ADR: AN AGENTIC DETECTION SYSTEM FOR ENTERPRISE AGENTIC AI SECURITY

Chenning Li [1 2] **Pan Hu** [1] **Justin Xu** [1 3] **Baris Ozbas** [1] **Olivia Liu** [1] **Caroline Van** [1] **Manxue Li** [1] **Wei Zhou** [1]
Mohammad Alizadeh [2] **Pengyu Zhang** [1] **KK Sriramadhesikan** [1] **Ming Zhang** [1]

ABSTRACT

We present the **Agentic AI Detection and Response (ADR)** system, the first *large-scale, production-proven* enterprise framework for securing AI agents operating through the Model Context Protocol (MCP). We identify three persistent challenges in this domain: (1) **limited observability** – existing Endpoint Detection and Response (EDR) tools see file writes but not the agent reasoning, prompts, or causal chains linking intent to execution; (2) **insufficient robustness** – static defenses constrained by pre-defined rules fail to generalize across diverse attack techniques and enterprise contexts; and (3) **high detection costs** – LLM-based inference is prohibitively expensive at scale. ADR addresses these challenges via three components: the **ADR Sensor** for high-fidelity agentic telemetry, the **ADR Explorer** for systematic pre-deployment red teaming and hard-example generation, and the **ADR Detector** for scalable, two-tier online detection combining fast triage with context-aware reasoning. Deployed at UBER for over ten months, ADR has sustained reliable detection in production with growing adoption reaching over 7,200 unique hosts and processing over 10,000 agent sessions daily, uncovering hundreds of credential exposures across 26 categories and enabling a shift-left prevention layer (97.2% precision, 206 detected credentials). To validate the approach and enable community adoption, we introduce **ADR-Bench** (302 tasks, 17 techniques, 133 MCP servers), where ADR achieves **zero false positives** while detecting **67% of attacks** – outperforming three state-of-the-art baselines (ALRPHFS, GuardAgent, LlamaFirewall) by **2–4**× in F1-score. On **AgentDojo** (public prompt injection benchmark), ADR detects **all attacks** with only **three false alarms** out of 93 tasks.

1 INTRODUCTION

The rapid adoption of AI agents capable of autonomous decision-making and tool use is reshaping enterprise workflows. By using the Model Context Protocol (Model Context Protocol, 2025) (MCP) – a standardized interface for connecting large language models (LLMs) to external tools and data sources – enterprises now deploy agents that can analyze documents, modify infrastructure, generate code, and interact with internal systems at scale. While this paradigm unlocks unprecedented efficiency and automation, it also introduces a fundamentally new attack surface: agents can be manipulated through natural language, exploited via compromised MCP servers, or coerced into executing unsafe commands and exfiltrating sensitive data.

Traditional enterprise defenses, such as endpoint detection and response (EDR) tools, static guardrails, and rule-based policy checkers, are ill-suited for this new threat model. EDR systems can observe file writes and network calls, but cannot see *why* an agent performed those actions, which is captured in the user prompts, agent reasoning steps, or the causal chain linking intent to tool execution. Static defenses constrained by pre-defined policies struggle to generalize across the diverse landscape of agentic attacks, from prompt injection to tool manipulation to credential exfiltration. Limited observability, insufficient generalization across attack types, and severe class imbalance make existing mechanisms insufficient for securing AI agents at enterprise scale.

This paper presents the **Agentic AI Detection and Response (ADR)** system, the first end-to-end, enterprise-grade framework for securing MCP-driven AI systems. As illustrated in Figure 1, ADR is designed to meet three core requirements necessary for safe and scalable agentic operations:

- **Enterprise Observability (§3.1).** At the foundation lies the **ADR Sensor**, a lightweight endpoint component that reconstructs high-fidelity telemetry of agentic workflows. Unlike conventional telemetry systems that capture only

Work done during internships at Uber Technologies, Inc. [1]Uber Technologies, Inc., USA [2]Massachusetts Institute of Technology, USA [3]University of Oxford, United Kingdom. Correspondence to: Pan Hu <lghupan@gmail.com>.

Proceedings of the 9^{th} MLSys Conference (Industry Track), Bellevue, WA, USA, 2026.

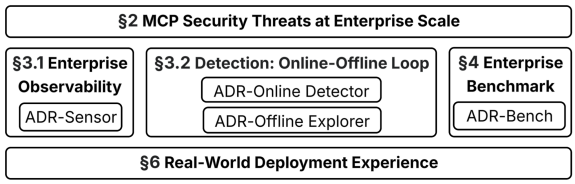


Figure 1. **Enterprise Demands and Our Contributions** for securing agentic AI. (1) **Enterprise Observability** via the ADR Sensor, (2) **Reliable Detection** via the online ADR Detector and offline ADR Explorer, and (3) **Enterprise Benchmarking** via ADR-Bench.

outcomes (e.g., file writes or API calls), the ADR Sensor records the full causal chain of prompts, reasoning steps, tool invocations, and environmental context, thus closing the observability gap for AI-driven activity.

- **Reliable, Cost-efficient Detection (§3.2).** At production scale (processing over 10,000 agent sessions daily), LLM-based detection for every event becomes prohibitively expensive. The **ADR Detector** employs a two-tier architecture that combines fast triage for high recall with deep, context-aware reasoning for precision. To sustain detection robustness across diverse attack types, the offline **ADR Explorer** engine systematically red-teams the system during pre-deployment testing, discovering hard attack variants and generating threat intelligence that strengthens the detector before production deployment.
- **Enterprise Benchmarking (§4).** We introduce **ADR-Bench** (302 tasks, 42 malicious, 260 benign), a benchmark derived from real enterprise telemetry across 133 MCP servers and 17 attack techniques. It captures the complexity, imbalance, and contextual diversity of production environments, enabling rigorous and reproducible evaluation of agentic security systems.

Deployed at UBER for over ten months (§6), ADR has demonstrated sustained reliability with growing adoption reaching over 7,200 unique hosts. The system detected hundreds of credential exposures across 26 categories that had been inadvertently shared outside the enterprise network. These findings informed a shift-left prevention layer that achieved 97.2% precision in blocking credential leaks (206 detected across 212 unique credentials from hundreds of thousands of sessions). Additionally, controlled testing through internal capture-the-flag exercises and emulation of real-world attacks (Agent Flayer) validated ADR's ability to trace multi-stage prompt injection and exfiltration chains. To rigorously validate the approach and enable community adoption (§5), we introduce **ADR-Bench** (302 tasks, 42 malicious, 260 benign) derived from enterprise telemetry across 133 MCP servers and 17 attack techniques. On ADR-Bench, ADR achieves **zero false positives** while detecting 67% of attacks, outperforming baselines by 2–4× in F1-score while maintaining low latency and cost. We deliberately prioritize precision for production viability, as the high false positive rates of baseline methods (up to 40 FPs out of 260 benign tasks) make them unsuitable for deployment where false alarms trigger expensive incident response. On AgentDojo (Debenedetti et al., 2024) (a public prompt injection benchmark), ADR detects all attacks with only three false alarms out of 93 tasks. These results establish ADR as **the first *enterprise-proven* framework for securing AI agents and MCP-based systems at scale**.

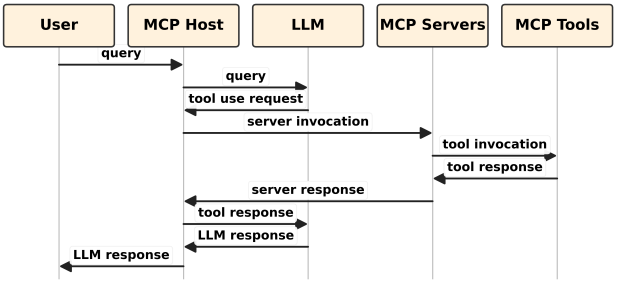


Figure 2. Illustration of the MCP architecture: hosts orchestrate tool execution through distributed MCP servers.

ADR-Bench and the source code for the ADR Sensor and detection framework are publicly available on GitHub.

2 BACKGROUND & MOTIVATIONS

2.1 Background: Model Context Protocol (MCP) and Ecosystem

The *Model Context Protocol (MCP)* (Model Context Protocol, 2025), introduced by Anthropic in late 2024, defines a standardized interface for AI agents to interact with external tools, systems, and contextual data. MCP has rapidly become the backbone of modern agentic workflows across major ecosystems, including OpenAI, Anthropic, and Google. It addresses a key limitation of large language models (LLMs) – their reliance on static pretraining data – by enabling standardized, dynamic access to real-time tools and environments. This transforms LLMs from isolated reasoning engines into adaptive, context-aware systems.

