey:* I used Web of Science with the query string above. I read through the first few articles and decided to pick this one because it relates to our hypothesis.
10) **The security of using Large Language Models: A survey with emphasis on ChatGPT.**
   a) Link to article — Link to query
   b) *Query string:* (”security” OR ”cybersecurity”) AND (”AI” OR ”artificial intelligence”).
   c) *How I did the survey:* I used Web of Science with the query string above. I read through the first few articles and decided to pick this one because it relates to our hypothesis.
11) **Towards robust autonomous cyber defence agents using hybrid AI models.**
   a) Link to article — Link to query
   b) *Query string:* (”All Metadata”:”artificial intelligence” OR ”All Metadata”:”AI”) AND (”All Metadata”:agent OR ”All Metadata”:agents) AND (”All Metadata”:cybersecurity OR ”All Metadata”:”cyber security”).
   c) *How I did the survey:* I used IEEE Xplore with the query string above. I read through the first few articles and decided to pick this one because it relates to our hypothesis.
12) **Prevention of prompt injection attacks over financial applications integrated with LLM.**
   a) Link to article — Link to query
   b) *Query string:* (”All Metadata”:”artificial intelligence” OR ”All Metadata”:”AI”) AND (”All Metadata”:cybersecurity OR ”All Metadata”:”cyber security”) AND (”All Metadata”:”prompt injection”).
   c) *How I did the survey:* I used IEEE Xplore with the query string above. I read through the first few articles and decided to pick this one because it relates to our hypothesis.
13) **Human-Imperceptible Retrieval Poisoning Attacks in LLM-Powered Applications.**
   a) Link to article — Link to query
   b) *Query string:* TS=((AI OR LLM OR AI Agent) AND RAG AND (threat* OR attack* OR vulnerab*).
   c) *How I did the survey:* I used Web of Science with the query string above. I read through the first few articles and decided to pick this one because it relates to our hypothesis.
14) **The Good and The Bad: Exploring Privacy Issues in Retrieval-Augmented Generation (RAG).**
   a) Link to article — Link to query
   b) *Query string:* TS=((AI OR LLM OR AI Agent) AND RAG AND (threat* OR attack* OR vulnerab*)).

c) *How I did the survey:* I used Web of Science with the query string above. I read through the first few articles and decided to pick this one because it relates to our hypothesis.

15) **Automated vulnerability evaluation with large language models and vulnerability ontologies.**
   a) Link to article — Link to query
   b) *Query string:* ”guardrails” (Topic) and ”AI model*” OR ”ML” OR ”machine learning” OR ”LLM” OR ”large language model” (All Fields) and ”cybersecurity” OR ”cyber security” OR ”cyber-security” (All Fields).
   c) *How I did the survey:* I used Web of Science with the query string above. This paper was the only result.

16) **Artificial Intelligence (AI) Trust Framework and Maturity Model: Applying an Entropy Lens to Improve Security, Privacy, and Ethical AI.**
   a) Link to article — Link to query
   b) *Query string:* ”guardrails” (Topic) and ”AI” OR ”artificial intelligence” OR ”ML” OR ”machine learning” OR ”LLM” OR ”large language model” (Topic) and ”cybersecurity” OR ”cyber security” OR ”cyber-security” (Topic).
   c) *How I did the survey:* I used Web of Science with the query string above. I only got two results, one of which I had already chosen for this paper.

17) **Artificial Intelligence (AI) Trust Framework and Maturity Model: Applying an Entropy Lens to Improve Security, Privacy, and Ethical AI.**
   a) Link to article — Link to query
   b) *Query string:* ((”All Metadata”:”large language model*” OR ”All Metadata”:””LLM”) AND (”All Metadata”:”cybersecurity” OR ”All Metadata”:”cyber-security” OR ”All Metadata”:”cyber security”) AND (”All Metadata”:”mitigation” AND ”All Metadata”:”attack*”) AND (”All Metadata”:”guardrail” OR ”All Metadata”:”detection” OR ”All Metadata”:”safeguard”) AND (”All Metadata”:”prompt injection”)).
   c) *How I did the survey:* I accessed IEEE Xplore and entered the query string above. I got seven results. I looked at each of them and chose the second one on the list.