unosecur

WHITEPAPER

Approving AI Agents for Production:

A Control Model for Agent Access

Table of contents

Approving AI Agents for Production 02

Why agent-by-agent approval fails 03

Where existing controls stop 03

What the research shows 05

The Approval Boundary 06

Visibility · Identity 08

Least privilege · Input protection 09

Output protection · Audit and compliance 10

Response · What changes for security teams 11

Where you stand: a maturity model 11

Conclusion 12

Why Unosecur Ark AI 13

Approving AI Agents for Production

AI agents now operate inside enterprise systems. Coding agents run on developer endpoints, and assistants and automated workflows call tools through the Model Context Protocol (MCP). They read tickets, query databases, send messages and change code, using access that belongs to real people.

Before an agent goes into production, security has to answer five questions.

- **Attribution:** Which identity does the agent act as, and can each of its actions be traced to a person?
- **Scope:** Which systems and tools can it reach, and is that scope limited to the task?
- **Persistence:** How long does its access last after the task ends, or after the user leaves?
- **Data exposure:** What sensitive data can flow through its prompts and tool calls, and is anything inspecting it?
- **Accountability:** Can its actions be reconstructed, and can it be contained without shutting AI down?

Most enterprises cannot answer these with evidence today. Identity providers authenticate the user at sign-in. Network controls see traffic, not intent. Neither sees the tool call, where the agent actually acts.

Reviewing agents one at a time does not close the gap. Agents and their tools change faster than reviews run, and when approval is slow, teams deploy without it. This paper proposes a different unit of approval: the Approval Boundary. Security approves seven controls once, and every agent operates inside them. Each control covers part of the agent lifecycle, is enforced outside the model and produces its own evidence. Unosecur Ark AI enforces all seven on one platform, tied to identity.

- **Visibility:** Every agent in use is discovered and inventoried.
- **Identity:** Every action runs under, and is attributed to, a named person or service account.
- **Least privilege:** Access is scoped to individual tools, granted just in time and measured against use.
- **Input protection:** Prompts and tool arguments are inspected before data leaves.

- **Output protection:** Tool responses and metadata are untrusted until inspected.
- **Audit and compliance:** Every action is logged, risk-scored and mapped to AI security frameworks.
- **Response:** One agent, or one MCP client, can be stopped without stopping anyone else.

Why agent-by-agent approval fails

Most security reviews for AI work one agent at a time. A team describes what the agent does, security assesses it, and the agent goes live or waits. That model breaks in three places.

- **Agents change after approval.** A new MCP server is added, a tool gains a write operation, or a tool description changes. The agent that was approved is no longer the agent that runs.
- **Reviews do not scale.** Every new use case needs a new review. Security becomes the bottleneck, and AI investment stalls in pilot.
- **Slow approval drives shadow AI.** When the sanctioned path is slow, teams install agents and add servers without it. The organization carries the risk, and nobody decided to accept it.

Underneath all three is one mismatch. The questions security must answer at sign-off are about runtime behavior. The evidence available today is about configuration. An approval model that works has to produce runtime evidence for every agent, including agents that do not exist yet.

Where existing controls stop

A tool call passes through the user, the AI client and its model, the authorization server, the MCP server, and the downstream system the tool reaches. The request travels out through each of them. The response travels back and enters the model's context, where it shapes the next call.

The MCP authorization specification covers part of this path. For remote servers it

requires OAuth 2.1, audience-bound tokens and no token passthrough to upstream systems. Authorization itself is optional, and local servers that use standard input and output take credentials from the environment instead. The MCP tools specification says servers must sanitize tool outputs and clients should validate tool results before passing them to the model. It does not define how. Each deployment decides.

Enterprise controls already touch parts of the path. Each was built for a different unit of work, and none sees a tool call end to end.

