A Survey on Agentic Security: Applications, Threats and Defenses

Asif Shahriar[1]*, Md Nafiu Rahman[1]*, Sadif Ahmed[1]*, Farig Sadeque[1], Md Rizwan Parvez[2]
[1]BRAC University [2]Qatar Computing Research Institute (QCRI)

Abstract

LLM-based agents are now used throughout cybersecurity. While these agents facilitate powerful and autonomous security applications, their autonomy opens up new attack surfaces, and the security community is actively building defenses to secure them. Yet the literature on this subject has grown quickly and unevenly. Existing surveys treat applications, threats, and defenses in isolation, leaving no unified account of how an agent's capabilities, vulnerabilities, and countermeasures interconnect. In this work we present the first holistic survey of the agentic security landscape, structuring the field around the fundamental pillars of Applications, Threats and Defenses. We provide a comprehensive taxonomy of over 260 papers, explaining how agents are used in downstream cybersecurity applications, inherent threats to agentic systems, and countermeasures designed to protect them. In addition, we provide detailed pillar-specific and cross-cutting analyses that show the security-lifecycle coverage of agentic applications, comparison between red-teaming and blue-teaming agents, and the adversarial use of red-teaming applications. On the threat side, we analyze the entry points and agent-loop stages that attacks target, their specificity to the agentic setting, and the threat models they assume. On the defense side, we analyze the prevailing defense strategies, their cost and security trade-offs, and where in the agent lifecycle they are deployed. We further map which defenses cover which attack classes and chart trends in agent architecture, backbone model usage, data modality coverage, and the growth of attack and defense research over time. Taken together, these findings indicate that agentic systems are structurally fragile by default and that securing them will require defenses that span the full agent lifecycle rather than single-layer fixes. A complete and continuously updated list of all surveyed papers is publicly available at `https://github.com/kagnlp/Awesome-Agentic-Security`.

1 Introduction

Definition: LLM Agent

We define an **LLM Agent** as a system whose core decision module is an LLM that *plans*, *invokes tools/APIs*, and *acts* in an external environment while *observing* feedback and *adapting* subsequent actions. It maintains *state* (short/long-term memory or a knowledge store) and may include explicit *self-critique/verification* and *governance* layers to satisfy task goals and safety constraints.

Since their introduction, Large Language Models (LLMs) have been used extensively in the domain of cybersecurity (Xu et al., 2025a; Wang et al., 2024b; Deng et al., 2024b). The transition of the research landscape from passive LLMs to autonomous LLM agents (Yao et al., 2023; Shinn et al., 2023; Schick et al., 2023) has made these models significantly more capable, allowing them not just to describe a solution but to execute it. This newfound agency has enabled LLM-agents to demonstrate remarkable capabilities across the full security spectrum, from reconnaissance and exploit generation to detection, forensics, and automated repair (Shen et al., 2025; Zhu et al., 2025c; Lin et al., 2025b). However, the same autonomy that drives

*Equal contribution. Author order does not matter.

these capabilities also enlarges the attack surface. A number of studies have shown that the very act of wrapping an LLM in an agentic framework significantly increases its vulnerability (Saha et al., 2025; Kumar et al., 2025; Chiang et al., 2025), as the safety alignment of the base model does not reliably transfer once the model is given tools, memory, and autonomy. In response, a growing body of research has focused on developing countermeasures to harden these systems through isolation, runtime monitoring, and formal verification (Debenedetti et al., 2025; Udeshi et al., 2025).

The rapid development of agentic security research (over 260 papers between 2024 and 2026) has created a fragmented landscape that lacks comprehensive analysis. While existing surveys provide valuable insights into specific aspects like security threats (Deng et al., 2024d), trustworthiness (Yu et al., 2025), enterprise governance (Raza et al., 2025) and core LLM safety (Ma et al., 2025), they fail to capture the complete picture, as shown in Table 1. This fragmentation leaves practitioners and researchers without a unified framework for understanding how agent capabilities, vulnerabilities, and defenses interconnect. For example, without a study that surveys both attacks on agentic systems and defensive countermeasures, it is hard to understand which threats are actually covered by existing defenses and which remain unmitigated. Similarly, without examining offensive applications and threats together, it is hard to evaluate whether a red-teaming agent built to find bugs can be used as an attack vector to jailbreak or poison agents.

In this work we present the first holistic survey of the agentic security landscape, structured to answer three key questions a security researcher would ask: *"What can agents do for my security?"* (Applications), *"How can they be attacked?"* (Threats), and *"How do I stop them?"* (Defenses). To this end, we define three pillars of taxonomy:
Applications (§2). Using LLM-agents in downstream cybersecurity tasks, including red teaming (autonomous vulnerability discovery), blue teaming (defending against threats), and domain-specific security (cloud, web).
Threats (§3). Security vulnerabilities inherent to agentic systems that attackers can exploit.
Defenses (§4). Techniques and countermeasures used to harden agentic systems against the threats.

Beyond cataloguing these pillars, we analyze each one and then present a cross-cutting analysis. For applications, we map every system onto the offensive and defensive lifecycles, study their differences, identify research clusters and examine the dual-use nature of red-teaming agents. For threats, we analyze the channels through which attacks enter and the agent-loop stages where they manifest, the specificity of each threat to the agentic setting, the threat models attackers assume, and the benchmarks used to measure them. For defenses, we analyze the defense strategies along scalability, robustness, overhead, and coverage, and examine where in the agent lifecycle protection is placed. Finally, we read across the full corpus along several axes, including the coverage of each attack class by existing defenses, agent architecture and cardinality, backbone model usage, data modality coverage, and the temporal distribution of attack and defense research.

Findings. Our pillar-wise and cross-cutting analyses reveal several notable findings.

- **Applications.** Most security applications of agentic systems cluster in the stages that give immediate, verifiable feedback (vulnerability discovery and exploitation on offense, detection and remediation on defense). Offensive agents are more autonomous than defensive agents, as the defensive ones are often bound to human approval. Many red-teaming agentic applications expose capabilities that can be used adversarially, and their offensive power comes from the agentic scaffold rather than the base model, which often refuses the same task when prompted directly.
- **Threats.** Roughly two-thirds of attacks are inherited or amplified from the base LLM, confirming that base-model safety alignment does not transfer to the agentic setting. Most attacks target the perception and action-selection stages of the agentic loop, while the reflection stage is the least attacked. Most attacks succeed black-box, indicating that agentic systems are structurally fragile by default. Adversarial benchmarks are fragmented: some attack surfaces (e.g., user query, tool response) are heavily benchmarked, while others (inter-agent communication in multi-agent systems and reflection output) are largely understudied, and there is a lack of universally comparable metrics beyond attack success rate.
- **Defenses.** No single defense strategy is robust across all important axes such as scalability, adversarial robustness, overhead, and coverage, which pushes the field toward hybrid, multi-layer architectures.

Table 1: Survey comparison. Legend: ✓ = covered; △ = partial/limited; ✗ = not covered.

Survey	Applications	Threats	Defenses	Benchmarks	Focus
Yu et al. (2025)	✗	✓	✓	✓	Trustworthiness, robustness, privacy
Raza et al. (2025)	✗	△	△	✗	Enterprise governance & risk
Deng et al. (2024d)	✗	✓	△	✗	Security threats
He et al. (2024)	✗	✓	✓	✗	Technical vulnerabilities
Ma et al. (2025)	△	✓	✓	△	Large-model safety (agents as subset)
Wang et al. (2025b)	✗	✓	✗	△	Safety risks during model development pipeline
de Witt (2025)	✗	△	△	✗	Theoretical basis for multi-agent risks
This Survey	✓	✓	✓	✓	**Holistic agentic security**

Stronger protection consistently costs latency and tokens, tracing a clear security-cost Pareto frontier. Defense placement is shifting toward an assume-breach posture, with post-computation monitoring and lifecycle-wide protection now dominant.

- **Cross-cutting insights.** Most defensive attempts are focused on injection and agent manipulation attacks, while prompt infection and pre-execution attacks remain the least covered. The field is migrating from monolithic agents to multi-agent and planner-executor designs, GPT remains a near-universal backbone, and coverage is skewed toward text and code over images, network traces and binaries.

Contributions. Our main contributions are as follows.

- **Holistic three-pillar survey.** We conduct an in-depth survey of agentic security through a comprehensive taxonomy of over 260 papers, organized around three interconnected pillars of application, threats and defenses, as presented in Fig. 1.
- **Focus on applications.** We provide a detailed review of how agents are actually used by security teams, covering offensive, defensive, and domain-specific tasks, an area largely overlooked by prior surveys.
- **Per-pillar analysis.** We study each pillar in depth, analyzing lifecycle coverage, autonomy, and dual-use in applications; attack entry points, failure stages, agent-specificity, threat models, and evaluation in threats; and defense strategies, cost trade-offs, and lifecycle placement in defenses.
- **Cross-cutting analysis.** We read across all three pillars to map which defenses cover which attack classes and to identify field-level trends and gaps, including the migration from monolithic to planner-executor and hybrid agents, the GPT backbone monopoly, uneven modality and threat coverage, and benchmark fragmentation.

2 Applications of Agents in Security

This section describes how agents are applied across the cybersecurity landscape, from offensive testing and exploit generation to defensive detection, forensics, and automated remediation.

2.1 Offensive Security Agents (Red-Teaming)

This subsection describes autonomous and reasoning-driven red-team agentic systems that conduct penetration testing, vulnerability discovery, fuzzing, and exploit adaptation.

2.1.1 Autonomous Penetration Testing

Recent autonomous agents can already carry out end to end penetration testing with adaptive planning and feedback. Deng et al. (2024b) propose PentestGPT, which uses reasoning, generation and parsing modules and keeps a Pentesting Task Tree to avoid context loss, and it beats baseline models on HackTheBox and

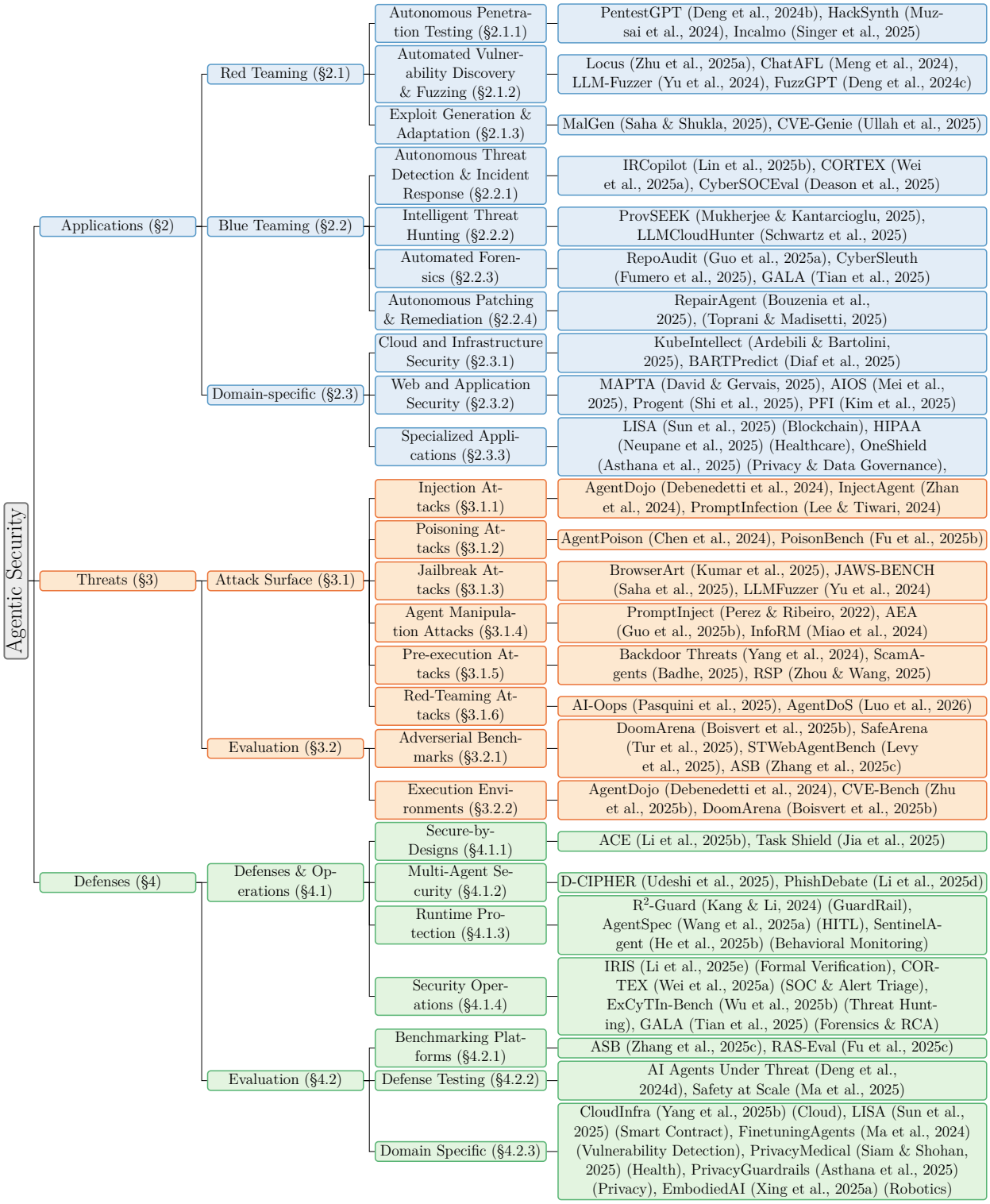


Figure 1: Overview of Agentic Security Taxonomy

VulnHub. Shen et al. (2025) present PentestAgent, a multi agent system for reconnaissance, search, planning and execution that uses retrieval augmented generation with a vector database to preserve state across the kill chain. Kong et al. (2025b) introduce VulnBot, which coordinates specialist agents with a Penetration

Task Graph and reaches high completion rates on AutoPenBench. Focusing on SSH access, Nieponice et al. (2025) develop ARACNE, which splits strategic planning from command interpretation and succeeds on ShelLM and OverTheWire Bandit. Happe & Cito (2025a) show that *cochise* can compromise Microsoft Active Directory accounts, which suggests that reasoning tuned models can sustain long attacks in messy enterprise networks. RedTeamLLM (Challita & Parrend, 2025) uses a summarize act loop to repair plans and handle context limits in entry level CTFs, while Co RedTeam (He et al., 2026) copies real red teaming workflows with coordinated discovery and exploitation stages plus long term memory to reach over 60% exploitation success across models. Mayoral-Vilches et al. (2025) present CAI, an open source framework organized by autonomy levels that performs well in live CTFs and bug bounty programs.

Other systems target specific stages of the penetration testing lifecycle. RapidPen (Nakatani, 2026) focuses on initial access with an IP to shell pipeline that uses ReAct style planning and a RAG exploit database to gain shells quickly and cheaply. AutoPentester (Ginige et al., 2025) improves task execution on custom virtual machines with a strategy analyzer, RAG command generation and automated result checks. PenHeal (Huang & Zhu, 2024) covers post exploitation and defense by pairing prompting based testing with a remediation tool that ranks fixes by cost and effectiveness. For privilege escalation, Probst et al. (2026) introduce a Linux benchmark, and show that guided GPT 4 Turbo agents can reach near human level success. Probst et al. (2026) then study how system changes and prompts can help small open weight agents match large cloud models on Linux privilege escalation.

Recent studies also compare these agents across tasks to explain why they work. HackSynth (Muzsai et al., 2024) evaluates a dual agent planner and summarizer on many CTF challenges and shows that success depends strongly on temperature and token limits. AutoPentest (Henke, 2025) uses Nmap and NIST NVD lookups to finish sub tasks at moderate cost efficiency. Huang et al. (2025) compare singular and modular designs and show that better context retention, coordination and multi step planning greatly improve modular agents. Deng et al. (2026a) analyze twenty eight LLM systems, separate capability failures from planning and state errors and introduce Excalibur, which adds difficulty aware planning. Happe & Cito (2025b) argue that pretrained foundation models already have enough pattern matching skill for autonomous hacking without RAG. Singer et al. (2025) present Incalmo, an attack layer that lets agents use declarative tasks instead of low level shell commands to steal assets across emulated networks. Luong et al. (2025) fine tune a large open weight model on chain of thought data inside a multi agent system and report nearly 80% sub task completion, which beats AutoPenBench (Gioacchini et al., 2024) and AI Pentest Benchmark (Isozaki et al., 2024) baselines.

Researchers have also built datasets, environments and benchmarks for stronger evaluation. Cybench (Zhang et al., 2025a) collects forty professional CTF tasks and splits them into subtasks for finer measurement. BountyBench (Zhang et al., 2026a) tests detect, exploit and patch tasks on twenty five real codebases tied to OWASP Top 10 bug bounty cases. EnIGMA (Abramovich et al., 2025) gives agents tools such as interactive debuggers and server connection utilities and reaches state of the art on Cybench. CTFAgent Zou et al. (2026) uses plan and execute control with a stateful task tree and special tools, and it works well in both fully automated and human in the loop settings. For training, CTF Dojo (Zhuo et al., 2025) provides large container based challenge environments with clear feedback, while Cyber Zero (Zhuo et al., 2026b) creates training data from public CTF writeups without runtime execution, and both improve benchmark results.

2.1.2 Vulnerability Discovery & Fuzzing

Zhu et al. (2025a) propose Locus, a synthesizer and validator agent that uses program analysis tools and symbolic execution to generate progress capturing predicates, speed up directed fuzzers, and uncover several unpatched bugs. Meng et al. (2024) build ChatAFL on top of AFLNet, query a foundation model for machine readable protocol grammars, enrich seed message sequences and break coverage plateaus, which raises state transition coverage and finds new bugs in mature protocol stacks. Ji et al. (2026) introduce FirmAgent, which combines hybrid fuzzing and static taint analysis to help LLM agents find IoT firmware vulnerabilities without special hardware. Wu et al. (2026) develop ChainFuzzer, a greybox framework that finds and reproduces workflow level vulnerabilities and unsafe dataflows in multi tool LLM agents. LLMFuzzer Yu et al. (2024), TitanFuzz Deng et al. (2023) and FuzzGPT Deng et al. (2024c) extend input generation with fuzzing or reasoning, where LLMFuzzer mutates seed jailbreak templates to expose transferable safety

failures, TitanFuzz (Deng et al., 2023) pairs generative and infilling models to improve code coverage and find dozens of deep learning bugs, and FuzzGPT (Deng et al., 2024c) uses in context learning on past bug triggering snippets to find more edge case defects.

Moving from discovery to exploitation, Fang et al. (2024) show that a simple generative agent can autonomously exploit 87% of recent real world CVEs, which highlights the gap between exploiting known flaws and finding new ones. Zhu et al. (2025d) extend this to multi agent zero day discovery with HPTSA, a hierarchical planner that creates sub agents for specific vulnerability classes and outperforms single agent baselines. Wang & Zhou (2025) present a two phase Android system for vulnerability discovery and validation that reaches strong coverage on Ghera and self validates 104 zero day flaws in production apps with auto generated proof of concept exploits. For network and threat analysis, Lin et al. (2026) describe ARTEMIS, a multi agent penetration testing framework with dynamic prompt generation and automatic triage that outperformed most human cybersecurity professionals in a live enterprise network study, and Xuan et al. (2026) show that TTPDetect can use a context explorer and reasoning rules to recognize adversary tactics, techniques and procedures in stripped malware binaries.

To evaluate these abilities, Lee et al. (2025b); Zhu et al. (2025c); Wang et al. (2025c) introduce benchmarks for exploitation and repair, with SEC bench building reproducible CVE artifacts at low cost per instance and showing that top code agents still do poorly on proof of concept generation and patching. Wang et al. (2026c) and Lau et al. (2026) expand this with large scale tests, where CyberGym asks agents to reproduce real memory safety bugs from crash traces and ZeroDayBench tests patching novel zero day vulnerabilities moved across repositories. At the repository level, Shen et al. (2026) evaluate secure code completion with dynamic testing, Ahmed et al. (2025) benchmark vulnerability localization on C and C++ CVEs, Yildiz et al. (2025) study just in time vulnerability detection by linking functions to commits that add or fix flaws, and Peng et al. (2025) propose an outcome driven framework for both functionality and security of LLM generated code.

2.1.3 Exploit Generation & Adaptation

Lupinacci et al. (2025) show that agents can be pushed to run malware through prompt injection, RAG backdoors and inter agent trust abuse, and they find that RAG backdoor attacks work on most tested commercial models while peer agents are treated as fully trusted. Saha & Shukla (2025) propose MalGEN, a multi stage multi agent pipeline that produces diverse malware samples aligned with MITRE ATT&CK in a controlled setting to study evasion tactics and stress test defenders. He et al. (2025a) describe an Agent in the Middle attack that intercepts and changes inter agent messages to inject malicious logic into multi agent frameworks, which shows that communication channels are a major attack surface. Ahmed (2025) introduce AttackLLM to automate malicious payload generation and measure the offensive power of large language models in adversarial settings.

Ullah et al. (2025) introduce CVE Genie, a role based multi agent framework that rebuilds environments and generates verifiable exploits, and it reproduces 428 CVEs across many languages and weakness classes at low average cost. Xiao et al. (2026a) present ReX, a prompt to pwn system that turns natural language intent into working exploits with interactive feedback loops and dynamic execution checks. Chen et al. (2026) advance exploit generation with a constraint guided multi agent system that uses specialized sub agents to solve hard path constraints and build reliable exploit chains.

Extending exploit generation to Web3, Gervais & Zhou (2026) show how agents can find and exploit smart contract logic flaws by analyzing on chain state inconsistencies. Fakih et al. (2025) present LLM4CVE, an iterative repair system that combines efficiently fine tuned models with proof of concept validation and reaches high human rated quality and strong similarity to ground truth patches on major open weight models.

While individual red-teaming systems differ in their implementation details, most surveyed offensive agents follow a common architectural pattern. They begin with reconnaissance and information gathering, construct an attack plan using a planner component, invoke specialized agents for vulnerability discovery, fuzzing, or exploit generation, and finally execute and validate attacks against the target environment. Many modern systems additionally maintain persistent state through memory, retrieval-augmented generation (RAG), exploit databases, or task graphs that enable long-horizon reasoning and iterative refinement across multiple attack stages.

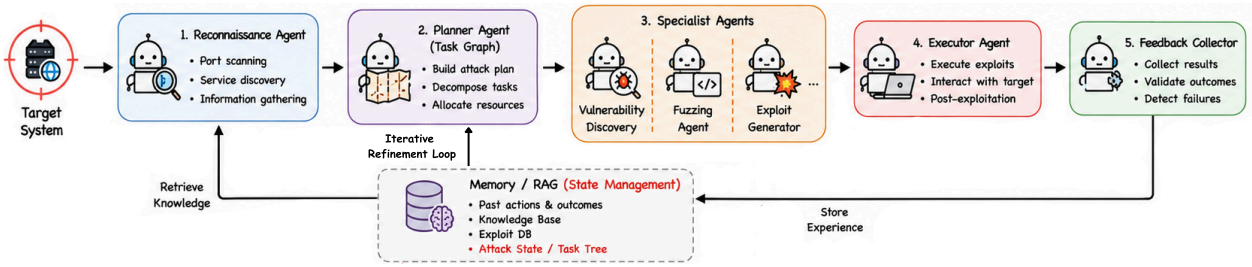


Figure 2: Architecture of a red-teaming agent pipeline. A reconnaissance agent gathers information and enumerates the attack surface, while a planner constructs an attack graph and decomposes objectives into tasks. Specialized agents perform vulnerability discovery, fuzzing, and exploit generation. An executor carries out actions, and a feedback collector validates outcomes.

As shown in Fig. 2, the dominant trend in offensive agent design is the transition from monolithic agents toward planner–executor and multi-agent architectures. Systems such as PentestAgent, VulnBot, ARACNE, HPTSA, and CAI increasingly separate reconnaissance, planning, vulnerability analysis, and execution into specialized components coordinated through shared memory and task graphs.

2.2 Defensive Security Agents (Blue-Teaming)

This subsection describes blue-team applications of automated agents for continuous monitoring, threat detection, incident response, threat hunting, and automated patching.

2.2.1 Autonomous Threat Detection & Incident Response

Agentic SOC frameworks monitor alerts, analyze threats and run response playbooks. Turcotte et al. (2025) propose an AI based SOC framework that uses density based clustering and isolation forests to prioritize mixed SIEM alerts and suppress repeated noise. Tellache et al. (2025) present a RAG based agent that combines natural language search over a threat intelligence vector store with structured VirusTotal lookups to enrich SIEM alerts and generate mitigation plans for Advanced Persistent Threats, and they validate it with AI and expert review on many enterprise alerts. Blefari et al. (2026) develop a RAG tool for intelligence teams that automates analysis of high risk communication patterns and produces clear preliminary reports. Wei et al. (2025a) introduce CORTEX, a divide and conquer triage system with separate agents for behavior analysis, evidence collection and reasoning that creates an auditable verdict and cuts false alerts more than single agent baselines. Li et al. (2024b) present an LLM agent for explainable intrusion detection in IoT networks that uses a reason then act pipeline with specialized tools for data extraction and classification.

For incident remediation, Lin et al. (2025b) propose IRCopilot, a structured role based design that mirrors real response teams, reduces context loss and hallucination, and outperforms baseline models across several benchmarks. Gao et al. (2026) present a lightweight agent system that combines perception, reasoning, planning and action, processes raw system logs and refines attack guesses through continuous in context adaptation. Liu (2025) compare centralized, decentralized and hybrid team models in tabletop response games and find that hybrid and argumentative setups improve coverage of multi stage attack chains. Singh et al. (2025) provide a real world study of active SOC operations and find that language models mostly serve as assistive co pilots rather than fully autonomous deciders.

To measure these abilities, Deason et al. (2025) introduce CyberSOCEval for threat reasoning on malware analysis and threat intelligence tasks, show gaps between commercial and open source models, and find that test time scaling helps cyber defense less than coding or math. Chakraborty et al. (2026) design a broader benchmark that tests whether AI agents can interpret cyber threat intelligence and turn narrative reports into validated detection rules across cloud and endpoint environments.

2.2.2 Intelligent Threat Hunting

Tseng et al. (2024) and Schwartz et al. (2025) show how LLMs can turn threat reports into SIEM and cloud detection rules, where the first extracts Indicators of Compromise to generate regular expressions with low false positives and the second pulls cloud indicators and attack techniques from text and images to create rule candidates with 92% precision and 98% recall, most of which compile into deployable query logic.

Mukherjee & Kantarcioglu (2025) introduce ProvSEEK, which combines a vectorized threat knowledge base with system provenance graphs and uses RAG, chain of thought reasoning and behavioral model filtration to improve detection precision and recall on multiple DARPA datasets with little extra token or latency cost as databases grow. Hans et al. (2025) use large language models to map low level intrusion logs to MITRE ATT&CK techniques and infer attacker cognitive biases such as risk tolerance and loss aversion.

Despite these advancements, applying intelligent agents to real-world operations presents significant challenges. Meng et al. (2025a) study failures in assisted threat intelligence workflows and find contradictions, scope drift and generalization gaps, then suggest structured prompting and verification fixes. Chona et al. (2026) introduce an open ended reinforcement learning benchmark that asks agents to find malicious event timestamps from raw Windows logs without hints, and they show that current frontier models fail badly at unsupervised threat hunting.

2.2.3 Automated Forensics & Root Cause Analysis

For repository auditing, Guo et al. (2025a) introduce RepoAudit, which combines memory with path sensitive demand driven traversal and a validator for data flow facts and path condition satisfiability to find dozens of true bugs in real world projects, including new bugs in high profile codebases, at very low cost. Alharthi & Yasaei (2025) and Fumero et al. (2025) build automated tools for cloud forensics and log analysis, and CyberSleuth uses a memory augmented multi agent design to process raw packet traces and application logs, identify the exact vulnerability in most incidents, and produce reports that industry experts rate as complete, useful and coherent.

Tian et al. (2025) present GALA, a multi modal root cause analysis workflow that combines statistical causal inference with iterative reasoning, trace weighted impact scoring, service dependency subgraph extraction and a re ranking agent to improve root cause accuracy over prior microservice baselines.

Pan et al. (2025) introduce the MAST taxonomy of multi agent failure modes and an automated judging pipeline for execution failure detection across orchestration frameworks, while Alharthi & Garcia (2025) present CIAF, an ontology driven framework that structures cloud logs and builds incident narratives from mixed evidence streams.

2.2.4 Autonomous Vulnerability Remediation

Researchers have proposed several agents for automated patch synthesis and vulnerability repair.Bouzenia et al. (2025) introduce RepairAgent, an autonomous bug repair pipeline that manages multiple repair tools with a dynamically updated prompt and finite state machine middleware, and it fixed 164 defects at low cost with reasonable token use per bug. Toprani & Madisetti (2025) present an infrastructure as code agent that auto generates deployment ready and policy compliant templates to fix issues found in continuous integration pipelines.

