cyberhaven

AI Security Buyer's Guide

Evaluating *AI Security* for the Agentic Era

Introduction

A fundamental change has occurred across enterprises in the last twelve months. Employees went from asking AI questions to handing it the keys to valuable data and applications.

AI has officially crossed a threshold: from a tool to prompt to an actor operating inside the IT environment. Today, AI agents read email, ship code, and query databases. And increasingly, they act without human oversight.

Cyberhaven Labs found that nearly 40% of AI interactions involve sensitive data, and the average employee shares sensitive data with AI tools once every three days.

What makes this moment different from every prior wave of technology disruption is the nature of the actor. In the network era, data had a home and security had a boundary. In the cloud era, that boundary dissolved. In the agentic era, the human is no longer guaranteed to be the center of an action, or even in the loop at all.

AI agents don't wait for approval. They take actions and chain workflows across systems faster than security teams can track. Prompt injection, tool poisoning, workflow hijacking: these aren't theoretical risks. They are happening now.

"By 2028, more than 50% of enterprises will use AI security platforms to secure third-party AI service usage and protect custom-built AI applications." — Gartner, Inc*

Most security programs were not designed for this. Endpoint detection and response (EDR) tools see system behavior, not data behavior. Legacy data loss prevention (DLP) fires on volume, not context. Cloud-based AI security tools are blind to agents running locally on endpoints. The result is a rapidly expanding attack surface with inadequate coverage.

This guide frames seven criteria for evaluating AI security programs, across both GenAI tools and autonomous agents, for security leaders and the practitioners responsible for making these programs work. Each criterion reflects what the current threat model actually requires, not what legacy architectures were designed to do.

* Gartner

Section 01

Why AI Security Requires a Different Approach

The risks introduced by generative AI tools and AI agents are not extensions of familiar categories. They represent a qualitative change in how data moves and who, or what, is moving it.

The table below identifies the challenges that most security programs are not currently equipped to address.

Challenge	Why It Matters
Shadow agents and MCP servers run on endpoints	Locally deployed agents generate no footprint in cloud-based or SaaS inventories. Security teams often know about approved tools; they rarely know about the open-source agent frameworks, custom Model Context Protocol (MCP) servers, or locally installed coding assistants a developer spun up independently.
Agents inherit employee identity and permissions	An AI agent running on an employee laptop gains access to whatever that employee can reach (e.g production databases, source code, cloud storage). Broad access can become a single point of failure.
Violations emerge across conversation threads, not single events	An agent manipulated through prompt injection does not reveal itself in any one message. It reveals itself across a full execution thread, or a chain of tool calls, data accesses, and downstream actions that no single alert would surface.
There is no native audit trail for agent behavior	Agents don't authenticate or log the way humans do. When an agent reads a file, transforms it, passes output to a second agent, and stores the result elsewhere, no connected record of those events exists by default.
Sensitive data flows through AI pipelines without review	PII, source code, financial records, and regulated data move through agent workflows with no visibility into what reaches an external model, what is stored in a vector database, or what is surfaced in a downstream output.
Block-first controls accelerate shadow adoption	Blanket restrictions don't eliminate AI usage. They push it toward unmonitored alternatives: Personal accounts, unapproved apps, and locally installed agents that operate outside security visibility entirely.
GenAI SaaS and endpoint agents are governed by different tools	Browser-extension and network-layer tools are blind to agents running locally in IDEs, CLIs, and desktop frameworks. Developer laptops are the most common deployment environment for agentic AI, and the largest blind spot in most current architectures.

