ISSUE BRIEF

FRONTIER MODEL FORUM

Emerging Security Practices for AI Agents

DATE

June 3, 2026

INTRODUCTION

AI agents based on the most advanced general-purpose models represent a qualitative shift in how software operates. Unlike traditional software or conversational AI, these agents combine the reasoning capabilities of frontier models with access to tools, enabling the agents to process data and instructions while acting directly on a user's behalf. The most performant agents can now reliably plan and execute long sequences of actions autonomously, invoking tools, querying external services, maintaining memory across sessions, and interacting with other agents to independently complete highly complex tasks. The unique capacity of AI agents to reason and take action on their own opens up significant new economic and scientific opportunities–yet also introduces significant security risks.

The same capabilities and affordances that make AI agents so powerful also pose a new class of security challenges that existing cybersecurity frameworkswere not designed to address.[1] Chief among these challenges is the risk of misaligned agent actions, where an agent, whether through misaligned instructions, adversarial inputs (such as through prompt injections), or compounding errors across a multi-step workflow, takes consequential actions outside the intended scope of its deployment. Equally concerning is the risk associated with access to private or sensitive data. As agents are granted broader tool access and pull from an expanding ecosystem of APIs and external data sources, the attack surface extends well beyond what any single user prompt might expose. Architectural choices around memory management, tool authorization, and delegation hierarchies for multi-agent systems, play a significant role in either containing or amplifying these vulnerabilities.

Securing AI agents is a shared responsibility that requires model

developers, deployers, and users to each make deliberate choices about agent architecture, scaffolding, and the boundaries of agent authority, as appropriate. Numerous collaborative efforts are already underway across the AI industry to establish shared standards and best practices.[2] However, agent security also raises tradeoffs around usability, user friction, and meaningful transparency.

This issue brief aims to offer a preliminary overview of agentic security, drawing on discussions with security experts within the Frontier Model Forum (FMF) and the broader AI security community. The brief distinguishes agentic security from other AI-related security issues and then outlines shared emerging practices across key layers of the agent stack – **models**, **system guardrails** and **architecture** choices, **harnesses**,[3] and **tools**–as well as several core security principles associated with each layer. Since AI agent development and deployment is advancing rapidly as security solutions strive to keep pace, the issue brief also concludes with key open technical challenges that require further research. This overview is intended to inform the development of more robust security measures and guide the secure deployment of agents for developers, deployers, and users in a constantly evolving field.

Overview of Agents

The term "AI agent" does not yet have a settled definition, but a useful working description is a system that uses external tools to perform goal-directed tasks on behalf of a user. Such systems range from simple single-agent deployments to complex multi-agent architectures, and vary widely in their degree of autonomy, reasoning capability, and task complexity.[4]

Multi-agent systems can be composed of deep layers of sub-agents, each of which generally coordinate, communicate, and allocate sub-tasks. Multi-agent systems can also coordinate parallel workstreams, introducing a distinct class of challenges rooted in the trust relationships and communication between agents.

Several distinct agent types have emerged in practice. Conversational assistants can execute code and connect to third-party services. Browsing agents navigate the open web. Coding agents operate in developer environments. Software development kits also allow third-party developers to build custom agents with their own tool sets. These agents differ substantially in architecture and the security controls appropriate for each differ accordingly.

Agent Security as a Shared Responsibility

Agent security is a shared responsibility distributed across the value chain, including developers who build the agent or its underlying system, deployers who serve it to users, third party entities who host services that agents interact with, tooling providers, and users themselves.[5]

Each of these actors has a unique role to play in securing agents responsibly:

- **Developers**, responsible for the underlying models that power agents, can anticipate the likely deployment types of their models and assess general risks and benefits as well as collaborate with actors across the value chain to facilitate downstream safeguards. They also have ongoing work to do to defend against prompt injection attacks, memory poisoning, and tool use safeguards.
- **Deployers** can draw on established, deterministic, and infrastructure-level security practices and controls, including sandboxing agent execution environments, applying least-privilege access controls, monitoring for anomalous behavior, validating inputs and outputs, and, where appropriate, maintaining audit logs that support incident investigation.
- **Tooling and harness providers** are entities that provide tools or harnesses that developers and deployers can integrate into their agentic systems.
- **Third party entities** are actors or surfaces that agents will interact with, such as merchants and host sites that may be able to detect an agent's scope of authority.
- **Users** acquire and use AI agents, and thus scope agent deployments and use cases. The potential damage from any agent failure scales directly with the level of trust and permissions a user has granted. As agent capabilities grow, so does the importance of equipping users with the knowledge to manage them safely and securely, helping them understand the risks associated with certain use cases, and facilitating the granting and revoking of permissions as the user deems fit. Some users may be more skilled and risk-tolerant than others and more empowered to apply AI agents for sensitive or high-stakes use cases.

Each actor will have different access to and control over the relevant agent security layers discussed below. Security guidance should be structured to the specific actor and the layers over which they have direct control and responsibility.

AGENT SECURITY: RISKS AND ATTACK VECTORS

Agentic security generally focuses on risks which may be realized on surfaces where AI agents are benignly deployed. These risks include attacks against agents by malicious actors using a range of methods, as well as accidental harms to users due to misaligned agent actions. Agents deployed within an organization, which are valuable for automating and accelerating a range of workflows, can also potentially pose insider threats,[6] while externally facing agents that interact with public channels are more exposed to untrusted inputs. Agentic security in this context is distinct from the risks that arise from malicious actors using and scaling agents to facilitate an offensive cyberattack.

Key attack vectors include:

- **Prompt injection**. AI agents are valuable because they can interact with multiple surfaces over time. However, this capability makes them subject to key attack vectors. One technically complex challenge and attack vector in agent security is prompt injection: an attack that arises when a system lacks a clear separation between trusted internal instructions and untrusted external data.[7] Prompt injection attacks can either be direct, where a user directly inputs malicious content into the AI system, overriding system instructions and guardrails, or indirect, where malicious instructions are embedded in external sources (like webpages, documents or tool outputs) that the AI-powered system retrieves and processes as legitimate instructions. Prompt injection predates AI agents, but agents dramatically amplify the stakes because a successful attack can result in internal systems updates or real-world action. For example, an email from an unknown sender contains hidden instructions that direct a user's email agent to collect sensitive messages from their inbox and forward them to the attacker. The industry is actively pursuing prompt injection defenses at multiple layers simultaneously.

- **Memory poisoning**. AI agents are valuable for long horizon tasks because they can retain context over time. However, attackers can exploit this capability by subtly manipulating an AI agent's memory over time, leading to long-term behavioral changes that are difficult to detect, and causing the agent to carry out an attacker's commands. This could cause the agent to make flawed decisions, leak sensitive information, or even carry out malicious actions. This differs from traditional attacks that target static code, as it exploits the agent's ability to learn and remember.

- **Tool supply chain compromise**. AI agents are useful because they leverage tools to achieve a range of tasks. Because tools return content directly into the model's reasoning context, a compromised tool is simultaneously a malicious code execution threat and a prompt injection vector. The risk posed by any given tool is a function of what data it can access, what state it can modify, and whether it can communicate outside established boundaries. Every tool an agent can access expands both its capability and its exposure. Agentic systems may therefore inherit security weaknesses from the tools they leverage that may be unknown to the AI developer.

In addition, some agent security risks can arise from an agent attempting to fulfill a benign request from the user in a way that was not the user's intent, which may lead to a range of harmful outcomes. While this issue brief focuses on how agent security approaches may mitigate attack vectors, we underscore that the layers, issues, and tradeoffs discussed may also be relevant to preventing accidental harm resulting from agents taking actions that are misaligned with user intent, even when this misalignment is not due to malicious attacks.