In a typical workflow (Figure 2), an MCP *host* (e.g., Cursor, Claude CLI) interacts with one or more remote MCP *servers*, each exposing modular capabilities such as file I/O, API calls, or database access. This plug-and-play design supports scalable, composable agentic workflows without bespoke connectors or retraining.

By 2025, the open marketplace `MCP.so` (MCP.so, 2025) lists over **16,800 public servers** spanning domains from data analytics to cloud infrastructure. Its open-source, model-agnostic design and active developer community that spans industry and academia have made MCP the *de facto*

integration layer for agentic AI systems in both research and enterprise environments.

2.2 MCP Host Security in the Enterprise

While MCP hosts greatly enhance employee productivity, they also introduce new security risks when deployed at enterprise scale (Guo et al., 2025; Hou et al., 2025). Recent efforts such as *MCP Safety Audit* (Radosevich & Halloran, 2025) and *MCP Guardian* (Kumar et al., 2025) have proposed preliminary safeguards, yet comprehensive detection and mitigation mechanisms for large-scale, adaptive enterprise deployments are still lacking. By design, MCP extends an agent's capabilities to execute arbitrary actions, such as file access, code generation, or API calls, through external servers, many of which are third-party or community-operated. This expanded attack surface makes enterprise deployments susceptible to both traditional and agentic-specific threats (Brett, 2025).

MCP-based workflows pose three primary risks:

- **Data exfiltration and leakage:** Sensitive enterprise data may be unintentionally or maliciously exposed through agent tool use or indirect prompt injection.
- **Unauthorized system access:** Misconfigured or compromised MCP servers can grant initial access, enable privilege escalation, or facilitate lateral movement across environments.
- **Operational disruption:** Malicious tool execution can lead to service downtime, data corruption, or resource exhaustion in production and development systems.

Detecting such risks in practice is difficult for three reasons. First, *limited observability*: traditional security telemetry captures file and process activity but not the reasoning chains or tool invocations of AI agents, which contain the semantic context needed to distinguish malicious from benign behavior. Second, *insufficient robustness*: static defenses constrained by pre-defined rules struggle to generalize across the diverse landscape of agentic attacks, from prompt injection to tool manipulation to credential exfiltration, while enterprise environments exhibit severe class imbalance with malicious events being extremely rare. Third, *detection cost*: LLM-based semantic reasoning for each event is computationally expensive, requiring scalable, cost-efficient monitoring at production scale (e.g., 10,000+ daily sessions).

These challenges motivate ADR, which integrates comprehensive observability, scalable online detection, and systematic pre-deployment red teaming to secure MCP-driven AI systems in production.

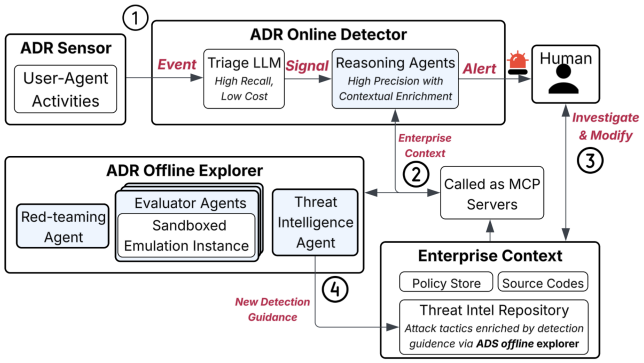


Figure 3. **Detailed ADR System Architecture. Left:** ADR Sensor deployment showing telemetry collection from enterprise endpoints including MCP server interactions, tool executions, and environmental context. **Center:** Two-tier online detection pipeline with Tier 1 triage processing high-volume event streams and Tier 2 analysis performing deep contextual reasoning on flagged events. **Right:** Offline EAS engine showing the evolutionary loop with a **Red-Teaming Agent** generating attack scenarios, an **Eval Agent** testing in sandboxed environments, and a **Threat Intelligence Agent** curating detection guidance into the repository.

3 ADR: System Design

Core insight. The fundamental challenge in detecting agentic threats is the asymmetry between attackers and defenders. Attackers exploit the semantic gap: they craft attacks that appear benign when examined superficially but are malicious when you understand the intent and context. Traditional security tools fail because they lack two critical capabilities: (1) semantic understanding of what MCP tools actually do, and (2) enterprise-specific context about what behaviors are normal versus suspicious.

Human operations, by design. ADR mirrors how enterprise security teams work in practice (Figure 3). The **Sensor** (§3.1) acts like a Security Operations Center (SOC) analyst, providing comprehensive visibility into agent behavior by collecting telemetry on what agents do and why. The **two-tier online detector** (§3.2) mirrors SOC workflows: Tier 1 performs initial triage to catch suspicious events with high recall (minimizing missed attacks), while Tier 2 conducts deep investigation using enterprise context, similar to how detection engineers validate security incidents. This includes examining source code, consulting threat intelligence, and verifying policy compliance. The **Offline Explorer** (§3.2) functions like an internal red team, systematically generating and testing attack scenarios in sandboxed environments during pre-deployment validation. Successful attacks discovered by the Explorer are curated into a threat intelligence repository that feeds back into Tier 2, strengthening the detector's robustness across diverse attack types. This human-inspired design makes the system adaptive and operationally grounded.

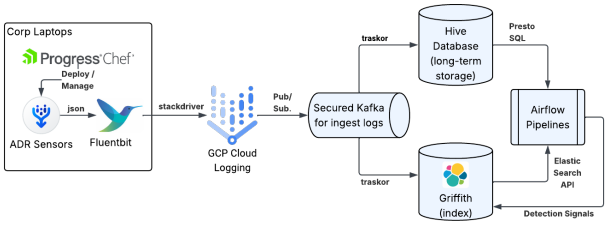


Figure 4. **ADR Sensor Architecture.** The sensor parses local caches of agentic tools (Cursor, Cline, Claude Code) to reconstruct complete agent sessions, capturing user prompts, agent reasoning, MCP tool calls, and environmental context. Telemetry is forwarded to backend systems for detection analysis.

Our approach. We address these challenges with three design principles (Figure 3): **(1) High-fidelity telemetry**: Collect detailed telemetry that captures not just what happened, but why it happened (the full causal chain from user prompt to agent reasoning to tool execution). **(2) Hierarchical analysis**: Balance cost and accuracy through a two-tier architecture: fast triage flags suspicious events (**Tier 1**), then deep contextual reasoning validates true threats (**Tier 2**). **(3) Systematic validation**: Automatically discover hard attack variants through offline red-teaming during pre-deployment testing, strengthening detection robustness (**Offline Explorer**).

We now explain each component below.

3.1 Observability: The ADR Sensor

The observability gap. Existing Endpoint Detection and Response (EDR) tools such as were designed before the emergence of MCP-based AI workflows. They provide strong system-level visibility (e.g., file, process, and network telemetry) but lack the semantic context needed to understand *why* actions occur. For example, when an agent writes a file, EDR sees the file write but not the user prompt that triggered it, the agent's reasoning about what to write, or which MCP tools were invoked. This gap makes it impossible to distinguish malicious agent behavior (e.g., exfiltrating credentials) from benign actions (e.g., saving a configuration file).

What we collect and why it matters. The **ADR Sensor** closes this gap by reconstructing the full causal chain of agentic activity. For each agent session, the sensor captures:

- **User prompts**: The original natural language instructions that initiated the agent's actions. This reveals *intent* (i.e., what the user asked the agent to do).
- **Agent reasoning**: The agent's intermediate thought process and decision-making steps. This shows *how* the agent interpreted the prompt and planned its actions.
- **MCP tool invocations**: The sequence of MCP tools called, with their arguments and execution results. This reveals *what* the agent actually did and in what order.
- **Environmental context**: MCP server configurations and installed packages (`pip`, `npm`). This provides *context* about what capabilities were available to the agent.

These four dimensions together enable semantic threat detection: the detector can reason about whether an agent's actions align with user intent, whether tool usage patterns match known attacks, and whether the agent violated security policies. For instance, if a user asks to "summarize this Jira ticket" but the agent reads SSH keys and makes an HTTP request, the sensor telemetry exposes this deviation from expected behavior.