Existing control	What enterprise controls already cover	What it misses in a tool call
Identity provider and SSO	Authenticates the user at sign-in	Which tool the user's agent calls afterwards, and what the tool returns
OAuth scopes	Limit what a token can request from a server	Individual tools inside a server, unless the server maps scopes to tools
Server allowlist	Limits which MCP servers a client may use	What an approved server's tools return, or how their descriptions change
API gateway	Routes and rate-limits service traffic	The user behind an agent's call, and natural-language instructions inside payloads
SIEM	Stores and correlates logs	Calls that were never logged with a user, client and tool attached
Endpoint protection	Detects malicious processes on the device	A legitimate coding agent using legitimate credentials to do the wrong thing

Unosecur Ark AI sits where these controls stop: on the tool call itself, under the identity of the person behind it.

What the research shows

Independent studies of MCP servers and applications point to the same gaps.

Finding	What it means for approval
91.8% of 414 internet-facing MCP servers lacked OAuth authentication	Calls to these servers carry no per-user identity
687 tool instances exposed shell execution with no access control	Approving a server can mean approving arbitrary command execution
Only 37.2% of 1,723 open-source MCP applications placed a blocking approval step before tool execution	Most tools run as soon as the model calls them
Tool poisoning appeared in 5.5% of 1,899 open-source MCP servers	Tool metadata can carry instructions before any tool runs
Tool poisoning attacks succeeded in 72.8% of cases against the most susceptible of 20 agents tested on 45 live servers	Agents follow instructions hidden in tool metadata

Two findings matter most for approval decisions.

The risk runs in both directions. Requests can exfiltrate data. Tool responses and metadata can inject instructions.

None of it trips a sign-in control. Each gap plays out as successful calls under valid credentials. No alarm fires, because nothing looks wrong.

The pattern maps to the OWASP Top 10 for Agentic Applications, chiefly Agent Goal Hijack, Tool Misuse and Exploitation, and Identity and Privilege Abuse.

Every one of these gaps opens after sign-in, on the tool call. That is where Unosecur Ark AI enforces the boundary.

The Approval Boundary

The Approval Boundary replaces review of each agent with seven controls security approves once. Unosecur Ark AI enforces each in code, outside the model, so no instruction given to the model can change it. Each produces evidence that it held.

Control	What security approves	How it is enforced	Evidence that it held
Visibility	Every agent in use is known	Discovery across endpoints, AI platforms, MCP traffic and custom agents. One live inventory of agents, owners and MCP clients	Agents discovered outside the sanctioned inventory
Identity	Every action runs as, and is attributed to, a named identity	SSO with OAuth token exchange to short-lived, per-user tokens, in place of shared keys	Share of actions attributed to a named user. Static keys remaining in client configurations
Least privilege	Which roles may use which tools, and which tools require just-in-time access	Tool-level authorization by user or agent. Just-in-time access with automatic expiry. Granted access measured against usage	Standing grants on sensitive tools. Granted tools never used
Input protection	What data may leave, and which operations are too risky to run unchecked	Data loss prevention on prompts and tool arguments. Intent analysis for bulk exports and deletions. Custom detection rules	Detections in prompts. Requests blocked or redacted, by server

Control	What security approves	How it is enforced	Evidence that it held
Output protection	That tool content is untrusted until inspected	Prompt injection and tool poisoning detection on responses and metadata. Redaction of sensitive data. Response size limits	Responses flagged, by server. Servers returning external content
Audit and compliance	What must be logged, and which frameworks it must map to	End-to-end activity logging with per-event risk scores. Per-agent risk posture. Continuous mapping to AI security frameworks	Compliance by framework. Time to reconstruct any tool call
Response	How narrowly an agent can be stopped, and who decides	Single-agent freeze, by hand or at a risk threshold. Owner warnings. Tickets. MCP client kill switch	Time to contain an agent. Agents stopped without wider disruption

Visibility

An agent nobody knows about cannot be approved. Agents arrive without a request, on laptops, in AI platforms and in production. Discovery that draws on endpoint management, AI platforms and MCP traffic brings them into one inventory without relying on opt-in, and keeps it current.

> **What it looks like:** A developer installs a coding agent and adds three MCP servers on a Friday. By Monday it is reading production data, and no request was ever filed.