Section 02

Six Criteria for Evaluating AI Security

CRITERIA 1

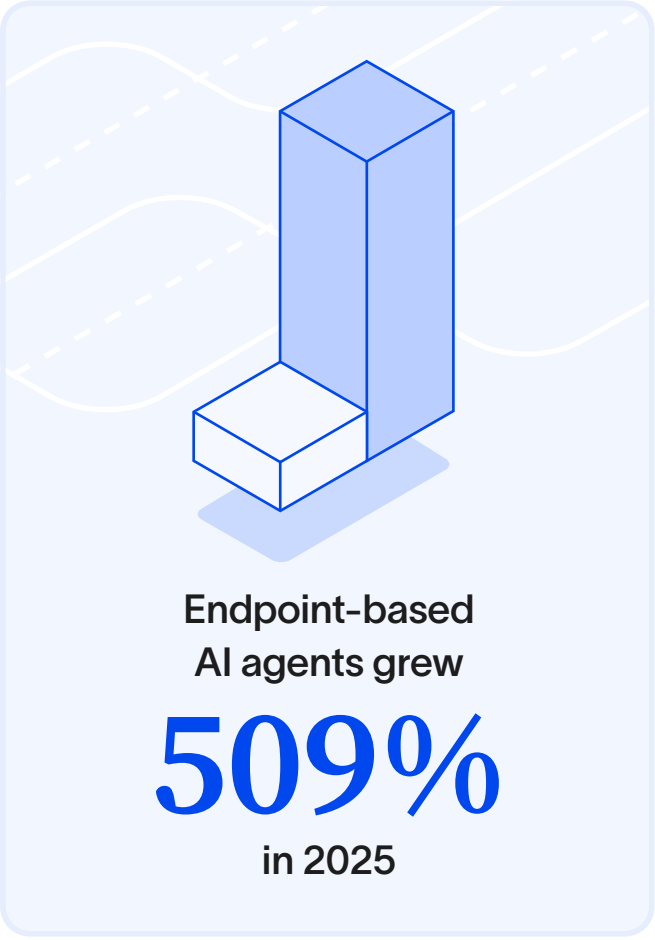
Continuous Agent and App Inventory Across Endpoints and SaaS

You cannot govern what you cannot see. The same dynamic that produced shadow IT is now producing shadow AI and shadow agents. Employees and developers deploy AI applications outside formal security review. Locally installed coding assistants, open-source agent frameworks, and custom MCP servers that generate no footprint in SaaS inventories and no telemetry in cloud-based security tools.

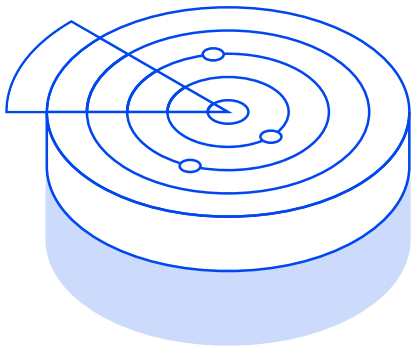
These tools are not accessed through a browser. Instead, they are installed directly on endpoints, with filesystem access, direct connections to code repositories, and in many cases, credentials to production systems.

Endpoint-based AI agents grew 509% in 2025, and enterprise adoption of coding assistants has jumped 357% year over year.

The visibility gap extends beyond tools employees actively choose to install. AI capabilities are now embedded directly inside enterprise SaaS applications, including generative features inside Salesforce, Slack, Google Workspace, and Microsoft 365 that employees use without recognizing them as AI tools at all. These embedded capabilities access the same sensitive data as any standalone AI application, but they generate no separate footprint and often fall outside the scope of AI governance programs focused on tool adoption. Effective inventory has to account for all three categories: standalone GenAI applications, locally installed agents, and embedded AI within the SaaS applications already in your environment.



An inventory that relies on user-reported intake, IT approval workflows, or browser-level telemetry will systematically miss the agents that carry the most risk. Effective visibility requires automatic, continuous discovery that reaches the endpoint.



"Nearly half of organizational data is considered sensitive or confidential. However, 32% of respondents to IDC's survey had more than 75% of sensitive data mapped and monitored."
— IDC*

Features to look for:

- Automatic discovery with no manual cataloging or user-reported intake required
- Coverage of locally installed tools, CLI frameworks, and MCP servers, in addition to AI features within SaaS applications
- Risk scoring across multiple dimensions for every discovered tool and agent
- Sanctioned, Unsanctioned, Tolerated, and Restricted classification at runtime
- Real-time updates as new tools and agents appear across the environment

How to evaluate:

Ask vendors whether they can enumerate Claude Code, locally installed coding assistants, and custom MCP servers running on developer endpoints without IT having approved or deployed them. If the answer requires browser telemetry or user intake, the inventory has a structural gap.