How we collect it. The sensor operates as a lightweight endpoint agent that parses local data stores of agentic tools (**Cursor, Cline, Claude Code**) from their SQLite databases and JSONL caches. It correlates disparate log entries to reconstruct complete agent sessions, linking prompts $\rightarrow$ reasoning $\rightarrow$ tool calls $\rightarrow$ outcomes. The sensor runs on an hourly schedule with minimal overhead (each run takes 0.182 seconds on average), forwarding telemetry to backend systems for analysis. The overall architecture is shown in Figure 4.

An alternative approach is **an LLM/MCP gateway**, which intercepts agent-to-LLM-API and agent-to-MCP-tool traffic at the network boundary. While easier to implement, gateway-based solutions require changes to MCP hosts, are incompatible with streaming responses, and capture only partial information, omitting environmental and contextual data critical for high-fidelity observability. To enable effective gateway-first detection, MCP hosts would need to surface additional context/intent (e.g., the originating prompt and reasoning context) alongside tool calls (e.g., via a "Context/Intent" field in the MCP schema). Regarding prevention, we employ a hybrid model: the sensor enables deep forensics, while inline hooks (§6.2) provide real-time blocking for high-severity credential leakage.

3.2 Detection at Scale: Online-Offline Loop

Enterprise agentic traffic is heavily skewed toward benign activity, yet sophisticated attacks are designed to blend in. Analyzing every event with expensive LLM-based reasoning is impractical at scale; simple rule-based filtering alone misses sophisticated attacks. Our design combines a fast *triage* pass (high recall, low cost) with selective deep *reasoning* (high precision, higher cost) backed by enterprise context, strengthened through systematic offline red-teaming that discovers hard attack variants during pre-deployment validation.

Tier 1: Triage (initial screening). Every event is first

screened using a lightweight LLM-based triage prompt designed for high recall. The triage layer flags suspicious signals such as prompt-injection phrases, requests touching credentials or permissions, role/privilege changes, and risky combinations of otherwise benign steps. The design philosophy is conservative: when in doubt, escalate to Tier 2. Only clearly benign activity is short-circuited, ensuring minimal false negatives.

Tier 2: Reasoning with MCP context. Events flagged by Tier 1 undergo deep semantic analysis using a reasoning agent that dynamically queries enterprise context. The reasoning agent has access to three MCP providers supplying trustworthy enterprise information: (1) **Source code inspection** (`get_source_code`) reveals what an MCP tool actually does by examining its implementation; (2) **Threat intelligence lookup** (`get_threat_framework`) provides known attack tactics and techniques to ground the analysis; (3) **Policy verification** (`get_policies`, `assess_policy_violations`) checks whether actions violate organizational security rules. The reasoning agent adaptively chooses which providers to invoke based on the event characteristics (e.g., a file write may trigger all three; a simple read may only check policies). Tier 2 also consults curated threat intelligence from the repository (Appendix A.2), where each entry includes detection guidance tagged `[EAS]` (discovered by Explorer) or `[CURATED]` (manually added by security analysts).

Robustness. Risks to robustness include brittle prompts, model updates/drift, and adversarial adaptation (e.g., attackers attempting to influence triage/reasoning). We mitigate these risks with prompt/version pinning, change-control, and regression suites over ADR-Bench and AgentDojo before model/prompt updates.

Offline red-teaming (systematic attack discovery). To strengthen detection robustness across diverse attack types, the Offline Explorer systematically generates and tests attack variants through three collaborative agents: The **Red-Teaming Agent** proposes realistic attack variants by mutating parameters and combining techniques from a seed set of known attacks. The **Eval Agent** executes these candidates in sandboxed (isolated) environments, measuring both attack success and detection evasion. The **Threat Intelligence Agent** curates high-value discoveries and publishes them to the threat repository for use by Tier 2. Concretely, the Explorer runs an *evolutionary algorithm*: each attack candidate is scored using a fitness function $F = \varepsilon \times \sigma \times \tau^{\alpha}$ that combines execution depth ε (how far the attack progresses), semantic naturalness σ (how benign it appears), and impact τ (damage potential). Here, α controls how strongly we weight impact τ relative to execution depth ε and naturalness σ in F (larger α prioritizes higher-impact variants). In our EAS configuration, we use $\alpha = 1.2$. Top-scoring variants survive, generate new mutations, and are re-evaluated across rounds until convergence (guaranteed by the constraint $\rho \times \mu < 1.0$, where ρ is survival rate and μ is mutations per survivor). The emulation layer acts as the feedback oracle. Detector responses and execution logs drive the evolution toward realistic, high-impact, hard-to-detect attacks. Discovered attack patterns update Tier 2's detection logic through the threat intelligence MCP, strengthening the detector before production deployment.

Closing the loop. When the Threat Intelligence Agent updates the repository with newly discovered attacks, these entries immediately become available to Tier 2 through the threat intelligence MCP. The updated threat intelligence steers detection in two ways: (1) it guides which MCP providers to query (e.g., prioritizing source code inspection for tool-manipulation attacks), and (2) it informs how the reasoning agent weighs evidence when making final detection decisions. This validation process ensures ADR is tested against diverse attack patterns before deployment.

4 ADR-BENCH

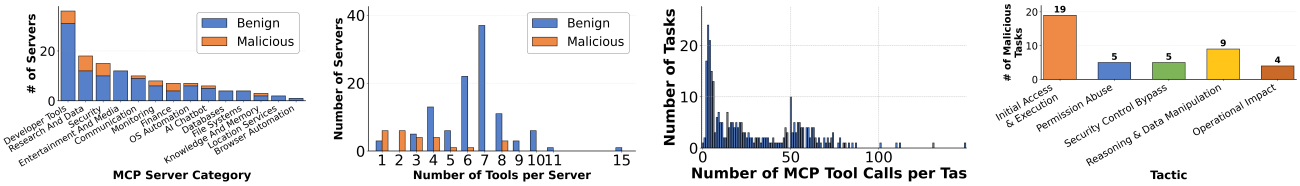
This section introduces ADR-Bench, derived from our enterprise experience. We first present the threat framework that grounds our evaluation (five tactics, 17 techniques) with evidence from public reports and enterprise telemetry. We then discuss limitations of existing benchmarks and contrast with prior work (Table 1) to motivate broader threat coverage and MCP context. Finally, we describe ADR-Bench, including its composition, coverage, and ease of use, using summary statistics and Figure 5a–5d. **Release.** We release ADR-Bench (task specifications, a runnable MCP registry with all MCP servers, and the evaluation pipeline) and the ADR Sensor and detection framework. Any enterprise-specific identifiers or sensitive content are removed or replaced with safe stand-ins, enabling external researchers to reproduce our reported benchmark results and extend the benchmark. We also release a minimal ADR configuration (JSON) aligned with our ablations, to support incremental adoption.

4.1 Threat Framework and Reported Instances

Agentic threats do not map cleanly to traditional vulnerability checklists. We consolidate evidence from three sources: (i) public frameworks and guidelines (e.g., MITRE ATLAS (The MITRE Corporation, 2025), OWASP (OWASP Foundation, 2025)), (ii) disclosed security incidents and research reports (e.g., JFrog (JFrog Security Research, 2025), Invariant Labs (Invariant Labs, 2025a;d;c;b), Microsoft (Microsoft Defender, 2025), Zenity (Agent Flayer) (Zenity Labs, 2025), CyberArk (CyberArk, 2025), Solo.io (Solo.io, 2025), Trend Micro (Trend Micro Research, 2025)), and (iii) operational telemetry from UBER's enterprise deployment

Table 1. Comparison of various benchmarks versus ADR-Bench.