AGENT SECURITY: RISKS AND ATTACK VECTORS

Several layers determine what an agent can do in practice, and impact an agent's

profile for the risks and attack vectors described above:

- the **model** underlying an agentic system
- The **agent system** surrounding and built on the model, including:
    - **system-level guardrails** and **architectural** choices that constrain agent behavior and mitigate security risks. These include probabilistic and deterministic controls that define the boundary of the agent
    - the **harness** that orchestrates planning, memory, and multi-step tool use
    - external **tools** the agent can invoke

Notably, each of those layers implicates different security questions and implementations. The model and harness layers introduce probabilistic, hard-to-predict failure modes that traditional security approaches were not designed to handle. The execution environment and tool access layer, by contrast, are well-suited to conventional, deterministic controls and can determine the scope of the damage when something goes wrong. For example, the same prompt injection that causes limited impact inside a tightly sandboxed agent with narrow tool permissions could trigger data exfiltration or unauthorized system changes if the agent has broad access to the filesystem and to stored credentials.

The industry is actively pursuing prompt injection defenses at multiple layers simultaneously and the level of maturity of each layer and its appropriateness for a particular AI deployment varies. Each layer warrants its own set of controls, addressed in turn below.

Model Controls

At the model level, some developers use fine-tuning strategies,[8] including reinforcement learning[9] to train agents to treat security as a priority in their reasoning. Some developers have implemented[10] instruction hierarchies that define how the model should behave when instructions from a system prompt conflict with instructions from untrusted users and third parties.[11]

System Controls

Deterministic controls

Deterministic controls at the infrastructure level are a core part of what data an agent can access within its execution environment. Sandboxing, filesystem scope, and network egress policy all determine what an agent can reach and whether data can leave the system. Existing security frameworks, such as the NIST Cybersecurity Framework, 800-53, NIST SSDF SP 800-218, SP 800-63, and ISO/IEC 27001, among others remain relevant for infrastructure security.[12]

A key line of defense is limiting what private data an agent can reach, and under what conditions. Some developers have published open-source standards, such as the Model Context Protocol,[13] for connecting agents to external data, tools, and software systems. These standards include controls that allow users to permit or restrict an agent's access to specific tools and processes on a per-task basis, including the option to grant only one-time access to sensitive information.[14] Some developers also offer a "logged out" mode, which allows agents to take actions without access to user credentials, reducing the potential for data leakage. However, the security of these controls ultimately depends on implementation discipline rather than protocol guarantees alone.[15] Environment controls come with a range of tradeoffs, as some internal users (i.e., users with a high degree of technical sophistication) may be better equipped to weigh and assess agent risks compared to others.

Developers have also developed novel design patterns, such as CaMeL, which converts user commands into executable code in a restricted Python-like language and uses a custom interpreter to track data provenance and enforce security policies.[16] Because CaMeL enforces these policies at execution time in the interpreter rather than through AI-based detection, it provides deterministic rather than probabilistic security guarantees. With CaMeL, untrusted data cannot silently hijack commands because the system always knows exactly where each piece of data originated.

Given that prompt injection research is still maturing, robust agent security should not rely on detection alone. Deterministic controls placed outside the agent's reasoning loop, including limits on tool access and permitted actions that cannot be overridden through context manipulation, remain valuable for mitigating harmful security outcomes.

Probabilistic controls

Rather than rely exclusively on deterministic controls, which on their own may not be able to strike the desired balance between system security and utility, developers may complement deterministic controls with probabilistic controls, such as classifiers that flag attacks at the input, output, and model activation levels.[17] AI-powered or other probabilistic controls can help reduce attacker success rates. This includes training classification models on known prompt injections and jailbreak attempts so they learn to recognize and resist malicious instructions.

Harness controls

The harness is the orchestration layer that enables a model to function as an agent. It manages prompts, tool access, execution flow across multi-step interactions, enabling the agent to plan, act, and maintain context across turns. Because agent actions pass through the harness, it also serves as a natural control point for monitoring, verification, logging, and enforcement.

