What is AI agent security?

AI agent security is the practice of protecting against both the risks of AI agent use and threats to agentic applications. It involves securing the agents themselves and the systems they interact with, helping to ensure that they operate as intended without being exploited for harmful purposes.

Agents are AI systems that are designed to function autonomously by planning, making decisions and calling external tools. It is critical to protect against both external cyberattacks and unintended actions taken by the agents. Because agentic AI is a rapidly developing field, the threat landscape is evolving in real time alongside the technology.

One defining feature of AI agents is their ability to perform tool-calling, in which they connect to an API, database, website or other tool and use it when needed. Tool-calling is typically orchestrated through AI agent frameworks and APIs.

In theory, agents use tools to augment their own capabilities in the planning and completion of complex tasks. For example, a customer service agent could interact with a customer, then connect to an internal database to access that customer's shopping history.

Multiagent systems take things one step further by combining several agents to delegate complex tasks into smaller chunks. A central planning agent manages the agentic workflow while worker agents complete their assigned portions of the task.

Autonomous AI decision-making and tool-calling combine to present a broad two-pronged attack surface. Hackers can manipulate the agent's behavior and cause it to misuse tools, or attack the tool itself through more traditional vectors such as SQL injection. AI agent security seeks to safeguard agentic AI systems against both types of threats.

Industry newsletter

The latest AI trends, brought to you by experts

Get curated insights on the most important—and intriguing—AI news. Subscribe to our weekly Think newsletter. See the IBM Privacy Statement.
We use your email to validate you are who you say you are, to create your IBMid, and to contact you for account related matters.
Business email

Your subscription will be delivered in English. You will find an unsubscribe link in every newsletter. You can manage your subscriptions or unsubscribe here. Refer to our IBM Privacy Statement for more information.

Your subscription will be delivered in English. You will find an unsubscribe link in every newsletter. You can manage your subscriptions or unsubscribe here. Refer to our IBM Privacy Statement for more information.

The agentic AI threat landscape

Agentic AI systems offer a greater range of vulnerabilities when compared to stand-alone AI models, such as large language models (LLMs), or traditional software applications. Even without the presence of an attacker, agents themselves can present security risks when not properly managed and maintained with clear guardrails, permissions and access controls.

The AI agent threat landscape covers:

- Expanded attack surface
- Autonomous actions at speed
- Unpredictable inference
- Lack of transparency

Expanded attack surface

Agents are often incorporated into larger systems that include APIs, databases, cloud-based systems and even other agents (multiagent systems). Each element in the agentic system presents its own suite of vulnerabilities. Attackers have a range of tools and exploits at their disposal to target potential weak points in the agentic workflow.

Autonomous actions at speed

Agentic automation means that agents act without receiving explicit instructions from a human user. Agents can act at speed, potentially liaising with other agents that are also doing the same thing at the same time. Each of these agent actions and outputs presents an attack opportunity and an amplification vector should an attacker succeed in compromising an agent or an entire agentic system.

Unpredictable inference

Inference is the process by which LLMs and other generative AI models, including agents, make decisions. In short, they use statistical modeling to “infer” the most likely output for any input. Because inference is probabilistic, model outputscannot be fully predicted, which introduces uncertainty into agent behavior.

As such, cybersecurity providers cannot perfectly anticipate what an agent will do. This unpredictability complicates the nature of agent threat mitigation as compared to traditional cybersecurity techniques.

Lack of transparency

Many AI models, such as OpenAI’s GPT models and Anthropic’s Claude, are not open source. It isn’t possible to “look inside” the model and figure out how it makes its decisions. And even open source models do not offer full transparency, given the inherently complex and opaque nature of how models arrive at outputs.

Cybersecurity personnel working with agentic systems might have a harder time conducting root cause analysis and formulating incident response plans.

Agentic AI vulnerabilities

The multifaceted nature of the agentic threat landscape introduces a range of vulnerabilities that attackers can exploit.

AI agent security vulnerabilities include:

- Prompt injection
- Tool and API manipulation
- Data poisoning
- Memory poisoning
- Privilege compromise
- Authentication and access control spoofing
- Remote code execution (RCE) attacks
- Cascading failures and resource overload

Prompt injection

Prompt injection is one of the most severe vulnerabilities of any large language model (LLM), not just AI agents. But with agents, the risk is magnified because agents can take autonomous actions. In a prompt injection attack, the attacker feeds adversarial inputs to the LLM that instruct it to behave in unintended ways. The agent can be instructed to ignore safety and ethics guidelines, send phishing emails, leak data or abuse tools.

An indirect prompt injection attack hides the malicious prompt in the agent's data source rather than feeding it to the model directly. When the agent calls on the data source, such as an external website, the malicious prompt is delivered to the model. Multimodal agents capable of fielding multiple data types are especially vulnerable to this type of attack—each form of data that the agent can process is a potential attack vector.

Goal manipulation versus agent hijacking

Goal manipulation and agent hijacking are often the desired outcomes for prompt injection attacks. With goal manipulation, attackers tweak the way an agent approaches tasks and makes decisions by altering its goals or thought process. Agent hijacking is an attack in which an attacker coerces an agent into performing unintended actions, such as accessing sensitive data.

Tool and API manipulation

Agentic AI is known for its ability to use tools and connect to APIs. But this same capability is also a vulnerability. Often through prompt injection, attackers trick an agent into misusing the tools to which it is connected.

Tool misuse can result in data leaks in which the agent exfiltrates sensitive user data to the attacker, or DDoS (distributed denial of service) attacks where the agent weaponizes its external connections. In such an attack, the agent coordinates a flood of connection requests to the target network, overloading it and forcing a shutdown.