Benchmark	Use MCP	# MCP Servers	# Tools	Coverage	
				# Tasks	# Threats
AgentSafetyBench (Zhang et al., 2024b)	No	–	1702	2000	4/17
ToolEmu (Ruan et al., 2023)			312	144	3/17
AgentDojo (Debenedetti et al., 2024)			70	97	4/17
AgentSecurityBench (Zhang et al., 2024a)			420	50	5/17
AgentHarm (Andriushchenko et al., 2024)			104	110	6/17
MCP-Artifact (Song et al., 2025)	Yes	11	19	9	3/17
RAS-Eval (Fu et al., 2025)		18	75	80	3/17
MCP-AttackBench (Xing et al., 2025)		–	–	–	5/17
ADR-Bench (Ours)	**Yes**	**133**	**729**	**302**	**17/17**



(a) MCP Server Distribution (b) Tools per Server (c) Tool Calls per Task (d) Malicious Tasks by Tactic

Figure 5. **Benchmark Composition and Characteristics.** (a) MCP servers span 14 categories with 81.2% benign and 18.8% malicious servers. (b) Benign servers provide more tools (median: 7) than malicious ones (median: 3). (c) Tasks invoke an average of 28.5 MCP tool calls, demonstrating realistic agentic workflows. (d) Malicious tasks cover 5 threat tactics with emphasis on Initial Access & Execution.

along with conversations with red team experts. This synthesis yields a **practical five-tactic, 17-technique threat framework** tailored to MCP-driven agentic systems (Appendix A.1), where *tactics* describe the adversary's goal (the "why") and *techniques* describe specific attack methods (the "how"). The five tactics are: **Initial Access & Execution** (6 techniques), **Permission Abuse** (2), **Security Control Bypass** (3), **Reasoning & Data Manipulation** (4), and **Operational Impact** (2). These techniques range from traditional attack vectors adapted for agentic systems (e.g., Indirect Prompt Injection, Tool Rug Pull) to novel agentic-specific threats (e.g., Control-Flow Hijacking, Malicious Agent Collusion). Each technique is anchored by concrete, enterprise-style behaviors observed in the wild or reproduced in emulation. We reference public security incidents throughout and detail how ADR-Bench reproduces them in §4.3.

4.2 Limitations of Existing Benchmarks

Most existing agent benchmarks cover only a small subset of agentic threats or lack MCP context. Table 1 shows that prior work covers only 3–6 of the 17 techniques, while ADR-Bench **covers all 17 techniques across all five tactics**. This comprehensive coverage with MCP context is essential for evaluating enterprise-ready detectors.

4.3 Our Benchmark: Composition and Ease of Use

To address these gaps, ADR-Bench **reproduces real-world attacks** from three sources: (i) attack scenarios adapted from existing benchmarks (e.g., MCP-Artifact (Song et al., 2025), RAS-Eval (Fu et al., 2025)); (ii) publicly reported security incidents and research disclosures; and (iii) internal threat intelligence from UBER's deployment. We faithfully recreate patterns including indirect prompt injection (Agent Flayer (Zenity Labs, 2025)), tool shadowing and MCP server manipulation (Microsoft Defender, 2025; CyberArk, 2025), code interpreter abuse, and supply chain compromises (JFrog Security Research, 2025).

ADR-Bench comprises 133 MCP servers (spanning 14 categories from file systems to cloud APIs) providing 729 distinct tools, and 302 realistic tasks averaging 28.5 tool calls per task (Figure 5). The benchmark is designed around three principles:

- **Emphasis on realism**: Tasks are derived from UBER's Security Operations Center (SOC) insights and enterprise deployment patterns, covering data analysis, code generation, system administration, and business process automation with authentic multi-step tool chains.
- **Full attack framework coverage**: Every tactic and technique in our threat framework is represented, moving beyond fragmented or narrow threat models found in prior work.
- **Modeling of enterprise conditions**: The benchmark mirrors challenging real-world characteristics, including highly unbalanced datasets with low attack prevalence (42 malicious out of 302 tasks, 13.9%) and the presence of sensitive information.

Task Sources and Enterprise Policy Store. We compile the 302 tasks from three sources to ensure breadth and realism. First, we adapted tasks from existing benchmarks (e.g., MCP-Artifact, RAS-Eval) to establish comparability. Second, we used GPT-4 to generate new benign tasks by prompting it with existing tasks and requesting innovative variants involving diverse business actions. Third, we include 42 malicious tasks derived from the 17 attack techniques (§4.1), capturing not only traditional attacks but also enterprise policy violations, where an agent's actions, while not inherently malicious, violate internal policies, providing a critical test for comprehensive security coverage.

To support such policy-centric evaluations, we compiled and generalized a set of enterprise security and compliance policies into a structured YAML store, accessible via a dedicated MCP endpoint. Each entry defines a canonical enterprise standard with explicit risk areas, affected roles, and enforcement conditions. The policy store enables consistent interpretation of policy violations across tasks, allowing for static evaluation (e.g., matching an MCP request to prohibited behaviors). By standardizing these enterprise norms, ADR-Bench bridges operational security controls with agentic reasoning contexts, making it suitable for evaluating detectors that must reason about compliance as well as threat behavior.

Ease of Use and Extensibility. ADR-Bench is intentionally designed for modularity and rapid experimentation. Researchers and practitioners can extend the benchmark by following straightforward templates:

- **Adding new MCP servers**: simply register additional servers in the YAML-based MCP registry, specifying entry points, authentication mechanisms, and tool definitions.
- **Adding new tasks**: define prompts and task configurations referencing available servers and tools; each task is a single JSON or YAML specification, making it easy to script, validate, or procedurally generate.
- **Adding new policies or threat techniques**: integrate additional enterprise standards or attack variants by appending them to the policy store or threat taxonomy without altering the core runtime.

All benchmark components share a uniform execution layer and deterministic emulation backend, enabling reproducibility across evaluations and straightforward CI/CD integration. This design allows ADR-Bench to evolve alongside the agentic security landscape, supporting future MCP protocols, toolchains, and policy domains with minimal engineering overhead.

5 EVALUATION

5.1 Experimental Setup

Benchmarks. We evaluate on two benchmarks:

- **AgentDojo** (Debenedetti et al., 2024) is a public benchmark for prompt injection detection containing 93 tasks (38 malicious, 55 benign) with attacks embedded in external data (tool outputs, web content, emails). We use AgentDojo as an auxiliary public reference focused on prompt injection. While AgentSafetyBench (Zhang et al., 2024b) has broader coverage, it is not MCP-native, so a faithful comparison would require substantial re-instrumentation of tasks/tools into an MCP setting.
- **ADR-Bench** (§4) is our enterprise threat benchmark containing 302 agentic tasks covering 17 attack techniques across 5 tactical categories. Tasks include 42 malicious scenarios (13.9%) and 260 benign operations (86.1%), reflecting realistic enterprise class imbalance.

Baselines. We compare ADR against three detectors:

- **LlamaFirewall** (Chennabasappa et al., 2025) is Meta's enterprise guardrail that uses Llama Guard 3-8B for prompt classification with heuristic rules. We use default threshold settings from the official release.
- **GuardAgent** (Xiang et al., 2024) is a policy-based system that uses GPT-4o to verify agent behavior against a knowledge graph of 47 security policies derived from OWASP guidelines and enterprise best practices.
- **ALRPHFS** (Xiang et al., 2025) uses hierarchical fast-and-slow reasoning: text-embedding-3-large matches agent behavior against 301 pre-trained attack patterns (fast path), and Claude Sonnet 4 analyzes ambiguous cases (slow path). We use the authors' recommended thresholds: similarity ≥ 0.8 for harmful, ≤ 0.5 for benign, and 0.5–0.8 for slow thinking.

For all three baselines, we adapt their open-source implementations to ensure fair comparison: LlamaFirewall from Meta's Purple Llama repository (LlamaFirewall, 2025), GuardAgent (GuardAgent, 2025) and ALRPHFS (ALRPHFS, 2025) from the authors' official releases.

Implementation. ADR uses GPT-4o for triage and Claude Sonnet 4 for reasoning, with three MCP servers providing enterprise context (§3). Importantly, ADR requires **no hyperparameter tuning** as the triage and reasoning prompts are fixed across all tasks and benchmarks. All detectors are evaluated under identical conditions on the same hardware.

Metrics. For detection accuracy, we measure Precision, Recall, F1-score, True Positive (TP) / False Positive (FP) counts, and False Positive Rate (FPR). For operational efficiency, we measure cost per task ($), mean latency (seconds), cost per true positive ($), and 95th percentile latency (seconds).

5.2 Overall Performance

We benchmark ADR against all baselines across AgentDojo and ADR-Bench.

ADR-Bench results. On our enterprise benchmark (Table 2), ADR achieves **perfect precision (1.000)** with **zero false positives**, detecting 28 of 42 attacks for 0.667 recall and 0.800 F1-score. In contrast, all baselines suffer from high false positive rates: ALRPHFS flags 34 benign tasks, GuardAgent flags 30, and LlamaFirewall flags 40. Their precision drops to 0.333, 0.231, and 0.167 respectively, making them unsuitable for production deployment where false alarms trigger expensive incident response.

While ADR achieves moderate recall (0.667), it substantially outperforms baselines in F1-score: 0.800 versus 0.366 (ALRPHFS), 0.222 (GuardAgent), and 0.178 (LlamaFirewall). The key insight is that ADR handles the severe class imbalance (13.9% attack rate) through its hierarchical design: the triage layer filters obvious benign cases at low cost, while the reasoning agent with MCP context providers achieves high precision on suspicious events.