To measure these systems more rigorously, Wang et al. (2025e) introduce an exploit driven evaluation pipeline that requires original proofs of concept to fail against the patch, which exposes major weaknesses in models that looked good under shallow metrics. Building on this idea, Wei et al. (2025c) create a large scale multilingual benchmark of real world vulnerabilities with containerized sandbox environments and show that even top tier repair agents have low success rates when they must pass both security and functionality tests.

Relying on functional testing alone can also create new risks in the development cycle. Chen et al. (2025b) introduce a red teaming framework that generates adversarial issue statements to trick program repair agents into producing malicious patches that pass unit tests while quietly adding serious security flaws.

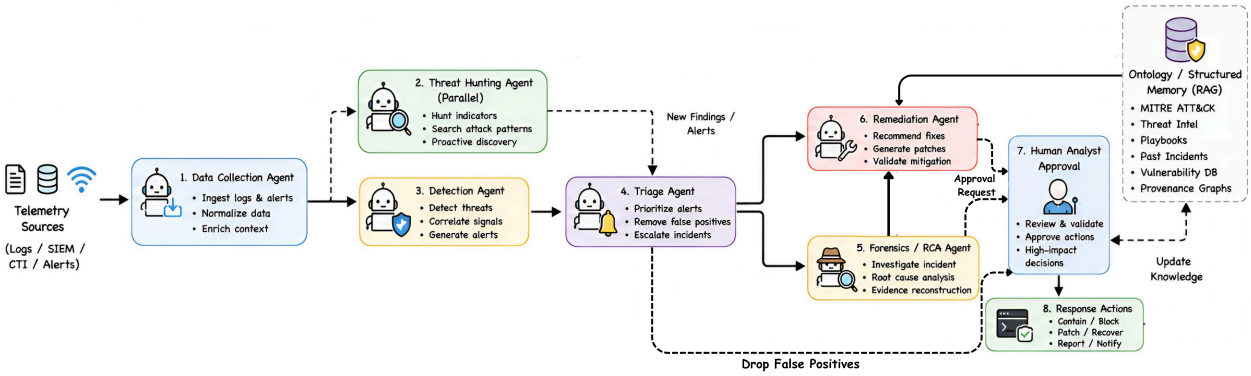


Figure 3: Architecture of a blue-teaming agent pipeline. A data collection agent processes security telemetry, alerts, and threat intelligence; detection and threat-hunting agents identify malicious activity; while a triage agent prioritizes incidents and filters false positives. Forensic analysis and remediation modules investigate causes and recommend mitigation or containment, with human oversight for high-impact decisions.

Although defensive agentic systems target diverse operational environments, their architectures typically follow a structured security-operations workflow. These systems ingest telemetry and alerts, perform automated detection and threat hunting, prioritize incidents through triage, investigate root causes, recommend remediation actions, and frequently incorporate human approval before executing high-impact responses. Long-term knowledge stores and threat-intelligence repositories play a central role in maintaining context across investigations.

Fig. 3 illustrates the common structure emerging across modern SOC-oriented agentic systems. Unlike offensive agents, defensive architectures emphasize telemetry processing, evidence aggregation, analyst collaboration, and approval-gated actions. This design reflects the operational requirement to minimize false positives and maintain human oversight over potentially disruptive response activities.

2.3 Domain-specific Applications

The section explains domain-specific agentic systems using language models for auditing, vulnerability detection, and policy-based hardening across sectors.

2.3.1 Cloud and Infrastructure Security

Agentic systems help secure cloud environments and infrastructure through automated scanning, hardening and remediation. To safely test Cloud Security Posture Management remediations, Yang et al. (2025b) propose a two-phase workflow that first explores a sandboxed replica of the target environment and then commits verified steps to production to reduce the impact of automated changes. KubeIntellect, introduced by Ardebili & Bartolini (2025), routes natural-language queries to sub-agents for logs, metrics and role-based access auditing, and can dynamically create new tools with a sandboxed code generator agent, achieving high tool-synthesis success and perfect reliability across hundreds of queries.

For container deployments, Ye et al. (2025b) present LLMSecConfig, which combines static analysis with retrieved best practices to automatically fix software container and orchestration misconfigurations while preserving the original operational intent.

Beyond traditional cloud environments, intelligent models also monitor and secure connected edge devices. Diaf et al. (2025) propose BARTPredict, a transformer-based forecaster that predicts Internet of Things traffic patterns and flags abnormal deviations as possible intrusions. A security analysis by Zhan et al. (2026) shows that large language model agent deployments on edge IoT hardware introduce attack surfaces such as transient failover blind spots and coordination-state divergence that are usually absent from centralized cloud setups.

2.3.2 Web and Application Security

David & Gervais (2025) propose MAPTA, a multi-agent web testing framework that combines reconnaissance, exploitation and verifier agents with sandboxed validation to ensure reported exploits are reproducible and safe for repeated testing. PTFusion (Wang et al., 2026a), improves context awareness in web penetration testing by fusing knowledge from multiple vulnerability signals. Similarly, Liu et al. (2025a) evaluate how effectively LLM agents reproduce web vulnerabilities from natural-language reports and highlight the gap between theoretical reasoning and practical exploit generation.

Regarding application interface risks, Mudryi et al. (2025) analyze browser-agent threats including prompt injection through page content, credential leakage via tool calls and clickjacking against agent interfaces, and propose layered defenses based on input sanitization, isolation and formal flow analysis. At the database level, Pedro et al. (2025) expose Prompt-to-SQL injections in LLM-integrated web applications where unsanitized user prompts become malicious database queries, and propose defenses integrated directly into the LangChain middleware.

At operating system level, Mei et al. (2025) design AIOS, which isolates intelligence kernels from user-space tools and manages resource access through a policy-aware scheduler. Building on this idea, Pirch et al. (2026) map established operating-system defenses to AI agent components to address isolation and privilege separation issues. Applying these concepts, Suwansathit et al. (2026) analyze vulnerabilities in the OpenClaw framework and show how decentralized trust boundaries enable complex exploitation chains across gateways and execution policies. To audit such architectures, Zhang et al. (2026b) propose a security analysis system that detects design flaws and runtime vulnerabilities in deployed LLM agent applications.

To secure execution environments, PFI (Kim et al., 2025) validates control-flow and data-flow boundaries within reasoning chains to prevent privilege escalation from untrusted content. Finally, Progent (Shi et al., 2025) is a runtime agent that enforces deterministic, programmable and fine-grained permissions and completely blocks successful attacks during red-team evaluations of tool-using frameworks.

2.3.3 Specialized Applications

This section reviews agentic security across finance, reverse engineering, operational technology, healthcare and enterprise privacy. In finance, Wei et al. (2025b) present LLM-SmartAudit. This framework uses a multi-agent conversational architecture with a buffer-of-thought mechanism to iteratively refine assessments and detect many smart contract vulnerabilities. Kevin & Yugopuspito (2025) introduce SmartLLM. This custom generative pipeline improves smart-contract vulnerability detection using chain-of-thought prompting tailored to common code weaknesses. Furthermore, hybrid and conversational systems Ma et al. (2024); Xia et al. (2024) improve explainability and exploit reproduction. They combine fine-tuned auditing models with interactive sessions to provide natural-language justifications alongside detected vulnerabilities.

Advancing binary analysis, Chen et al. (2025a) present ReCopilot. This expert language model is trained on domain-specific data and paired with static program analysis. It helps security analysts perform reverse engineering tasks on stripped binaries like decompilation and variable recovery. MalParse (Walton et al., 2024) an analysis tool that focuses on code semantics, uses a hierarchical code summarization pipeline and strategic prompt engineering to accurately categorize Android applications and extract actionable insights into malicious behavior.

To secure critical infrastructure, Sahu et al. (2026) propose a deception architecture for operational technology environments. It uses reinforcement learning to dynamically control network routing. It also uses language models as protocol-aware honeypots to generate realistic DNP3 outstation responses. In the highly regulated healthcare sector, Neupane et al. (2025) propose a compliance-focused multi-agent framework based on context protocols. It enforces field-level sanitization, role-based access and immutable audit trails to structurally embed compliance.

Finally, to address data protection and enterprise privacy, Asthana et al. (2025) develop OneShield, a multilingual privacy-guardrail system. It detects sensitive information across text streams and flags open-source dependency risks. This system is validated through comparative deployment studies across multiple real-world language corpora.

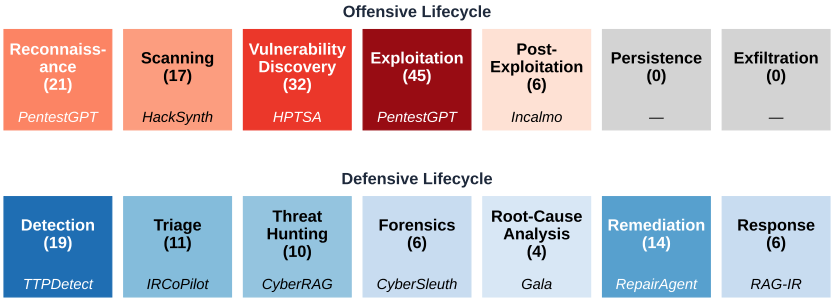


Figure 4: Security-Lifecycle Coverage Map of Agentic Security Research. Offensive work is concentrated in vulnerability discovery and exploitation, while defensive research focuses on detection and remediation.

2.4 Analysis of the Application Landscape

We analyze the application corpus along three axes: the stages of the security lifecycle addressed by existing systems, the contrast between red-teaming (offensive) and blue-teaming (defensive) agents and the prevalence of dual-use capabilities. Together, these dimensions provide both a structural overview of the current landscape and a deeper understanding of how agentic security applications shape operational capabilities, risks, and deployment contexts.

2.4.1 Security-Lifecycle Coverage Map

We first investigate how existing applications are distributed across the offensive kill chain and defensive security lifecycle. This analysis provides a structural map of research coverage, highlighting areas of concentration and identifying stages that remain comparatively underexplored.

To systematically characterize the current state of agentic security research, we map the application papers onto two parallel execution chains: an offensive lifecycle spanning reconnaissance through exfiltration, and a defensive lifecycle spanning initial detection through remediation and response. Fig. 4 visualizes this mapping by summarizing the paper counts and representative papers associated with each stage. We reviewed the methodology of every paper to determine which lifecycle stages each agent supports. A single agent was assigned to multiple stages whenever its architecture covered a broader operational scope. The resulting distribution reveals a strong concentration of research in a small number of stages and a complete absence of coverage in others.

On the offensive side, the literature clusters around tasks that reward code generation, stateless reasoning or immediate environmental feedback. Research is strongest in the early intelligence-gathering stages. We identified 21 papers focused on reconnaissance and 17 focused on scanning. Systems such as PentestGPT (Deng et al., 2024b), VulnBot (Kong et al., 2025b), AutoPentester (Ginige et al., 2025), and HackSynth (Muzsai et al., 2024) perform well in these stages because they can parse unstructured environmental data, orchestrate standard tools such as Nmap and systematically enumerate digital attack surfaces. Reconnaissance and scanning are among the most heavily covered stages in the offensive pipeline. Coverage increases further in vulnerability discovery. We identified 32 papers that support this stage. Systems such as HPTSA (Zhu et al., 2025d), MalGen (Saha & Shukla, 2025), and EnIGMA (Abramovich et al., 2025) use the coding and analysis capabilities of language models to inspect codebases, identify weaknesses and generate candidate attack paths. Exploitation is the most heavily represented stage in the entire dataset, with 45 papers. Representative systems include PentestGPT (Deng et al., 2024b), HPTSA (Zhu et al., 2025d), AttackLLM (Ahmed, 2025), MalGen (Saha & Shukla, 2025), and EnIGMA (Abramovich et al., 2025). These systems generate payloads, adapt attack strategies and iteratively validate proof-of-concept exploits in controlled environments. Coverage drops sharply after exploitation. Post-exploitation is represented by only 6 papers. Systems such as DarkAgent (Lupinacci et al., 2025), Incalmo (Singer et al., 2025), and Probst et al. (2026) explore tasks such as privilege escalation and lateral movement. However, such work remains uncommon

in the broader literature. Persistence currently has no dedicated agentic frameworks in our dataset, and no existing system focuses on maintaining long-term post-compromise access. Exfiltration is likewise absent; we found no autonomous frameworks for stealthy data extraction from compromised environments. This gap likely reflects both ethical concerns around autonomous malware development and the difficulty current models face in sustaining stealth and operational awareness over extended periods.

On the defensive side, the blue-teaming literature shows a similar concentration in stages that depend heavily on data processing and text analysis. Detection is the most common defensive stage, with 19 papers. Representative systems include TTPDetect (Xuan et al., 2026) and LLMCloudHunter (Schwartz et al., 2025). These systems process large volumes of telemetry and threat intelligence to identify potentially malicious activity. Alert triage is represented by 11 papers. Systems such as IRCoPilot (Lin et al., 2025b), CORTEX (Wei et al., 2025a), and AACT (Turcotte et al., 2025) prioritize incidents, reduce false positives and help analysts focus on the most important alerts. Threat hunting receives moderate attention with 10 papers. Representative systems include CyberRAG, CTI-REALM (Chakraborty et al., 2026), and ProvSeek (Mukherjee & Kantarcioglu, 2025). These systems support proactive investigations by combining retrieval, reasoning and threat intelligence analysis. Digital forensics is represented by 6 papers. Systems such as CyberSleuth (Fumero et al., 2025) and ProvSeek (Mukherjee & Kantarcioglu, 2025) attempt to reconstruct attack timelines and analyze evidence collected from system logs and provenance records. Root-cause analysis remains one of the least explored defensive stages, with only 4 papers. Systems such as Gala (Tian et al., 2025), CyberSleuth (Fumero et al., 2025), and LLMs in the SOC (Singh et al., 2025) attempt to trace incidents back to their underlying causes. This remains difficult because it requires long chains of reasoning across multiple sources of evidence. Remediation has comparatively strong coverage with 14 papers. Representative systems include RepairAgent (Bouzenia et al., 2025), PenHeal (Huang & Zhu, 2024), and LLM4CVE (Fakih et al., 2025). These systems build on automated program repair techniques to generate patches and mitigation strategies for discovered vulnerabilities. Response remains relatively sparse with only 6 papers. Systems such as IRCoPilot (Lin et al., 2025b) support operational decision making during incidents. Fully autonomous response remains uncommon because actions such as quarantining hosts or disabling network connectivity can have significant operational consequences.

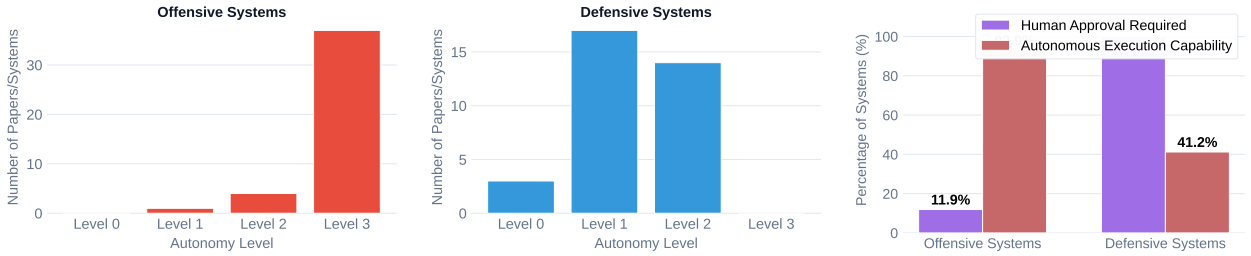
This lifecycle analysis shows that agentic security research remains concentrated in stages where success can be validated through immediate feedback, such as vulnerability discovery, exploitation, detection and remediation. On the other hand, stages that requires sustained reasoning, stealth or long-term operational awareness receive little attention. The gaps which are formed as a result highlight important directions for future research and underscore how unevenly current capabilities are distributed across the security lifecycle.

2.4.2 Red-Teaming vs Blue Teaming (SOC Agents)

Offensive and defensive LLM-based cybersecurity agents are developing along different lines. In red-teaming, the main goal is to support long attack chains, autonomous testing, and multi-stage exploitation. In blue-teaming and SOC settings, the focus is on handling large-scale telemetry, supporting analysts, and keeping humans in control. This difference shapes memory design, tool use, failure patterns, and evaluation practice.

Memory designs. Offensive agents mainly use context retention focused designs. Because attack workflows often span many steps, these systems try to reduce context loss across reconnaissance, planning, exploitation, and follow-up actions. A common example is the Reasoning, Generation, Parsing design used in PentestGPT(Deng et al., 2024b). Another common pattern is task graph coordination, which helps preserve state across attack phases. Defensive agents, in contrast, rely more on retrieval and structure focused memory. They need to manage large volumes of logs, alerts, and evidence, so retrieval augmented generation and ontology driven memory are more useful. CIAF (Alharthi & Garcia, 2025) uses ontology based cloud log structuring, while ProvSEEK (Mukherjee & Kantarcioglu, 2025) uses RAG for evidence refinement and verification.

Tool governance and autonomy. Offensive agents show a strong trend toward high autonomy. Recent systems increasingly aim for fully autonomous execution in penetration testing, fuzzing, and exploit generation. AutoPentest (Henke, 2025) and PentestGPT (Deng et al., 2024b) are examples of this direction. These agents often work independently inside sandboxed environments. Defensive agents follow a different design philosophy. In operational SOCs, LLMs are usually used as analyst copilots rather than fully autonomous



(a) Offensive system autonomy. (b) Defensive system autonomy. (c) Human oversight and execution.

Figure 5: Autonomy level distribution and execution capabilities across offensive and defensive agentic security systems. Offensive systems are heavily concentrated at Level 3 autonomy and largely operate without human approval, whereas defensive systems remain clustered at Levels 1 and 2, constrained by human-in-the-loop controls and limited execution authority.

systems. IRCopilot (Lin et al., 2025b) and CORTEX (Wei et al., 2025a) are representative examples, since they emphasize collaboration, alert triage, and approval gated decision making.

Failure modes. The main failure modes of offensive agents are planning errors and context loss. These systems can struggle with multi-step reasoning, long-horizon coordination, and maintaining state over extended workflows. They are also exposed to prompt injection and jailbreak attacks. In the worst case, an agent can be pushed into unsafe autonomous malware execution. Defensive agents face a different set of problems. False positives are a major issue because they can create alert fatigue, and CORTEX directly targets this problem. Other failure modes include contradictions in CTI pipelines and hallucinated findings in auditing agents such as RepoAudit (Guo et al., 2025a).

Offense-Defense autonomy asymmetry. Beyond architectural and operational differences, the literature reveals a significant asymmetry in autonomy between offensive and defensive agentic systems. A recurring concern is that offensive agents are advancing toward fully autonomous operation more rapidly than defensive agents.

To examine this trend, we classify all surveyed systems using a four-level autonomy framework. Level 0 systems provide recommendations without meaningful autonomous action. Level 1 systems function primarily as copilots that assist human operators. Level 2 systems can independently execute portions of a workflow but require approval at critical decision points. Level 3 systems autonomously plan, reason, and execute end-to-end tasks with minimal or no human intervention.

Figs. 5a and 5b summarize the autonomy distribution across surveyed systems, while Fig. 5c compares human oversight requirements and execution capabilities.

The autonomy distribution reveals a clear imbalance. Among offensive systems, 37 of 42 systems (88.1%) operate at Level 3 autonomy. Four systems operate at Level 2 and only one system appears at Level 1. No offensive systems fall into Level 0. Representative examples include PTFusion (Wang et al., 2026a), HPTSA (Zhu et al., 2025d), VulnBot (Kong et al., 2025b), and Incalmo (Singer et al., 2025), all of which autonomously plan and execute multi-stage attack workflows.

Defensive systems exhibit the opposite pattern. None of the 34 surveyed defensive systems reach Level 3 autonomy. Instead, 17 systems (50%) operate at Level 1, 14 systems operate at Level 2, and 3 systems remain at Level 0. Systems such as IRCopilot (Lin et al., 2025b), CyberRAG (Blefari et al., 2026), and LLM-Assisted CTI (Meng et al., 2025a) primarily function as analyst copilots. CORTEX (Wei et al., 2025a) and CyberSleuth (Fumero et al., 2025) provide limited workflow automation under human supervision.

Human oversight requirements further highlight this divide. As shown in Fig. 5c, only 11.9% of offensive systems require human approval during execution. In contrast, 100% of defensive systems maintain human-in-the-loop controls. Defensive agents almost universally include approval checkpoints, clarification prompts, or read-only operating modes.

Execution authority follows a similar pattern. Among offensive systems, 92.9% can autonomously execute commands or modify target environments. In contrast, only 41.2% of defensive systems possess execution capabilities. Even when execution is permitted, actions are generally restricted to low-risk tasks such as alert handling or predefined remediation workflows.

This divergence reflects fundamentally different design objectives. Offensive agents such as PTFusion (Wang et al., 2026a), HPTSA (Zhu et al., 2025d), VulnBot (Kong et al., 2025b), and Incalmo (Singer et al., 2025) are optimized for end-to-end task completion and therefore prioritize autonomous planning and execution. Defensive systems prioritize explainability, safety, accountability, and operational risk management. As a result, developers intentionally restrict autonomy and preserve human oversight.

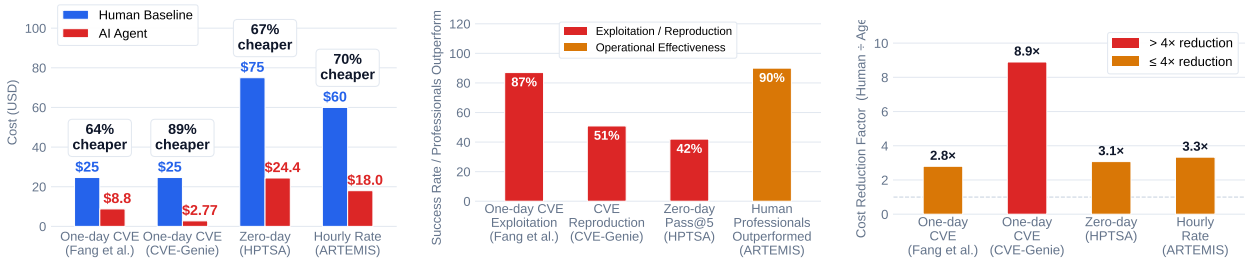
The autonomy gap between offensive and defensive agents reflects not only differences in technical capability but also differences in acceptable risk. While offensive research increasingly rewards end-to-end automation, defensive deployments remain constrained by safety, accountability, and operational concerns. Bridging this gap will be critical for the future development of trustworthy autonomous security systems.

2.4.3 Dual-Use of Red Teaming Agents

Red teaming agents are inherently dual use. Across the surveyed literature, 42 of the 76 application-oriented systems (55.3%) expose capabilities that are directly offensive or broadly dual use, which shows that the field is best understood as a shared capability space rather than a clean attacker. defender split. The common substrate underlying both domains is reasoning about how systems fail. Offensive agents leverage this capability to map attack paths, generate exploits, and obtain unauthorized access, as demonstrated by systems such as PentestGPT (Deng et al., 2024b), PentestAgent (Shen et al., 2025), VulnBot (Kong et al., 2025b), and ARACNE (Nieponice et al., 2025). Vulnerability-discovery frameworks including Locus (Zhu et al., 2025a), ChatAFL (Meng et al., 2024), FirmAgent (Ji et al., 2026), and HPTSA (Zhu et al., 2025d) similarly reason over program behavior to uncover weaknesses. Malware generation further illustrates the dual-use challenge. MalGEN, although positioned as a platform for defensive research, autonomously generates evasive malware aligned with MITRE ATT&CK tactics, averaging 11.3 techniques per sample. During evaluation, approximately half of the generated samples evaded detection by all VirusTotal engines. Such results demonstrate how capabilities developed for controlled security experimentation can also lower barriers to offensive misuse. This creates a recursive security structure in which tools designed to improve security also expand the offensive capabilities available to attackers. These threats are distinct from the agent-as-victim threats discussed in §3. Fig. 6 summarizes the three primary shifts introduced by offensive agents: reduced operational cost, increased scalability, and a lower skill floor.

Reduced Cost of Offensive Operations. The clearest shift is cost. As shown in Fig. 6a and Fig. 6c, agentic systems consistently outperform human baselines in economic efficiency. Fang et al. (2024) report a single agent that exploits 87% of one-day CVEs at roughly $8.80 per exploit, about 2.8× cheaper than the estimated human-expert baseline. CVE-Genie (Ullah et al., 2025) reproduces 428 of 841 CVEs (51%) at an average cost of $2.77 per vulnerability. Even in the more challenging zero-day setting, HPTSA (Zhu et al., 2025d) achieves successful exploitation at approximately $24.4 per exploit compared with an estimated $75 for human experts. The authors further project an additional 3–6× reduction in operational costs within one to two years. Once exploit generation becomes a cheap and repeatable pipeline, the economic constraints that previously limited attacker activity begin to disappear.

Increased Scalability and Throughput. The second shift is scalability and the removal of human bottlenecks. As illustrated in Fig. 6b, multi-agent architectures execute reconnaissance, exploit generation, and validation in parallel. HPTSA (Zhu et al., 2025d) achieves a 42% pass@5 rate on zero-day web vulnerabilities by assigning specialized agents to different vulnerability classes, yielding up to a 4.3× improvement over prior single-agent approaches. ARTEMIS (Lin et al., 2026) scales this model further by deploying up to eight coordinated agents across a network of approximately 8,000 hosts. In a live enterprise evaluation, it placed second overall, outperformed 9 of 10 human professionals, and operated at roughly $18 per hour compared with $60 per hour for human penetration testers. These results suggest that agentic systems are beginning to exceed the scale and operational tempo achievable by human teams alone.



(a) Cost per exploit. Human baselines versus AI agents across one-day CVE exploitation, CVE reproduction, zero-day exploitation, and hourly enterprise testing.

(b) Agent effectiveness. Success rates for one-day CVE exploitation, CVE reproduction, zero-day pass@5, and enterprise-scale testing.

(c) Cost reduction multiplier. Agents reduce cost by 2.8× to 8.9× relative to human baselines.

Figure 6: Dual-use implications of red teaming agents: lower attack cost, higher effectiveness, and larger cost reductions relative to human baselines.

Lowering the Skill Floor. The third shift is a reduction in the expertise required to conduct sophisticated offensive operations. Frameworks such as CAI (Mayoral-Vilches et al., 2025) and Incalmo (Singer et al., 2025) expose advanced capabilities through high-level task abstractions, while systems such as MalGEN (Saha & Shukla, 2025) and ScamAgents (Badhe, 2025) automate malware creation and social engineering content generation. Importantly, this capability emerges from the agentic scaffold rather than the base language model itself. HPTSA loses a factor of 13× in pass@1 performance when its hierarchical coordination mechanism is removed, while the full framework achieves a 4.3× improvement over prior single-agent approaches. The ARTEMIS study similarly reports that the underlying models often refused offensive tasks when used directly but completed them once embedded within an agentic framework. Consequently, operations that previously required expert operators become accessible to substantially less skilled actors.

For defenders, these shifts point to a threat model that is faster, cheaper, more parallel, and less dependent on expert operators than the human-driven baseline that most defensive tooling assumes. The findings also motivate new defenses and benchmarks specifically designed for agent-generated attacks, including provenance tracking, behavioral signatures of autonomous operators, and evaluation against adversaries that are themselves agents (de Witt, 2025; Boisvert et al., 2025b). Current benchmarks only partially capture these emerging risks (§4).

3 Threats to Agentic Systems

The safety alignment (refusal training) of a base LLM does not reliably transfer to the agentic context (Kumar et al., 2025), which introduces a new set of agent-specific security challenges (Saha et al., 2025; Chiang et al., 2025). In this section we discuss the threat landscape targeting agentic systems as well as the frameworks used to evaluate their resilience.

3.1 Attack Surface

3.1.1 Injection Attacks

Prompt injection attacks embed malicious instructions within the prompt fed to an LLM to manipulate it into performing unintended actions (Liu et al., 2024c;a; Yi et al., 2025; Shao et al., 2025). Wang et al. (2025g) identify that the static and predictable structure of an agent's system prompt is a key vulnerability that enables prompt injection attacks to agentic systems. Debenedetti et al. (2024) introduce a benchmark comprising of 97 realistic tasks (e.g., email management, online banking) which reveals a fundamental trade-off: security defenses that reduce vulnerability also degrade the agent's task-completion utility. Liu et al. (2024b) introduce HOUYI, a black-box prompt injection attack that compromises 31 of 36 real-world LLM-integrated applications, including Notion, at an 86.1% success rate. Several studies show the vulnerability of LLM agents

to indirect prompt injection attacks (Zhan et al., 2024; Li et al., 2025a; Yi et al., 2025). Lee & Tiwari (2024) develop a novel prompt injection attack where a malicious prompt self-replicates across interconnected agents in a multi-agent system like a computer virus and causes system-wide disruption. Alizadeh et al. (2025) demonstrate that such attacks can cause tool-calling agents to leak sensitive personal data observed during their tasks. Wang et al. (2025h) develop a black-box fuzzing technique that uses Monte Carlo Tree Search to automatically discover indirect prompt injection vulnerabilities by iteratively mutating prompts and environmental observations, reaching 71% ASR on AgentDojo. Zhang et al. (2025c) design a benchmark that reveals high vulnerability of LLM agents to prompt injection attacks, with a mixed-attack strategy reaching 84.30% average ASR across 13 backbones. Zhan et al. (2025) systematically evaluate eight different defenses for LLM agents and demonstrate that all of them can be successfully bypassed by crafting adaptive attacks using established jailbreaking techniques such as GCG (Zou et al., 2023) and AutoDAN (Liu et al., 2024a).