* IDC Spotlight: Rethinking Data Security and Insider Risk for Trusted AI Adoption

CRITERIA 2

Full Execution Lifecycle Observability, Not Prompt Logging

Prompt-level inspection tells you what a user typed. It tells you nothing about what the agent did with that input across a multi-step workflow. Violations in agentic AI emerge across conversation threads and chains of tool calls, not in individual messages. An agent manipulated through prompt injection may access a sensitive file in step three of a workflow, pass that data to an external tool in step five, and produce a compliance violation in step seven, with no single action appearing anomalous in isolation.

Effective observability requires a flight-recorder model, meaning a complete reconstruction of the agent's execution lifecycle, including the data it accessed, the tool calls it invoked, the actions it took, and the full multi-turn conversation context that connects those steps. That context is what makes violations visible and what makes investigations tractable.

Multi-agent workflows compound the challenge. When multiple agents work in sequence, each handoff is a potential gap in coverage. Observability that traces individual agents but cannot correlate across connected workflows will miss exactly the incidents that matter most.

Features to look for:

- Full conversation thread reconstruction, not isolated prompt-response pairs
- Tool call logging with data access tracing per agent action
- Multi-agent correlation across connected workflows
- Telemetry architecture built for autonomous software behavior, not human-paced event logging
- Forensic-grade context available for investigations without manual log correlation

How to evaluate:

Give a vendor a scenario: An agent reads a credentials file, passes output to a second agent, and that agent transmits data to an external model. Ask them to show you the complete audit trail. If the answer requires manually correlating logs from multiple systems, the observability architecture has a gap.

CRITERIA 3

Data Lineage That Follows Data Through Agent Pipelines

Every other AI security approach tells you what an agent did. Lineage tells you where the data came from, what it contained, and where it went next. That distinction is the difference between an alert and an investigation.

An extract from a production database carries the same sensitivity context whether it ends up in a spreadsheet, a Slack message, an embedding database, or an agent's output. Without lineage, enforcement fires on the fact that data moved. With lineage, it fires on whether that movement was risky, based on the full history of the data involved, not just its current location or format.

This matters acutely for agentic AI, because agents are precisely the mechanism through which data escapes its original context. An agent with read access to a filesystem can access confidential information, transform it, store it elsewhere, and surface it to users who would not otherwise have access to the source. Without lineage connecting those steps, the exposure is invisible until it becomes an incident.

Features to look for:

- Lineage tracking from data origin through agent pipelines, not only file-level tagging
- Persistence across copy/paste, format changes, and application transitions
- Lineage as a policy condition for enforcement decisions, not only a reporting feature
- Ability to trace data through MCP tool calls to external models and vector databases
- Lineage shared across AI Security, DLP, DSPM, and IRM so context is available across the full program

How to evaluate:

Ask whether the vendor can trace data from a regulated cloud source through an MCP tool call to an external model, and produce a complete record of that chain, without relying on file-level tagging or network-layer detection. If they cannot, enforcement decisions at the agent level will be based on incomplete information.

CRITERIA 4

Context-Aware Controls That Enforce Without Blocking Productivity

Block-first controls generate alert fatigue, push employees toward personal accounts and shadow AI usage, and create friction without meaningfully reducing risk. When an agent is blocked from completing a task, the user does not stop using agents. They find a workaround, often one that operates outside security visibility entirely.

Effective controls are context-aware. They can distinguish a developer using synthetic test data from an agent exfiltrating production PII through the same channel. They block, warn, or redact based on what is actually happening, and they explain the risk in plain language so employees understand the policy rather than resenting it.