Data poisoning

Data poisoning is the introduction of malicious data into an agent's training dataset or external data sources. Data determines how an agent learns, reasons and behaves. Corrupting its training data or data inputs can result in unintended behavior, such as data leakage.

For example, a coding agent might call on an external code library for reference. Slopsquatting—a portmanteau of "AI slop" and "typo squatting"—is when someone deliberately registers a code library name that is similar to that of a legitimate library. The intent is for the model to accidentally pull a subset of code from the fake library and add it to its generated code.

Along with tool misuse, data poisoning is a component of supply chain exploitation: where an attacker infiltrates and corrupts the system surrounding an AI agent.

Memory poisoning

Memory poisoning is the corruption of an agent's persistent memory: the data it retains that keeps it informed about what it has recently been doing. Memory poisoning attacks are intended to shape the agent's future behavior by altering its understanding about prior actions.

Privilege compromise

An agent situated at the center of an automated workflow has system permissions that enable it to access the data and tools it needs for its assigned tasks. If agents are not monitored, they might retain or be granted excessive permissions beyond what they require.

If these privileges aren't removed when the agent no longer needs them, they're no longer adding value—but they are still a potential attack vector. Attackers can exploit an agent's permissions to send messages, execute transactions, give itself more permissions, alter systems, read sensitive data and more.

Authentication and access control spoofing

If attackers manage to steal agent credentials, they can pose as those agents to compromise the systems to which the agent has access. Spoofing the agent's identity gives attackers the same permissions that the agent has—anything the agent can do, the unauthorized user can do now as well.

Weak authentication protocols combine with machine learning to yield rapid lateral movement: when attackers move deeper into a network after an initial breach. Lateral movement opens the door to data exfiltration, phishing attacks, malware distribution and more. Attackers can also adjust the way the agent behaves to alter its future actions.

Remote code execution (RCE) attacks

Remote code execution (RCE) is a type of cyberattack in which an attacker injects malicious code into a system from a different location. With agents, attackers can have the agent run malicious code that gives the attacker access to the code execution environment. A common real-world example involves an attacker extracting user credentials from a compromised agent's host system.

Cascading failures and resource overload

Cascading failures and resource overload both result in the overwhelming of the agentic system. In a multiagent system, cascading failures occur when a compromised agent's output negatively affects the next agent in the network until the entire system is down.

Resource overload is similar to a DDoS attack against an agent: attackers overburden the agent with requests that exceed its throughput, potentially disrupting runtime altogether. From an end user's perspective, the agent-powered application appears to be down.

AI agent security measures

Despite the wide and varied threat landscape, agentic AI systems can be secured with effective countermeasures and AI guardrails. Adopting a proactive security posture and following the current best practices for vulnerability management can help ML and cybersecurity professionals secure AI agents and stay ahead of enterprising cybercriminals.

AI agent security best practices include:

- Zero trust architecture
- The principle of least privilege
- Context-aware authentication
- Data encryption
- Microsegmenting
- Prompt hardening
- Prompt validation

Zero trust architecture

Zero trust architecture (ZTA) is an approach to cybersecurity that assumes that no device on a network is trustworthy by default. Instead, every single network access request must be authenticated and authorized before it can proceed. Continuous monitoring and multi-factor authentication (MFA) help ward against threats.

Imagine the network as a website and an access request as a user of that site. With ZTA, there is no option on the login screen to check a box and have the site "remember me next time." The user must enter their password—and fulfill other MFA challenges—every time they want to log in.

By choosing to "never trust, always verify," ZTA reduces an attacker's capacity for lateral movement, reducing the attack surface and buying more time for security to respond.

The principle of least privilege

The principle of least privilege states that every device or agent in a network should have the lowest possible permissions needed for their responsibilities. It's equivalent to putting everyone and everything on a strict "need-to-know" basis. Role-based access control (RBAC) and attribute-based access control (ABAC) are two methods for maintaining privilege levels and increasing data security.

Context-aware authentication

Context-aware authentication allows agents to retrieve data only if the user is permitted to access it. Access permissionscan adjust dynamically depending on the agent's role, permissions or even the time of day.

Data encryption

In addition to minimizing access with the principle of least privilege, data can be further protected against compromised agents through encryption. Data in transit and at rest should both be encrypted with AES-256 encryption or similar. Data containing sensitive information, such as personally identifiable information (PII), should be anonymized as well to further protect employees and customers.

Microsegmenting

Microsegmenting is the design practice of breaking up networks and environments into individual segments. When agents can execute code, they should do so in sandboxed environments to prevent lateral movement. Strict runtime controls further strengthen the environment to contain the agent within the sandbox.

Prompt hardening

Prompt hardening is the AI security practice of giving LLMs strict, limited instructions that leave little room for misinterpretation. By constraining an agent to a narrow lane, ML systems designers can help limit an attacker's ability to trick the agent into performing unintended behaviors.

Prompt hardening techniques include disallowing the agent to disclose its instructions and having it automatically decline any requests that fall outside its restricted scope.

Prompt validation

Prompt validation checks prompts against predefined rules before they are passed to the agent. Also known as prompt sanitization or input validation, this practice helps insulate agents from prompt injection attacks. Similarly, outputs should be validated before use in case the agent is compromised.

Adversarial training

Adversarial training teaches models to recognize potential attacks by mixing deceptive inputs into the training data. Adversarial training is in ongoing development and has yet to become a standard set of training protocols.