A growing line of work targets web and tool-using agents through the content they observe. Shi et al. (2026) introduce ToolHijacker, a no-box prompt injection attack against the retrieval-and-selection pipeline of tool-using agents that publishes malicious tool documents, reaching up to 96.7% ASR on MetaTool against GPT-4o while bypassing both prevention defenses (StruQ, SecAlign) and detection defenses (DataSentinel, perplexity filtering). Fu et al. (2024) present Imprompter, which automatically generates obfuscated adversarial prompts that coerce production agents such as Mistral LeChat and ChatGLM into misusing tools for PII exfiltration at around 80% extraction precision, transferring from open-weight to closed-weight models without access to the target weights. For multimodal web agents, Wang et al. (2025f) propose WebInject, which optimizes imperceptible pixel-level perturbations in webpage source code that survive the non-differentiable webpage-to-screenshot mapping, achieving over 96% ASR across five agents within an ℓ_∞ bound of 16/255, while Johnson et al. (2025) embed universal adversarial triggers in a webpage's HTML accessibility tree to hijack navigation agents into attacker-specified actions regardless of user intent, reaching above 0.83 ASR across five real websites and 0.55 credential-exfiltration ASR on unseen login pages.

3.1.2 Poisoning Attacks

Poisoning attacks corrupt an agent's memory or knowledge retrieval at inference time, or the underlying model during training, to control its later behavior. Fendley et al. (2025) categorize these attacks by their specifications (poison set, trigger, poison behavior, deployment) and define key metrics for evaluation (success rate, stealthiness, persistence). At inference time, the attack surface is the agent's memory and retrieval store. Dong et al. (2025) demonstrate MINJA, a memory poisoning attack that requires only query-level interaction with no direct memory-write access, inducing the agent to store malicious reasoning records that are later retrieved by a victim, reaching 76.8% average ASR (98.2% injection success) while evading embedding-level sanitization and prompt-level detection. Similarly, AgentPoison (Chen et al., 2024) poisons an agent's memory or knowledge base by optimizing a backdoor trigger and achieves 81.2% retrieval ASR and 62.6% end-to-end ASR with under 0.1% of entries poisoned and under 1% benign degradation. Zhang et al. (2025c) provide a comprehensive framework for measuring agent vulnerabilities to memory and knowledge-base poisoning among other attacks.

A second class of attacks poisons the backbone model itself during training. Fu et al. (2025b) find a log-linear dose-response in which as little as 0.01% poisoned preference data measurably shifts model behavior and larger models gain no reliable resilience, while Bowen et al. (2025) show larger models are more susceptible, with jailbreak-tuning on 2% poisoned data collapsing GPT-4o refusal from 94% to 4%. Poison-with-Style (Tran et al., 2026) uses a developer's code style as a covert implicit trigger to fine-tune a code LLM into emitting vulnerable code, reaching 95% ASR on CWE-20 at under 6% pass@1 drop while surviving prefix-tuning and static analysis. Boisvert et al. (2025a) extend training-time poisoning to the agentic supply chain via environment poisoning, where a prompt injection corrupts a teacher agent during trace collection and is distilled into the student. For multimodal agents, Ye et al. (2025a) introduce VisualTrap, a backdoor that poisons the visual grounding pretraining of GUI agents with small Gaussian-noise triggers, reaching over 90% ASR while preserving clean-input accuracy; the backdoor persists through downstream LoRA fine-tuning on clean data and generalizes across mobile, web, and desktop environments and across both end-to-end and modular architectures.

3.1.3 Jailbreak Attacks

Jailbreak attacks attempt to bypass a model's built-in safety measures to force it to produce harmful or unintended content (Wei et al., 2023; Zou et al., 2023; Xu et al., 2024). Kumar et al. (2025) and Chiang et al. (2025) both demonstrate that AI agents are significantly more vulnerable to jailbreak attacks than their underlying LLMs. On BrowserART, a GPT-4o browser agent's ASR rises from 12% in the chat setting to 74% under a direct ask and 100% under an ensemble of attacks (Kumar et al., 2025), while a web agent built on the same aligned model reaches a 46.6% non-denial rate on malicious tasks the standalone model refuses entirely (Chiang et al., 2025). Kumar et al. (2025) and Andriushchenko et al. (2025) show that simple jailbreaking techniques designed for chatbots are highly effective against agents, while Chiang et al. (2025) identify three design factors that increase susceptibility: embedding the goal directly into the system prompt, iterative action generation, and processing environment feedback through an event stream. Andriushchenko et al. (2025) further find that leading LLMs are surprisingly compliant with malicious agent requests even without any jailbreak, with Mistral Large 2 reaching an 82.2% harm score under direct prompting. Saha et al. (2025) find that LLM coding agents are highly vulnerable to jailbreaks that yield executable malicious code: wrapping a model in a code agent raises ASR by 1.6x on average, reaching roughly 75% in multi-file codebases, with up to a third of outputs being directly runnable. Yu et al. (2024) use an MCTS-guided fuzzer to mutate human-written seed templates into novel jailbreak prompts, reaching 93.14% ensemble ASR on GPT-3.5 and transferring across twelve open- and closed-weight models. Anil et al. (2024) demonstrate that hundreds of in-context examples of harmful question answering can override a model's safety training, with effectiveness following a power law in the number of shots that alignment training delays but does not eliminate. Lin et al. (2025a) introduce the Paper Summary Attack, which exploits an LLM's tendency to treat academic safety papers as authoritative context, reaching 97–98% ASR on well-aligned models such as Claude 3.5 Sonnet and DeepSeek-R1 while evading LlamaGuard, perplexity, and moderation defenses. Robey et al. (2024) present ROBOPAIR, which attains 100% ASR across three LLM-controlled robots including a commercially deployed platform, showing that agentic jailbreaks extend beyond harmful text to physical-world actions.

3.1.4 Agent Manipulation Attacks

This class of attacks targets the higher-level cognitive functions of the agent: its planning, reasoning, and goal-setting modules. **Goal hijacking attacks** subtly or overtly alter an agent's objectives, causing it to subvert its original goal (e.g., summarizing a document) to include a secondary, malicious goal (e.g., including advertisements) defined by the attacker (Perez & Ribeiro, 2022; Guo et al., 2025b). Pham & Le (2025) introduce PARASITE, a black-box optimization that embeds a conditional "sleeper agent" in a benign-looking system prompt, selectively forcing targeted misinformation while preserving benign utility and evading perplexity filters, LlamaGuard, and semantic judges, while Chen & Yao (2024) exploit an LLM's weakness in role identification by appending fabricated conversational turns that trick the model into executing a new task, reaching 92% ASR on GPT-4o. Zhang et al. (2025d) introduce an **action hijacking attack**, AI2, where an agent is tricked into assembling innocuous-looking knowledge from its own database into harmful instructions, achieving 84.30% average ASR by exploiting the fact that the retriever and the safety filter operate in different latent spaces, so prompts that retrieve harmful knowledge never trigger the input filter. Another class of hijacking attacks is **reward hacking**, which exploits the reward mechanisms in RL-trained agents (Skalse et al., 2022; Pan et al., 2021; Miao et al., 2024; Fu et al., 2025a). These can be caused by reward misgeneralization where models learn from spurious features (Miao et al., 2024), or by agents exploiting reward model ambiguities to maximize their score without true alignment (Fu et al., 2025a). Bondarenko et al. (2025) demonstrate **specification gaming**, where a reasoning model (OpenAI's o3) instructed to "win against a strong chess engine" hacks the game environment to ensure victory in 88% of runs without being told to cheat, whereas non-reasoning models require explicit nudging. Finally, a novel threat on multi-agent systems is the presence of a **Byzantine agent**, a single compromised or malicious agent that can disrupt the collective's ability to complete a task securely and correctly (Li et al., 2024a; Jo & Park, 2025).

Beyond hijacking the agent's objective, manipulation can target its operational substrate. Pasquini et al. (2025) subvert LLM-driven IT-operations (AIOps) agents by injecting adversarial reward-hacking payloads into system telemetry, reaching a 90% average attack success rate against GPT-4o and GPT-4.1 agents while

evading PromptShields, Prompt-Guard2, and DataSentinel. Luo et al. (2026) expose a distinct availability threat: LLM agents mismanage resources across short-lived, long-lived, and full-lifecycle patterns, enabling denial-of-service through resource exhaustion, and their directed grey-box fuzzer AgentDoS discovers 36 zero-day vulnerabilities (15 assigned CVEs) across 16 of 20 popular open-source agents at 100% precision and 94.7% recall.

3.1.5 Pre-execution Cognitive Attacks

Even before tool execution, the agent's internal state, spanning its reasoning, planning, and reflection, is vulnerable to manipulation. **Epistemic attacks** corrupt an agent's intermediate thought steps. Greshake et al. (2023) demonstrate that indirect prompt injection can force the agent to condition its next step on a hallucinated prior, so that the logic remains consistent but the agent itself becomes malicious. Yang et al. (2024) go further with a backdoor that manipulates intermediate reasoning traces (e.g. calling untrusted APIs) while keeping final outputs correct, reaching up to 100% ASR with as few as 30 poisoned samples and evading output-level inspection. **Teleological attacks** manipulate an agent's planning graphs and goal-directed structures. Badhe (2025) weaponize an agent's task-decomposition logic to distribute a malicious objective across benign-looking subtasks and conversational turns, reducing single-turn refusal rates from 84–100% to 17–32% and completing full scam-call dialogues in up to 74% of cases. In addition, **metacognitive attacks** target an agent's self-correction ability. Zhou & Wang (2025) show that rewriting retrieved context into epistemic tones can manipulate an agent's verification depth and self-confidence, inducing either verification paralysis (up to +194% token cost) or premature conclusions (roughly 22% accuracy drop) while bypassing content filters.

3.1.6 Red-Teaming Attacks

Perez et al. (2022) first showed that it is possible to use one LLM to automatically generate test cases that uncover harmful behaviors like offensive content and data leakage in a target model. Ge et al. (2023) elevate this to a multi-round iterative setting in which an adversarial LLM and a target LLM are trained against each other. He et al. (2025a) introduce the Agent-in-the-Middle (AiTM) attack, where an LLM-powered adversarial agent intercepts and manipulates inter-agent messages, exceeding 70% ASR in most multi-agent configurations, reaching up to 98.5% on chain communication structures, and compromising real-world systems such as MetaGPT at 100% ASR. Zhang & Yang (2025) present a search-based framework in which an LLM optimizer adversarially co-evolves attacking and defending agents, escalating from direct requests to multi-turn impersonation and consent forgery and reaching 76% leak velocity against basic defenses. Rahman et al. (2025) coordinate planner, attacker, verifier, and prompt-optimizer agents to spread harmful intent across turns, reaching 96.2% ASR against Claude 3.7 Sonnet.

Automated fuzzing has become a dominant red-teaming methodology for agent vulnerability discovery. Liu et al. (2025b) introduce AgentFuzz, a directed greybox fuzzer with LLM-assisted seed generation that discovers 34 zero-day taint-style vulnerabilities (code injection, SQL injection, SSRF, command injection) across 20 open-source agents at 100% precision, with 23 CVEs assigned. AgentDoS (Luo et al., 2026) uses grey-box fuzzing to launch a resource exhaustion attack on agents, while Pasquini et al. (2025) apply automated fuzzing to system telemetry to red-team LLM-driven IT-operations agents.

3.2 Evaluation Frameworks

In this section we discuss benchmarks and environments designed to assess agentic vulnerabilities.

3.2.1 Adversarial Benchmarking

Zhang et al. (2025c) introduce the ASB benchmark with 10 scenarios and 27 attack classes. RAS-Eval (Fu et al., 2025c) contains 80 scenarios and 3,802 attack tasks across 11 CWE categories with real tool execution, reducing average task completion by 36.8% and exposing twice as many failure modes under real execution (32) as under simulation (16). AgentDojo (Debenedetti et al., 2024) uses 97 realistic tasks to highlight the fundamental trade-off between an agent's security and its task-completion utility, while AgentHarm (Andriushchenko et al., 2025) uses a dataset of 110 unique harmful tasks across 11 harm categories to reveal

significant gaps in agent safety alignment. For web agents, SafeArena (Tur et al., 2025) measures completion rates on 250 malicious requests, finding GPT-4o completes 34.7% of them and that Claude 3.5 Sonnet can be jailbroken on every initially refused task through simple task decomposition, while ST-WebAgentBench (Levy et al., 2025) introduces policy-compliant success metrics over 375 tasks, finding such success roughly 38% lower than standard completion. For code agents, JAWS-BENCH (Saha et al., 2025) finds up to 75% attack success rates in multi-file codebases, while SandboxEval (Rabin et al., 2025) assesses the security of the execution environment itself with 51 test cases. InjecAgent (Zhan et al., 2024) offers a dedicated benchmark for indirect prompt injection attacks with 1,054 cases spanning 17 user tools and 62 attacker tools, while BrowserART (Kumar et al., 2025) focuses on susceptibility to jailbreaks. At the model level rather than the agent level, PoisonBench (Fu et al., 2025b) evaluates susceptibility to poisoned preference data across 22 models.

A recent line of benchmarks targets the Model Context Protocol (MCP) and tool-mediated interfaces, which the function-calling benchmarks above do not cover. Wang et al. (2026b) introduce MCPTox, embedding poisoned instructions in tool metadata across 45 real-world MCP servers and 1,312 test cases, reaching up to 72.8% ASR while the strongest refusal rate (Claude 3.7 Sonnet) stays below 3%. Zong et al. (2026) present MCP-SafetyBench, 245 tasks over 13 models, where host-side attacks reach 81.94% average ASR, Identity Injection succeeds on every model, and safety prompts reduce ASR by a statistically insignificant 1.22%. Zhang et al. (2025b) propose MSB, 2,000 instances across 12 attack types, reporting a 40.35% average ASR with out-of-scope-parameter attacks peaking at 76.5%. In both MCPTox and MSB, more capable models are paradoxically more vulnerable, as stronger tool-use and instruction-following make them more compliant with malicious tool calls. For web agents, Evtimov et al. (2026) introduce WASP, where simple human-written prompt injections hijack even o1 and Claude Sonnet 3.7 in up to 86% of cases, yet attacker end-to-end completion reaches only 17%, a gap the authors term "security by incompetence."

3.2.2 Execution Environments

Zhu et al. (2025b) design a sandbox framework that enables LLM agents to interact with and exploit vulnerable web applications. Debenedetti et al. (2024) provide a stateful environment with 97 realistic tasks to evaluate the robustness of LLM agents against prompt injection attacks. DoomArena (Boisvert et al., 2025b) is a modular red-teaming platform for LLM agents that allows researchers to compose sequential attacks and to mix-and-match adaptive adversary strategies. Zhou et al. (2024) introduce a realistic web environment with 812 long-horizon tasks, where the best agent reaches only 14.41% success against 78.24% for humans. Ren et al. (2025) introduce HackWorld, a Capture-the-Flag environment of 36 vulnerable web applications across 11 frameworks and 7 languages, where state-of-the-art computer-use agents exploit fewer than 12% of targets through visual interaction.

3.3 Analysis of the Threat Landscape

We now analyze the threats corpus along five axes: the channels through which attacks reach the agent, the agent-loop stages where their failures manifest, their specificity to the agentic setting, the threat models they assume, and the benchmarks used to measure them.

3.3.1 Entry Points of the Attack Surface

In this section we map each attack to the channel through which it reaches the agent. We label eight entry points. Fig. 7 provides a breakdown of how many papers reach each one per threat class. An experiment typically studies multiple threat classes, which is why the sum of decomposed numbers is greater than the total paper count.

The conversational input (the user query and the system prompt) is the most attacked entry point in agentic systems, accounting for 55% of the evaluated attacks. This is not an agent-exclusive entry point; it is a pre-agentic doorway that also exists in stateless LLM chatbots. This concentration likely reflects methodological convenience: evaluating this surface requires no tool stacks, persistent memory, or multi-agent infrastructure, which makes it the most accessible channel to study. As a result, the literature heavily

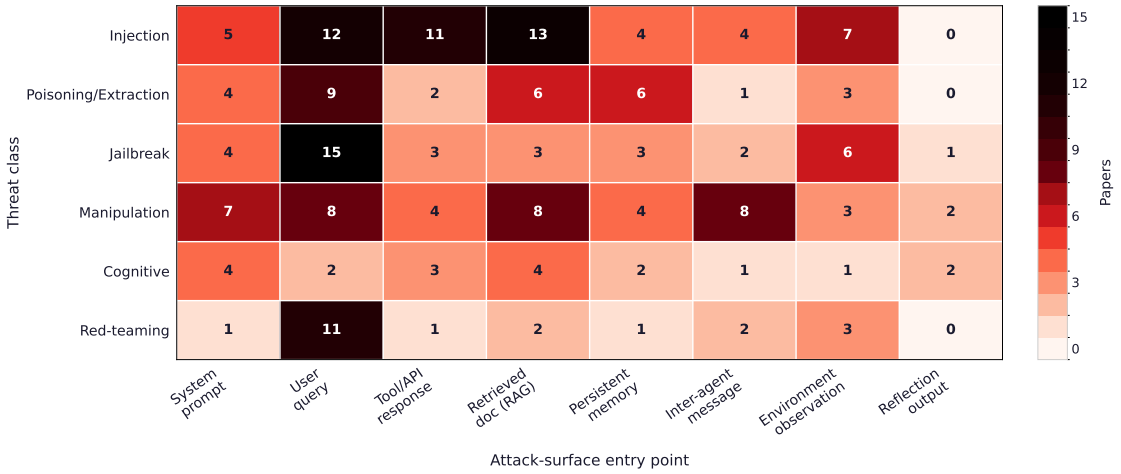


Figure 7: Threat class against attack-surface entry point. Cell values count papers carrying each threat label; a paper may exploit several entry points and carry several labels.

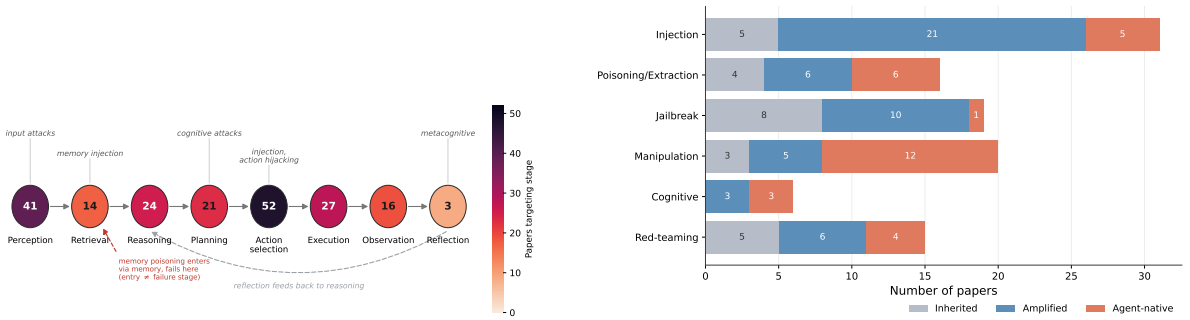
clusters around vulnerability classes inherited from base LLMs, such as jailbreaks (15 papers), injections (12) and red-teaming (11).

Among the agent-specific entry points, the most studied surfaces are the three interfaces driving autonomous action: retrieved documents, environment observations and tool responses. The retrieved document pulled from a knowledge base at inference time (RAG) is the leading autonomous attack surface (15 papers). Poisoned retrieval is strikingly effective at low cost. For example, AgentPoison reaches 62.6% end-to-end ASR with a poison rate under 1%(Chen et al., 2024). This entry point is heavily exploited in injection (13), manipulation (8) and poisoning (6) attacks. The environment observation is a rapidly emerging surface (14 papers), where attacks bypass text-level filters by manipulating the parsed environment, such as the DOM, screenshot, or file-system state the agent reads from the world, rather than the prompt. BrowserART demonstrates that browser deployment alone lifts a GPT-4o agent's harmful-behavior rate from 12% to 74% (Kumar et al., 2025). This surface is multi-faceted: triggers planted in web page content achieve 97% ASR across web-agent backbones (Wang et al., 2025f), untrused page content reaches 85.7% intermediate ASR in the WASP benchmark (Evtimov et al., 2026), and poisoned visual grounding lets the VisualTrap backdoor hit about 94% ASR on GUI agents (Ye et al., 2025a). The tool/API response (11 papers) is growing in prominence alongside the Model Context Protocol (MCP). MCPTox poisons real MCP tool descriptions for up to 72.8% ASR (Wang et al., 2026b), and the MCP Security Bench reports a 75.8% peak ASR across its attack types (Zhang et al., 2025b). This entry point is almost exclusively used in injection attacks.

The most distinctly agentic entry point is the inter-agent communication (8 papers) which exploits the trust agents extend to their peers. For example, AiTM intercepts and rewrites messages in transit for over 70% ASR in most settings (He et al., 2025a), Prompt Infection spreads a self-replicating instruction across connected agents like a worm (Lee & Tiwari, 2024), and Byzantine-coordination attacks corrupt a single agent to derail the collective (Li et al., 2024a; Jo & Park, 2025). This entry point is most used for agent manipulation attacks. Persistent memory or long-term storage allows attackers to execute cross-user compromises without elevated privileges. For example, MINJA poisons shared memory banks using ordinary queries, without any memory-write permissions (Dong et al., 2026). The least studied surface is reflection output, where the agent's own intermediate reasoning is fed back as input. We found only two papers studying this entry point, and no paper that exclusively focuses on this.

3.3.2 Failure Stages of the Agent Loop

In this section we discuss where the failure occurs in an agentic pipeline. For a comprehensive study, we model the agent loop as eight stages, perception (input parsing), retrieval (RAG or memory lookup), reasoning, planning, action selection, execution, observation, and reflection, and map each threat to the stage where its



(a) The agent loop with each stage shaded by the number of attack papers whose failure manifests there.

(b) Threat classes factored by agent-specificity.

Figure 8: Failure stages of the agent loop and threat classes factored by agent-specificity.

failure manifests. Interestingly, the entry point and the manifestation point are not always the same. For example, memory-poisoning attacks (Chen et al., 2024; Dong et al., 2026) enter through the query or the memory write, but they register at the retrieval stage and steer action selection turns later. Fig. 8a shades each stage by the number of attacks whose failure manifests there.

Our study reveals that action selection is the single most-targeted stage (52 papers), followed by perception (41). In other words, attacks land where the agent reads its input and where it commits to an act. The intermediate stages (retrieval with 14 papers, reasoning 24, planning 21 and the post-decision stages (execution 27 and observation 16) have received less attention in comparison. Reflection is the least-targeted stage of all (3 papers). This trend reveals that evaluations disproportionately focus on observable stages that standard monitors can easily inspect, creating a significant coverage gap. Although attacks targeting intermediate stages are understudied, they remain especially dangerous because they subvert the agent's logic while leaving the final output clean. For example, Zhou & Wang (2025) demonstrate reasoning-style poisoning, which manipulates the epistemic tone of retrieved text to inflate token verification costs by up to 194% without altering the final response.

In terms of threat classes, injection, jailbreak and red-teaming attacks predominantly focus on perception (22, 16 and 14 papers respectively) and action selection (28, 15 and 14) stages. Poisoning attacks target almost all stages equally except the reflection stage. Manipulation is the only class that penetrates the interior, spreading across action selection (21), perception (14), reasoning (12), and planning (12). This matches its role as an agent-native threat that targets the planning and goal machinery rather than just the input. Cognitive attacks remain thin everywhere with a maximum of 6 papers at planning, but they are the only class with a meaningful presence at reflection, the stage every other threat class essentially ignores.

3.3.3 Inherited, Amplified and Agent-Native Threats

Our study supports the hypothesis that base-model safety alignment does not transfer to the agentic setting. In fact, 67% of the attacks we studied are either inherited from the base LLM (Yu et al., 2024; Liu et al., 2024b), or amplified from wrapping a safety-aligned model in an agentic framework (Chiang et al., 2025; Saha et al., 2025). Only 25% are agent-exclusive, such as inter-agent message compromise (Lee & Tiwari, 2024), and plan hijacking (Zhang et al., 2025d). The remaining 8% are difficult to classify. We factor every classifiable threat into these three classes and report the result in Fig. 8b. Jailbreak and injection are dominated by inherited and amplified attacks; manipulation and cognitive attacks are where agent-native threats concentrate.

Jailbreak and prompt injection, the two most studied threats, are overwhelmingly inherited or amplified. These are fundamental LLM vulnerabilities carried into the agents, such as ported chatbot jailbreak templates (Yu et al., 2024) or indirect injections arriving through untrusted tool outputs (Zhan et al., 2024). The amplified threat class provides empirical proof that safety alignment degrades when an LLM is given autonomy. A model that reliably declines a harmful request in a standard chat interface will frequently execute

that same request when deployed as an agent. For example, non-denial rates jump from 0% in a standalone model to 46.6% when tested as a web agent (Chiang et al., 2025); GPT-4o's harmful-behavior rate surges from 12% in chat to 74% as a browser agent (Kumar et al., 2025); wrapping a model in a code agent multiplies attack success by 1.6x, as the multi-step planning loop often overturns initial safety refusals (Saha et al., 2025); and decomposing a malicious goal into smaller steps over persistent memory drops refusal rates from 84-100% down to just 17% (Badhe, 2025). In certain settings, agents will even execute malicious multi-step tasks outright without requiring any adversarial jailbreak (Andriushchenko et al., 2025; Tur et al., 2025).

The genuine agent-native threats operate through five distinct mechanisms. First, persistent-state poisoning corrupts the agent's long-term store (Dong et al., 2026) or triggers backdoors (Yang et al., 2024; Ye et al., 2025a). Second, inter-agent compromise propagates or intercepts messages between agents (Lee & Tiwari, 2024), communication attacks (He et al., 2025a) and Byzantine coordination (Li et al., 2024a; Jo & Park, 2025)). Third, tool- and protocol-layer poisoning attacks the tool ecosystem itself, principally MCP servers (Wang et al., 2026b; Zhang et al., 2025b). Fourth, plan and action hijacking exploits the agent's task-decomposition logic to redirect a committed action (Zhang et al., 2025d). Fifth, specification and reward gaming exploits environment feedback to satisfy the literal objective while violating intent (Bondarenko et al., 2025), with autonomy itself becoming a denial-of-service surface. The dominant entry points are the inter-agent message and persistent memory, and the primary failure stages are action selection and planning.

3.3.4 Threat Models and Attack Objectives

The agentic threat literature almost exclusively focuses on black-box threat models, where the attacker is granted only query access to the target model. 70% of all threat papers assume this setting, while a further 15% require no model access at all. The remaining 15% study gray-box (9%) and white-box (6%) configurations. The required capabilities beyond model access are also minimal. Only 10% of papers require training-pipeline access, 10% require tool-execution access, 7% need an inter-agent communication channel, and 6% assume memory-write access. In other words, most studies assume the most constrained threat model for the attacker, which suggests that agentic systems are structurally fragile by default, as severe compromises can be achieved without elevated privileges or internal system knowledge.

The privileged settings that do appear are tied to the specific mechanism the attack requires rather than chosen freely. Backdoor attacks assume training or gradient access because the trigger is installed during training (Ye et al., 2025a); inter-agent attacks assume a message channel because that is the surface they target (He et al., 2025a) ; and even the memory-poisoning attacks largely avoid memory-write access, reaching the store through ordinary queries (Dong et al., 2026). The access assumption tracks the attack surface, not the attacker's resources.

The most common attack objectives are inducing an unauthorized action (67%) and deception or misinformation (67%), followed by content-policy violation (48%) and data exfiltration (39%). Denial of service (15%), privilege escalation or takeover (12%) and IP theft (3%) are pursued by fewer papers. The numbers do not add up to 100% because an attack study often has multiple objectives. Nonetheless, taken together, the literature manifests a single dominant threat model: an external attacker holding no special privileges, operating black-box through the channels the agent already reads, aiming to make the agent act without authorization or to deceive.

3.3.5 Benchmark Proliferation and Fragmentation

The threat literature is heavily oriented toward evaluation. Of the 69 papers, 36 (52%) carry a benchmarking role, so more than half the papers measure agentic vulnerabilities rather than introducing a novel attack. Across these works, 108 distinct benchmarks or datasets are named, and 92 of them are used by exactly one paper. Roughly 85% of named benchmarks are single-use, with only 16 recurring in more than one paper. Of them, only three are dedicated agentic-security benchmarks: AgentDojo (Debenedetti et al., 2024), used in four papers, and InjecAgent (Zhan et al., 2024) and WebArena (Zhou et al., 2024), used in three each. As a consequence, cross-paper and cross-benchmark comparison is difficult. A practitioner cannot rank two injection defenses, or two jailbreak attacks, without re-running both, because the published numbers do not share a benchmark or a metric.