The goal is not to eliminate risk by eliminating AI usage. It is to enforce governance at the moment of action, with enough context to be accurate, and enough clarity to change behavior over time.

Features to look for:

- Block, warn, and redact options at the prompt and response level
- Plain-English policy explanations instead of generic block pages
- Option for the user to revise and proceed when a prompt contains sensitive data
- Context-aware enforcement based on data sensitivity, agent risk score, and action type, not only content pattern matching
- Coaching that improves employee behavior over time rather than generating alerts that analysts ignore

How to evaluate:

Ask vendors to demonstrate what a user sees when a prompt contains sensitive data. A generic block page is not an acceptable answer. The response should identify the specific risk, explain the policy, and give the employee a path forward.

CRITERIA 5

Unified Coverage and Policy Enforcement Across GenAI and Agentic AI

AI usage does not stay neatly within one surface or one application. An employee might start a workflow in a browser-based GenAI application and continue it through a locally installed agent. A developer might paste sensitive data into Copilot in a browser, then invoke Claude Code from a CLI on the same file. Governing these two interactions with separate tools, separate policy engines, and separate consoles creates structural gaps at exactly the seams where data flows most freely.

Effective AI security reaches every surface where sensitive data is at risk: browser-based GenAI SaaS applications, locally installed coding assistants, unmanaged devices, contractor machines, and ChromeOS environments that cannot support a full endpoint agent. Coverage and policy need to be consistent across all of them.

This is not only a coverage problem. It is a policy coherence problem. When two tools govern the same data movement differently because they do not share a classification layer or a policy engine, enforcement is inconsistent by design. Security teams end up managing exceptions in one product that the other product would have blocked.

Features to look for:

- → Browser-level coverage with content inspection and cloud account context for GenAI SaaS governance
- → Endpoint-level coverage for locally installed agents, CLI frameworks, and MCP servers
- → Detection of personal versus corporate AI tool usage at runtime, without requiring endpoint deployment on every device
- → A single policy engine spanning GenAI SaaS and endpoint agents, managed from one console
- → Consistent classification context available regardless of whether the interaction starts in a browser or a local agent framework

How to evaluate:

Ask vendors how they handle a workflow that spans a browser-based AI tool and a locally installed coding assistant. Two separate consoles, two separate policy engines, or two separate classification systems is the wrong answer. Then ask whether the vendor can distinguish a user's personal ChatGPT account from a corporate-provisioned one, and whether that detection requires an endpoint agent on every covered device.

CRITERIA 6

Platform Coverage Across DLP, DSPM, IRM, and AI Security

AI security is not a standalone problem. An AI agent that exfiltrates data is also a DLP incident. An agent with access to a misconfigured cloud bucket is also a data security posture management (DSPM) finding. An employee using a personal AI tool to move sensitive data before they resign is also an insider risk signal.

Organizations that deploy AI security as an isolated capability will miss the connections across those categories that make investigations coherent and enforcement precise. The incident that looks like an AI security event in one product may look like a DLP policy violation in another and an insider risk indicator in a third. Without a shared data layer connecting those products, security teams reconstruct the same event chain three times from three different systems.

"Trusted AI outcomes require trusted data foundations. Unified visibility, contextual analysis, and insider risk management are essential." — IDC*

The strongest programs build AI security on the same platform as DLP, DSPM, and IRM, sharing a data lineage foundation, a unified classification layer, and a single policy engine. That architecture is the difference between a security program that can reason about AI risk in context and one that generates alerts that cannot be connected to anything.

Features to look for:

- AI Security built on the same platform and data layer as DLP, DSPM, and IRM
- Shared lineage layer that connects AI security findings to the broader data security program
- Single classification framework and policy engine, not siloed modules requiring separate configuration
- Unified console and reporting for AI security, DLP, and IRM without additional interfaces
- No duplicate policy engineering for organizations already running DLP or IRM on the same platform

How to evaluate:

Ask vendors whether an AI security alert can be correlated with a DLP event on the same data movement, in the same interface, without manual log correlation. Then ask whether the same classification context that drives DLP enforcement also drives AI security enforcement. If the answer to either question is no, the platform is a collection of products, not a unified program.