Malicious inputs, such as prompt injections, can be stored in memory, propagate

across agents, and reappear in later decision cycles, enabling risks that unfold gradually over time rather than at the moment of input.

Attributes of AI agents that make them valuable to users and deployers also expose new security challenges. For example, persistent memory, a record of previous turns or interactions taken by an agent in a given workflow or past workflow, is valuable for advanced AI agents. However, it also can act as an accelerant to other risks by allowing a compromised state to persist and propagate across tasks.[18] If corrupt information enters an agent's memory, an agent can carry a flawed premise forward indefinitely, acting on it long after the original source is gone.

As discussed above, developers have built detection systems on top of models that act as prompt shields or chain-of-thought auditors to inspect agent reasoning for signs of prompt injection.[19] Other detection techniques transform untrusted input text to make its provenance visually obvious to the model, helping it distinguish external data from trusted system instructions.[20] To complement this approach, developers have built link safety systems that prevent agents from automatically fetching URLs that aren't already publicly indexed, blocking a common attack vector in which an adversary tricks an agent into visiting a custom URL that exfiltrates sensitive data through query parameters.[21]

Tool controls

Agents are designed to act autonomously and update their behavior based on what they observe. Both sides of that loop, incoming observations and outgoing actions, run largely through third-party services. This is what makes agents useful for real work, and it is also where a significant portion of their attack surface lies.

Some developers have addressed tool supply chain compromise risk by routing third-party integration through directories of vetted tools, while still allowing users and enterprises to add their own.[22] Anything outside the directory can be tested in a sandbox before it is granted access to real data.

SECURITY PRINCIPLES

Foundational security practices remain central to a defense-in-depth approach to agent security.

Restrict agent scope

Firms have articulated best practices that complement security controls, including limiting the actions an agent can take and the resources it can access to what is strictly necessary to minimize the blast radius of a compromised system.[23] A useful framework for thinking about AI agent security risk is what some practitioners call the "lethal trifecta for AI agents."[24] The lethal trifecta outlines the three agent capabilities that, when combined, dramatically increase the risk of an attacker accessing private data. Those three capabilities are: access to private data, exposure to untrusted

content, and the ability to communicate externally.

One proposed mitigation offered while robustness to prompt injection research is still maturing is to prohibit agents from satisfying all three lethal trifecta properties at once.[25] This may be beneficial for use cases deemed higher risk, as it ensures that any given agent either lacks access to private data, is isolated from untrusted content, or cannot send messages, emails, or make requests to external systems. The email prompt injection example in the "Agent Security: Risks and Attack Vectors" section above can illustrate this principle. That attack is only possible because the agent can read untrusted input (emails from unknown senders), access sensitive data (the inbox), and communicate externally (send emails). Limiting an agent to only process emails from trusted senders, scoping its email inbox access, or allowing outbound email only to approved recipients minimizes the risk of sensitive data being compromised. Restricting agent scope, however, may limit agent usability, and deployers should weigh tradeoffs between agent usability and security controls appropriately.

Implement adaptable human oversight

Human oversight is a core safeguard against the risks that come with granting agents broad autonomy. Requiring confirmation before an agent executes high-stakes or irreversible actions gives users the opportunity to catch and correct errors before they propagate, whether those errors stem from flawed reasoning, misunderstood instructions, or malicious prompt injection. This means maintaining visibility into agent reasoning and actions, and building in checkpoints at which a user can approve a step, redirect the agent, or take over the operation entirely.

However, multi-agent systems challenge the traditional human-in-the-loop model, as agents operate in parallel, share context, and influence each other in ways that make output-by-output review impractical. This raises key questions around user fatigue and the level of friction that users should encounter as they engage in oversight of agents. While human-in-the-loop can be a meaningful way to ensure agent oversight, there are open questions about whether certain types of human-in-the-loop implementations (which some users may ignore or passively click through) create the illusion of oversight, which may yield security issues.