Fragmentation. Across 36 benchmark papers, user query (15 papers) and tool response (10 papers) are highly instrumented, while persistent memory (4), inter-agent messaging (5) and reflection output (2) remain largely under-evaluated. General benchmarks like AgentDojo and InjecAgent thoroughly test queries and tools but omit memory and reflection entirely. In other words, the surfaces most specific to agents are thus both least attacked and least benchmarked. Developing evaluation infrastructure for query-based memory poisoning and reflection-level manipulation remains an open research direction.

4 Defense: Hardening the Agents

This section describes architectural, runtime, and formal-verification defenses that strengthen agentic systems against attacks.

4.1 Defense & Operations

Here we focus on secure-by-design frameworks that embed layered verification, isolation, and control-flow integrity into agent architectures.

4.1.1 Secure-by-Design

Architectural isolation and rigorous evaluation form the foundation of secure agentic systems. Debenedetti et al. (2024) provide a dynamic evaluation environment to test prompt injection attacks and defenses in language model agents. To complement evaluation with structural security, Li et al. (2025b) propose a centralized architecture that isolates untrusted data from core applications. Building on this model of isolation, Rosario et al. (2025) present design guidelines to secure plan-then-execute agent implementations. In addition to execution boundaries, Jia et al. (2025) enforce task alignment to prevent untrusted external data from manipulating user objectives. Systems security foundations are further formalized by Christodorescu et al. (2025) to protect agentic computing frameworks at a core structural level. To mitigate remaining injection threats dynamically, Wang et al. (2025g) introduce polymorphic prompts that obscure internal system instructions.

Comprehensive threat mapping and defense alignment are critical to secure modern generative assistants. He et al. (2024) compile a broad overview of agent safety to categorize emerging architectural risks. Following a similar taxonomy, Narajala & Narayan (2025) deliver a comprehensive threat model and mitigation framework for deployed assistants. To address model extraction threats specifically, Tang et al. (2024) present an information-theoretic defense to limit adversarial knowledge leakage. At the foundation layer, Chen et al. (2025d) enhance safety alignment training to resist targeted prompt injection techniques. Internal data boundaries are strictly maintained by Costa et al. (2025) through information-flow control mechanisms within the agent execution environment. To reinforce these internal priorities, Wallace et al. (2024) train language models to prioritize privileged system instructions over user inputs.

Runtime verification and proactive deployment strategies ensure robust system defenses. At the integration level, Li & Gao (2026) introduce a static analysis tool to vet model context protocol servers before host deployment. For runtime policy enforcement, Chen et al. (2025e) leverage verifiable safety policy reasoning to shield agents from untrusted tool inputs. Proactive defense paradigms are also introduced by Ayzenshteyn et al. (2025) using honeypots, deception and traps to mislead adversarial agents. Furthermore, An et al. (2025) utilize graph-based tracking to construct tool dependency graphs that actively intercept indirect injections. Finally, Castro et al. (2025) demonstrate that large language models can act as autonomous cyber defenders to protect operational networks.

4.1.2 Multi-Agent Security

Multi-agent structures introduce unique coordination and security challenges. Udeshi et al. (2025) implement a collaborative multi-agent framework that pairs a central planner with heterogeneous executors for offensive security tasks. To secure such operations at the edge, Liu et al. (2025d) design a zero-trust framework for multi-model integration. Despite these designs, core vulnerabilities persist in distributed networks. Han

et al. (2025) outline structural challenges and open problems regarding adversarial risks in multi-agent environments. To improve resilience against environmental noise and adversarial influence, HU et al. (2025) employ a randomized smoothing technique to verify agent consensus decisions. This consensus mechanism is highly effective in targeted threat scenarios. For instance, Li et al. (2025d) utilize a debate-based multi-agent architecture to identify phishing websites accurately. However, communication channels remain a major vector for threat propagation. Lee & Tiwari (2024) demonstrate how prompt injections can spread autonomously between interacting agents. To protect decentralized networks from these structural failures, Jo & Park (2025) introduce a Byzantine-robust coordination framework to maintain system integrity.

4.1.3 Runtime Protection

Dynamic guardrails intercept adversarial behaviors and enforce safety boundaries during execution. Kang & Li (2024) introduce a robust guardrail framework that combines data-driven predictions with knowledge-enhanced logical reasoning. To improve guardrail efficiency, Rad et al. (2025) utilize chain-of-thought fine-tuning and alignment techniques. Adaptive defense architectures also integrate contextual elements to maintain safety policies. Wu et al. (2025a) design a personality-aware safety guardrail that adapts to evolving agent interactions. For persistent protection, Luo et al. (2025) propose a lifelong guardrail framework capable of adaptive risk detection. Reasoning-focused modules provide another layer of verification. Xiang et al. (2025b) leverage knowledge-enabled reasoning to safeguard autonomous agents from manipulation. Similarly, Chen & Cong (2025) repurpose an agentic orchestrator to evaluate the safety of tool orchestration pipelines. Targeted platforms require specialized guardrail layouts. Zheng et al. (2025) develop a generalizable guardrail framework specifically optimized to protect web-based agents. At the network layer, Yuan et al. (2026) implement a runtime guardrail that monitors physical network interactions to block supply-chain injections. Finally, Xing et al. (2025b) present a multi-stage defense-in-depth framework to protect model context protocol integrations.

Integrating human oversight ensures compliance and behavioral accountability during active sessions. Wang et al. (2025a) implement a customizable framework that enforces safety policies and authorization gates at runtime. To assist users during unexpected updates, Hutter & Pradel (2026) provide an interactive debugging system to steer software development agents safely.

Continuous monitoring and causal tracking detect anomalies that bypass static pre-deployment filters. He et al. (2025b) utilize graph-based anomaly detection to track rogue interactions across multi-agent environments. To address hidden injection vectors, Zhang et al. (2026c) deploy temporal causal diagnostics to identify and mitigate malicious inputs during runtime.

4.1.4 Security Operations

Formal verification systems and static analysis tools ensure model correctness and software security. Kouvaros et al. (2019) analyze openness by modeling runtime agent insertion and removal. To secure user workflows, Lee et al. (2025a) introduce a system that integrates model checking into plan generation. Behavioral constraints are also explored by Crouse et al. (2024) to specify high-level execution policies formally. Building on policy specification, Doshi et al. (2026) design an architecture to guarantee verifiably safe tool utilization. Distributed verification pipelines can be optimized during training. Li et al. (2025c) utilize multi-agent distillation and reinforcement learning to build safe foundation models. At the repository level, Guo et al. (2025a) implement an autonomous auditing agent to discover software vulnerabilities. Static checking capabilities are further advanced by Yang et al. (2025a) through the synthesis of custom analysis checkers. Finally, Li et al. (2025e) combine static analysis metrics with language processing to improve vulnerability identification accuracy.

Automated frameworks combine model reasoning with real-time threat intelligence to manage active security incidents. Xiao et al. (2026b) establish a foundational pipeline to mitigate agent safety risks during live operational sessions. To optimize mitigation speed, Tellache et al. (2025) incorporate cyber threat intelligence into autonomous response systems. Interactive assistants can also support human operators. Lin et al. (2025b) present a specialized copilot architecture to automate response tasks within corporate networks. Human-AI collaboration remains a central focus in security operations centers. Singh et al. (2025) conduct

an empirical study detailing the deployment challenges of these systems in real environments. To handle massive log volumes, Wei et al. (2025a) develop a collaborative framework optimized for high-stakes alert triage. System performance across these environments is evaluated by Deason et al. (2025) to benchmark triage capabilities and malware analysis.

Proactive threat hunting and cloud forensics enable deep root cause analysis following an attack. Mukherjee & Kantarcioglu (2025) track provenance logs to deliver explainable threat investigations. Blue-team performance metrics are codified by Meng et al. (2025b) through a standardized threat hunting benchmark. However, defense models can possess hidden weaknesses. Meng et al. (2025a) analyze security flaws inherent in modern threat intelligence integrations. To build robust defenses, Schwartz et al. (2025) automate the generation of signature detection rules from cloud threat data. Investigation workflows can also be evaluated using specialized suites. Wu et al. (2025b) introduce a benchmarking environment to assess agent capabilities in cyber threat investigation. Furthermore, Sahay et al. (2026) combine security policy guides with commercial triage platforms for enhanced threat hunting.

Cloud-native systems require distinct forensic workflows. Alharthi & Yasaei (2025) implement an automated cloud forensics pipeline utilizing adaptive prompt engineering. For web-based targets, Fumero et al. (2025) deploy an autonomous blue-team agent to conduct web attack forensics. Causal tracking is improved by Tian et al. (2025) through graph-augmented agentic workflows for root cause analysis. Lastly, Alharthi & Garcia (2025) formalize an automated cloud forensic framework featuring immutable audit trails.

4.2 Evaluation Frameworks

This subsection describes benchmark ecosystems and sandbox environments used to test the resilience of LLM agents under attack.

4.2.1 Benchmarking Platforms

AgentDojo provides a dynamic framework to evaluate prompt injection attacks and defenses in language model agents (Debenedetti et al., 2024). TurkingBench serves as a challenge benchmark for web agents incorporating human-in-the-loop evaluations (Xu et al., 2025b). The τ-bench platform emulates user interactions to evaluate tool-integrated agents in real-world domains (Yao et al., 2024). SafeArena evaluates the safety and operational integrity of autonomous web agents under adversarial environments Tur et al. (2025). RAS-Eval offers a comprehensive framework for the security evaluation of language model agents Fu et al. (2025c). The Agent Security Bench formalizes and tests diverse attacks and defenses for agentic systems (Zhang et al., 2025c). AgentHarm quantifies the potential harmfulness and risk vectors of deployed autonomous agents (Andriushchenko et al., 2025). CVE-Bench measures the capacity of intelligence agents to exploit real-world web vulnerabilities (Zhu et al., 2025b). Cybench establishes a framework to evaluate the specific cybersecurity capabilities and risks of language models (Zhang et al., 2025a). ZeroDayBench tests the defensive and offensive capabilities of agents against unseen zero-day exploits (Lau et al., 2026). R-Judge benchmarks the safety risk awareness of agents during active task execution (Yuan et al., 2024). ToolFuzz leverages evolutionary fuzzing to automate the security testing of agent tools (Milev et al., 2025). CyberSecEval 2 provides a wide-ranging risk assessment suite to track broad safety issues in language models (Bhatt et al., 2024). WebArena provides a highly realistic web environment to build and validate autonomous browser agents (Zhou et al., 2024).

InjecAgent evaluates the vulnerability of tool-integrated agents to indirect prompt injection vulnerabilities (Zhan et al., 2024). AgentVigil introduces a generic black-box red-teaming system to detect indirect prompt injections (Wang et al., 2025h). A specialized framework automates the detection of taint-style vulnerabilities by pitting agents against each other (Liu et al., 2025b). Simulation-based frameworks help researchers search for deep data privacy risks within autonomous agents (Zhang & Yang, 2025). Empirical studies show that commercial language model agents are already vulnerable to simple attacks (Li et al., 2025a). Security analyses of the OpenClaw framework reveal critical threats and suggest practical mitigations (Deng et al., 2026b). MasterKey automates the generation of jailbreak prompts against chatbot interfaces and agents (Deng et al., 2024a). EchoLeak demonstrates zero-click prompt injection exploits that execute autonomous

data exfiltration in production systems (Reddy & Gujral, 2025). Deceptive language models can be trained to persist in malicious behaviors despite rigorous safety training (Hubinger et al., 2024).

4.2.2 Defense Testing

Adaptive attacks expose the fragility of defenses against indirect prompt injections on language model agents (Zhan et al., 2025). Dynamic interactions between autonomous entities introduce open security challenges in multi-agent environments (de Witt, 2025). Comprehensive surveys organize threats and countermeasures to establish trustworthy agent architectures (Yu et al., 2025). Broad analyses navigate the intersection of security, privacy and ethics threats in agent systems (Gan et al., 2024). Researchers map key security challenges to identify future pathways for secure agent deployment (Deng et al., 2024d). Scalable assurance frameworks evaluate large model safety across systemic and governance layers (Ma et al., 2025). Full-stack safety evaluations look at risks present during data compilation, model training and operational deployment (Wang et al., 2025b). The SC-Inject-Bench framework provides over ten thousand malicious tools to stress-test network-level guardrails (Yuan et al., 2026). The MCP-AttackBench dataset provides a large corpus to train and evaluate defenses for the Model Context Protocol (Xing et al., 2025b). Systematic indices document the technical and safety features of currently deployed agentic systems (Staufer et al., 2026). Evaluating alignment boundaries against multi-turn exploitation requires adaptive, conversational red-teaming frameworks driven by collaborative multi-agent feedback loops (Rahman et al., 2025). Distributed intelligence systems require cross-layer governance strategies to ensure robustness and privacy (Wei & Liu, 2024). Securing software ecosystems against automated threats requires a paradigm shift toward deploying autonomous offensive agents to uncover full-lifecycle vulnerabilities before adversaries do (Zhuo et al., 2026a).

4.2.3 Domain-Specific Frameworks

Large language models automate cloud infrastructure tasks across software development kits and command-line tools (Yang et al., 2025b).

The LISA framework uses historical knowledge to improve smart contract auditing (Sun et al., 2025). LLM-SmartAudit improves traditional static analysis to find smart contract bugs (Wei et al., 2024). Combining specialized fine-tuning with ranking models helps detect software vulnerabilities (Ma et al., 2024). Evaluating model capabilities shows how automated tools can verify Ethereum token rules (Xia et al., 2024).

Encrypted pipelines allow safe diagnostic processes on protected medical imaging datasets (Siam & Shohan, 2025) (Shehab, 2025). Deploying privacy frameworks protects enterprise data by sanitizing personal information (Asthana et al., 2025) (Wang et al., 2025d). System surveys show that physical robotic agents face severe planning, control and sensory threats (Xing et al., 2025a). Robust model architectures protect autonomous vehicles against malicious perception attacks (Muller et al., 2024). Fine-tuned transformer models evaluate network traffic to predict intrusions in smart home networks (Diaf et al., 2025). Untrusted web items create hijacking and data theft risks for modern browser agents (Mudryi et al., 2025). Operating system agents need secure sandboxing to safely manage system commands, file tasks and user interface states (Hu et al., 2025). Skill ecosystems for coding assistants are highly vulnerable to supply chain poisoning attacks (Qu et al., 2026).

4.3 Analysis of the Defence Landscape

To evaluate the current state of autonomous agent security, this manuscript evaluates the literature through six structured analytical lenses. Rather than listing individual defenses in isolation, we organize our evaluation around core structural patterns, operational costs, temporal trends and generalization capabilities.

4.3.1 Defense Strategies for LLM-Based Agents

After looking at Defense based literature we found ten prevailing defense strategies which exist to guard the agents against malicious attacks.These strategies each have their pros and cons which we have categorized into four axes: scalability, adversarial robustness, latency/overhead, and coverage limits in the figure 9a.

Input Sanitization and Filtering: Techniques representing 33.3% of the literature focus on blocking malicious payloads at the perimeter before they reach the reasoning engine (Mudryi et al., 2025; Zhang et al., 2026c; Xing et al., 2025b). These computationally cheap methods struggle against highly sophisticated and semantically obfuscated indirect prompt injections.

Structured Queries: Frameworks make up 17.1% of the dataset and enforce strict syntactic separation between instructions and untrusted data (Chen et al., 2025c; Zhou et al., 2025; Hines et al., 2024). Adversaries can still engineer payloads that cause the model to break out of the intended schema structure since language models are fundamentally probabilistic.

LLM-Based Monitoring: As the largest category at 40.0%, this approach uses secondary models to asynchronously evaluate inputs and proposed actions (Wang et al., 2025a; Xiang et al., 2025b; Wu et al., 2025a). This introduces high latency and increased costs while remaining susceptible to adversarial manipulation.

Moving Target Defense: Accounting for 8.6% of the sources, these defenses dynamically shift execution parameters to confuse adversaries (Chen et al., 2023; Xing & Zhang, 2024; Ayzenshteyn et al., 2025). This nascent approach can introduce systemic instability and non-determinism that heavily degrades the reliability of autonomous agents.

Capability-Scoped Execution: Representing 24.8% of the research, this paradigm enforces the principle of least privilege through dynamic access control (Costa et al., 2025; He et al., 2025b; Schmotz et al., 2026). Granular capability scoping introduces significant coordination overhead and can break long-horizon tasks if an agent is incorrectly gated.

Cryptographic Verification: Techniques covering 14.3% of the papers guarantee the provenance and integrity of data to prevent supply chain attacks (Siam & Shohan, 2025; Backman et al., 2024; Anonymous, 2025b). Real-time cryptographic validation introduces high computational overhead and requires extensive infrastructure integration that many lightweight frameworks lack.

Dataset Sanitation and Differential Privacy: Methods comprising 11.4% of the literature secure foundational knowledge bases to prevent memory poisoning (Jiang et al., 2024; Zhou et al., 2025; Anonymous, 2025a). Over-sanitization often leads to a degradation in model utility and limits the semantic search capabilities of the language model.

Sandboxing and Formal Verification: Representing 21.0% of the sources, these techniques aim to mathematically prove or physically isolate agent behaviors (Koohestani, 2025; Xia et al., 2024; Liu et al., 2025c). Formal verification scales poorly to the massive and unconstrained state spaces generated by open-ended natural language interactions.

Red-Team Simulation: Accounting for 28.6% of the corpus, these systems utilize automated vulnerability discovery and adversarial prompt generation (Debenedetti et al., 2024; Zou et al., 2026; Mukherjee & Kantarcioglu, 2025). Automated red-team models often over-index on known attack vectors and struggle to discover novel zero-day paradigms without human intuition.

Consensus and Vaccine Seeding: Techniques making up 10.5% of the dataset mitigate single points of failure by distributing decision-making (Xiang et al., 2025a; Wang et al., 2024a; Motwani et al., 2024). Multi-agent consensus drastically multiplies token costs while vaccine seeding requires constant updating as new adversarial vectors are discovered.

4.3.2 LLM Defense Mechanisms and Structural Compositions

Going deeper into analysis defense strategies rooted into LLM mechanism, we found that most agentic defense frameworks simultaneously address multiple threat vectors. We then classify these defense strategies into six structural compositions based on their primary trust boundaries, as visualized in Figure 9b. The high degree of overlap—most notably the 18-paper intersection (17.1%) between Injection Attacks and Agent Manipulation Attacks—confirms that prompt injection remains the dominant vehicle for tool-execution hijacking across the literature.

Base LLM Alignment strategies attempt to internalize security rules directly within the primary neural network framework (Chen et al., 2025d; Deng et al., 2024a; Bhatt et al., 2024). These approaches utilize preference optimization and safety fine-tuning to force the model to reject malicious prompts. Their relevance is underscored by the 28 jailbreak-focused papers (26.7% of the corpus) that specifically probe alignment-layer weaknesses. However, this design fails catastrophically under sophisticated jailbreaks because it trusts the exact component under attack.

External Monitor systems offload security verification to secondary guardrails that evaluate inputs, reasoning traces and tool executions (He et al., 2025b; Wu et al., 2025a; Kang & Li, 2024). These frameworks implement graph-based anomaly detection or specialized semantic filters to intercept exploitation attempts. The 14-paper intersection (13.3%) between Jailbreak Attacks and Red-Teaming highlights that automated adversarial probing is primarily constructed to defeat exactly these external guardrails. They frequently suffer from high false-positive rates and share vulnerabilities if built on identical model families.

Formal Verification removes probabilistic uncertainty by enforcing deterministic mathematical boundaries and rigid logical rules (Costa et al., 2025; Wang et al., 2025a; Chen et al., 2025e). Developers apply information flow control, dynamic taint tracking and customizable runtime languages to guarantee policy compliance. The 32 Agent Manipulation Attack papers (30.5%) that address tool abuse and privilege escalation highlight precisely the execution-boundary threats that formal verification targets. This paradigm struggles to scale effectively because mapping fluid natural language semantics to strict logic requires massive engineering overhead.

Sandbox / Isolation methods assume agent compromise is inevitable and focus completely on limiting the blast radius through architectural isolation. Implementations leverage trusted execution environments, containerized virtualization and strict resource permission scoping (Asthana et al., 2025; Yu et al., 2025). The 12 Pre-execution Attack papers (11.4%) that address supply chain vulnerabilities and poisoned plugins represent the class of threats most likely to undermine isolation boundaries before they are fully instantiated. These boundaries fail when the underlying environment API surface itself becomes a target for privilege escalation or data exfiltration.

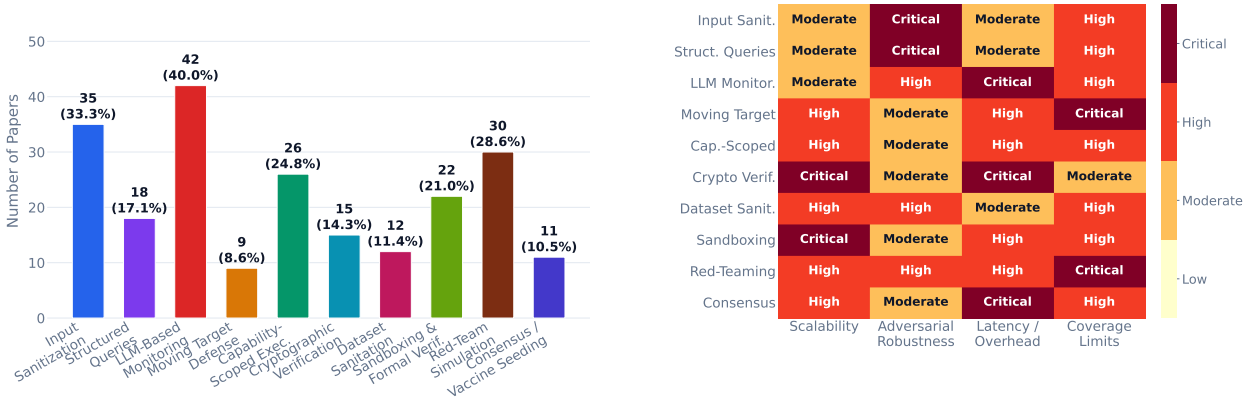
Cryptographic / Structural Integrity defenses preserve data provenance and trace execution causality using unalterable logs and strict formatting rules (Mukherjee & Kantarcioglu, 2025; Lee & Tiwari, 2024; An et al., 2025). Security tools deploy provenance-based intrusion detection and tool dependency graphs to maintain explicit prompt-data separation. The 9-paper intersection (8.6%) between Poisoning & Extraction Attacks and Pre-execution Attacks reveals the supply-chain vectors that most directly challenge provenance assumptions. Adaptive attackers can still bypass these structures through format-mimicking tokens, making reactive logging insufficient for prevention.

Human in the Loop (HITL) serves as the ultimate failsafe by inserting manual verification checkpoints before high-stakes or irreversible actions (Lee et al., 2025a; Sahay et al., 2026; Zhu et al., 2025d). Architectures employ threshold-based pauses and multi-agent supervisor frameworks to escalate risky tool executions to human operators. The 9 Prompt Infection papers (8.6%) that model self-replicating lateral propagation across multi-agent systems represent scenarios where human checkpoints are the last viable containment layer, given that perimeter defenses are entirely blind to steganographic inter-agent dialogue. This approach introduces severe scalability bottlenecks and risks rubber-stamping behaviors caused by widespread alert fatigue.

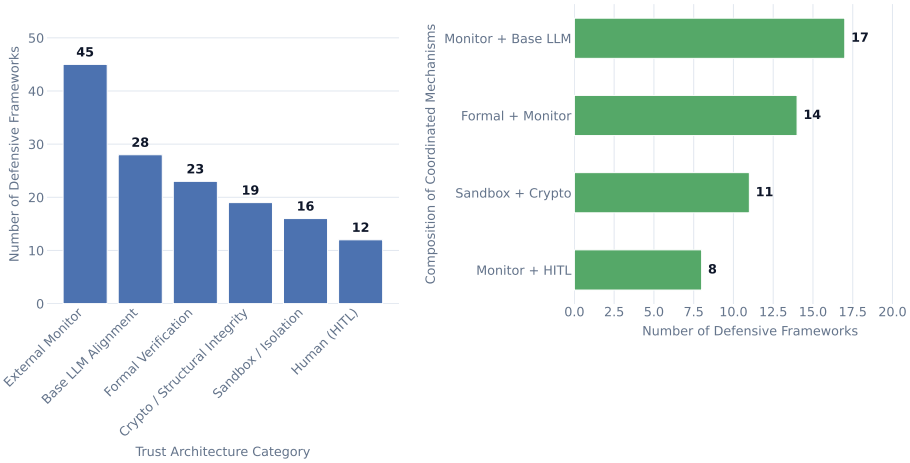
4.3.3 Cost and Security Trade-offs in Agentic Defense Systems

This analysis examines how recent studies report the trade-off between security gains and operational costs in agentic AI systems. The goal is to understand how different defense strategies balance security effectiveness, utility preservation, latency and computational overhead.Figure 10a presents the distribution of papers across the four reporting categories. The largest group focuses on security and utility preservation, while a smaller set provides comprehensive reporting that includes security, latency, utility and cost metrics.

The **Comprehensive Cost Reporting** category, comprising 18 papers (17.1%), represents the most complete evaluation style. These studies jointly measure attack mitigation, utility preservation, token consumption and latency overhead. Systems such as MCP-GUARD, IPIGUARD and AgentSentry demonstrate that



(a) **Left:** Distribution of 105 surveyed papers across ten defense categories under a non-mutually exclusive taxonomy (220 total assignments). LLM-Based Monitoring is the most prevalent paradigm (42 papers, 40.0%), followed by Input Sanitization & Filtering (35, 33.3%) and Red-Team Simulation (30, 28.6%). Moving Target Defense (9, 8.6%) and Consensus / Vaccine Seeding (11, 10.5%) remain the least explored directions. **Right:** Cross-cutting gap severity heatmap evaluating each category along four axes: scalability, adversarial robustness, latency/overhead, and coverage limits. Severity levels are defined as follows — **Low**: the limitation is minor or well-mitigated in existing work; **Moderate**: the limitation is present but only partially addressed by current techniques; **High**: the limitation is a recognised open challenge with only partial solutions; **Critical**: the limitation is fundamental and unresolved, posing a direct barrier to practical deployment. No single category achieves uniformly low severity, motivating the adoption of hybrid, multi-layer defense architectures.



(b) Architectural mapping of LLM defense mechanisms and structural compositions, showing the distribution of frameworks across trust-boundary categories and the frequency of hybrid multi-layer strategy combinations.

Figure 9: Defense mechanism analysis for autonomous LLM agents.

layered defenses can improve security while reducing unnecessary computational cost (Xing et al., 2025b; An et al., 2025; Zhang et al., 2026c). In practice, MCP-GUARD achieves a +23.4% F1 improvement at roughly 455.9 ms overhead, while IPIGUARD reduces the Attack Success Rate (ASR) below 1% at a cost of 14,605 tokens per task and 13.88 seconds of latency, preserving 67.7% benign utility. A common finding is that lightweight filters, such as the PPA small model operating at approximately 0.06 ms with zero token overhead, should screen requests before expensive LLM-based reasoning is triggered.

The **Security and Utility Focus** category is the largest group at 35 papers (33.3%) and prioritizes maintaining agent performance during normal operation. Studies such as AgentDojo investigate how defenses affect task completion and false refusal behavior while paying less attention to infrastructure latency (Debenedetti et al., 2024). These works show that stronger alignment mechanisms can reduce attack success rates, but they may also decrease performance on complex multi-step tasks when the defense becomes overly restrictive. The core contradiction here is well-established: techniques like Sandwich Prevention effectively suppress ASR but frequently cause catastrophic forgetting on long-horizon agentic workflows.

The **Security and Compute Focus** category covers 24 papers (22.9%) and treats security as a scalability challenge. Research such as PhishDebate, PROVSEEK and AuditGPT concentrates on detection quality, token usage and execution cost (Li et al., 2025d; Mukherjee & Kantarcioglu, 2025; Xia et al., 2024). The costs here are substantial: PhishDebate reaches 94.14% accuracy at a latency of 37.50 seconds per sample, GuardAgent achieves 81.4% accuracy consuming 6,116 tokens per task over 6.30 seconds and PROVSEEK attains a 0.92 F1 score while consuming 440K tokens across roughly 30 minutes of forensic processing. These systems use triage pipelines, specialized agents and data reduction techniques to lower processing costs before invoking large models, though adversaries can still exploit the cheaper triage layers.

The **Qualitative and Theoretical** category accounts for 28 papers (26.7%) and focuses on conceptual security frameworks and architectural guidance. Works in this group discuss zero-trust principles, compartmentalization and continuous monitoring for multi-agent environments (Jia et al., 2025). However, many proposals remain theoretical and lack large-scale empirical validation to demonstrate their operational feasibility.