* IDC Spotlight: Rethinking Data Security and Insider Risk for Trusted AI Adoption

Section 03

AI Security Readiness Checklist

Use the following checklist to assess your current program posture before evaluating vendors or selecting a platform. Each question corresponds directly to one of the seven criteria above.

A "no" answer identifies a gap. An organization without agent inventory cannot build observability, and an organization without observability cannot enforce context-aware controls. The program is only as strong as its weakest foundation.

Discovery and inventory

Can you enumerate every AI app and agent across your organization, including locally installed tools that do not appear in your SaaS inventory?	○ Yes	○ No
Can you classify each tool as sanctioned, unsanctioned, tolerated, or restricted, based on automated discovery rather than user-reported intake?	○ Yes	○ No
Do you have risk scores for every AI application and agent in your environment, updated continuously as new tools appear?	○ Yes	○ No

Controls and enforcement

Can you enforce policy at the data level, based on data sensitivity and agent risk score, rather than blocking tools or destinations by category?	○ Yes	○ No
Can you distinguish a developer using synthetic test data from an agent accessing production PII through the same channel?	○ Yes	○ No
Do your controls explain the risk to employees in plain language, with a path forward, rather than presenting a generic block page?	○ Yes	○ No

Observability and investigation

Can you monitor AI app and agent activity in real time?	○ Yes	○ No
Can you reconstruct the full execution lifecycle of an agent interaction: The data it accessed, the tool calls it made, the actions it took, and the multi-turn conversation context connecting those steps?	○ Yes	○ No
Can you correlate activity across multiple agents working in sequence, without manually stitching together logs from different systems?	○ Yes	○ No

Coverage and policy coherence

Does your AI security coverage reach browser-based GenAI tools, web-based and local agents.	○ Yes	○ No
Can you detect whether a user is accessing AI tools through a personal account versus a corporate-provisioned account, without requiring a full endpoint deployment on every device?	○ Yes	○ No
Do your GenAI and agentic AI policies share a classification layer, so the same data context drives enforcement across both surfaces?	○ Yes	○ No

Conclusion

Govern AI Without Slowing Down Business

The goal of this guide is not to slow an organization's AI adoption. It is to give security teams the tools to govern it accurately, without defaulting to blunt controls that drive employees toward unmonitored alternatives and create new data risks.

Every consideration in this guide points toward the same underlying requirement that enforcement has to follow the data, not the tool. The landscape of AI tools will keep changing. New agent frameworks, new MCP servers, and new local models will appear faster than any allowlist can track. Organizations that anchor their AI security program to a tool-centric model will spend their budget chasing coverage gaps. Organizations that anchor it to data will scale with adoption rather than against it.

Discovery, observability, and controls are the three capabilities any AI security program needs to function. Discovery tells you what is running. Observability tells you what it is doing with sensitive data. Controls let you act on that information before damage occurs. Each depends on the others, and none works in isolation.

The connective tissue across all three is lineage. Every other AI security approach tells you what an agent did. Lineage tells you where that data came from, what it contained, and where it went next. That is the difference between an alert and an investigation, and it is only possible when AI security is built on a lineage foundation that has already been mapping data across endpoints, SaaS, and cloud environments.

That foundation also matters when you evaluate platforms. AI security that was retrofitted onto a legacy DLP engine, or bolted onto a cloud-based monitoring product, cannot provide endpoint-level visibility into locally installed agents. It cannot trace lineage through multi-step agentic workflows. And it cannot enforce context-aware controls at machine speed without generating the alert fatigue that erodes security programs over time.

Cyberhaven was built around data, not perimeters. The platform delivers AI Security, DLP, DSPM, and IRM from a unified console, governed by a single policy engine, grounded in the same lineage layer. For organizations navigating the shift from GenAI tools to autonomous agents, that architecture is the difference between a security program that keeps pace with AI adoption and one that is always one deployment behind.

Request a demo