LOOKING AHEAD

These practices represent an early but evolving foundation for AI agent security. As agents grow more capable and widely deployed, existing safeguards will need continuous reassessment through rigorous evaluation.

Several open research questions will continue to shape the field:

- **Adaptive Least Privilege**. Agents operating across multi-step tasks often require shifting permissions across tools and services. Adaptive least privilege, where permissions are granted only when required for a specific task and

revoked immediately afterward, especially in higher risk deployments, may be a promising approach. However, this approach comes with tradeoffs regarding agent usability, particularly as a main benefit of these tools involves recurring tasks (such as daily calendar updating based on inbox updates).

- **Privacy-Preserving Behavioral Monitoring**. Current monitoring focuses on behavior and event logging, but strategies designed around human-initiated actions may prove insufficient as autonomous agents run longer, more complex workflows involving multiple sub-agents. New approaches will be needed to maintain visibility of agent behavior without compromising user privacy.

- **Adversarial Robustness: Benchmarking, Red Teaming, and Training**. Many agent security issues lack standardized benchmarks or adversarial testing methodologies. Drawing from cybersecurity practice, adversarial testing loops can strengthen model robustness over time.[26] Systems can be trained against simulated attackers to probe for vulnerabilities like prompt injection or unsafe tool invocation. Scaling this approach will require comprehensive evaluation suites and more sophisticated automated red-teaming methods capable of testing the full range of ways agents can fail or be exploited.

- **Agent Reliability and Benchmarking**. Standardized benchmarks and testing methodologies are needed to measure agent reliability on sensitive tasks, even when no adversarial inputs are present.

- **Identity Management**. Traditional Identity and Access Management frameworks, such as OAuth 2.0, Security Assertion Markup Language, and OpenID Connect, were designed for more deterministic systems, assuming predictable application behavior and a single authenticated principal, whether human or a static machine identity. Establishing clearer signals of trust, potentially through the development of web standards and verification protocols that allow websites to explicitly declare content intended for AI consumption may be necessary.[27]

- **AI agents for defensive purposes**. The ecosystem may continue to adopt AI agents for defensive use cases and security workflows, and some research shows the promise of applying AI agents to real-world penetration testing, threat detection, automated vulnerability scanning,[28] root cause analysis,[29] cloud and infrastructure security, and Security Operations Center security workflows.[30]

- **Human oversight questions.** Oversight may shift from reviewing individual responses to governing workflows, including persistent supervisory monitoring with risk-tiered escalation where humans intervene only for consequential, sensitive, or low-confidence actions while routine steps run autonomously. A possible future state is that agent actions should generate auditable records of inputs, rationale, and approvals to ensure accountability,

reversibility, and compliance. The effect of user fatigue on the success of Human-in-the-loop should also be examined.

Looking ahead, agents will likely introduce unanticipated uses and interaction patterns, and agent security practices must evolve alongside them. The FMF looks forward to collaborating with the AI security community to develop and promote guidance that keeps pace with this rapidly changing landscape.

FOOTNOTES

1. For example, the NIST Cybersecurity Framework, first released in 2014, was not designed for the capabilities of current AI agents.
2. See the Cloud Security Alliance, "Securing Autonomous AI Agents" (February 4, 2026); OWASP, "OWASP Top 10 for Agentic Applications for 2026" (December 9, 2025); and Coalition for Secure AI, "The Future of Agentic Security: From Chatbots to Autonomous Swarms" (March 16, 2026).
3. A harness is the software infrastructure surrounding an AI model. It manages everything (tool execution, memory, context management, state persistence) except the model reasoning.
4. U.S. Government Accountability Office. "Science & Tech Spotlight: AI Agents." Published September 10, 2025.
5. Chan, A., et al. "Visibility into AI Agents". In: *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency (*2024).
6. We define "insider threat" as the potential for an insider to use their authorized access or understanding of an organization, whether intentionally or unintentionally, to negatively affect the integrity, confidentiality, and availability of an organization's data or systems. For additional information, see the U.S. Cybersecurity and Infrastructure Security Agency (CISA), "Insider Threat Mitigation".
7. U.K. National Cyber Security Centre (NCSC). "Prompt Injection Is Not SQL Injection (It May Be Worse)". Published December 8, 2025.
8. Chen, S., Zharmagambetov, A., Wagner, D., & Guo, C. (2025). "Meta secalign: A secure foundation llm against prompt injection attacks". *arXiv*. Published July 3, 2025.
9. Google. "Indirect Prompt Injections and Google's Layered Defense Strategy for Gemini." Published June 2025.
10. OpenAI. "Understanding Prompt Injections: A Frontier Security Challenge." Published November 7, 2025.
11. Wallace E, Xiao K, Leike R, et al. "The Instruction Hierarchy: Training LLMs to Prioritize Privileged Instructions." *arXiv*. Published April 19, 2024.
12. NIST Cybersecurity Framework, NIST 800-53, NIST SSDF SP 800-218, NIST SP 800-63, and ISO/IEC 27001.