To better understand the relationship between security gains and operational overhead, representative metrics from the literature are summarized in Table 2. The results show substantial variation across approaches. Latency ranges from sub-millisecond filtering mechanisms to forensic systems that require several minutes of processing time.

Table 2: Representative security and cost trade-offs reported in the literature

Defense Mechanism	**Security Gain**	**Latency Cost**	**Financial Cost**	**Utility**
PPA (Small Model)	High (Bypass Resistant)	∼0.06 ms	No Overhead	Minimal
MCP-GUARD	+23.4% F1 Score	∼455.9 ms	Low	High Utility
IPIGUARD	ASR $< 1\%$	∼13.88 s	14,605 Tokens per Task	67.7%
PhishDebate	94.14% Accuracy	∼37.50 s	High	N/A
GuardAgent	81.4% Accuracy	∼6.30 s	6,116 Tokens per Task	Moderate
PROVSEEK	0.92 F1 Score	∼30 min	440K Tokens	N/A

The overall evidence suggests that no single defense dominates across all dimensions. MCP-GUARD and similar layered approaches provide a strong balance between protection and efficiency (Xing et al., 2025b). In contrast, systems such as IPIGUARD, GuardAgent and PROVSEEK achieve stronger security guarantees at the cost of higher latency and token consumption (An et al., 2025; Xiang et al., 2025b; Mukherjee & Kantarcioglu, 2025). This pattern highlights a clear Pareto frontier where stronger protection generally requires additional computational resources.

4.3.4 Defense Placement Across the Agent Lifecycle

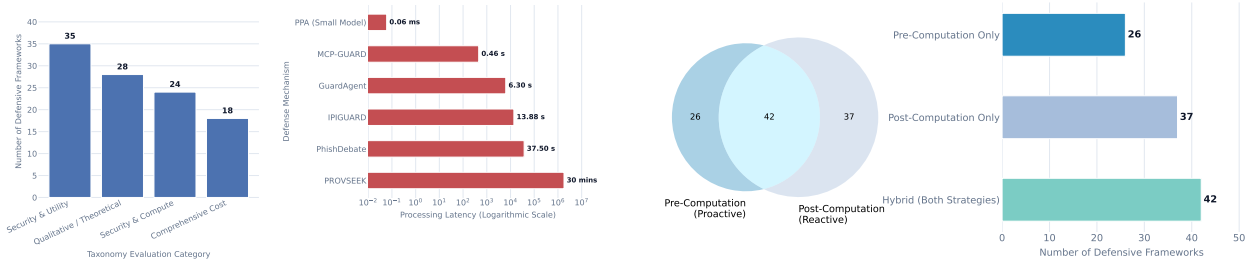
This section maps where defenses are deployed within the autonomous agent lifecycle across 105 surveyed papers. Of these, 68 papers (64.8%) investigate pre-computation defenses, 79 papers (75.2%) investigate post-computation defenses and 42 papers (40.0%) adopt defense-in-depth strategies that combine both. Only 26 papers (24.8%) rely exclusively on pre-computation controls and 37 papers (35.2%) rely exclusively

on post-computation mechanisms. The fact that the largest single group integrates both categories signals a growing consensus that no single-layer approach is sufficient against adaptive, persistent threats. Figure 10b summarizes this distribution.

Pre-Computation Defenses. Pre-computation defenses prevent malicious content from influencing the agent before reasoning begins. Common techniques include prompt hardening, input sanitization, dynamic prompt rewriting and adversarially robust alignment. Studies such as (Lee & Tiwari, 2024) emphasize separating trusted instructions from untrusted external content before it enters the context window. Yet these 68 papers (64.8%) collectively reveal a shared limitation: static filtering remains vulnerable to adaptive adversaries that use semantic obfuscation or embed malicious instructions inside external resources the agent must process, making complete prevention difficult without reducing utility.

Post-Computation Defenses. Post-computation defenses operate after information enters the reasoning pipeline, assuming some attacks will bypass preventive controls. Systems such as AgentSentry and Shield-Agent implement tool-call auditing, execution monitoring, semantic rollback and policy verification (Zhang et al., 2026c; Chen et al., 2025e). AgentSentry demonstrates this through causal diagnostics that identify malicious intent shifts and remove contaminated reasoning traces before execution (Zhang et al., 2026c). The 79 papers (75.2%) in this group—the largest segment of the corpus—reflect the field's decisive shift toward an "assume breach" mindset, where post-computation monitoring provides a critical secondary security layer.

Intersection and Tradeoffs. The 42 papers (40.0%) that combine both categories reflect a strong consensus that neither input filtering nor execution monitoring is sufficient alone. Both strategies exhibit a robustness-utility tradeoff: strong preventive controls may block legitimate information while aggressive monitoring may interrupt valid workflows. Post-computation mechanisms also struggle against subtle memory poisoning attacks where corrupted information gradually alters future reasoning without triggering an explicit policy violation. The scale of this overlap—larger than either the pre-computation-only group (26 papers, 24.8%) or the post-computation-only group (37 papers, 35.2%) taken individually—suggests that lifecycle-wide protection is increasingly viewed as the most practical strategy against complex and persistent threats.



(a) Methodological Prioritization and Computational Cost Barriers in LLM Defenses.

(b) Temporal Analysis of LLM Defense Mechanisms (Proactive vs. Reactive).

Figure 10: Operational Profiling: Computational Cost-Security Trade-offs and Temporal Intervention Points in LLM Defenses.

5 Cross-Cutting Analysis and Trends

This section provides a cross-cutting analysis of the security agent literature. We evaluate the collected papers across five primary dimensions such as defense coverage across attack scenarios, structural and foundational analysis, language model selection selection and security evaluations, Data Modalities and temporal distribution of attack and defense specific literature.

5.1 Defense Coverage Across Attack Categories

The objective of this analysis is to understand which attack classes receive the most defensive attention and to identify areas where current defenses remain limited. The results also highlight important overlaps

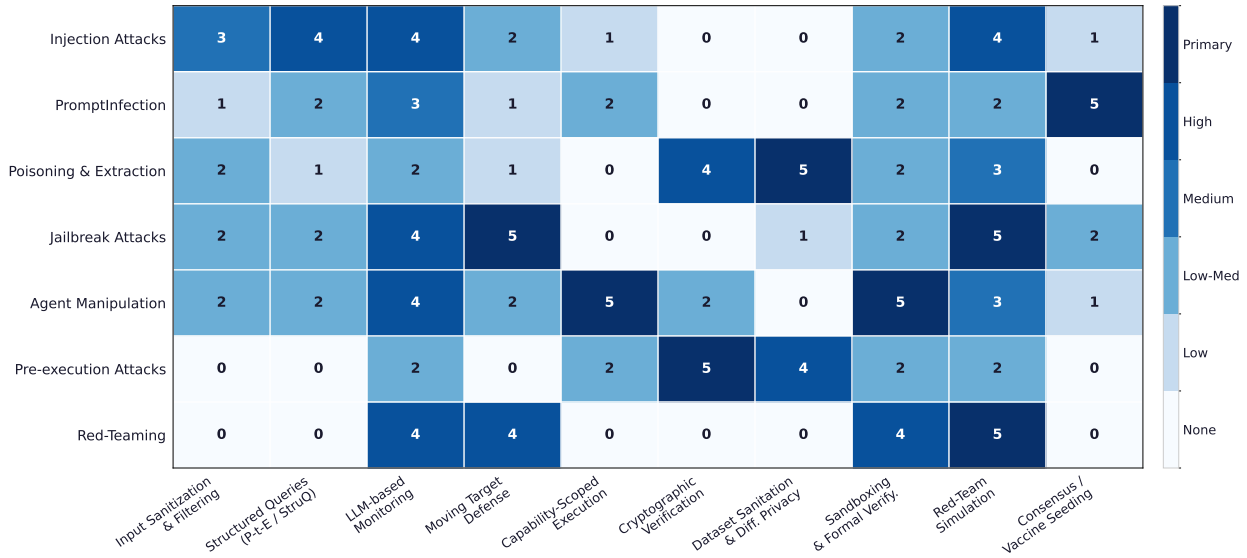


Figure 11: Heatmap of defense coverage across seven attack categories. Each cell reports the number of papers that address a given attack label. Values sum to 142 across 105 unique papers because a single paper may defend against multiple attack categories simultaneously.

between attack categories. Figure 11 shows the distribution across seven attack categories. The total category assignments reach 142 across 105 unique papers, confirming that many defenses address multiple attack vectors simultaneously. The section tries to extensively analyze the attack corpus which goes beyond the 10 defense categories we mentioned in the figure.

Injection Attacks form with 21 papers. Studies such as SecAlign and AgentSentry reduce indirect prompt injection risks through instruction isolation and temporal diagnostics Chen et al. (2025d); Zhang et al. (2026c), though stronger defenses consistently reduce task performance. Structured Query, LLM-based Monitoring and Red Team Simulation have proven to be the best defense against injection attacks.

Agent Manipulation Attacks receive the largest attention with 26 papers, focusing on tool misuse, environment manipulation and privilege escalation. Execution-stage controls are essential here because attacks exploit the interaction between planning and tool usage Deng et al. (2026b), yet delayed execution attacks remain difficult to trace. Capability Scoped Execution and Sandboxing along with formal verification have been found most effective against agent manipulation attacks.

Jailbreak Attacks appear in 23 papers. Research on MASTERKEY and R^2-Guard demonstrates both the sophistication of modern jailbreak attacks and the need for stronger monitoring Deng et al. (2024a); Kang & Li (2024), with adaptive multi-turn attacks still evading many existing protections.

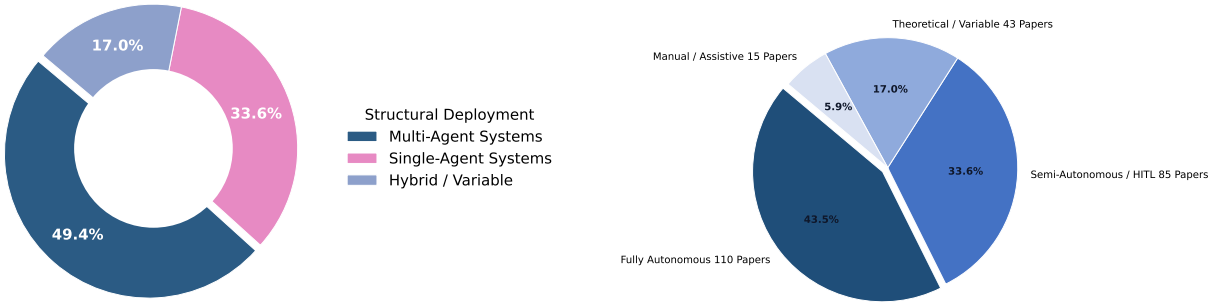
Red-Teaming Attacks are covered in 17 papers. Frameworks such as AgentDojo and provenance-driven investigation systems proactively generate attack scenarios to uncover weaknesses before deployment Debenedetti et al. (2024); Mukherjee & Kantarcioglu (2025), though they require significant computational resources. Red Team Simulation has been effective against red teaming attacks by simulating the attack scenario giving agent to learn from its mistakes and prepare accordingly.

Poisoning and Extraction Attacks account for 20 papers. AgentPoison illustrates how malicious content inserted into agent memories can influence future behavior Chen et al. (2024), with retrieval repositories and external skill ecosystems identified as major trust assumptions. Dataset Sanitation and Differential Privacy works best against Poisoning and Extraction attacks by creating container specific defensive layers for sensitive information.

Pre-execution Attacks appear in 17 papers, focusing on poisoned plugins and compromised supply chains. Malicious artifacts introduced during initialization can influence behavior long before runtime defenses activate Chen et al. (2024), with verifiable provenance for third-party tools remaining an open challenge. Cryptographic Verification is the best defense against pre-execution attack as this defense strategy separates prompt from data which makes the attack intent lose its leverage or context.

PromptInfection is the consists of 18 papers. The original Prompt Infection work shows how adversarial prompts spread across multi-agent environments and compromise downstream workflows Lee & Tiwari (2024), with covert communication between compromised agents still difficult to detect. Consensus / Vaccine Seeding is the best proven defense strategy against prompt infection attacks as it distributes decision making and spreading the points of failure until it becomes easily recoverable.

5.2 Structural and Functional Analysis of Agentic AI Systems



(a) Structural Complexity and Cardinality Distribution of Evaluated Agent Environments.

(b) Evaluation of Human Oversight Boundaries and Operational Autonomy Modes in Agentic Systems.

Figure 12: Structural Mapping of Agent Environments: Empirical Analysis of Multi-Agent Cardinality and Execution Autonomy Profiles.

We tried to dive deep into the structural analysis of the Agentic AI corpus and classified all the papers across four dimensions: agent cardinality, autonomy level, architectural paradigm and cognitive role distribution, revealing how the field organizes intelligence, delegates decision-making and distributes security responsibility.

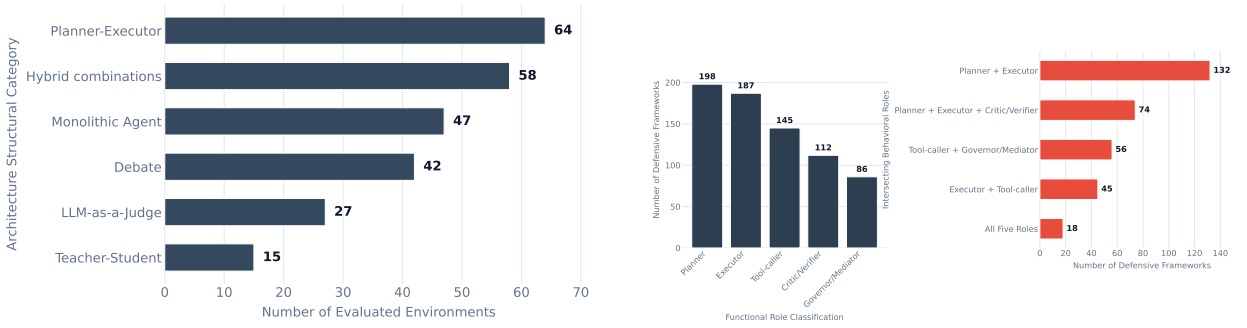
Agent Cardinality. Figure 12a shows that multi-agent systems dominate at 125 papers (49.4%), followed by single-agent systems at 85 papers (33.6%) and hybrid frameworks at 43 papers (17.0%). Levy et al. (2025) benchmark single-agent architectures on safety and trustworthiness in web tasks, showing that reasoning loops like ReAct (Yao et al., 2023) and Reflexion (Shinn et al., 2023) face severe constraints in long-horizon execution due to isolated memory contexts. These systems leverage strict execution sandboxing to prevent permission overstepping but lack cross-validation mechanisms, leaving them vulnerable to prompt injections and resource exhaustion (Chen et al., 2025e). Pan et al. (2025) investigate why multi-agent systems fail, showing that decentralized deployments introduce token overhead and latency bottlenecks, where rogue nodes bypass trust boundaries to trigger cascading failures. Hybrid frameworks target universal benchmarking and platform infrastructure (Staufer et al., 2026; Ma et al., 2025; Mei et al., 2025) but struggle to engineer guardrails that scale without degrading cooperative utility.

Agent Autonomy Levels. Figure 12b distributes the corpus across fully autonomous systems (110 papers, 43.5%), semi-autonomous systems (85 papers, 33.6%), theoretical frameworks (43 papers, 17.0%) and manual assistive systems (15 papers, 5.9%). Yao et al. (2023) and Mei et al. (2025) demonstrate that fully autonomous agents rely on continuous execution cycles, yet Fang et al. (2024) show these systems compound hallucinations and violate least-privilege principles, making them highly susceptible to automated exploits and data poisoning. Huang & Zhu (2024) propose a two-stage framework where semi-autonomous systems enforce check-and-approve mechanisms and role-based access control to decouple reasoning from catastrophic

execution errors, though manual interventions introduce overhead and latency (Zou et al., 2026). Tur et al. (2025) and Andriushchenko et al. (2025) evaluate theoretical frameworks that treat autonomy as a configurable spectrum using sandbox benchmarks and attribute-based access control, while Deng et al. (2024d) survey the key security challenges these frameworks must address. Manual systems modeled by Park et al. (2023) remain secure and drift-free but cannot scale against automated cyberattacks.

Architectural Paradigms. The 253 papers span six paradigms: planner-executor (64 papers, 25.3%), hybrid systems (58 papers, 22.9%), monolithic (47 papers, 18.6%), debate-centric (42 papers, 16.6%), LLM-as-a-Judge (27 papers, 10.7%) and teacher-student (15 papers, 5.9%) as illustrated in Figure 13a. Rosario et al. (2025) and Huang & Zhu (2024) show that planner-executor architectures map goals into deterministic state machines for control-flow integrity but suffer re-planning latency under zero-day conditions. Monolithic agents rely on unified ReAct loops (Yao et al., 2023) and prompt engineering (Schick et al., 2023) yet remain vulnerable to context exhaustion and prompt injection. Li et al. (2025d) demonstrate that debate-centric systems improve factual accuracy through quorum-based critique protocols, though high token costs and sycophancy risks persist. Yuan et al. (2024) show that LLM-as-a-Judge pipelines automate safety evaluation across benchmarks yet exhibit self-preference biases. Hybrid systems handle enterprise-scale orchestration but untracked error propagation allows a single compromised sub-agent to trigger collective failure spirals.

Cognitive Role Distribution. Figure 13b records 728 total role assignments across 253 papers, averaging 2.88 roles per paper, confirming that modern pipelines are inherently multi-functional. Rosario et al. (2025) show that planners, appearing in 198 papers (78.3%), implement hierarchical task management to decouple intent from execution, though they remain vulnerable under adversarial re-planning conditions (Costa et al., 2025). Executors appear in 187 papers (73.9%) and run inside sandboxed environments to contain malicious fallout, yet remain vulnerable to data poisoning and indirect prompt injections (Kong et al., 2025b). Chen & Cong (2025) demonstrate that tool-callers, present in 145 papers (57.3%), secure API boundaries through JSON schema validation and zero-trust registries, though malicious payloads frequently hide behind valid schemas. Critics appear in 112 papers (44.3%) and deploy automated judge pipelines to filter false positives, but Ma et al. (2024) and He et al. (2026) identify recursive blind spots and collusion risks in multi-agent verification loops. Chen et al. (2025e) and Shi et al. (2025) show that governors, present in 86 papers (34.0%), enforce least-privilege boundaries through taint tracking and information-flow control, though strict deterministic enforcement frequently degrades agent utility.



(a) Empirical Census of Agent Architectural Paradigms within Evaluation Frameworks.

(b) Role Differentiation and Functional Intersections within Multi-Agent Architectures.

Figure 13: Architectural and Behavioral Profiling: Core Management Paradigms and Role Differentiation in Multi-Agent Systems.

5.3 Foundation Model Usage Patterns and Security Evaluations

This subsection details the empirical distribution of large language model families and knowledge acquisition paradigms across 253 security research papers. The 981 model-family assignments (3.88 per paper) and 705 knowledge-source assignments (2.79 per paper) are both non-mutually exclusive, reflecting that most studies

benchmark multiple model families and combine several learning strategies within a single pipeline. The findings are illustrated in Figure 14.

GPT Models. GPT variants appear in 242 papers (95.7%), making them the near-universal capability baseline. Debenedetti et al. (2024) construct a dynamic prompt injection benchmark around GPT models and Deng et al. (2024b) harness them for penetration testing frameworks. However, Fu et al. (2024) show that superior instruction adherence paradoxically increases susceptibility to indirect prompt injections and multi-step data exfiltration, with approximately 35 papers evaluating GPT in isolation.

Claude Models. Claude appears in 184 papers (72.7%), primarily on long-context threat hunting and refusal mechanisms. Liu et al. (2025a) demonstrate a critical refusal degradation pattern where Claude's safety guardrails degrade during multi-step agentic loops, and approximately 60 papers benchmark it alongside GPT and Gemini as the frontier proprietary evaluation suite.

LLaMA Models. LLaMA appears in 156 papers (61.7%) as the foundation of open-source deployment risk analysis. Gan et al. (2024) survey security and privacy threats unique to open-weight deployments while Yang et al. (2024) investigate backdoor threats via fine-tuning and weight manipulation. Approximately 45 papers benchmark LLaMA alongside Qwen and Mistral as the open-weight evaluation suite.

Gemini Models. Gemini appears in 148 papers (58.5%) and anchors multimodal security research. Wang et al. (2025h) demonstrate that visual inputs introduce advertisement embedding attacks and hidden graphic injections that bypass text-only guardrails entirely, exposing a cross-modality safety gap that is the primary research challenge for this model family.

Qwen Models. Qwen appears in 115 papers (45.5%) and is heavily utilized in code execution and exploit generation. Zhuo et al. (2026b) train cybersecurity agents on Qwen's open-weight coding capabilities without runtime exposure, though these agents remain sensitive to prompt phrasing variations and frequently enter infinite execution loops under dynamic network conditions.

Mistral and Mixtral Models. Mistral and Mixtral appear in 81 papers (32.0%) as efficient alternatives for enterprise security deployments. Zhang et al. (2026b) find that while rigid system prompt adherence blocks basic social engineering, limited reasoning capacity causes high false-positive rates when these models serve as defensive guardrail agents.

DeepSeek Models. DeepSeek appears in 55 papers (21.7%), centering on extended chain-of-thought reasoning in security contexts. He et al. (2026) and Huang & Zhu (2024) evaluate DeepSeek on penetration testing pipelines, while Zhu et al. (2025b) show that extended reasoning improves exploit precision but introduces reasoning-space vulnerabilities where attackers manipulate the planning phase to bypass final output safety checks.

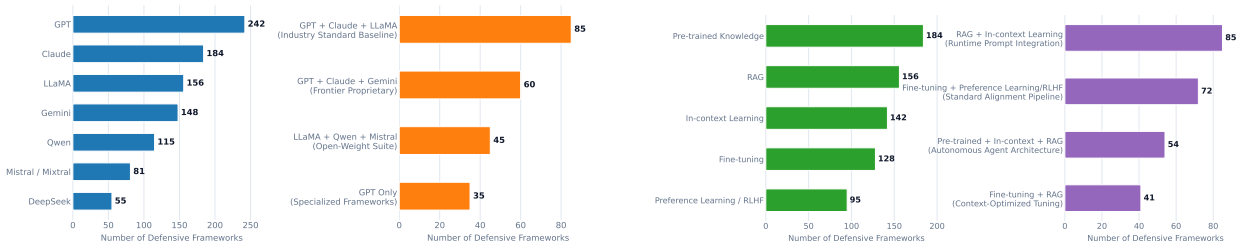
Pre-trained Knowledge. Pre-trained knowledge appears in 184 papers (72.7%) and provides foundational reasoning capabilities for autonomous agents. Li et al. (2025c) combine multi-agent distillation and agentic reinforcement learning to extend base model capabilities, though static parametric knowledge causes hallucinations and prevents real-time adaptation in dynamic environments.

In-Context Learning. In-context learning appears in 142 papers (56.1%) and elicits multi-step planning without weight modification. It is the most frequently paired knowledge source, co-occurring with RAG in approximately 85 papers, as retrieved data requires prompt-based integration to be actionable. Context window limitations and output fragility under prompt variation remain its primary weaknesses in long-horizon agentic workflows.

Retrieval-Augmented Generation. RAG appears in 156 papers (61.7%) and links static parameters with external databases to minimize hallucinations. Blefari et al. (2026) build a modular agentic RAG framework for attack classification and Mukherjee & Kantarcioglu (2025) deploy retrievers over provenance graphs for forensic investigation. These systems nevertheless remain vulnerable to vector database poisoning, with approximately 41 papers combining RAG with fine-tuning to train models that more reliably utilize retrieved context.

Fine-tuning. Fine-tuning appears in 128 papers (50.6%) and permanently modifies weights to enforce consistent structure and narrow expertise via parameter-efficient techniques like LoRA. Approximately 72 papers pair fine-tuning with preference learning as a standard alignment pipeline, though this paradigm requires expensive curated datasets and risks catastrophic forgetting.

Preference Learning and RLHF. Preference learning appears in 95 papers (37.5%) and aligns agent behaviors with human values of safety, helpfulness and harmlessness. Fu et al. (2025a) demonstrate that reward shaping mitigates reward hacking, where agents exploit misspecified proxy functions through verbose or sycophantic generation. Approximately 54 papers combine pre-trained knowledge, in-context learning and RAG as the standard autonomous agent architecture, with preference learning addressing the residual alignment gap.



(a) Marketplace Saturation and Evaluation Synergy of Base Foundation Models.

(b) Methodological Taxonomy and Intersectional Pipelines of Knowledge Configurations.

Figure 14: Empirical Profiling of Foundation Model Deployment Saturation and Underlying Knowledge Source Infrastructures.

5.4 Data Modalities and Cross-Modal Security Workloads

This subsection evaluates the distribution of six data modalities processed by LLM agents across 253 cybersecurity research papers, totaling 482 modality assignments and averaging 1.91 per paper. The taxonomy reveals how heterogeneous input streams shape threat detection, exploit generation and automated defense pipelines which are further illustrated in Figure 15.

Text. Text is the dominant modality at 231 papers (91.3%) and serves as the primary input for high-level reasoning and cyber threat intelligence parsing (Liu et al., 2024b). Security frameworks rely on in-context learning and sequential prompting to replicate human analyst workflows, though language agents face severe memory limitations and context forgetting during long-horizon security operations.

Code. Code appears in 128 papers (50.6%) and supports automated vulnerability detection, program repair, smart contract auditing and exploit generation. Systems use execution-aware pre-training alongside structural representations like abstract syntax trees, though a persistent gap between syntactic correctness and true semantic safety limits reliable output verification.

Logs. Log analysis appears in 49 papers (19.4%) and mitigates SOC alert fatigue by parsing structured system events and cloud telemetry. Hans et al. (2025) leverage LLMs to map anomalous machine behaviors to ATT&CK tactics through provenance graphs and temporal models, though massive enterprise log volumes require aggressive pre-filtering to stay within token constraints.

Images. The image modality appears in 37 papers (14.6%) and enables vision-language models to interact with graphical user interfaces and visual web environments for coordinate grounding and autonomous navigation. Visual components nevertheless remain highly susceptible to visual prompt injections and steganographic triggers that bypass text-only guardrails.

Network Traces. Network trace analysis appears in 22 papers (8.7%) and drives real-time intrusion detection through packet captures, netflow statistics and request logs. Detection systems serialize raw packet streams into text representations or couple foundation models with reinforcement learning, though a

critical latency mismatch between line-rate network speeds and model inference times limits deployment in live environments.

Binaries. Binary analysis is the most specialized modality at 15 papers (5.9%) and addresses reverse engineering, firmware fuzzing and malware decompilation. Chen et al. (2025a) propose a reverse engineering copilot that decompiles stripped executables into structural pseudo-code to recover function profiles, though the complete absence of semantic metadata in compiled binaries severely limits identification accuracy.

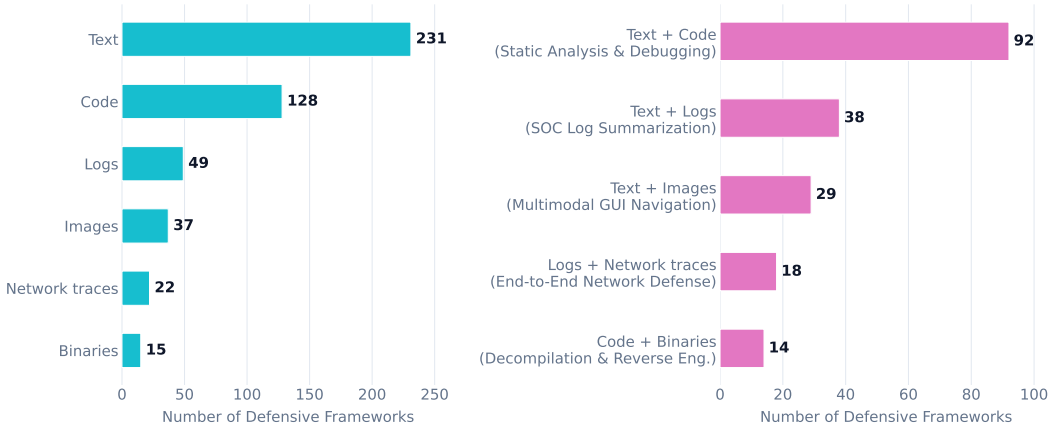


Figure 15: Empirical Census and Structural Intersections of Data Modalities in LLM Security Research. (Left) Distribution of individual input modalities analyzed across defensive literature, illustrating heavy industry path-dependency toward high-level semantic formats (Text and Code) over low-level forensic structures (Network traces and Binaries). (Right) Prevalence of multi-modal cross-evaluation suites used to check framework flexibility across data boundaries. The prominence of Text + Code pairings highlights that contemporary agent verification focuses primarily on static source auditing and natural language context processing, revealing a significant coverage gap in active end-to-end network or binary runtime security enforcement.