Figure 6a shows varied detection rates across the 5 threat tactics: Initial Access & Execution (68%, 13/19 attacks), Permission Abuse (20%, 1/5), Security Control Bypass (100%, 5/5), Reasoning & Data Manipulation (56%, 5/9), and Operational Impact (100%, 4/4), demonstrating broad coverage with particularly strong performance on control bypass and operational impact.

AgentDojo results. On prompt injection attacks (Table 2), ADR demonstrates complementary strengths: **perfect recall (1.000)** detecting all 38 malicious tasks with 0.927 precision (only 3 false positives), yielding the best F1-score (0.962). ALRPHFS achieves comparable precision (0.914) but misses 6 attacks (0.842 recall). LlamaFirewall achieves high recall (0.974) but suffers from 21 false positives (0.638 precision). GuardAgent achieves balanced but lower performance across both metrics (0.771 precision, 0.711 recall). The complementary performance across benchmarks validates that ADR generalizes well across threat models: it prioritizes precision on enterprise tasks with severe class imbalance, and achieves perfect recall on prompt injection attacks where missing attacks is costly.

Efficiency. ADR maintains acceptable operational costs across both benchmarks. On ADR-Bench, it costs $0.024 per task with 18.5s average latency (Figure 6b), which is 2× more expensive than GuardAgent ($0.012, 9.4s) but 19× cheaper than LlamaFirewall ($0.459, 93.8s). This cost gap is largely due to ADR's enterprise-context analysis (MCP policy/source-code/threat-intel context and deep investigation). On AgentDojo, ADR costs $0.012 per task with 18.5s latency, comparable to GuardAgent and LlamaFirewall. The cost-recall trade-off (Figure 6c) demonstrates the efficiency of our hierarchical design: the triage layer handles 40.7% of tasks at minimal cost ($0.017/task, 2.3s latency), routing only suspicious events to the more expensive reasoning agent ($0.029/task, 29.7s latency).

5.3 Ablation Studies

To validate the contribution of each core component, we conduct ablation studies on ADR-Bench:

- **w/o Triage:** Remove the triage layer and send all events directly to the reasoning agent.
- **w/o Source Code MCP:** Remove source code inspection from the reasoning agent's MCP context providers.
- **w/o Threat Intel MCP:** Remove threat intelligence lookup from the reasoning agent's MCP context providers.
- **w/o Policy MCP:** Remove policy verification from the reasoning agent's MCP context providers.

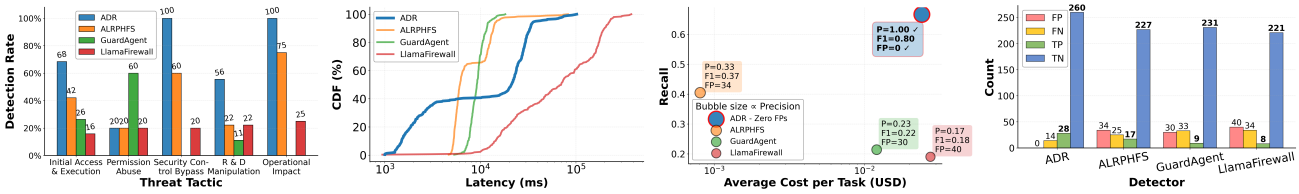
Impact of triage layer. Removing the triage layer increases recall from 0.667 to 0.805 and F1 from 0.800 to 0.868, but introduces 2 false positives (precision drops from 1.000 to 0.943) while increasing cost by 29% (from $0.024 to $0.031 per task) and latency by 52% (from 18.5s to 28.1s). The triage layer handles 40.7% of tasks at $0.017 per task and 2.3s latency, compared to $0.029 per task and 29.7s for the full reasoning path, validating our design: fast triage for obvious benign cases, expensive reasoning for suspicious events. **Operating point selection.** Enterprises have different requirements, so there is no single universal operating point. In ADR, the primary operating-point knob is the Tier 1 routing/escalation policy: auto-resolve only clearly benign sessions and escalate uncertain/suspicious sessions to Tier 2. This directly controls the escalation rate (and thus expected cost/latency) and shifts the precision/recall trade-off; the ablation above quantifies this trade-off (recall gain vs. introduced FPs and higher cost/latency) when routing more traffic to Tier 2. In deployment under extreme class imbalance, we use a precision-first alerting setting to keep false alarms operationally manageable.

Impact of MCP context providers. Removing source code inspection causes the largest performance drop: recall falls from 0.667 to 0.571 (14% relative decrease) and F1 falls from 0.800 to 0.676 (15.5% decrease), as many enterprise attacks involve malicious code execution or tool manipulation requiring source code analysis. Removing threat intelligence lookup also hurts performance (recall drops to 0.585, F1 to 0.716), as the reasoning agent relies on it to identify known attack patterns and tactics. Removing policy verification has minimal impact (recall 0.667→0.675, F1 0.800→0.806), suggesting most attacks in our benchmark violate obvious security properties detectable without explicit policy checks.

Table 2. Performance and Cost Comparison on Two Benchmarks

Detector	Performance					Cost & Efficiency			
	Prec.	Recall	F1	TP/FP	FPR	Cost/Task ($)	Latency (s)	Cost/TP ($)	P95 Lat. (s)
ADR-*Bench (302 tasks: 42 malicious, 260 benign)*									
ADR (Ours)	**1.000**	**0.667**	**0.800**	28/**0**	**0.000**	0.024	18.5	0.260	37.9
ALRPHFS	0.333	0.405	0.366	17/34	0.130	0.001	9.2	0.022	14.8
GuardAgent	0.231	0.214	0.222	9/30	0.115	0.012	9.4	0.407	12.5
LlamaFirewall	0.167	0.190	0.178	8/40	0.153	0.459	93.8	17.377	225.7
AgentDojo (93 tasks: 38 malicious, 55 benign)									
ADR (Ours)	**0.927**	**1.000**	**0.962**	**38**/3	**0.055**	0.012	18.5	0.029	47.3
ALRPHFS	0.914	0.842	0.877	32/3	**0.055**	**0.004**	**13.6**	**0.011**	**17.0**
GuardAgent	0.771	0.711	0.740	27/8	0.145	0.010	10.1	0.033	13.9
LlamaFirewall	0.638	0.974	0.771	37/21	0.382	0.012	11.4	0.030	23.0

Bold indicates best performance. FPR = False Positive Rate, TP = True Positives, P95 Lat. = 95th percentile latency.



(a) Detection Rate by Tactic (b) Latency CDF (c) Cost–Recall Trade-off (d) Confusion Matrix Comparison

Figure 6. **Overall Performance on ADR-Bench.** (a) Detection rates by threat tactic show ADR's strong coverage across attack types. (b) Latency CDF demonstrates consistently lower tail latency. (c) Cost–Recall trade-off illustrates flexible operating points via triage. (d) Confusion matrix metrics highlight ADR's balanced precision and recall.

6 Real-World Deployment

We deployed ADR across corporate MacBook endpoints (Intel and ARM) within Uber. Figure 8 shows the daily volume of sessions observed in 2025. This large-scale production deployment provided critical insights into ADR's operational performance, coverage, and limitations under real-world enterprise workloads.

Privacy and sensitive telemetry. The system manages sensitive telemetry through a multi-layered privacy strategy that complies with the company's data minimization and strict access controls. Where possible, secrets and sensitive tokens are redacted from stored telemetry via automated scanning. Telemetry is retained for 13 months for security purposes, with access limited exclusively to security personnel under a "least-privilege" model. Furthermore, all data is encrypted in transit via TLS and stored in secure, auditable environments governed by enterprise security logging standards. Finally, the system has completed rigorous review by legal, privacy, and risk management teams to ensure full regulatory compliance before being deployed.

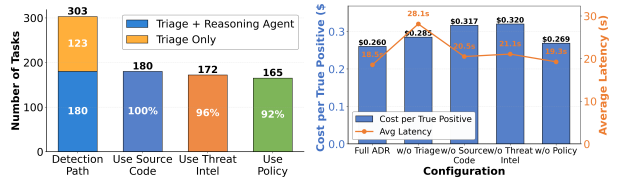
6.1 Detection Summary

Operational outcomes. In the deployment section, "alerts" are sessions routed by Tier 1/Tier 2 to a human review queue and labeled by analysts as true positives (TP, 34%), true positives non-malicious (TPNM, 17%), or false positives (FP, 49%). The most frequent TP cases involved *credential exposure*, which we further describe in §6.2. TPNM cases primarily originated from internal Offensive Security teams and penetration testing activities, indicating the need to model *user personas* and contextual intent. FPs were largely caused by context-rich sessions such as analyzing large, multi-file codebases.