In Unosecur Ark AI: Hooks for Cursor, Claude Code, Codex CLI, Gemini CLI, OpenCode and Windsurf are enforced through Kandji, Jamf Pro, Microsoft Intune or JumpCloud. Assistants are covered through compliance APIs, and every agent, MCP client and owner sits in one live inventory.

Identity

An action that cannot be tied to a person cannot be governed, revoked or explained. Shared keys and service accounts are the usual cause, turning every agent call into borrowed credentials. Per-user, short-lived tokens restore attribution and make per-user policy possible.

> **What it looks like:** An incident review traces a customer export to an MCP server. The log shows one shared API key, used by forty people.

In Unosecur Ark AI: Users sign in through SSO with SAML or OIDC and receive short-lived, per-user tokens. Every tool call is recorded against the person behind it.

Least privilege

Server-level grants give a read task the rights to write and delete. When an agent is steered by external content, its follow-up action stays within access already granted. Tool-level authorization reduces the blast radius of any one call. Just-in-time access eliminates standing privilege on sensitive tools. Measuring granted access against usage shows what can be removed.

> **What it looks like:** A support agent needs to read tickets. The same server exposes a delete tool, and the grant covers both.

In Unosecur Ark AI: Access policies grant all or selected tools on each server to a user or an agent. Users request temporary access to specific tools, and admins approve it. The Access Analyzer compares each user's granted tools with the tools they use, and removes unused ones in one click.

Input protection

A permitted call can still carry the wrong data. Prompts and tool arguments are inspected for credentials, PII and regulated identifiers, and high-risk operations such as bulk exports and deletions are flagged or blocked before they execute. The OWASP AI Agent Security Cheat Sheet recommends classifying actions by risk and gating destructive ones.

> **What it looks like:** A support agent needs to read tickets. The same server exposes a delete tool, and the grant covers both.

In Unosecur Ark AI: Access policies grant all or selected tools on each server to a user or an agent. Users request temporary access to specific tools, and admins approve it. The Access Analyzer compares each user's granted tools with the tools they use, and removes unused ones in one click.

Input protection

A permitted call can still carry the wrong data. Prompts and tool arguments are inspected for credentials, PII and regulated identifiers, and high-risk operations such as bulk exports and deletions are flagged or blocked before they execute. The OWASP AI Agent Security Cheat Sheet recommends classifying actions by risk and gating destructive ones.

> **What it looks like:** A developer pastes a stack trace containing a live cloud key into a coding agent. The prompt leaves the laptop before anyone sees it.

In Unosecur Ark AI: Data loss prevention covers prompts and tool arguments, with global and country-specific data types plus custom entities, and can flag, block or redact. Tool intent analysis flags or blocks bulk exports and deletions. Custom content rules catch data specific to your organization.

Output protection

The OWASP MCP Security Cheat Sheet treats every tool response as untrusted, including responses from approved servers. Responses and tool metadata are inspected for injected instructions before the model acts on them, sensitive data is redacted, and payload size is capped so a single read cannot return an entire dataset.

> **What it looks like:** An agent reads a web page through an approved server. Hidden text on the page tells it to send repository secrets to an outside address.

In Unosecur Ark AI: Prompt injection detection inspects tool outputs and tool metadata on every governed server. Data loss prevention redacts sensitive data in responses, and bulk response limits cap what a single call can return.

Audit and compliance

When an incident occurs, the questions are which identity, under whose authority, touched what and when. The record has to exist before the question is asked. Per-event risk scoring surfaces what matters in the stream, per-agent posture shows which agents drift, and framework mapping turns the same record into audit evidence.

> **What it looks like:** An auditor asks which agents touched regulated data last quarter, and under whose access. The answer takes three weeks of log pulls.

In Unosecur Ark AI: An immutable activity log records what each agent asked and did, with a risk score for every event. Sessions rebuild multi-step agent runs turn by turn. The compliance view maps findings to controls across OWASP Agentic, OWASP LLM, NIST AI RMF and MITRE ATLAS, with the evidence behind each control.