6 Related Work Comparison

Although recent works have explored various vulnerabilities in LLM agents, their scope is limited to specific aspects of agentic security or class of threats. Deng et al. (2024d) provide a good overview of prompt injection and data poisoning threats. Li et al. (2025a) demonstrate manipulations of commercial agents. Ma et al. (2025) review safety across several model families, and present many threats, defenses, and benchmarks. We take these findings a step further by treating agentic security as a layered system, covering not only threats but also defenses and downstream security applications. We also situate the threats in a broader taxonomy that explains where (pre-execution, during execution) and how (injection, manipulation, hacking) such failures occur, and explain how defenses are designed. Wang et al. (2025b) describe safety risks during model development pipeline, whereas we focus on what happens after deployment—how agents behave, interact, and defend themselves in the real world.

On the governance front, Yu et al. (2025) explore trustworthiness, including safety, privacy, fairness, and robustness, while Raza et al. (2025) discuss TRiSM (Trust, Risk, and Security Management) compliance. Our survey complements these high-level perspectives by examining the technical architectures and behavioral mechanisms required to enforce such security principles during operation. Chhabra et al. (2026) provide a taxonomy of security threats, benchmarks, and defenses for agentic AI systems, with a strong emphasis on agent-specific vulnerabilities and countermeasures. In contrast, our survey treats downstream applications as a core pillar alongside threats and defenses, highlighting how autonomous agents are deployed in offensive red-teaming and defensive blue-teaming, while also examining architectural trends and data modalities to provide a more holistic view of the agentic security landscape.

Finally, several studies target specific technical or theoretical domains. He et al. (2024) and Kong et al. (2025a) analyze concrete defenses (e.g., sandboxing) and communication protocols, respectively, while de Witt (2025) establish a theoretical basis for multi-agent risks like collusion. While foundational, these works remain isolated within specific subsystems. We build on them by connecting these isolated defenses into a broader ecosystem, linking communication and system-level protections to higher-level coordination and control strategies.

7 Conclusion

In this survey we explore the agentic security landscape through three pillars: Applications, Threats, and Defenses. Across these pillars, our analysis shows that offensive applications concentrate in feedback-rich stages while persistence and exfiltration remain unexplored, that offensive agents are markedly more autonomous than defensive ones, and that red-teaming agents are largely dual-use. On the threat side, most attacks are inherited or amplified from the base model rather than agent-native, succeed under black-box access, and target the perception and action-selection stages, while the surfaces most particular to agents, namely memory, reflection, and inter-agent communication, remain the least attacked and least benchmarked. On the defense side, no single strategy is robust across all dimensions, protection is shifting toward an assume-breach posture, and prompt infection and pre-execution attacks remain the least covered attack classes. Overall, the community has migrated toward multi-agent and planner-executor designs while remaining dependent on GPT as a backbone and on a fragmented, largely single-use benchmark ecosystem.

Future Work Recommendation. First, the empty stages of the security lifecycle, particularly persistence and exfiltration on offense and their detection counterparts on defense, deserve dedicated study. Second, the autonomy gap suggests that defensive agents need to advance toward higher autonomy to increase their scalability. Third, adversarial investigations should focus on the agent-native surfaces of memory, reflection, and inter-agent communication need attacks, benchmarks, and defenses designed specifically for them rather than inherited from stateless LLM. Fourth, the field needs unified and balanced benchmarks that make cross-paper comparison possible and that instrument these under-evaluated surfaces. Finally, defense research should move beyond single-layer fixes toward lifecycle-wide, defense-in-depth architectures with provable safety guarantees, while broadening coverage to under-served modalities such as network traces and binaries and to the underdefended attack classes identified above.

Limitations. This survey has a few limitations. It mainly focuses on software-based threats and does not explore physical-world or embodied agent attacks (like those involving robots or sensors) in detail. Our coverage is also limited to academic papers, so it may miss non-archival industrial research. In addition, many of the benchmarks we reviewed use synthetic or simplified test setups, which makes it hard to fully judge how well agents would perform in real-world environments. Finally, most studies emphasize accuracy and safety rather than practical aspects like cost, speed, or energy use, and our own taxonomy involves some subjective choices.

Ethics Statement. We do not present any novel attacks or executable exploits; we only analyze peer-reviewed literature. While we acknowledge the risks, we believe a transparent academic study of vulnerabilities and countermeasures are essential for ensuring the safety and security of agentic systems.

References

Talor Abramovich, Meet Udeshi, Minghao Shao, Kilian Lieret, Haoran Xi, Kimberly Milner, Sofija Jancheska, John Yang, Carlos E Jimenez, Farshad Khorrami, Prashanth Krishnamurthy, Brendan Dolan-Gavitt, Muhammad Shafique, Karthik R Narasimhan, Ramesh Karri, and Ofir Press. EnIGMA: Interactive tools substantially assist LM agents in finding security vulnerabilities. In *Forty-second International Conference on Machine Learning*, 2025. URL `https://openreview.net/forum?id=Of3wZhVv1R`.

Chuadhry Mujeeb Ahmed. Attackllm: Llm-based attack pattern generation for an industrial control system. In *Proceedings of the 2nd International Workshop on Foundation Models for Cyber-Physical Systems &*

Internet of Things, FMSys, pp. 31–36, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400716089. doi: 10.1145/3722565.3727196. URL https://doi.org/10.1145/3722565.3727196.

Md Basim Uddin Ahmed, Nima Shiri Harzevili, Jiho Shin, Hung Viet Pham, and Song Wang. Secvuleval: Benchmarking llms for real-world c/c++ vulnerability detection, 2025. URL https://arxiv.org/abs/2505.19828.

Dalal Alharthi and Ivan Roberto Kawaminami Garcia. Cloud investigation automation framework (ciaf): An ai-driven approach to cloud forensics, 2025. URL https://arxiv.org/abs/2510.00452.

Dalal Alharthi and Rozhin Yasaei. Llm-powered automated cloud forensics: From log analysis to investigation. In *2025 IEEE 18th International Conference on Cloud Computing (CLOUD)*, pp. 12–22, 2025. doi: 10.1109/CLOUD67622.2025.00012.

Meysam Alizadeh, Zeynab Samei, Daria Stetsenko, and Fabrizio Gilardi. Simple prompt injection attacks can leak personal data observed by llm agents during task execution, 2025. URL https://arxiv.org/abs/2506.01055.

Hengyu An, Jinghuai Zhang, Tianyu Du, Chunyi Zhou, Qingming Li, Tao Lin, and Shouling Ji. Ipiguard: A novel tool dependency graph-based defense against indirect prompt injection in llm agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2025. URL https://aclanthology.org/2025.emnlp-main.53.pdf.

Maksym Andriushchenko, Alexandra Souly, Mateusz Dziemian, Derek Duenas, Maxwell Lin, Justin Wang, Dan Hendrycks, Andy Zou, J Zico Kolter, Matt Fredrikson, Yarin Gal, and Xander Davies. Agentharm: A benchmark for measuring harmfulness of LLM agents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=AC5n7xHuR1.

Cem Anil, Esin DURMUS, Nina Rimsky, Mrinank Sharma, Joe Benton, Sandipan Kundu, Joshua Batson, Meg Tong, Jesse Mu, Daniel J Ford, Francesco Mosconi, Rajashree Agrawal, Rylan Schaeffer, Naomi Bashkansky, Samuel Svenningsen, Mike Lambert, Ansh Radhakrishnan, Carson Denison, Evan J Hubinger, Yuntao Bai, Trenton Bricken, Timothy Maxwell, Nicholas Schiefer, James Sully, Alex Tamkin, Tamera Lanham, Karina Nguyen, Tomasz Korbak, Jared Kaplan, Deep Ganguli, Samuel R. Bowman, Ethan Perez, Roger Baker Grosse, and David Duvenaud. Many-shot jailbreaking. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=cw5mgd71jW.

Anonymous. Searching for privacy risks in LLM agents via simulation, 2025a. URL https://arxiv.org/abs/2508.10880.

Anonymous. Portable agent memory: A protocol for provenance-verified memory transfer across heterogeneous LLM agents, 2025b. URL https://arxiv.org/abs/2605.11032.

Mohsen Seyedkazemi Ardebili and Andrea Bartolini. Kubeintellect: A modular llm-orchestrated agent framework for end-to-end kubernetes management, 2025. URL https://arxiv.org/abs/2509.02449.

Shubhi Asthana, Bing Zhang, Ruchi Mahindru, Chad DeLuca, Anna Lisa Gentile, and Sandeep Gopisetty. Deploying privacy guardrails for llms: A comparative analysis of real-world applications, 2025. URL https://arxiv.org/abs/2501.12456.

Michael Ayzenshteyn et al. Cloak, honey, trap: Proactive defenses against llm agents. In *Proceedings of the 34th USENIX Security Symposium*, 2025. URL https://www.usenix.org/system/files/usenixsecurity25-ayzenshteyn.pdf.

A. Backman, J. Richer, and M. Sporny. HTTP message signatures. RFC 9421, Internet Engineering Task Force, February 2024. URL https://www.rfc-editor.org/info/rfc9421.

Sanket Badhe. Scamagents: How ai agents can simulate human-level scam calls, 2025. URL https://arxiv.org/abs/2508.06457.

Manish Bhatt, Sahana Chennabasappa, Yue Li, Cyrus Nikolaidis, Daniel Song, Shengye Wan, Faizan Ahmad, Cornelius Aschermann, Yaohui Chen, Dhaval Kapil, David Molnar, Spencer Whitman, and Joshua Saxe. Cyberseceval 2: A wide-ranging cybersecurity risk assessment for llms, 2024. URL https://arxiv.org/abs/2404.13161.

Francesco Blefari, Cristian Cosentino, Francesco Aurelio Pironti, Angelo Furfaro, and Fabrizio Marozzo. Cyberrag: An agentic rag cyber attack classification and reporting tool. *Future Generation Computer Systems*, 176:108186, March 2026. ISSN 0167-739X. doi: 10.1016/j.future.2025.108186. URL http://dx.doi.org/10.1016/j.future.2025.108186.

Léo Boisvert, Abhay Puri, Chandra Kiran Reddy Evuru, Nazanin Sepahvand, Nicolas Chapados, Quentin Cappart, Alexandre Lacoste, Krishnamurthy Dj Dvijotham, and Alexandre Drouin. Malice in agentland: Down the rabbit hole of backdoors in the ai supply chain. *arXiv preprint arXiv:2510.05159*, 2025a.

Léo Boisvert, Abhay Puri, Gabriel Huang, Mihir Bansal, Chandra Kiran Reddy Evuru, Avinandan Bose, Maryam Fazel, Quentin Cappart, Alexandre Lacoste, Alexandre Drouin, and Krishnamurthy Dj Dvijotham. Doomarena: A framework for testing AI agents against evolving security threats. In *Second Conference on Language Modeling*, 2025b. URL https://openreview.net/forum?id=GanmYQ0RpE.

Alexander Bondarenko, Denis Volk, Dmitrii Volkov, and Jeffrey Ladish. Demonstrating specification gaming in reasoning models, 2025. URL https://arxiv.org/abs/2502.13295.

Islem Bouzenia, Premkumar Devanbu, and Michael Pradel. RepairAgent: An Autonomous, LLM-Based Agent for Program Repair . In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*, pp. 2188–2200, Los Alamitos, CA, USA, May 2025. IEEE Computer Society. doi: 10.1109/ICSE55347.2025.00157. URL https://doi.ieeecomputersociety.org/10.1109/ICSE55347.2025.00157.

Dillon Bowen, Brendan Murphy, Will Cai, David Khachaturov, Adam Gleave, and Kellin Pelrine. Scaling trends for data poisoning in llms. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39 (26):27206–27214, Apr. 2025. doi: 10.1609/aaai.v39i26.34929. URL https://ojs.aaai.org/index.php/AAAI/article/view/34929.

Sebastián R. Castro, Roberto Campbell, Nancy Lau, Octavio Villalobos, Jiaqi Duan, and Alvaro A. Cardenas. Large language models are autonomous cyber defenders, 2025. URL https://arxiv.org/abs/2505.04843.

Arjun Chakraborty, Sandra Ho, Adam Cook, and Manuel Meléndez. Cti-realm: Benchmark to evaluate agent performance on security detection rule generation capabilities, 2026. URL https://arxiv.org/abs/2603.13517.

Brian Challita and Pierre Parrend. Redteamllm: an agentic ai framework for offensive security, 2025. URL https://arxiv.org/abs/2505.06913.

Bocheng Chen, Advait Paliwal, and Qiben Yan. Jailbreaker in jail: Moving target defense for large language models. In *Proceedings of the 10th ACM Workshop on Moving Target Defense (MTD'23)*, pp. 29–32, New York, NY, USA, 2023. Association for Computing Machinery. URL https://arxiv.org/abs/2310.02417.

Guoqiang Chen, Huiqi Sun, Daguang Liu, Zhiqi Wang, Qiang Wang, Bin Yin, Lu Liu, and Lingyun Ying. Recopilot: Reverse engineering copilot in binary analysis, 2025a. URL https://arxiv.org/abs/2505.16366.

Jizhou Chen and Samuel Lee Cong. Agentguard: Repurposing agentic orchestrator for safety evaluation of tool orchestration, 2025. URL https://arxiv.org/abs/2502.09809.

Simin Chen, Yixin He, Suman Jana, and Baishakhi Ray. Red teaming program repair agents: When correct patches can hide vulnerabilities, 2025b. URL https://arxiv.org/abs/2509.25894.

Siyi Chen, Tianhan Luo, Shijian Wu, Xiangyu Liu, Yilin Zhou, Qi Li, and Wenyuan Xu. A multi-agent framework for automated exploit generation with constraint-guided comprehension and reflection, 2026. URL https://arxiv.org/abs/2604.05130.

Sizhe Chen, Julien Piet, Chawin Sitawarin, and David Wagner. StruQ: Defending against prompt injection with structured queries. In *34th USENIX Security Symposium (USENIX Security 25)*, Seattle, WA, 2025c. USENIX Association. URL https://arxiv.org/abs/2402.06363.

Sizhe Chen, Arman Zharmagambetov, David Wagner, and Chuan Guo. Meta secalign: A secure foundation llm against prompt injection attacks, 2025d. URL https://arxiv.org/abs/2507.02735.

Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. Agentpoison: Red-teaming LLM agents via poisoning memory or knowledge bases. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=Y841BRW9rY.

Zhaorun Chen, Mintong Kang, and Bo Li. Shieldagent: Shielding agents via verifiable safety policy reasoning, 2025e. URL https://arxiv.org/abs/2503.22738.

Zheng Chen and Buhui Yao. Pseudo-conversation injection for llm goal hijacking, 2024. URL https://arxiv.org/abs/2410.23678.

Anshuman Chhabra, Shrestha Datta, Shahriar Kabir Nahin, and Prasant Mohapatra. Agentic ai security: Threats, defenses, evaluation, and open challenges. *IEEE Access*, 14:49455–49482, 2026. doi: 10.1109/ACCESS.2026.3675554.

Jeffrey Yang Fan Chiang, Seungjae Lee, Jia-Bin Huang, Furong Huang, and Yizheng Chen. Why are web ai agents more vulnerable than standalone llms? a security analysis. In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*, 2025. URL https://openreview.net/forum?id=4KoMbO2RJ9.

Alankrit Chona, Igor Kozlov, and Ambuj Kumar. Cyber defense benchmark: Agentic threat hunting evaluation for llms in secops, 2026. URL https://arxiv.org/abs/2604.19533.

Mihai Christodorescu et al. Systems security foundations for agentic computing (camel), 2025. URL https://arxiv.org/abs/2512.01295.

Manuel Costa, Boris Köpf, Aashish Kolluri, Andrew Paverd, Mark Russinovich, Ahmed Salem, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Béguelin. Securing ai agents with information-flow control, 2025. URL https://arxiv.org/abs/2505.23643.

Maxwell Crouse, Ibrahim Abdelaziz, Ramon Astudillo, Kinjal Basu, Soham Dan, Sadhana Kumaravel, Achille Fokoue, Pavan Kapanipathi, Salim Roukos, and Luis Lastras. Formally specifying the high-level behavior of llm-based agents, 2024. URL https://arxiv.org/abs/2310.08535.

Isaac David and Arthur Gervais. Multi-agent penetration testing ai for the web (mapta), 2025. URL https://arxiv.org/abs/2508.20816.

Christian Schroeder de Witt. Open challenges in multi-agent security: Towards secure systems of interacting ai agents, 2025. URL https://arxiv.org/abs/2505.02077.

Lauren Deason, Adam Bali, Ciprian Bejean, Diana Bolocan, James Crnkovich, Ioana Croitoru, Krishna Durai, Chase Midler, Calin Miron, David Molnar, Brad Moon, Bruno Ostarcevic, Alberto Peltea, Matt Rosenberg, Catalin Sandu, Arthur Saputkin, Sagar Shah, Daniel Stan, Ernest Szocs, Shengye Wan, Spencer Whitman, Sven Krasser, and Joshua Saxe. Cybersoceval: Benchmarking llms capabilities for malware analysis and threat intelligence reasoning, 2025. URL https://arxiv.org/abs/2509.20166.

Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=m1YYAQjO3w.

Edoardo Debenedetti, Ilia Shumailov, Tianqi Fan, Jamie Hayes, Nicholas Carlini, Daniel Fabian, Christoph Kern, Chongyang Shi, Andreas Terzis, and Florian Tramèr. Defeating prompt injections by design, 2025. URL https://arxiv.org/abs/2503.18813.

Gelei Deng, Yi Liu, Yuekang Li, Kailong Wang, Ying Zhang, Zefeng Li, Haoyu Wang, Tianwei Zhang, and Yang Liu. Masterkey: Automated jailbreaking of large language model chatbots. In *Proceedings of the Network and Distributed System Security (NDSS) Symposium*, 2024a. URL https://www.ndss-symposium.org/wp-content/uploads/2024-188-paper.pdf.

Gelei Deng, Yi Liu, Víctor Mayoral-Vilches, Peng Liu, Yuekang Li, Yuan Xu, Tianwei Zhang, Yang Liu, Martin Pinzger, and Stefan Rass. PentestGPT: Evaluating and harnessing large language models for automated penetration testing. In *33rd USENIX Security Symposium (USENIX Security 24)*, pp. 847–864, Philadelphia, PA, August 2024b. USENIX Association. ISBN 978-1-939133-44-1. URL https://www.usenix.org/conference/usenixsecurity24/presentation/deng.

Gelei Deng, Yi Liu, Yuekang Li, Ruozhao Yang, Xiaofei Xie, Jie Zhang, Han Qiu, and Tianwei Zhang. What makes a good llm agent for real-world penetration testing?, 2026a. URL https://arxiv.org/abs/2602.17622.

Xinhao Deng et al. Taming openclaw: Security analysis and mitigation of autonomous llm agent threats, 2026b. URL https://arxiv.org/abs/2603.11619.

Yinlin Deng, Chunqiu Steven Xia, Haoran Peng, Chenyuan Yang, and Lingming Zhang. Large language models are zero-shot fuzzers: Fuzzing deep-learning libraries via large language models. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2023, pp. 423–435, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702211. doi: 10.1145/3597926.3598067. URL https://doi.org/10.1145/3597926.3598067.

Yinlin Deng, Chunqiu Steven Xia, Chenyuan Yang, Shizhuo Dylan Zhang, Shujing Yang, and Lingming Zhang. Large language models are edge-case generators: Crafting unusual programs for fuzzing deep learning libraries. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ICSE '24, New York, NY, USA, 2024c. Association for Computing Machinery. ISBN 9798400702174. doi: 10.1145/3597503.3623343. URL https://doi.org/10.1145/3597503.3623343.

Zehang Deng, Yongjian Guo, Changzhou Han, Wanlun Ma, Junwu Xiong, Sheng Wen, and Yang Xiang. Ai agents under threat: A survey of key security challenges and future pathways, 2024d. URL https://arxiv.org/abs/2406.02630.

Alaeddine Diaf, Abdelaziz Amara Korba, Nour Elislem Karabadji, and Yacine Ghamri-Doudane. Bartpredict: Empowering iot security with llm-driven cyber threat prediction, 2025. URL https://arxiv.org/abs/2501.01664.

Shen Dong, Shaochen Xu, Pengfei He, Yige Li, Jiliang Tang, Tianming Liu, Hui Liu, and Zhen Xiang. Memory injection attacks on LLM agents via query-only interaction. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=QINnsnppv8.

Shen Dong, Shaochen Xu, Pengfei He, Yige Li, Jiliang Tang, Tianming Liu, Hui Liu, and Zhen Xiang. Memory injection attacks on llm agents via query-only interaction. *Advances in Neural Information Processing Systems*, 38:46697–46731, 2026.

Aarya Doshi, Yining Hong, Congying Xu, Eunsuk Kang, Alexandros Kapravelos, and Christian Kästner. Towards verifiably safe tool use for llm agents, 2026. URL https://arxiv.org/abs/2601.08012.

Ivan Evtimov, Arman Zharmagambetov, Aaron Grattafiori, Chuan Guo, and Kamalika Chaudhuri. Wasp: Benchmarking web agent security against prompt injection attacks. In *Advances in Neural Information Processing Systems*, 2026.

Mohamad Fakih, Rahul Dharmaji, Halima Bouzidi, Gustavo Quiros Araya, Oluwatosin Ogundare, and Mohammad Abdullah Al Faruque. Llm4cve: Enabling iterative automated vulnerability repair with large language models, 2025. URL https://arxiv.org/abs/2501.03446.

Richard Fang, Rohan Bindu, Akul Gupta, and Daniel Kang. Llm agents can autonomously exploit one-day vulnerabilities, 2024. URL https://arxiv.org/abs/2404.08144.

Neil Fendley, Edward W. Staley, Joshua Carney, William Redman, Marie Chau, and Nathan Drenkow. A systematic review of poisoning attacks against large language models, 2025. URL https://arxiv.org/abs/2506.06518.

Jiayi Fu, Xuandong Zhao, Chengyuan Yao, Heng Wang, Qi Han, and Yanghua Xiao. Reward shaping to mitigate reward hacking in rlhf, 2025a. URL https://arxiv.org/abs/2502.18770.

Tingchen Fu, Mrinank Sharma, Philip Torr, Shay B Cohen, David Krueger, and Fazl Barez. Poisonbench: Assessing large language model vulnerability to poisoned preference data. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=21kAulloDG.

Xiaohan Fu, Shuheng Li, Zihan Wang, Yihao Liu, Rajesh K. Gupta, Taylor Berg-Kirkpatrick, and Earlence Fernandes. Imprompter: Tricking llm agents into improper tool use, 2024. URL https://arxiv.org/abs/2410.14923.

Yuchuan Fu, Xiaohan Yuan, and Dongxia Wang. Ras-eval: A comprehensive benchmark for security evaluation of llm agents in real-world environments, 2025c. URL https://arxiv.org/abs/2506.15253.

Stefano Fumero, Kai Huang, Matteo Boffa, Danilo Giordano, Marco Mellia, Zied Ben Houidi, and Dario Rossi. Cybersleuth: Autonomous blue-team llm agent for web attack forensics, 2025. URL https://arxiv.org/abs/2508.20643.

Yuyou Gan, Yong Yang, Zhe Ma, Ping He, Rui Zeng, Yiming Wang, Qingming Li, Chunyi Zhou, Songze Li, Ting Wang, Yunjun Gao, Yingcai Wu, and Shouling Ji. Navigating the risks: A survey of security, privacy, and ethics threats in llm-based agents, 2024. URL https://arxiv.org/abs/2411.09523.

Yiran Gao, Kim Hammar, and Tao Li. In-context autonomous network incident response: An end-to-end large language model agent approach, 2026. URL https://arxiv.org/abs/2602.13156.

Suyu Ge, Chunting Zhou, Rui Hou, Madian Khabsa, Yi-Chia Wang, Qifan Wang, Jiawei Han, and Yuning Mao. Mart: Improving llm safety with multi-round automatic red-teaming, 2023. URL https://arxiv.org/abs/2311.07689.

Arthur Gervais and Liyi Zhou. Ai agent smart contract exploit generation, 2026. URL https://arxiv.org/abs/2507.05558.

Yasod Ginige, Akila Niroshan, Sajal Jain, and Suranga Seneviratne. Autopentester: An llm agent-based framework for automated pentesting. In *2025 IEEE 24th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pp. 163–174, 2025. doi: 10.1109/Trustcom66490.2025.00026.

Luca Gioacchini, Marco Mellia, Idilio Drago, Alexander Delsanto, Giuseppe Siracusano, and Roberto Bifulco. Autopenbench: Benchmarking generative agents for penetration testing, 2024. URL https://arxiv.org/abs/2410.03225.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, AISec '23, pp. 79–90, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702600. doi: 10.1145/3605764.3623985. URL https://doi.org/10.1145/3605764.3623985.

Jinyao Guo, Chengpeng Wang, Xiangzhe Xu, Zian Su, and Xiangyu Zhang. Repoaudit: An autonomous LLM-agent for repository-level code auditing. In *Forty-second International Conference on Machine Learning*, 2025a. URL https://openreview.net/forum?id=TXcifVbFpG.

Qiming Guo, Jinwen Tang, and Xingran Huang. Attacking llms and ai agents: Advertisement embedding attacks against large language models, 2025b. URL https://arxiv.org/abs/2508.17674.

Shanshan Han, Qifan Zhang, Yuhang Yao, Weizhao Jin, and Zhaozhuo Xu. Llm multi-agent systems: Challenges and open problems, 2025. URL https://arxiv.org/abs/2402.03578.

Soham Hans, Stacy Marsella, Sophia Hirschmann, and Nikolos Gurney. Security logs to att&ck insights: Leveraging llms for high-level threat understanding and cognitive trait inference, 2025. URL https://arxiv.org/abs/2510.20930.

Andreas Happe and Jürgen Cito. Can llms hack enterprise networks? autonomous assumed breach penetration-testing active directory networks. *ACM Trans. Softw. Eng. Methodol.*, September 2025a. ISSN 1049-331X. doi: 10.1145/3766895. URL https://doi.org/10.1145/3766895. Just Accepted.

Andreas Happe and Jürgen Cito. On the surprising efficacy of llms for penetration-testing, 2025b. URL https://arxiv.org/abs/2507.00829.

Pengfei He, Yuping Lin, Shen Dong, Han Xu, Yue Xing, and Hui Liu. Red-teaming LLM multi-agent systems via communication attacks. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 6726–6747, Vienna, Austria, July 2025a. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.349. URL https://aclanthology.org/2025.findings-acl.349/.

Pengfei He, Ash Fox, Lesly Miculicich, Stefan Friedli, Daniel Fabian, Burak Gokturk, Jiliang Tang, Chen-Yu Lee, Tomas Pfister, and Long T. Le. Co-redteam: Orchestrated security discovery and exploitation with llm agents, 2026. URL https://arxiv.org/abs/2602.02164.

Xu He, Di Wu, Yan Zhai, and Kun Sun. Sentinelagent: Graph-based anomaly detection in multi-agent systems, 2025b. URL https://arxiv.org/abs/2505.24201.

Yifeng He, Ethan Wang, Yuyang Rong, Zifei Cheng, and Hao Chen. Security of ai agents, 2024. URL https://arxiv.org/abs/2406.08689.

Julius Henke. Autopentest: Enhancing vulnerability management with autonomous llm agents, 2025. URL https://arxiv.org/abs/2505.10321.

Keegan Hines, Gary Lopez, Matthew Hall, Federico Zarfati, Yonatan Zunger, and Emre Kiciman. Defending against indirect prompt injection attacks with spotlighting. In *Proceedings of the Conference on Applied Machine Learning in Information Security (CAMLIS 2024)*, volume 3920 of *CEUR Workshop Proceedings*, pp. 48–62, 2024. URL https://arxiv.org/abs/2403.14720.

Jinwei HU, Yi DONG, Zhengtao DING, and Xiaowei HUANG. Enhancing robustness of llm-driven multi-agent systems through randomized smoothing. *Chinese Journal of Aeronautics*, pp. 103779, 2025. ISSN 1000-9361. doi: https://doi.org/10.1016/j.cja.2025.103779. URL https://www.sciencedirect.com/science/article/pii/S1000936125003851.

Xueyu Hu, Tao Xiong, Biao Yi, Zishu Wei, Ruixuan Xiao, Yurun Chen, Jiasheng Ye, Meiling Tao, Xiangxin Zhou, Ziyu Zhao, Yuhuai Li, Shengze Xu, Shenzhi Wang, Xinchen Xu, Shuofei Qiao, Zhaokai Wang, Kun Kuang, Tieyong Zeng, Liang Wang, Jiwei Li, Yuchen Eleanor Jiang, Wangchunshu Zhou, Guoyin Wang, Keting Yin, Zhou Zhao, Hongxia Yang, Fan Wu, Shengyu Zhang, and Fei Wu. Os agents: A survey on mllm-based agents for computer, phone and browser use, 2025. URL https://arxiv.org/abs/2508.04482.