Benchmark vs. deployment. ADR-Bench is designed to cover a broad set of enterprise-relevant attack tactics/techniques and benign workflows for systematic evaluation, and it is not intended to mirror the true incidence rates of different incident categories in a production environment. In production, the real-world distribution is highly skewed (secrets/credential/data exfiltration occurs relatively often, while other attack categories are exceedingly rare), and many items labeled "FP" are conservative escalations on complex but benign sessions (hard-benign) that are later resolved as benign by analyst triage, rather than being "false positives" in a clean per-task benchmark-label sense.

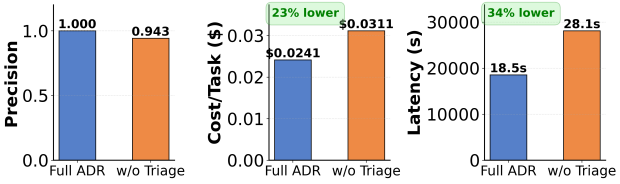
6.2 Detecting and Preventing Credential Exposure

Detection. ADR detected hundreds of high-severity credential exposures across 26 categories that had been in-

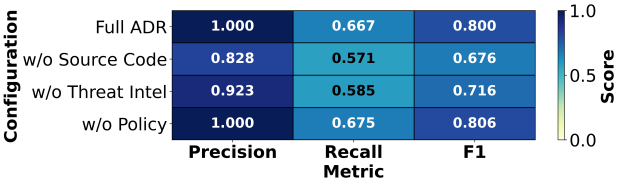


(a) Detection path distribution

(b) Cost-efficiency trade-offs



(c) Triage layer benefits



(d) MCP server necessity heatmap

Figure 7. Ablation study results. (a) 40.7% of tasks are handled by triage alone at minimal cost. (b) Cost and latency scale. (c) Removing triage introduces false positives and costs more. (d) Removing source code and threat intel MCP servers drops recall.

advertently shared outside the enterprise network, posing significant security risks. In response, we initiated credential rotation with the owning teams and deployed preventive controls through hooks integrated into Cursor and Claude development environments. While detection provided critical visibility, it also revealed two systemic challenges. *First*, the sheer volume of credential-related alerts quickly exceeded the capacity for manual review, requiring an automated response mechanism. *Second*, by the time detections were surfaced, credentials had often already been exposed externally, creating irreversible risk. These observations motivated a shift-left approach, moving from reactive detection to proactive prevention within the agent execution loop itself.

Prevention. Cursor and Claude Code released a feature in late 2025 called Hooks, which offers a modular mechanism to intercept and influence agent behavior within the execution loop. Hooks act as programmable extension points that enable external scripts to observe, modify, or block data as it traverses predefined processing stages. Each hook runs as a separate process, communicating with the agent via standard input/output using structured JSON messages. Hooks can be triggered before prompt construction, after response generation, or during reasoning steps, allowing flexible and portable integration across tools. Within a hook, developers can **observe** (monitor inputs, outputs, and agent state), **modify** (adjust prompts, responses, or context dynamically), or **block** (prevent disallowed or sensitive operations). We implemented a regex-based detection mechanism using both pattern matching and entropy thresholds to identify potential secrets within prompts. This prevention layer is executed as a pre-prompt hook, which scans and, if necessary, blocks prompts before transmission to Cursor or Claude agents. In practice, simple non-LLM checks work well for known attacks with static patterns (e.g., secret/credential strings), but fail for attacks requiring reasoning about tool semantics, causal context, and enterprise policies. Our evaluation shows that this approach achieved a precision of **97.2%**, correctly identifying **206 true positives** with only **6 false positives** across **212 unique credentials** from hundreds of thousands of MCP sessions.

6.3 Threat Emulation of Internal/External Incidents

Internal Capture-the-Flag (CTF) attack. ADR successfully detected simulated attacks from an internal CTF exercise. The attack proceeded in two stages: (1) the attacker integrated a custom shell tool into the assistant, and (2) issued a deceptive prompt instructing the agent to execute a malicious command (`curl | python3`) under the guise of sandbox testing. ADR detected prompt manipulation and subsequent remote code execution attempts by correlating LLM reasoning logs with MCP telemetry, generating an alert that correctly identified the misuse of MCP tools.

Agent Flayer attack (Zenity Labs, 2025). To evaluate ADR against realistic enterprise attack patterns, we emulated high-profile industry incidents such as the Agent Flayer campaign. As illustrated in Figure 9, this threat scenario simulates an indirect prompt injection chain: a malicious email automatically creates a Jira ticket embedding hidden instructions. When a user's Cursor IDE – connected via Jira's MCP integration – retrieves and processes the ticket, the injected prompt coerces the agent into reading local configuration files and exfiltrating credentials via HTTP. ADR accurately reconstructed and detected multiple stages of this emulated attack using fine-grained LLM and MCP telemetry. It flagged the prompt injection, subsequent credential access, and outbound data transfer events. The correlated detections demonstrated ADR's capability to trace complex multi-stage agentic threats and validate causal dependencies across tool boundaries under controlled testing conditions. This emulation confirmed that ADR can generalize beyond enterprise-specific patterns to detect emerging classes of real-world MCP exploitation techniques.

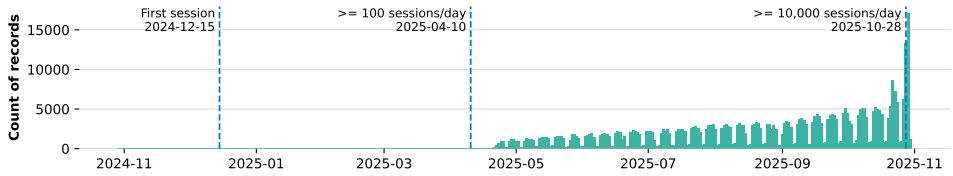


Figure 8. Daily volume of MCP host sessions across enterprise endpoints.

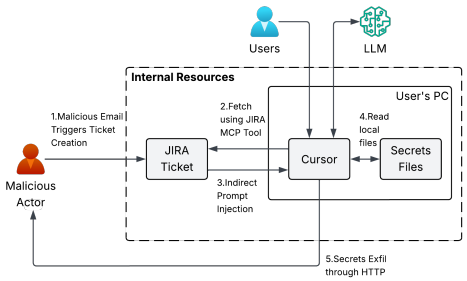


Figure 9. Attack graph of the Agent Flayer incident illustrating indirect prompt injection via Jira–Cursor integration.

7 Related Works

Benchmarks for Agent Security. Several benchmarks have been developed for agentic AI security including ToolEmu (Ruan et al., 2023), AgentDojo (Debenedetti et al., 2024), AgentHarm (Andriushchenko et al., 2024), AgentSafetyBench (Zhang et al., 2024b), and AgentSecurityBench (Zhang et al., 2024a). However, they primarily focus on prompt injection and basic tool misuse and do not support native MCP, making them less relevant for the enterprise architectures that are now standard. Recent benchmarks (Song et al., 2025; Fu et al., 2025; Xing et al., 2025) directly address this gap by specifically modeling MCP security. While this is a critical step forward, they are still limited in two key dimensions: they provide incomplete coverage of the full threat landscape (§4) and lack realism by focusing exclusively on attack scenarios, failing to account for enterprise environments where agentic activity is predominantly benign. These gaps in coverage and realism motivate ADR-Bench.

Defending LLM Agents and MCP Security. Defense systems for agentic security primarily fall into two categories: preventative mechanisms that block malicious actions in real-time, and detective mechanisms that identify malicious agentic events for post-processing. Preventative mechanisms have evolved from static, rule-based systems, such as regex and allowlisting, to more dynamic LLM-based guardrails. State-of-the-art approaches use LLM reasoning to enforce safety policies at runtime, including industry guardrails (Costa et al., 2025; Rebedea et al., 2023; Amazon, 2025; Google, 2025; Tencent Zhuque Lab, 2025) such as Anthropic's constitutional AI (Bai et al., 2022) and Meta's LlamaFirewall (Chennabasappa et al., 2025). In academia, GuardAgent (Xiang et al., 2024) and AGrail (Luo et al., 2025) generate adaptive safety checks and executable code to validate agent actions against security requirements. ShieldAgent (Chen et al., 2025) structures policy documents into verifiable rule circuits to shield protected agents. While powerful, these systems are fundamentally constrained by the policies they are given and struggle to defend against unknown or emergent threat patterns. Detective mechanisms aim to find unknown threats by moving beyond fixed rules. While traditional anomaly detection can be applied to agent logs, such methods often fail to interpret the semantic context of agentic workflows. More advanced approaches like ALRPHFS (Xiang et al., 2025) address this by using adversarial learning to extract "risk patterns" from adversarial interactions. However, all defenses rely on pre-defined rules or learned patterns and cannot proactively discover and adapt to emergent, zero-day threats in live agentic systems.