13. Anthropic. "Introducing the Model Context Protocol." Published November 25, 2024.
14. Anthropic Support. "Getting started with custom connectors using remote MCP." Published 2025.
15. National Security Agency Artificial Intelligence Security Center (NSA AISC). "Model Context Protocol (MCP): Security Design Considerations for AI-Driven Automation." Published May 2026.
16. Beurer-Kellner, L., Buesser, B., Creţu, A.M., et al. "Design Patterns for Securing LLM Agents against Prompt Injections." *arXiv*. Published June 10, 2025.
17. Meta, Llama Prompt Guard 2; Google, "Indirect prompt injections & Google's layered defense strategy for Gemini" (March 26, 2026); Cunningham, H., Wei, J., Wang, Z., et al, *arXiv*, "Constitutional Classifiers++: Efficient Production-Grade Defenses against Universal Jailbreaks" (January 8, 2026).
18. Palo Alto Networks. "OpenClaw (formerly Moltbot, Clawdbot) May Signal the Next AI Security Crisis." Published January 29, 2026.
19. Beurer-Kellner, L., Buesser, B., Creţu, A.M., et al, *arXiv*, "Design Patterns for Securing LLM Agents against Prompt Injections" (June 10, 2025); Microsoft Learn, "Prompt Shields in Azure AI Content Safety" (January 30, 2026).
20. Hines, K., Lopez, G., Hall, M., et al. "Defending Against Indirect Prompt Injection Attacks with Spotlighting." *arXiv*. Published March 20, 2024.
21. OpenAI. "Keeping your data safe when an AI agent clicks a link." OpenAI Security. Published January 28, 2026.
22. Anthropic. "Response to NIST Request for Information: Security Considerations for Artificial Intelligence Agents" (Docket No. NIST-2025-0035), submitted March 9, 2026.
23. Díaz, S., Kern, C., Olive, K. "Google's Approach for Secure AI Agents." Google. 2025.
24. Willison, S. "The Lethal Trifecta for AI Agents: Private Data, Untrusted Content, and External Communication." Published June 16, 2025.
25. Meta. "Agents Rule of Two: A Practical Approach to AI Agent Security." Published October 31, 2025.
26. Dziemian, M. "How Vulnerable Are AI Agents to Indirect Prompt Injections? Insights from a Large-Scale Public Competition." *ArXiv*. Published March 16, 2026.
27. Franklin, M., Tomašev, N., Jacobs, J., et al. "AI Agent Traps." *SSRN*. Published March 28, 2026.
28. Meta Engineering. Introducing AutoPatchBench: A Benchmark for AI-Powered Security Fixes. Published April 29, 2025.
29. Lin, J. W., et al. "Comparing AI Agents to Cybersecurity Professionals in Real-World Penetration Testing." *arXiv*. Published December 10, 2025.
30. Deason, L., et al. "CyberSOCEval: Benchmarking LLMs capabilities for malware analysis and threat intelligence reasoning." Published September 15, 2025.