Junjie Huang and Quanyan Zhu. Penheal: A two-stage llm framework for automated pentesting and optimal remediation. In *Proceedings of the Workshop on Autonomous Cybersecurity*, AutonomousCyber '24, pp. 11–22, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400712296. doi: 10.1145/3689933.3690831. URL https://doi.org/10.1145/3689933.3690831.

Lanxiao Huang, Daksh Dave, Tyler Cody, Peter A. Beling, and Ming Jin. From capabilities to performance: Evaluating key functional properties of LLM architectures in penetration testing. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 15879–15905, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.802. URL https://aclanthology.org/2025.emnlp-main.802/.

Evan Hubinger, Carson Denison, Jesse Mu, Albert Webson, Jared Kaplan, Christopher Olah, Ethan Perez, and Neil Hubert. Sleeper agents: Training deceptive llms that persist through safety training, 2024. URL https://arxiv.org/abs/2401.05566.

Robert Hutter and Michael Pradel. Agentstepper: Interactive debugging of software development agents, 2026. URL https://arxiv.org/abs/2602.06593.

Isamu Isozaki, Manil Shrestha, Rick Console, and Edward Kim. Towards automated penetration testing: Introducing llm benchmark, analysis, and improvements, 2024. URL https://arxiv.org/abs/2410.17141.

Jiangan Ji, Chao Zhang, Shuitao Gan, Lin Jian, Hangtian Liu, Tieming Liu, Lei Zheng, and Zhipeng Jia. Firmagent: Leveraging fuzzing to assist llm agents with iot firmware vulnerability discovery. In *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2026. doi: 10.14722/ndss.2026.231943. URL https://www.ndss-symposium.org/ndss-paper/firmagent-leveraging-fuzzing-to-assist-llm-agents-with-iot-firmware-vulnerability-discovery/.

Feiran Jia, Tong Wu, Xin Qin, and Anna Squicciarini. The task shield: Enforcing task alignment to defend against indirect prompt injection in LLM agents. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 29680–29697, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1435. URL https://aclanthology.org/2025.acl-long.1435/.

Changyue Jiang, Xudong Pan, Geng Hong, Chenfu Bao, and Min Yang. RAG-Thief: Scalable extraction of private data from retrieval-augmented generation applications with agent-based attacks, 2024. URL https://arxiv.org/abs/2411.14110.

Yongrae Jo and Chanik Park. Byzantine-robust decentralized coordination of llm agents, 2025. URL https://arxiv.org/abs/2507.14928.

Sam Johnson, Viet Pham, and Thai Le. Manipulating llm web agents with indirect prompt injection attack via html accessibility tree. *arXiv preprint arXiv:2507.14799*, 2025.

Mintong Kang and Bo Li. r^2-guard: Robust reasoning enabled llm guardrail via knowledge-enhanced logical reasoning, 2024. URL https://arxiv.org/abs/2407.05557.

Jun Kevin and Pujianto Yugopuspito. Smartllm: Smart contract auditing using custom generative ai, 2025. URL https://arxiv.org/abs/2502.13167.

Juhee Kim, Woohyuk Choi, and Byoungyoung Lee. Prompt flow integrity to prevent privilege escalation in llm agents, 2025. URL https://arxiv.org/abs/2503.15547.

Dezhang Kong, Shi Lin, Zhenhua Xu, Zhebo Wang, Minghao Li, Yufeng Li, Yilun Zhang, Hujin Peng, Zeyang Sha, Yuyuan Li, Changting Lin, Xun Wang, Xuan Liu, Ningyu Zhang, Chaochao Chen, Muhammad Khurram Khan, and Meng Han. A survey of llm-driven ai agent communication: Protocols, security risks, and defense countermeasures, 2025a. URL https://arxiv.org/abs/2506.19676.

He Kong, Die Hu, Jingguo Ge, Liangxiong Li, Tong Li, and Bingzhen Wu. Vulnbot: Autonomous penetration testing for a multi-agent collaborative framework, 2025b. URL https://arxiv.org/abs/2501.13411.

Roham Koohestani. AgentGuard: Runtime verification of AI agents, 2025. URL https://arxiv.org/abs/2509.23864. Accepted at ASE 2025 Workshop on Agentic Software Engineering (AgenticSE).

Panagiotis Kouvaros, Alessio Lomuscio, Edoardo Pirovano, and Hashan Punchihewa. Formal verification of open multi-agent systems. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '19, pp. 179–187, Richland, SC, 2019. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450363099.

Priyanshu Kumar, Elaine Lau, Saranya Vijayakumar, Tu Trinh, Elaine T Chang, Vaughn Robinson, Shuyan Zhou, Matt Fredrikson, Sean M. Hendryx, Summer Yue, and Zifan Wang. Aligned LLMs are not aligned browser agents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=NsFZZU9gvk.

Nancy Lau, Louis Sloot, Jyoutir Raj, Giuseppe Marco Boscardin, Evan Harris, Dylan Bowman, Mario Brajkovski, Jaideep Chawla, and Dan Zhao. Zerodaybench: Evaluating llm agents on unseen zero-day vulnerabilities for cyberdefense, 2026. URL https://arxiv.org/abs/2603.02297.

Christine P. Lee, David Porfirio, Xinyu Jessica Wang, Kevin Chenkai Zhao, and Bilge Mutlu. Veriplan: Integrating formal verification and llms into end-user planning. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, CHI '25, New York, NY, USA, 2025a. Association for Computing Machinery. ISBN 9798400713941. doi: 10.1145/3706598.3714113. URL https://doi.org/10.1145/3706598.3714113.

Donghyun Lee and Mo Tiwari. Prompt infection: Llm-to-llm prompt injection within multi-agent systems, 2024. URL https://arxiv.org/abs/2410.07283.

Hwiwon Lee, Ziqi Zhang, Hanxiao Lu, and Lingming Zhang. Sec-bench: Automated benchmarking of llm agents on real-world software security tasks, 2025b. URL https://arxiv.org/abs/2506.11791.

Ido Levy, Ben Wiesel, Sami Marreed, Alon Oved, Avi Yaeli, and Segev Shlomov. ST-WebAgentBench: A benchmark for evaluating safety & trustworthiness in web agents. In *ArXiv*, 2025. arXiv:2410.06703.

Ang Li, Yin Zhou, Vethavikashini Chithrra Raghuram, Tom Goldstein, and Micah Goldblum. Commercial llm agents are already vulnerable to simple yet dangerous attacks, 2025a. URL https://arxiv.org/abs/2502.08586.

Evan Li, Tushin Mallick, Evan Rose, William Robertson, Alina Oprea, and Cristina Nita-Rotaru. Ace: A security architecture for llm-integrated app systems, 2025b. URL https://arxiv.org/abs/2504.20984.

Simin Li, Jun Guo, Jingqiao Xiu, Ruixiao Xu, Xin Yu, Jiakai Wang, Aishan Liu, Yaodong Yang, and Xianglong Liu. Byzantine robust cooperative multi-agent reinforcement learning as a bayesian game. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=z6KS9D1dxt.

Weizhen Li, Jianbo Lin, Zhuosong Jiang, Jingyi Cao, Xinpeng Liu, Jiayu Zhang, Zhenqiang Huang, Qianben Chen, Weichen Sun, Qiexiang Wang, Hongxuan Lu, Tianrui Qin, Chenghao Zhu, Yi Yao, Shuying Fan, Xiaowan Li, Tiannan Wang, Pai Liu, King Zhu, He Zhu, Dingfeng Shi, Piaohong Wang, Yeyi Guan, Xiangru Tang, Minghao Liu, Yuchen Eleanor Jiang, Jian Yang, Jiaheng Liu, Ge Zhang, and Wangchunshu Zhou. Chain-of-agents: End-to-end agent foundation models via multi-agent distillation and agentic rl, 2025c. URL https://arxiv.org/abs/2508.13167.

Wenhao Li, Selvakumar Manickam, Yung wey Chong, and Shankar Karuppayah. Phishdebate: An llm-based multi-agent framework for phishing website detection, 2025d. URL https://arxiv.org/abs/2506.15656.

Xiaofan Li and Xing Gao. A first look at the security issues in the model context protocol ecosystem. In *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2026. URL https://arxiv.org/abs/2510.16558.

Yanjie Li, Zhen Xiang, Nathaniel D. Bastian, Dawn Song, and Bo Li. IDS-agent: An LLM agent for explainable intrusion detection in iot networks. In *NeurIPS 2024 Workshop on Open-World Agents*, 2024b. URL https://openreview.net/forum?id=iiK0pRyLkw.

Ziyang Li, Saikat Dutta, and Mayur Naik. IRIS: LLM-assisted static analysis for detecting security vulnerabilities. In *The Thirteenth International Conference on Learning Representations*, 2025e. URL https://openreview.net/forum?id=9LdJDU7E91.

Justin W. Lin, Eliot Krzysztof Jones, Donovan Julian Jasper, Ethan Jun shen Ho, Anna Wu, Arnold Tianyi Yang, Neil Perry, Andy Zou, Matt Fredrikson, J. Zico Kolter, Percy Liang, Dan Boneh, and Daniel E. Ho. Comparing ai agents to cybersecurity professionals in real-world penetration testing, 2026. URL https://arxiv.org/abs/2512.09882.

Liang Lin, Zhihao Xu, Xuehai Tang, Shi Liu, Biyu Zhou, Fuqing Zhu, Jizhong Han, and Songlin Hu. Paper summary attack: Jailbreaking llms through llm safety papers, 2025a. URL https://arxiv.org/abs/2507.13474.

Xihuan Lin, Jie Zhang, Gelei Deng, Tianzhe Liu, Xiaolong Liu, Changcai Yang, Tianwei Zhang, Qing Guo, and Riqing Chen. Ircopilot: Automated incident response with large language models, 2025b. URL https://arxiv.org/abs/2505.20945.

Bin Liu, Yanjie Zhao, Guoai Xu, and Haoyu Wang. Llm agents for automated web vulnerability reproduction: Are we there yet?, 2025a. URL https://arxiv.org/abs/2510.14700.

Fengyu Liu, Yuan Zhang, Jiaqi Luo, Jiarun Dai, Tian Chen, Letian Yuan, Zhengmin Yu, Youkun Shi, Ke Li, Chengyuan Zhou, Hao Chen, and Min Yang. Make agent defeat agent: automatic detection of taint-style vulnerabilities in llm-based agents. In *Proceedings of the 34th USENIX Conference on Security Symposium*, SEC '25, USA, 2025b. USENIX Association. ISBN 978-1-939133-52-6.

Fengyu Liu, Yuan Zhang, Jiaqi Luo, Jiarun Dai, Tian Chen, Letian Yuan, Zhengmin Yu, Youkun Shi, Ke Li, Chengyuan Zhou, Hao Chen, and Min Yang. Make agent defeat agent: Automatic detection of Taint-Style vulnerabilities in LLM-based agents. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 3767–3786, Seattle, WA, 2025c. USENIX Association. ISBN 978-1-939133-52-6. URL https://www.usenix.org/conference/usenixsecurity25/presentation/liu-fengyu.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. AutoDAN: Generating stealthy jailbreak prompts on aligned large language models. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=7Jwpw4qKkb.

Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Zihao Wang, Xiaofeng Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yan Zheng, and Yang Liu. Prompt injection attack against llm-integrated applications, 2024b. URL https://arxiv.org/abs/2306.05499.

Yinqiu Liu, Ruichen Zhang, Haoxiang Luo, Yijing Lin, Geng Sun, Dusit Niyato, Hongyang Du, Zehui Xiong, Yonggang Wen, Abbas Jamalipour, Dong In Kim, and Ping Zhang. Secure multi-llm agentic ai and agentification for edge general intelligence by zero-trust: A survey, 2025d. URL https://arxiv.org/abs/2508.19870.

Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. Formalizing and benchmarking prompt injection attacks and defenses. In *Proceedings of the 33rd USENIX Conference on Security Symposium*, SEC '24, USA, 2024c. USENIX Association. ISBN 978-1-939133-44-1.

Zefang Liu. Autobnb: Multi-agent incident response with large language models. In *2025 13th International Symposium on Digital Forensics and Security (ISDFS)*, pp. 1–6, 2025. doi: 10.1109/ISDFS65363.2025.11012055.

Jiaqi Luo, Jiarun Dai, Fengyu Liu, Songyang Peng, Youkun Shi, Tong Bu, Geng Hong, Xudong Pan, and Yuan Zhang. Autonomy comes with costs: Detecting denial-of-service vulnerabilities caused by resource abusing in llm-based agents. In *35th USENIX Security Symposium (USENIX Security 26)*, 2026.

Weidi Luo, Shenghong Dai, Xiaogeng Liu, Suman Banerjee, Huan Sun, Muhao Chen, and Chaowei Xiao. Agrail: A lifelong agent guardrail with effective and adaptive safety detection, 2025. URL https://arxiv.org/abs/2502.11448.

Phung Duc Luong, Le Tran Gia Bao, Nguyen Vu Khai Tam, Dong Huu Nguyen Khoa, Nguyen Huu Quyen, Van-Hau Pham, and Phan The Duy. xoffense: An ai-driven autonomous penetration testing framework with offensive knowledge-enhanced llms and multi agent systems, 2025. URL https://arxiv.org/abs/2509.13021.

Matteo Lupinacci, Francesco Aurelio Pironti, Francesco Blefari, Francesco Romeo, Luigi Arena, and Angelo Furfaro. The dark side of llms: Agent-based attacks for complete computer takeover, 2025. URL https://arxiv.org/abs/2507.06850.

Wei Ma, Daoyuan Wu, Yuqiang Sun, Tianwen Wang, Shangqing Liu, Jian Zhang, Yue Xue, and Yang Liu. Combining fine-tuning and llm-based agents for intuitive smart contract auditing with justifications, 2024. URL https://arxiv.org/abs/2403.16073.

Xingjun Ma, Yifeng Gao, Yixu Wang, Ruofan Wang, Xin Wang, Ye Sun, Yifan Ding, Hengyuan Xu, Yunhao Chen, Yunhao Zhao, Hanxun Huang, Yige Li, Yutao Wu, Jiaming Zhang, Xiang Zheng, Yang Bai, Yiming Li, Zuxuan Wu, Xipeng Qiu, Jingfeng Zhang, Xudong Han, Haonan Li, Jun Sun, Cong Wang, Jindong Gu, Baoyuan Wu, Siheng Chen, Tianwei Zhang, Yang Liu, Mingming Gong, Tongliang Liu, Shirui Pan, Cihang Xie, Tianyu Pang, Yinpeng Dong, Ruoxi Jia, Yang Zhang, Shiqing Ma, Xiangyu Zhang, Neil Gong, Chaowei Xiao, Sarah Erfani, Tim Baldwin, Bo Li, Masashi Sugiyama, Dacheng Tao, James Bailey, and Yu-Gang Jiang. Safety at scale: A comprehensive survey of large model and agent safety. *Foundations and Trends® in Privacy and Security*, 8(3-4):254–469, 2025. ISSN 2474-1558. doi: 10.1561/3300000051. URL http://dx.doi.org/10.1561/3300000051.

Víctor Mayoral-Vilches, Luis Javier Navarrete-Lozano, María Sanz-Gómez, Lidia Salas Espejo, Martiño Crespo-Álvarez, Francisco Oca-Gonzalez, Francesco Balassone, Alfonso Glera-Picón, Unai Ayucar-Carbajo, Jon Ander Ruiz-Alcalde, Stefan Rass, Martin Pinzger, and Endika Gil-Uriarte. Cai: An open, bug bounty-ready cybersecurity ai, 2025. URL https://arxiv.org/abs/2504.06017.

Kai Mei, Xi Zhu, Wujiang Xu, Wenyue Hua, Mingyu Jin, Zelong Li, Shuyuan Xu, Ruosong Ye, Yingqiang Ge, and Yongfeng Zhang. Aios: Llm agent operating system. In *Conference on Language Modeling*, 2025. URL https://arxiv.org/pdf/2403.16971.

Ruijie Meng, Martin Mirchev, Marcel Böhme, and Abhik Roychoudhury. Large language model guided protocol fuzzing. In *Proceedings of the 31st Annual Network and Distributed System Security Symposium (NDSS)*, 2024.

Yuqiao Meng, Luoxi Tang, Feiyang Yu, Jinyuan Jia, Guanhua Yan, Ping Yang, and Zhaohan Xi. Uncovering vulnerabilities of llm-assisted cyber threat intelligence, 2025a. URL https://arxiv.org/abs/2509.23573.

Yuqiao Meng, Luoxi Tang, Feiyang Yu, Xi Li, Guanhua Yan, Ping Yang, and Zhaohan Xi. Benchmarking llm-assisted blue teaming via standardized threat hunting, 2025b. URL https://arxiv.org/abs/2509.23571.

Yuchun Miao, Sen Zhang, Liang Ding, Rong Bao, Lefei Zhang, and Dacheng Tao. Inform: Mitigating reward hacking in rlhf via information-theoretic reward modeling, 2024. URL https://arxiv.org/abs/2402.09345.

Ivan Milev, Mislav Balunović, Maximilian Baader, and Martin Vechev. Toolfuzz – automated agent tool testing, 2025. URL https://arxiv.org/abs/2503.04479.

Sumeet Ramesh Motwani, Mikhail Baranchuk, Martin Strohmeier, Vijay Bolina, Philip H.S. Torr, Lewis Hammond, and Christian Schroeder de Witt. Secret collusion among generative AI agents: Multi-agent deception via steganography. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://arxiv.org/abs/2402.07510.

Mykyta Mudryi, Markiyan Chaklosh, and Grzegorz Wójcik. The hidden dangers of browsing ai agents, 2025. URL https://arxiv.org/abs/2505.13076.

Kunal Mukherjee and Murat Kantarcioglu. Llm-driven provenance forensics for threat investigation and detection, 2025. URL https://arxiv.org/abs/2508.21323.

Ruoyu Muller, Ruoyu Song, Chang Wang, Yong Zhan, and J. Paul. Enhancing llm-based autonomous driving agents to mitigate perception attacks, 2024. URL https://arxiv.org/abs/2409.14488.

Lajos Muzsai, David Imolai, and András Lukács. Hacksynth: Llm agent and evaluation framework for autonomous penetration testing, 2024. URL https://arxiv.org/abs/2412.01778.

Sho Nakatani. Rapidpen: Fully automated ip-to-shell penetration testing with llm-based agents, 2026. URL https://arxiv.org/abs/2502.16730.

Vineeth Sai Narajala and Om Narayan. Securing agentic ai: A comprehensive threat model and mitigation framework for generative ai agents, 2025. URL https://arxiv.org/abs/2504.19956.

Subash Neupane, Shaswata Mitra, Sudip Mittal, and Shahram Rahimi. Towards a hipaa compliant agentic ai system in healthcare, 2025. URL https://arxiv.org/abs/2504.17669.

Tomas Nieponice, Veronica Valeros, and Sebastian Garcia. Aracne: An llm-based autonomous shell pentesting agent, 2025. URL https://arxiv.org/abs/2502.18528.

Alexander Pan, Kush Bhatia, and Jacob Steinhardt. The effects of reward misspecification: Mapping and mitigating misaligned models. In *Deep RL Workshop NeurIPS 2021*, 2021. URL https://openreview.net/forum?id=mp1AstNFvQ5.

Melissa Z Pan, Mert Cemri, Lakshya A Agrawal, Shuyi Yang, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Kannan Ramchandran, Dan Klein, Joseph E. Gonzalez, Matei Zaharia, and Ion Stoica. Why do multiagent systems fail? In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*, 2025. URL https://openreview.net/forum?id=wM521FqPvI.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

Dario Pasquini, Evgenios M Kornaropoulos, Giuseppe Ateniese, Omer Akgul, Athanasios Theocharis, and Petros Efstathopoulos. When aiops become" ai oops": Subverting llm-driven it operations via telemetry manipulation. *arXiv preprint arXiv:2508.06394*, 2025.

Rodrigo Pedro, Miguel E. Coimbra, Daniel Castro, Paulo Carreira, and Nuno Santos. Prompt-to-sql injections in llm-integrated web applications: Risks and defenses. In *Proceedings of the IEEE/ACM 47th International Conference on Software Engineering*, ICSE '25, pp. 1768–1780. IEEE Press, 2025. ISBN 9798331505691. doi: 10.1109/ICSE55347.2025.00007. URL https://doi.org/10.1109/ICSE55347.2025.00007.

Jinjun Peng, Leyi Cui, Kele Huang, Junfeng Yang, and Baishakhi Ray. Cweval: Outcome-driven evaluation on functionality and security of llm code generation, 2025. URL https://arxiv.org/abs/2501.08200.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models, 2022. URL https://arxiv.org/abs/2202.03286.

Fábio Perez and Ian Ribeiro. Ignore previous prompt: Attack techniques for language models, 2022. URL https://arxiv.org/abs/2211.09527.

Viet Pham and Thai Le. Cain: Hijacking llm-humans conversations via malicious system prompts, 2025. URL https://arxiv.org/abs/2505.16888.

Lukas Pirch, Micha Horlboge, Patrick Großmann, Syeda Mahnur Asif, Klim Kireev, Thorsten Holz, and Konrad Rieck. Toward securing ai agents like operating systems, 2026. URL https://arxiv.org/abs/2605.14932.

Benjamin Probst, Andreas Happe, and Jürgen Cito. Enhancing linux privilege escalation attack capabilities of local llm agents, 2026. URL https://arxiv.org/abs/2604.27143.

Yubin Qu, Yi Liu, Tongcheng Geng, Gelei Deng, Yuekang Li, Leo Yu Zhang, Ying Zhang, and Lei Ma. Supply-chain poisoning attacks against llm coding agent skill ecosystems, 2026. URL https://arxiv.org/abs/2604.03081.

Rafiqul Rabin, Jesse Hostetler, Sean McGregor, Brett Weir, and Nick Judd. Sandboxeval: Towards securing test environment for untrusted code, 2025. URL https://arxiv.org/abs/2504.00018.

Melissa Kazemi Rad, Huy Nghiem, Sahil Wadhwa, Andy Luo, and Mohammad Shahed Sorower. Refining input guardrails: Enhancing LLM-as-a-judge efficiency through chain-of-thought fine-tuning and alignment. In *AAAI 2025 Workshop on Preventing and Detecting LLM Misinformation (PDLM)*, 2025. URL https://openreview.net/forum?id=UNPzbCKovl.

Salman Rahman, Liwei Jiang, James Shiffer, Genglin Liu, Sheriff Issaka, Md Rizwan Parvez, Hamid Palangi, Kai-Wei Chang, Yejin Choi, and Saadia Gabriel. X-teaming: Multi-turn jailbreaks and defenses with adaptive multi-agents. *arXiv preprint arXiv:2504.13203*, 2025.

Shaina Raza, Ranjan Sapkota, Manoj Karkee, and Christos Emmanouilidis. Trism for agentic ai: A review of trust, risk, and security management in llm-based agentic multi-agent systems, 2025. URL https://arxiv.org/abs/2506.04133.

Pavan Reddy and Aditya Sanjay Gujral. Echoleak: The first real-world zero-click prompt injection exploit in a production llm system, 2025. URL https://arxiv.org/abs/2509.10540.

Xiaoxue Ren, Penghao Jiang, Kaixin Li, Zhiyong Huang, Xiaoning Du, Jiaojiao Jiang, Zhenchang Xing, Jiamou Sun, and Terry Yue Zhuo. Hackworld: Evaluating computer-use agents on exploiting web application vulnerabilities. *arXiv preprint arXiv:2510.12200*, 2025.

Alexander Robey, Zachary Ravichandran, Vijay Kumar, Hamed Hassani, and George J. Pappas. Jailbreaking llm-controlled robots, 2024. URL https://arxiv.org/abs/2410.13691.

Ron F. Del Rosario, Klaudia Krawiecka, and Christian Schroeder de Witt. Architecting resilient llm agents: A guide to secure plan-then-execute implementations, 2025. URL https://arxiv.org/abs/2509.08646.

Bikash Saha and Sandeep Kumar Shukla. Malgen: A generative agent framework for modeling malicious software in cybersecurity, 2025. URL https://arxiv.org/abs/2506.07586.

Shoumik Saha, Jifan Chen, Sam Mayers, Sanjay Krishna Gouda, Zijian Wang, and Varun Kumar. Breaking the code: Security assessment of ai code agents through systematic jailbreaking attacks, 2025. URL https://arxiv.org/abs/2510.01359.

Rishikesh Sahay, Bell Eapen, Weizhi Meng, Md Rasel Al Mamun, Nikhil Kumar Dora, Manjusha Sumasadan, Sumit Kumar Tetarave, and Rod Soto. Policy-guided threat hunting: An llm-enabled framework with splunk soc triage, 2026. URL https://arxiv.org/abs/2603.23966.

Abhijeet Sahu, Shuva Paul, and Richard Macwan. Network and device level cyber deception for contested environments using rl and llms, 2026. URL https://arxiv.org/abs/2603.17272.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=Yacmpz84TH.

David Schmotz, Luca Beurer-Kellner, Sahar Abdelnabi, and Maksym Andriushchenko. Skill-Inject: Measuring agent vulnerability to skill file attacks, 2026. URL https://arxiv.org/abs/2602.20156.

Yuval Schwartz, Lavi Ben-Shimol, Dudu Mimran, Yuval Elovici, and Asaf Shabtai. Llmcloudhunter: Harnessing llms for automated extraction of detection rules from cloud-based cti. In *Proceedings of the ACM on Web Conference 2025*, WWW '25, pp. 1922–1941, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400712746. doi: 10.1145/3696410.3714798. URL https://doi.org/10.1145/3696410.3714798.

Zedian Shao, Hongbin Liu, Jaden Mu, and Neil Gong. Enhancing prompt injection attacks to llms via poisoning alignment. In *Proceedings of the 18th ACM Workshop on Artificial Intelligence and Security*, pp. 13–27, 2025.

Mohammed A. Shehab. Agentic-ai healthcare: Multilingual, privacy-first framework with mcp agents. *arXiv preprint*, arXiv:2510.02325, 2025. URL https://arxiv.org/abs/2510.02325. preprint, submitted 25 Sep 2025, cs.CR / cs.AI.

Chihao Shen, Connor Dilgren, Purva Chiniya, Luke Griffith, Yu Ding, and Yizheng Chen. Secrepobench: Benchmarking code agents for secure code completion in real-world repositories, 2026. URL https://arxiv.org/abs/2504.21205.

Xiangmin Shen, Lingzhi Wang, Zhenyuan Li, Yan Chen, Wencheng Zhao, Dawei Sun, Jiashui Wang, and Wei Ruan. Pentestagent: Incorporating llm agents to automated penetration testing. In *Proceedings of the 20th ACM Asia Conference on Computer and Communications Security*, ASIA CCS '25, pp. 375–391, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400714108. doi: 10.1145/3708821.3733882. URL https://doi.org/10.1145/3708821.3733882.

Jiawen Shi, Zenghui Yuan, Guiyao Tie, Pan Zhou, Neil Zhenqiang Gong, and Lichao Sun. Prompt injection attack to tool selection in llm agents. In *Proceedings of the Network and Distributed System Security (NDSS) Symposium 2026*, 2026. doi: 10.14722/ndss.2026.230675. URL https://dx.doi.org/10.14722/ndss.2026.230675.

Tianneng Shi, Jingxuan He, Zhun Wang, Linyu Wu, Hongwei Li, Wenbo Guo, and Dawn Song. Progent: Programmable privilege control for llm agents, 2025. URL https://arxiv.org/abs/2504.11703.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

Abdullah Al Siam and Sadequzzaman Shohan. Privacy-preserving ai for encrypted medical imaging: A framework for secure diagnosis and learning, 2025. URL https://arxiv.org/abs/2507.21060.

Brian Singer, Keane Lucas, Lakshmi Adiga, Meghna Jain, Lujo Bauer, and Vyas Sekar. Incalmo: An autonomous llm-assisted system for red teaming multi-host networks, 2025. URL https://arxiv.org/abs/2501.16466.

Ronal Singh, Shahroz Tariq, Fatemeh Jalalvand, Mohan Baruwal Chhetri, Surya Nepal, Cecile Paris, and Martin Lochner. Llms in the soc: An empirical study of human-ai collaboration in security operations centres, 2025. URL https://arxiv.org/abs/2508.18947.

Joar Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward hacking. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Leon Staufer, Kevin Feng, Kevin Wei, Luke Bailey, Yawen Duan, Mick Yang, A. Pinar Ozisik, Stephen Casper, and Noam Kolt. The 2025 ai agent index: Documenting technical and safety features of deployed agentic ai systems, 2026. URL https://arxiv.org/abs/2602.17753.

Izaiah Sun, Daniel Tan, and Andy Deng. Lisa technical report: An agentic framework for smart contract auditing, 2025. URL https://arxiv.org/abs/2509.24698.

Surada Suwansathit, Yuxuan Zhang, and Guofei Gu. A security analysis of the openclaw ai agent framework, 2026. URL https://arxiv.org/abs/2603.27517.