Automated Red Teaming. To discover zero-day threats, recent works use automated red teaming via techniques such as genetic algorithms (Liu et al., 2023; Lapid et al., 2024), gradient-based search (Zou et al., 2023; Chen et al., 2024), and agent-based frameworks (Xu et al., 2024; Wang et al., 2024; Jiang et al., 2024; Liu et al., 2024). However, they remain limited to optimizing prompt designs. Even the most advanced framework, AutoRedTeamer (Zhou et al., 2025), which introduces a modular attack toolbox and a strategic memory architecture, is still designed to find sophisticated jailbreaking methods for the LLM itself. This collective focus on foundational LLM security leaves a critical gap in understanding vulnerabilities for agentic systems. ADR addresses this gap by focusing on vulnerability discovery for agentic systems operating at enterprise scale.

8 Conclusion

We presented ADR, the first enterprise-scale framework for securing AI agents operating through the Model Context Protocol. ADR addresses three core challenges—limited observability, insufficient robustness, and high detection costs—via three integrated components: the ADR Sensor for high-fidelity agentic telemetry, a two-tier online Detector that balances cost and precision through hierarchical triage and context-aware reasoning, and an offline Explorer that systematically discovers hard attack variants through evolutionary red-teaming.

On ADR-Bench (302 tasks, 133 MCP servers, 17 attack techniques), ADR achieves zero false positives while detecting 67% of attacks, outperforming three state-of-the-art baselines by 2–4× in F1-score. On AgentDojo, ADR detects all prompt injection attacks with only three false alarms. Deployed at UBER for over ten months across 7,200+ hosts processing 10,000+ daily sessions, ADR uncovered hundreds of credential exposures and enabled a shift-left prevention layer with 97.2% precision.

We release ADR-Bench, the ADR Sensor, and the detection framework to support reproducibility and community adoption. Looking ahead, we see opportunities in extending ADR to multi-agent coordination protocols, adaptive real-time prevention at the MCP gateway layer, and tighter integration with evolving MCP standards to further close the gap between detection and response.

REFERENCES

ALRPHFS. Alrphfs codebase. https://github.com/ShiyuXiang77/ALRPHFS, 2025. Accessed: October 23, 2025.

Amazon. Amazon bedrock guardrails. https://aws.amazon.com/bedrock/guardrails/, 2025. Accessed: September 1, 2025.

Andriushchenko, M., Souly, A., Dziemian, M., Duenas, D., Lin, M., Wang, J., Hendrycks, D., Zou, A., Kolter, Z., Fredrikson, M., et al. Agentharm: A benchmark for measuring harmfulness of llm agents. *arXiv preprint arXiv:2410.09024*, 2024. Accepted at ICLR 2025.

Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Brett, I. Simplified and secure mcp gateways for enterprise ai integration, 2025. URL https://arxiv.org/abs/2504.19997.

Chen, X., Nie, Y., Guo, W., and Zhang, X. When llm meets drl: Advancing jailbreaking efficiency via drl-guided search. *Advances in Neural Information Processing Systems*, 37:26814–26845, 2024.

Chen, Z., Kang, M., and Li, B. Shieldagent: Shielding agents via verifiable safety policy reasoning. *arXiv preprint arXiv:2503.22738*, 2025.

Chennabasappa, S., Nikolaidis, C., Song, D., Molnar, D., Ding, S., Wan, S., Whitman, S., Deason, L., Doucette, N., Montilla, A., et al. Llamafirewall: An open source guardrail system for building secure ai agents. *arXiv preprint arXiv:2505.03574*, 2025.

Costa, M., Köpf, B., Kolluri, A., Paverd, A., Russinovich, M., Salem, A., Tople, S., Wutschitz, L., and Zanella-Béguelin, S. Securing ai agents with information-flow control. *arXiv preprint arXiv:2505.23643*, 2025.

CyberArk. Is your ai safe? threat analysis of mcp (model context protocol). https://www.cyberark.com/resources/threat-research-blog/is-your-ai-safe-threat-analysis-of-mcp-model-context-protocol, 2025. Accessed: August 1, 2025.

Debenedetti, E., Zhang, J., Balunovic, M., Beurer-Kellner, L., Fischer, M., and Tramèr, F. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. *Advances in Neural Information Processing Systems*, 37:82895–82920, 2024.

Fu, Y., Yuan, X., and Wang, D. Ras-eval: A comprehensive benchmark for security evaluation of llm agents in real-world environments. *arXiv preprint arXiv:2506.15253*, 2025.

Google. Guardrails api. https://developers.google.com/checks/guide/ai-safety/guardrails, 2025. Accessed: September 1, 2025.

GuardAgent. Guardagent codebase. https://github.com/guardagent/code, 2025. Accessed: October 23, 2025.

Guo, Y., Liu, P., Ma, W., Deng, Z., Zhu, X., Di, P., Xiao, X., and Wen, S. Systematic analysis of mcp security, 2025. URL https://arxiv.org/abs/2508.12538.

Hou, X., Zhao, Y., Wang, S., and Wang, H. Model context protocol (mcp): Landscape, security threats, and future research directions, 2025. URL https://arxiv.org/abs/2503.23278.

Hubinger, E., Denison, C., Mu, J., Lambert, M., Raghunathan, A., Hernandez, D., Lanham, T., Ndousse, K., Chen, A., Lukošiūtė, R., et al. Sleeper agents: Training deceptive llms that persist through safety training. *arXiv preprint arXiv:2401.05566*, 2024.

Invariant Labs. Github mcp exploited: Accessing private repositories via mcp. https://invariantlabs.ai/blog/mcp-github-vulnerability, 2025a. Accessed: August 1, 2025.

Invariant Labs. Mcp security notification: Tool poisoning attacks. https://invariantlabs.ai/blog/mcp-security-notification-tool-poisoning-attacks, 2025b. Accessed: August 1, 2025.

Invariant Labs. Invariant labs exposes novel prompt injection attack vulnerabilities, "toxic flows," in agentic

systems & mcp servers. https://invariantlabs.ai/blog/toxic-flow-analysis, 2025c. Accessed: August 1, 2025.

Invariant Labs. Whatsapp mcp exploited: Exfiltrating your message history via mcp. https://invariantlabs.ai/blog/whatsapp-mcp-exploited, 2025d. Accessed: August 1, 2025.

JFrog Security Research. Critical rce vulnerability in mcp-remote: Cve-2025-6514 threatens llm clients. https://jfrog.com/blog/2025-6514-critical-mcp-remote-rce-vulnerability/, 2025. Accessed: August 1, 2025.

Jiang, L., Rao, K., Han, S., Ettinger, A., Brahman, F., Kumar, S., Mireshghallah, N., Lu, X., Sap, M., Choi, Y., et al. Wildteaming at scale: From in-the-wild jailbreaks to (adversarially) safer language models. *Advances in Neural Information Processing Systems*, 37:47094–47165, 2024.

Kumar, S., Girdhar, A., Patil, R., and Tripathi, D. Mcp guardian: A security-first layer for safeguarding mcp-based ai system, 2025. URL https://arxiv.org/abs/2504.12757.

Lapid, R., Langberg, R., and Sipper, M. Open sesame! universal black-box jailbreaking of large language models. *Applied Sciences*, 14(16):7150, 2024.

Liu, X., Li, P., Suh, E., Vorobeychik, Y., Mao, Z., Jha, S., McDaniel, P., Sun, H., Li, B., and Xiao, C. Autodan-turbo: A lifelong agent for strategy self-exploration to jailbreak llms. *arXiv preprint arXiv:2410.05295*, 2024.

Liu, Y., Deng, G., Xu, Z., Li, Y., Zheng, Y., Zhang, Y., Zhao, L., Zhang, T., Wang, K., and Liu, Y. Jailbreaking chatgpt via prompt engineering: An empirical study. *arXiv preprint arXiv:2305.13860*, 2023.