Response

Most teams already have a kill switch of sorts: lock out the person or switch AI off. It costs so much that it gets pulled late. Containment has to be precise: one agent stopped, its owner warned, the case routed to the right team, and every other team still working.

> **What it looks like:** One agent starts calling tools far outside its normal pattern at 2 a.m. The only option on the table is to switch AI off for the whole engineering team.

In Unosecur Ark AI: A security analyst chooses the response. Unosecur Freeze Agent stops one agent in seconds, by hand or automatically at your risk threshold. Owner warnings and tickets sit on the same record, and an MCP client kill switch cuts off a whole client.

What changes for security teams

New agents and new use cases that stay inside the boundary need no fresh review. Review is triggered only when the boundary changes: a new server, a new sensitive tool, or a server that begins returning external content. Security stops approving agents one by one and governs the limits they all share.

Where you stand: a maturity model

Each control moves through three stages. Most organizations sit at different stages for different controls. Production readiness means every control at Proven.

Control	Exposed	Governed	Proven
Visibility	Agents found by chance or by survey	Agents inventoried through endpoint telemetry	Inventory continuously current, with unsanctioned agents surfaced as they appear
Identity	Shared keys and service accounts	SSO with per-user tokens on most servers	Every action under a short-lived, per-user token and attributed to a person
Least privilege	Server-wide, standing grants	Tool-level policies by role	Just-in-time access on privileged tools, and access right-sized against usage
Input protection	Prompts and arguments uninspected	Data loss prevention on tool arguments	Prompts and arguments inspected, with high-risk operations gated before execution

Control	Exposed	Governed	Proven
Output protection	Tool output trusted by default	Injection checks on selected servers	Every response and all tool metadata inspected, redacted and size-limited
Audit and compliance	Logs without user or tool context	Attributed logs for tool calls	Risk-scored activity, continuous framework compliance, and an audit trail on demand
Response	Switch AI off, or do nothing	Manual revocation of user access	One agent frozen in seconds, or one MCP client cut off, with everyone else working

Conclusion

AI agents act through real people's access, one tool call at a time. Approving them one at a time cannot keep pace, and refusing to approve them pushes them out of sight. An Approval Boundary approved once, enforced at every call and measured continuously gives security a way to say yes, and the evidence to show afterwards that the yes held. Unosecur Ark AI puts that boundary into production.

Why Unosecur Ark AI

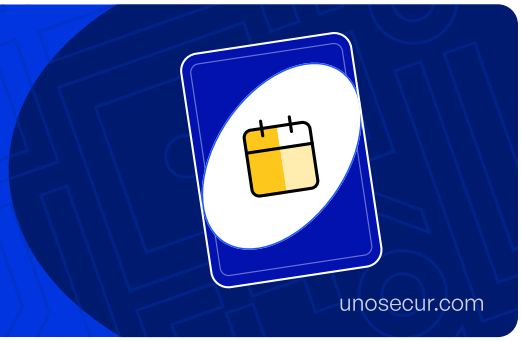
Unosecur Ark AI enforces all seven controls of the Approval Boundary on one platform. It finds agents across endpoints, AI platforms, LLM gateways, tracing platforms, identity providers, code repositories, MCP traffic and custom agents, and maps an owner to each. It records every prompt and tool call against the identity behind it, and scores the risk of every event and every agent.

At the MCP layer, it checks access policy on every tool call, injects upstream credentials so agents never hold them, and inspects requests and responses for sensitive data and injected instructions. When an agent goes rogue, a security analyst can freeze that one agent, warn its owner or raise a ticket, while every other team keeps working.

Ready to secure your AI agents?

Speak to the Unosecur team for an agentic posture assessment.

Book a demo



unosecur.com

Trusted by security leaders

ottobock.

"Being risk-efficient means **identifying risk, mitigating and remediating it** [with Unosecur]"

Henning Christiansen
Chief Information Security Officer, Ottobock