Minxue Tang, Anna Dai, Louis DiValentin, Aolin Ding, Amin Hass, Neil Zhenqiang Gong, Yiran Chen, and Hai "Helen" Li. ModelGuard: Information-Theoretic defense against model extraction attacks. In *33rd USENIX Security Symposium (USENIX Security 24)*, pp. 5305–5322, Philadelphia, PA, August 2024. USENIX Association. ISBN 978-1-939133-44-1. URL https://www.usenix.org/conference/usenixsecurity24/presentation/tang.

Amine Tellache, Abdelaziz Amara Korba, Amdjed Mokhtari, Horea Moldovan, and Yacine Ghamri-Doudane. Advancing autonomous incident response: Leveraging llms and cyber threat intelligence, 2025. URL https://arxiv.org/abs/2508.10677.

Yifang Tian, Yaming Liu, Zichun Chong, Zihang Huang, and Hans-Arno Jacobsen. Gala: Can graph-augmented large language model agentic workflows elevate root cause analysis?, 2025. URL https://arxiv.org/abs/2508.12472.

Dheer Toprani and Vijay K. Madisetti. Llm agentic workflow for automated vulnerability detection and remediation in infrastructure-as-code. *IEEE Access*, 13:69175–69181, 2025. doi: 10.1109/ACCESS.2025.3560911.

Khang Tran, Yazan Boshmaf, Issa Khalil, NhatHai Phan, Ting Yu, and Md Rizwan Parvez. Poison with style: A practical poisoning attack on code large language models. *arXiv preprint arXiv:2605.27631*, 2026.

PeiYu Tseng, ZihDwo Yeh, Xushu Dai, and Peng Liu. Using llms to automate threat intelligence analysis workflows in security operation centers, 2024. URL https://arxiv.org/abs/2407.13093.

Ada Defne Tur, Nicholas Meade, Xing Han Lù, Alejandra Zambrano, Arkil Patel, Esin DURMUS, Spandana Gella, Karolina Stanczak, and Siva Reddy. Safearena: Evaluating the safety of autonomous web agents. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=7TrOBcxSvy.

Melissa Turcotte, François Labrèche, and Serge-Olivier Paquette. Automated alert classification and triage (aact): An intelligent system for the prioritisation of cybersecurity alerts, 2025. URL https://arxiv.org/abs/2505.09843.

Meet Udeshi, Minghao Shao, Haoran Xi, Nanda Rani, Kimberly Milner, Venkata Sai Charan Putrevu, Brendan Dolan-Gavitt, Sandeep Kumar Shukla, Prashanth Krishnamurthy, Farshad Khorrami, Ramesh Karri, and Muhammad Shafique. D-cipher: Dynamic collaborative intelligent multi-agent system with planner and heterogeneous executors for offensive security, 2025. URL https://arxiv.org/abs/2502.10931.

Saad Ullah, Praneeth Balasubramanian, Wenbo Guo, Amanda Burnett, Hammond Pearce, Christopher Kruegel, Giovanni Vigna, and Gianluca Stringhini. From cve entries to verifiable exploits: An automated multi-agent framework for reproducing cves, 2025. URL https://arxiv.org/abs/2509.01835.

Eric Wallace, Kai Xiao, Reimar Leike, Lilian Weng, Johannes Heidecke, and Alex Beutel. The instruction hierarchy: Training llms to prioritize privileged instructions, 2024. URL https://arxiv.org/abs/2404.13208.

Brandon J Walton, Mst Eshita Khatun, James M Ghawaly, and Aisha Ali-Gombe. Exploring Large Language Models for Semantic Analysis and Categorization of Android Malware . In *2024 Annual Computer Security Applications Conference Workshops (ACSAC Workshops)*, pp. 248–254, Los Alamitos, CA, USA, December 2024. IEEE Computer Society. doi: 10.1109/ACSACW65225.2024.00035. URL https://doi.ieeecomputersociety.org/10.1109/ACSACW65225.2024.00035.

Haoyu Wang, Christopher M. Poskitt, and Jun Sun. Agentspec: Customizable runtime enforcement for safe and reliable llm agents, 2025a. URL https://arxiv.org/abs/2503.18666.

Jingyi Wang et al. An efficient multiparty threshold ECDSA protocol against malicious adversaries for blockchain-based LLMs. *IET Information Security*, 2024a. doi: 10.1049/2024/2252865. URL https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/2024/2252865.

Junjie Wang, Yuchao Huang, Chunyang Chen, Zhe Liu, Song Wang, and Qing Wang. Software testing with large language models: Survey, landscape, and vision. *IEEE Transactions on Software Engineering*, 50 (4):911–936, 2024b.

Kun Wang, Guibin Zhang, Zhenhong Zhou, Jiahao Wu, Miao Yu, Shiqian Zhao, Chenlong Yin, Jinhu Fu, Yibo Yan, Hanjun Luo, Liang Lin, Zhihao Xu, Haolang Lu, Xinye Cao, Xinyun Zhou, Weifei Jin, Fanci Meng, Shicheng Xu, Junyuan Mao, Yu Wang, Hao Wu, Minghe Wang, Fan Zhang, Junfeng Fang, Wenjie Qu, Yue Liu, Chengwei Liu, Yifan Zhang, Qiankun Li, Chongye Guo, Yalan Qin, Zhaoxin Fan, Kai Wang, Yi Ding, Donghai Hong, Jiaming Ji, Yingxin Lai, Zitong Yu, Xinfeng Li, Yifan Jiang, Yanhui Li, Xinyu Deng, Junlin Wu, Dongxia Wang, Yihao Huang, Yufei Guo, Jen tse Huang, Qiufeng Wang, Xiaolong Jin, Wenxuan Wang, Dongrui Liu, Yanwei Yue, Wenke Huang, Guancheng Wan, Heng Chang, Tianlin Li, Yi Yu, Chenghao Li, Jiawei Li, Lei Bai, Jie Zhang, Qing Guo, Jingyi Wang, Tianlong Chen, Joey Tianyi Zhou, Xiaojun Jia, Weisong Sun, Cong Wu, Jing Chen, Xuming Hu, Yiming Li, Xiao Wang, Ningyu Zhang, Luu Anh Tuan, Guowen Xu, Jiaheng Zhang, Tianwei Zhang, Xingjun Ma, Jindong Gu, Liang Pang, Xiang Wang, Bo An, Jun Sun, Mohit Bansal, Shirui Pan, Lingjuan Lyu, Yuval Elovici, Bhavya Kailkhura, Yaodong Yang, Hongwei Li, Wenyuan Xu, Yizhou Sun, Wei Wang, Qing Li, Ke Tang, Yu-Gang Jiang, Felix Juefei-Xu, Hui Xiong, Xiaofeng Wang, Dacheng Tao, Philip S. Yu, Qingsong Wen, and Yang Liu. A comprehensive survey in llm(-agent) full stack safety: Data, training and deployment, 2025b. URL https://arxiv.org/abs/2504.15585.

Peiran Wang, Xiaogeng Liu, and Chaowei Xiao. CVE-bench: Benchmarking LLM-based software engineering agent's ability to repair real-world CVE vulnerabilities. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4207–4224, Albuquerque, New Mexico, April 2025c. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.212. URL https://aclanthology.org/2025.naacl-long.212/.

Shouju Wang, Fenglin Yu, Xirui Liu, Xiaoting Qin, Jue Zhang, Qingwei Lin, Dongmei Zhang, and Saravan Rajmohan. Privacy in action: Towards realistic privacy mitigation and evaluation for llm-powered agents. *arXiv preprint*, arXiv:2509.17488, 2025d. URL https://arxiv.org/abs/2509.17488. preprint, submitted 22 Sep 2025, cs.CR / cs.AI.

Weizhe Wang, Wei Ma, Qiang Hu, Yao Zhang, Jianfei Sun, Bin Wu, Yang Liu, Guangquan Xu, and Lingxiao Jiang. Vulnrepaireval: An exploit-based evaluation framework for assessing large language model vulnerability repair capabilities, 2025e. URL https://arxiv.org/abs/2509.03331.

Wenhao Wang, Hao Gu, Zhixuan Wu, Hao Chen, Xingguo Chen, and Fan Shi. Ptfusion: Llm-driven context-aware knowledge fusion for web penetration testing. *Information Fusion*, 127:103731, 2026a. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2025.103731. URL https://www.sciencedirect.com/science/article/pii/S1566253525007936.

Xilong Wang, John Bloch, Zedian Shao, Yuepeng Hu, Shuyan Zhou, and Neil Zhenqiang Gong. Webinject: Prompt injection attack to web agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 2010–2030, 2025f.

Zhilong Wang, Neha Nagaraja, Lan Zhang, Hayretdin Bahsi, Pawan Patil, and Peng Liu. To protect the llm agent against the prompt injection attack with polymorphic prompt, 2025g. URL https://arxiv.org/abs/2506.05739.

Zhiqiang Wang, Yichao Gao, Yanting Wang, Suyuan Liu, Haifeng Sun, Haoran Cheng, Guanquan Shi, Haohua Du, and Xiangyang Li. Mcptox: A benchmark for tool poisoning on real-world mcp servers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 35811–35819, 2026b.

Zhun Wang, Vincent Siu, Zhe Ye, Tianneng Shi, Yuzhou Nie, Xuandong Zhao, Chenguang Wang, Wenbo Guo, and Dawn Song. Agentvigil: Generic black-box red-teaming for indirect prompt injection against llm agents, 2025h. URL https://arxiv.org/abs/2505.05849.

Zhun Wang, Tianneng Shi, Jingxuan He, Matthew Cai, Jialin Zhang, and Dawn Song. Cybergym: Evaluating AI agents' real-world cybersecurity capabilities at scale. In *The Fourteenth International Conference on Learning Representations*, 2026c. URL https://openreview.net/forum?id=2YvbLQEdYt.

Ziyue Wang and Liyi Zhou. Agentic discovery and validation of android app vulnerabilities, 2025. URL https://arxiv.org/abs/2508.21579.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: how does llm safety training fail? In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Bowen Wei, Yuan Shen Tay, Howard Liu, Jinhao Pan, Kun Luo, Ziwei Zhu, and Chris Jordan. Cortex: Collaborative llm agents for high-stakes alert triage, 2025a. URL https://arxiv.org/abs/2510.00311.

Wenqi Wei and Ling Liu. Trustworthy distributed ai systems: Robustness, privacy, and governance, 2024. URL https://arxiv.org/abs/2402.01096.

Zhiyuan Wei, Jing Sun, Zijiang Zhang, Xianhao Zhang, Meng Lia, and Zhe Hou. Llm-smartaudit: Advanced smart contract vulnerability detection, 2024. arXiv preprint.

Zhiyuan Wei, Jing Sun, Yuqiang Sun, Ye Liu, Daoyuan Wu, Zijian Zhang, Xianhao Zhang, Meng Li, Yang Liu, Chunmiao Li, Mingchao Wan, Jin Dong, and Liehuang Zhu. Advanced smart contract vulnerability detection via llm-powered multi-agent systems. *IEEE Transactions on Software Engineering*, 51(10): 2830–2846, 2025b. doi: 10.1109/TSE.2025.3597319.

Zichao Wei, Jun Zeng, Ming Wen, Zeliang Yu, Kai Cheng, Yiding Zhu, Jingyi Guo, Shiqi Zhou, Le Yin, Xiaodong Su, and Zhechao Ma. Patcheval: A new benchmark for evaluating llms on patching real-world vulnerabilities, 2025c. URL https://arxiv.org/abs/2511.11019.

Jiangrong Wu, Zitong Yao, Yuhong Nan, and Zibin Zheng. Chainfuzzer: Greybox fuzzing for workflow-level multi-tool vulnerabilities in llm agents, 2026. URL https://arxiv.org/abs/2603.12614.

Yaozu Wu, Jizhou Guo, Dongyuan Li, Henry Peng Zou, Wei-Chieh Huang, Yankai Chen, Zhen Wang, Weizhi Zhang, Yangning Li, Meng Zhang, Renhe Jiang, and Philip S. Yu. Psg-agent: Personality-aware safety guardrail for llm-based agents, 2025a. URL https://arxiv.org/abs/2509.23614.

Yiran Wu, Mauricio Velazco, Andrew Zhao, Manuel Raúl Meléndez Luján, Srisuma Movva, Yogesh K Roy, Quang Nguyen, Roberto Rodriguez, Qingyun Wu, Michael Albada, Julia Kiseleva, and Anand Mudgerikar. Excytin-bench: Evaluating llm agents on cyber threat investigation, 2025b. URL https://arxiv.org/abs/2507.14201.

Shihao Xia, Shuai Shao, Mengting He, Tingting Yu, Linhai Song, and Yiying Zhang. Auditgpt: Auditing smart contracts with chatgpt, 2024. URL https://arxiv.org/abs/2404.04306.

Chong Xiang, Tong Wu, Zexuan Zhong, David Wagner, Danqi Chen, and Prateek Mittal. Certifiably robust RAG against retrieval corruption. In *Proceedings of the Network and Distributed System Security (NDSS) Symposium 2025*, 2025a. URL https://arxiv.org/abs/2405.15556.

Zhen Xiang, Linzhi Zheng, Yanjie Li, Junyuan Hong, Qinbin Li, Han Xie, Jiawei Zhang, Zidi Xiong, Chulin Xie, Carl Yang, Dawn Song, and Bo Li. Guardagent: Safeguard llm agents by a guard agent via knowledge-enabled reasoning, 2025b. URL https://arxiv.org/abs/2406.09187.

ZeKe Xiao, Qin Wang, Yuekang Li, and Shiping Chen. Prompt to pwn: Automated exploit generation for smart contracts, 2026a. URL https://arxiv.org/abs/2508.01371.

Zibo Xiao, Jun Sun, and Junjie Chen. Air: Improving agent safety through incident response, 2026b. URL https://arxiv.org/abs/2602.11749.

Wenpeng Xing, Minghao Li, Mohan Li, and Meng Han. Towards robust and secure embodied ai: A survey on vulnerabilities and attacks, 2025a. URL https://arxiv.org/abs/2502.13175.

Wenpeng Xing, Zhonghao Qi, Yupeng Qin, Yilin Li, Caini Chang, Jiahui Yu, Changting Lin, Zhenzhen Xie, and Meng Han. Mcp-guard: A multi-stage defense-in-depth framework for securing model context protocol in agentic ai, 2025b. URL https://arxiv.org/abs/2508.10991.

Zhixin Xing and Zheng Zhang. Dynamic attention: Enhancing inherent robustness of transformers against adversarial attacks. Network and Distributed System Security (NDSS) Symposium 2024, 2024. URL https://arxiv.org/abs/2311.17400. NDSS 2024, doi:10.14722/ndss.2024.24115.

Hanxiang Xu, Shenao Wang, Ningke Li, Kailong Wang, Yanjie Zhao, Kai Chen, Ting Yu, Yang Liu, and Haoyu Wang. Large language models for cyber security: A systematic literature review. *ACM Trans. Softw. Eng. Methodol.*, September 2025a. ISSN 1049-331X. doi: 10.1145/3769676. URL https://doi.org/10.1145/3769676.

Kevin Xu, Yeganeh Kordi, Tanay Nayak, Adi Asija, Yizhong Wang, Kate Sanders, Adam Byerly, Jingyu Zhang, Benjamin Van Durme, and Daniel Khashabi. TurkingBench: A challenge benchmark for web agents. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3694–3710, Albuquerque, New Mexico, April 2025b. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.188. URL https://aclanthology.org/2025.naacl-long.188/.

Zihao Xu, Yi Liu, Gelei Deng, Yuekang Li, and Stjepan Picek. A comprehensive study of jailbreak attack versus defense for large language models, 2024. URL https://arxiv.org/abs/2402.13457.

Zhou Xuan, Xiangzhe Xu, Mingwei Zheng, Louis Zheng-Hua Tan, Jinyao Guo, Tiantai Zhang, Le Yu, Chengpeng Wang, and Xiangyu Zhang. Identifying adversary tactics and techniques in malware binaries with an llm agent, 2026. URL https://arxiv.org/abs/2602.06325.

Chenyuan Yang, Zijie Zhao, Zichen Xie, Haoyu Li, and Lingming Zhang. Knighter: Transforming static analysis with llm-synthesized checkers. In *Proceedings of the ACM SIGOPS 31st Symposium on Operating Systems Principles*, SOSP '25, New York, NY, USA, 2025a. Association for Computing Machinery. doi: 10.1145/3731569.3764827. URL https://doi.org/10.1145/3731569.3764827.

Wenkai Yang, Xiaohan Bi, Yankai Lin, Sishuo Chen, Jie Zhou, and Xu Sun. Watch out for your agents! investigating backdoor threats to LLM-based agents. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=Nf4MHF1pi5.

Zhenning Yang, Archit Bhatnagar, Yiming Qiu, Tongyuan Miao, Patrick Tser Jern Kon, Yunming Xiao, Yibo Huang, Martin Casado, and Ang Chen. Cloud infrastructure management in the age of ai agents. *ACM SIGOPS Operating Systems Review*, 59(1), 2025b. doi: 10.1145/3759441.3759443. URL https://arxiv.org/pdf/2506.12270v1.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains, 2024. URL https://arxiv.org/abs/2406.12045.

Ziang Ye, Yang Zhang, Wentao Shi, Xiaoyu You, Fuli Feng, and Tat-Seng Chua. Visualtrap: A stealthy backdoor attack on GUI agents via visual grounding manipulation. In *Second Conference on Language Modeling*, 2025a. URL https://openreview.net/forum?id=7HPuAkgdVm.

Ziyang Ye, Triet Huynh Minh Le, and M. Ali Babar. Llmsecconfig: An llm-based approach for fixing software container misconfigurations, 2025b. URL https://arxiv.org/abs/2502.02009.

Jingwei Yi, Yueqi Xie, Bin Zhu, Emre Kiciman, Guangzhong Sun, Xing Xie, and Fangzhao Wu. Benchmarking and defending against indirect prompt injection attacks on large language models. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, KDD '25, pp. 1809–1820, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400712456. doi: 10.1145/3690624.3709179. URL https://doi.org/10.1145/3690624.3709179.

Alperen Yildiz, Sin G Teo, Yiling Lou, Yebo Feng, Chong Wang, and Dinil Mon Divakaran. Benchmarking LLMs and LLM-based agents in practical vulnerability detection for code repositories. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 30848–30865, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1490. URL https://aclanthology.org/2025.acl-long.1490/.

Jiahao Yu, Xingwei Lin, Zheng Yu, and Xinyu Xing. LLM-Fuzzer: Scaling assessment of large language model jailbreaks. In *33rd USENIX Security Symposium (USENIX Security 24)*, pp. 4657–4674, Philadelphia, PA, August 2024. USENIX Association. ISBN 978-1-939133-44-1. URL https://www.usenix.org/conference/usenixsecurity24/presentation/yu-jiahao.

Miao Yu, Fanci Meng, Xinyun Zhou, Shilong Wang, Junyuan Mao, Linsey Pan, Tianlong Chen, Kun Wang, Xinfeng Li, Yongfeng Zhang, Bo An, and Qingsong Wen. A survey on trustworthy llm agents: Threats and countermeasures. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '25, pp. 6216–6226, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400714542. doi: 10.1145/3711896.3736561. URL https://doi.org/10.1145/3711896.3736561.

Tongxin Yuan, Zhiwei He, Lingzhong Dong, Yiming Wang, Ruijie Zhao, Tian Xia, Lizhen Xu, Binglin Zhou, Fangqi Li, Zhuosheng Zhang, Rui Wang, and Gongshen Liu. R-judge: Benchmarking safety risk awareness for llm agents, 2024. URL https://arxiv.org/abs/2401.10019.

Zhuowen Yuan, Zhaorun Chen, Zhen Xiang, Nathaniel D. Bastian, Seyyed Hadi Hashemi, Chaowei Xiao, Wenbo Guo, and Bo Li. Shieldnet: Network-level guardrails against emerging supply-chain injections in agentic systems, 2026. URL https://arxiv.org/abs/2604.04426.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 10471–10506, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.624. URL https://aclanthology.org/2024.findings-acl.624/.

Qiusi Zhan, Richard Fang, Henil Shalin Panchal, and Daniel Kang. Adaptive attacks break defenses against indirect prompt injection attacks on LLM agents. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 7101–7117, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.395. URL https://aclanthology.org/2025.findings-naacl.395/.

Zhonghao Zhan, Krinos Li, Yefan Zhang, and Hamed Haddadi. Systems-level attack surface of edge agent deployments on iot. In *Proceedings of the Sixth European Workshop on Machine Learning and Systems*, EuroMLSys '26, pp. 99–108, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400726057. doi: 10.1145/3805621.3807636. URL https://doi.org/10.1145/3805621.3807636.

Andy K Zhang, Neil Perry, Riya Dulepet, Joey Ji, Celeste Menders, Justin W Lin, Eliot Jones, Gashon Hussein, Samantha Liu, Donovan Julian Jasper, Pura Peetathawatchai, Ari Glenn, Vikram Sivashankar, Daniel Zamoshchin, Leo Glikbarg, Derek Askaryar, Haoxiang Yang, Aolin Zhang, Rishi Alluri, Nathan Tran, Rinnara Sangpisit, Kenny O Oseleononmen, Dan Boneh, Daniel E. Ho, and Percy Liang. Cybench: A framework for evaluating cybersecurity capabilities and risks of language models. In *The Thirteenth*

International Conference on Learning Representations, 2025a. URL https://openreview.net/forum?id=tc90LV0yRL.

Andy K Zhang, Joey Ji, Celeste Menders, Riya Dulepet, Thomas Qin, Ron Yifeng Wang, Junrong Wu, Kyleen Liao, Jiliang Li, Jinghan Hu, Sara Hong, Nardos Demilew, Shivatmica Murgai, Jason Khiem Tran, Nishka Kacheria, Ethan Jun shen Ho, Denis Liu, Lauren McLane, Olivia Beyer Bruvik, Dai-Rong Han, Seungwoo Kim, Akhil Vyas, Cuiyuanxiu Chen, Ryan Li, Weiran Xu, Jonathan Z Ye, Prerit Choudhary, Siddharth M. Bhatia, Vikram Sivashankar, Yuxuan Bao, Dawn Song, Dan Boneh, Daniel E. Ho, and Percy Liang. Bountybench: Dollar impact of AI agent attackers and defenders on real-world cybersecurity systems. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2026a. URL https://openreview.net/forum?id=pIsP4lMlFd.

D Zhang, Z Li, X Luo, X Liu, P Li, and W Xu. Mcp security bench (msb): Benchmarking attacks against model context protocol in llm agents. arxiv 2025. *arXiv preprint arXiv:2510.15994*, 2025b.

Haiyue Zhang, Yi Nian, and Yue Zhao. Agent audit: A security analysis system for llm agent applications, 2026b. URL https://arxiv.org/abs/2603.22853.

Hanrong Zhang, Jingyuan Huang, Kai Mei, Yifei Yao, Zhenting Wang, Chenlu Zhan, Hongwei Wang, and Yongfeng Zhang. Agent security bench (asb): Formalizing and benchmarking attacks and defenses in llm-based agents. In *ICLR*, 2025c. URL https://openreview.net/forum?id=V4y0CpX4hK.

Tian Zhang et al. Agentsentry: Mitigating indirect prompt injection via temporal causal diagnostics, 2026c. URL https://arxiv.org/abs/2602.22724.

Yanzhe Zhang and Diyi Yang. Searching for privacy risks in llm agents via simulation, 2025. URL https://arxiv.org/abs/2508.10880.

Yuyang Zhang, Kangjie Chen, Jiaxin Gao, Ronghao Cui, Run Wang, Lina Wang, and Tianwei Zhang. Towards action hijacking of large language model-based agent, 2025d. URL https://arxiv.org/abs/2412.10807.

Boyuan Zheng, Zeyi Liao, Scott Salisbury, Zeyuan Liu, Michael Lin, Qinyuan Zheng, Zifan Wang, Xiang Deng, Dawn Song, Huan Sun, and Yu Su. Webguard: Building a generalizable guardrail for web agents, 2025. URL https://arxiv.org/abs/2507.14293.

Pengcheng Zhou, Yinglun Feng, and Zhongliang Yang. Privacy-aware RAG: Secure and isolated knowledge retrieval, 2025. URL https://arxiv.org/abs/2503.15548.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents, 2024. URL https://arxiv.org/abs/2307.13854.

Xingfu Zhou and Pengfei Wang. Reasoning-style poisoning of llm agents via stealthy style transfer: Process-level attacks and runtime monitoring in rsv space, 2025. URL https://arxiv.org/abs/2512.14448.

Jie Zhu, Chihao Shen, Ziyang Li, Jiahao Yu, Yizheng Chen, and Kexin Pei. Locus: Agentic predicate synthesis for directed fuzzing, 2025a. URL https://arxiv.org/abs/2508.21302.

Yuxuan Zhu, Antony Kellermann, Dylan Bowman, Philip Li, Akul Gupta, Adarsh Danda, Richard Fang, Conner Jensen, Eric Ihli, Jason Benn, Jet Geronimo, Avi Dhir, Sudhit Rao, Kaicheng Yu, Twm Stone, and Daniel Kang. CVE-bench: A benchmark for AI agents' ability to exploit real-world web application vulnerabilities. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=3pk0p4NGmQ.

Yuxuan Zhu, Antony Kellermann, Dylan Bowman, Philip Li, Akul Gupta, Adarsh Danda, Richard Fang, Conner Jensen, Eric Ihli, Jason Benn, Jet Geronimo, Avi Dhir, Sudhit Rao, Kaicheng Yu, Twm Stone, and Daniel Kang. Cve-bench: A benchmark for ai agents' ability to exploit real-world web application vulnerabilities, 2025c. URL https://arxiv.org/abs/2503.17332.

Yuxuan Zhu, Antony Kellermann, Akul Gupta, Philip Li, Richard Fang, Rohan Bindu, and Daniel Kang. Teams of llm agents can exploit zero-day vulnerabilities, 2025d. URL https://arxiv.org/abs/2406.01637.

Terry Yue Zhuo, Dingmin Wang, Hantian Ding, Varun Kumar, and Zijian Wang. Training language model agents to find vulnerabilities with CTF-dojo. In *NeurIPS 2025 Fourth Workshop on Deep Learning for Code*, 2025. URL https://openreview.net/forum?id=sQPeclxRBG.

Terry Yue Zhuo, Yangruibo Ding, Wenbo Guo, and Ruijie Meng. To defend against cyber attacks, we must teach ai agents to hack. *arXiv preprint arXiv:2602.02595*, 2026a.

Terry Yue Zhuo, Dingmin Wang, Hantian Ding, Varun Kumar, and Zijian Wang. Cyber-zero: Training cybersecurity agents without runtime. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=1gRTeAik4G.

Xuanjun Zong, Zhiqi Shen, Lei Wang, Yunshi Lan, and Chao Yang. MCP-safetybench: A benchmark for safety evaluation of large language models with real-world MCP servers. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=7XYjeL46co.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models, 2023. URL https://arxiv.org/abs/2307.15043.

Yuwen Zou, Jia Liu, and Wenjun Fan. Ctfagent: An llm-powered agent for ctf challenge solving. *Journal of Information Security and Applications*, 96:104305, 2026. ISSN 2214-2126. doi: https://doi.org/10.1016/j.jisa.2025.104305. URL https://www.sciencedirect.com/science/article/pii/S2214212625003424.

A Paper Collection Methodology

To ensure a comprehensive and reproducible review of the agentic security landscape, we employed a multi-stage paper collection methodology combining automated searches, manual curation, and snowballing techniques.

Automated Database Search

We conducted an automated search across major academic repositories, including ACL Anthology, IEEE Xplore, ACM Digital Library, and arXiv, covering publications from January 2023 to May 2026. Using a Boolean query, we combined two groups keywords related to agentic systems and security concepts.

Search Query Structure `(Group 1 Keywords) AND (Group 2 Keywords)`

- **Group 1 (Agent-related):** `("LLM agent" OR "AI agent" OR "agentic AI" OR "autonomous agent" OR "multi-agent system")`
- **Group 2 (Security-related):** `("security" OR "threat" OR "vulnerability" OR "attack" OR "defense" OR "red team" OR "blue team" OR "penetration testing" OR "fuzzing" OR "jailbreak" OR "prompt injection" OR "poisoning" OR "hardening" OR "adversarial")`

Manual Curation

To identify relevant work that our keyword search may had missed, we manually scanned the proceedings of top-tier security (e.g., USENIX, ACM CCS, Oakland) and AI (e.g., ACL, EMNLP, NeurIPS, ICLR, ICML) conferences from the same period.

Inclusion and Exclusion Criteria

Inclusion Criteria The paper's primary subject must be **LLM-based agents**. It must have a substantial focus on a **technical security aspect**, aligning with one of our three pillars. The work must be a peer-reviewed publication or a highly-cited preprint.

Exclusion Criteria We excluded papers on general LLM safety that do not address agentic systems, studies on non-LLM agents, works centered on high-level ethics or policy without technical details, and non-technical articles.

Snowballing

Finally, we performed backward and forward snowballing on the curated set of included papers. We reviewed the reference lists of these key papers to identify foundational or related works we might have missed.