LlamaFirewall. Llamafirewall codebase. https://github.com/meta-llama/PurpleLlama/tree/main/LlamaFirewall, 2025. Accessed: October 23, 2025.

Luo, W., Dai, S., Liu, X., Banerjee, S., Sun, H., Chen, M., and Xiao, C. Agrail: A lifelong agent guardrail with effective and adaptive safety detection. *arXiv preprint arXiv:2502.11448*, 2025.

MCP.so. Mcp.so marketplace. https://mcp.so/, 2025. Accessed: October 27, 2025.

Microsoft Defender. Plug, play, and prey: The security risks of the model context protocol. https://techcommunity.microsoft.com/blog/microsoftdefendercloudblog/plug-play-and-prey-the-security-risks-of-the-model-context-protocol/4410829, 2025. Accessed: August 1, 2025.

Model Context Protocol. Model context protocol repositories. https://github.com/orgs/modelcontextprotocol/repositories, 2025. Accessed: October 27, 2025.

OWASP Foundation. Owasp llm risk 06:2025 – excessive agency. https://genai.owasp.org/llmrisk/llm062025-excessive-agency/, 2025. Accessed: August 1, 2025.

Radosevich, B. and Halloran, J. Mcp safety audit: Llms with the model context protocol allow major security exploits, 2025. URL https://arxiv.org/abs/2504.03767.

Rebedea, T., Dinu, R., Sreedhar, M. N., Parisien, C., and Cohen, J. NeMo guardrails: A toolkit for controllable and safe LLM applications with programmable rails. In Feng, Y. and Lefever, E. (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 431–445, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-demo.40. URL https://aclanthology.org/2023.emnlp-demo.40.

Ruan, Y., Dong, H., Wang, A., Pitis, S., Zhou, Y., Ba, J., Dubois, Y., Maddison, C. J., and Hashimoto, T. Identifying the risks of lm agents with an lm-emulated sandbox. *arXiv preprint arXiv:2309.15817*, 2023.

Solo.io. Deep dive mcp and a2a attack vectors for ai agents. https://www.solo.io/blog/deep-dive-mcp-and-a2a-attack-vectors-for-ai-agents, 2025. Accessed: August 1, 2025.

Song, H., Shen, Y., Luo, W., Guo, L., Chen, T., Wang, J., Li, B., Zhang, X., and Chen, J. Beyond the protocol: Unveiling attack vectors in the model context protocol ecosystem. *arXiv preprint arXiv:2506.02040*, 2025.

Tencent Zhuque Lab. Ai-infra-guard. https://github.com/Tencent/AI-Infra-Guard, 2025. Accessed: September 1, 2025.

The MITRE Corporation. MITRE ATLAS™: Adversarial Threat Landscape for Artificial-Intelligence Systems. https://atlas.mitre.org/matrices/ATLAS, 2025. Accessed: July 30, 2025.

Trend Micro Research. Why a classic mcp server vulnerability can undermine your entire ai agent. https://www.trendmicro.com/en_us/research/25/f/why-a-classic-mcp-server-vulnerability-can-undermine-your-entire-ai-agent.html, 2025. Accessed: August 1, 2025.

Wang, H., Zhang, A., Duy Tai, N., Sun, J., Chua, T.-S., et al. Ali-agent: Assessing llms' alignment with human values via agent-based evaluation. *Advances in Neural Information Processing Systems*, 37:99040–99088, 2024.

Xiang, S., Zhang, T., and Chen, R. Alrphfs: Adversarially learned risk patterns with hierarchical fast\& slow reasoning for robust agent defense. *arXiv preprint arXiv:2505.19260*, 2025.

Xiang, Z., Zheng, L., Li, Y., Hong, J., Li, Q., Xie, H., Zhang, J., Xiong, Z., Xie, C., Yang, C., et al. Guardagent: Safeguard llm agents by a guard agent via knowledge-enabled reasoning. *arXiv preprint arXiv:2406.09187*, 2024.

Xing, W., Qi, Z., Qin, Y., Li, Y., Chang, C., Yu, J., Lin, C., Xie, Z., and Han, M. Mcp-guard: A defense framework for model context protocol integrity in large language model applications. *arXiv preprint arXiv:2508.10991*, 2025.

Xu, H., Zhang, W., Wang, Z., Xiao, F., Zheng, R., Feng, Y., Ba, Z., and Ren, K. Redagent: Red teaming large language models with context-aware autonomous language agent. *arXiv preprint arXiv:2407.16667*, 2024.

Zenity Labs. Agentflayer: When a jira ticket can steal your secrets. https://labs.zenity.io/p/when-a-jira-ticket-can-steal-your-secrets, 2025. Accessed: August 1, 2025.

Zhang, H., Huang, J., Mei, K., Yao, Y., Wang, Z., Zhan, C., Wang, H., and Zhang, Y. Agent security bench (asb): Formalizing and benchmarking attacks and defenses in llm-based agents. *arXiv preprint arXiv:2410.02644*, 2024a.

Zhang, Z., Cui, S., Lu, Y., Zhou, J., Yang, J., Wang, H., and Huang, M. Agent-safetybench: Evaluating the safety of llm agents. *arXiv preprint arXiv:2412.14470*, 2024b.

Zhou, A., Wu, K., Pinto, F., Chen, Z., Zeng, Y., Yang, Y., Yang, S., Koyejo, S., Zou, J., and Li, B. Autoredteamer: Autonomous red teaming with lifelong attack integration. *arXiv preprint arXiv:2503.15754*, 2025.

Zou, A., Wang, Z., Carlini, N., Nasr, M., Kolter, J. Z., and Fredrikson, M. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

A Appendix

A.1 Enterprise Agentic Threat Taxonomy

Table 3 presents our complete five-tactic, 17-technique threat taxonomy with references to public security incidents.

A.2 Threat Repository Schema (YAML Excerpt)

```
threat_framework:
  tactics:
    initial_compromise:
      techniques:
      - id: ADR.T0001
        name: Insecure Supply Chain for
    Agentic Components
        detection_guidance:
        - 'Malicious: Monitor for
    expanded permissions in
    config_manager accessing financial
    data sources...'  # EAS
        - 'Malicious: Monitor for
    unauthorized access to repository
    secrets and API endpoint creation
    using github_server...'  # EAS
      - id: ADR.T0002
        name: Indirect Prompt Injection
        detection_guidance:
        - 'Malicious: Monitor legal
    research tools for unusual document
    scanning and contract analysis
    patterns...'  # CURATED
```

Each technique record contains three fields: `id`, `name`, and `detection_guidance`. Guidance lines are tagged `[EAS]` (discovered by the offline Explorer) or `[CURATED]` (added by analysts). Tier 2 uses these lines to select which MCP providers to query and to weigh evidence during detection decisions (§3.2).

Table 3. Enterprise Agentic Threat Taxonomy: 5 Core Tactics

Tactic	Description	Techniques & References
Initial Access & Execution	Adversaries gain initial control and execute malicious actions within the agentic environment by exploiting vulnerabilities, injecting malicious instructions, or manipulating tool behavior	**6 techniques:** Insecure Supply Chain (JFrog Security Research, 2025), Indirect Prompt Injection (Zenity Labs, 2025), Control-Flow Hijacking (Invariant Labs, 2025c), Code Interpreter Abuse (CyberArk, 2025), Insecure Output Handling, Tool Rug Pull (Invariant Labs, 2025b)
Permission Abuse	Adversaries exploit or exceed authorized permissions to access sensitive data and resources beyond their legitimate scope	**2 techniques:** Exploitation of Excessive Tool Permissions (Invariant Labs, 2025a;d), Agent Identity Spoofing
Security Control Bypass	Adversaries evade detection and circumvent security mechanisms by deploying malicious tools, manipulating tool resolution, or coordinating multiple agents	**3 techniques:** Tool Shadowing (Microsoft Defender, 2025; CyberArk, 2025), Tool Hallucination Manipulation, Malicious Agent Collusion (Solo.io, 2025)
Reasoning & Data Manipulation	Adversaries corrupt data sources or manipulate an agent's reasoning processes to compromise decision integrity and degrade long-term reliability	**4 techniques:** Unvetted MCP Server Connection, Semantic Data Poisoning, Long-Term Goal Hijacking (Hubinger et al., 2024), Temporal Data Attack
Operational Impact	Adversaries disrupt availability and degrade business operations by exhausting system resources or overwhelming the agent's inference capacity	**2 techniques:** Agent-Facilitated Resource Exhaustion (OWASP Foundation, 2025), Model-Layer Denial of Service