Cortex Agentic Endpoint Security

Securing the Next Endpoint

Introduction

AI agents are changing what ordinary endpoints can do. The endpoint has become the most vulnerable frontier for enterprise exploitation, not because attackers found a new door, but because enterprises quietly built thousands of new ones. Once an agent can access local files, browser sessions, credentials, applications, tools, or execution environments, it can act through the endpoint on a user's behalf. The endpoint remains the same laptop, workstation, server, or virtual machine security teams already protect. What changes is the degree of autonomy operating through it.

Today, 45% of Fortune 500 companies already have AI agents in production.

Agents can enter enterprise use through many paths, including browser extensions, desktop applications, workflow platforms, developer tools, MCP servers, agent marketplaces, and custom frameworks. Some introduce software or components onto the endpoint. Others use capabilities already present. In either case, security teams must account for more than whether an application or process exists. They need to understand which agent is active, what capabilities it can use, what it's attempting to do, and whether that action should be allowed.

Existing endpoint controls can detect many of the behaviors that result, including process execution, file changes, network activity, and other signs of compromise. What they miss, however, is the agent-specific context and policy controls needed to govern autonomous activity before it becomes an endpoint event. A destructive shell command, for example, may look like any other command once it executes. However, evaluating the action after the fact misses the core issue; the security question needs to begin earlier: Which agent initiated it, what gave the agent that capability, what context influenced the decision, and should the action have been permitted at all?

Agentic endpoint security addresses that gap.

Cortex Agentic Endpoint Security (AES) extends endpoint protection to the software, components, and autonomous activity associated with AI agents. It gives security teams four capabilities:

- **Gain continuous visibility:** Discover AI agents, models, extensions, MCP servers, packages, and other supported agentic software across managed endpoints, including where they appear and which versions are present.
- **Assess changing risk:** Continuously evaluate discovered items and individual versions as publishers, dependencies, behavior, and other risk conditions change.
- **Govern what enters and remains:** Apply guardrails, policies, and remediation to block, restrict, or remove software that violates organizational requirements.
- **Stop harmful agent actions:** Evaluate supported agent activity at runtime and block actions such as unauthorized file access, unsafe tool use, or destructive command execution before they reach the host.

Together, those capabilities address two related dimensions of the agentic endpoint: the software and components that give agents capabilities and the actions agents take with those capabilities once they are running.

The chapters ahead examine how the agentic endpoint emerged, why existing controls leave gaps, and how Cortex AES gives the SOC the visibility, context, governance, and runtime enforcement needed to secure autonomous activity at the endpoint.

*Daily usage: 82% of enterprise leaders now use Generative AI weekly, WHAIR

The Agentic Blind Spot

Every enterprise endpoint now runs a software stack no one fully sees. Driven by developer tools, vibe coding, and employee adoption, AI agents dynamically expand their capabilities using unreviewed extensions, MCP servers, packages, and skills. Because these integrations are pulled in at scale during real-time workflows, risk enters the environment long before traditional security controls can evaluate it.

Simultaneously, these agents operate with delegated human authority. Given a high-level goal, an agent autonomously determines which files to open, scripts to execute, or tools to invoke using the logged-in user's existing credentials and browser sessions. This creates severe runtime exposure (such as prompt injection or context manipulation) even when using entirely legitimate software.

The agentic blind spot sits directly at this intersection. Traditional security controls cannot monitor the dynamic, unreviewed software layer being built on the fly, nor can they interpret the agentic intent behind commands executing under normal user permissions.

Two Dimensions of Agentic Endpoint Risk

Securing the agentic endpoint consequently requires visibility into two related dimensions.

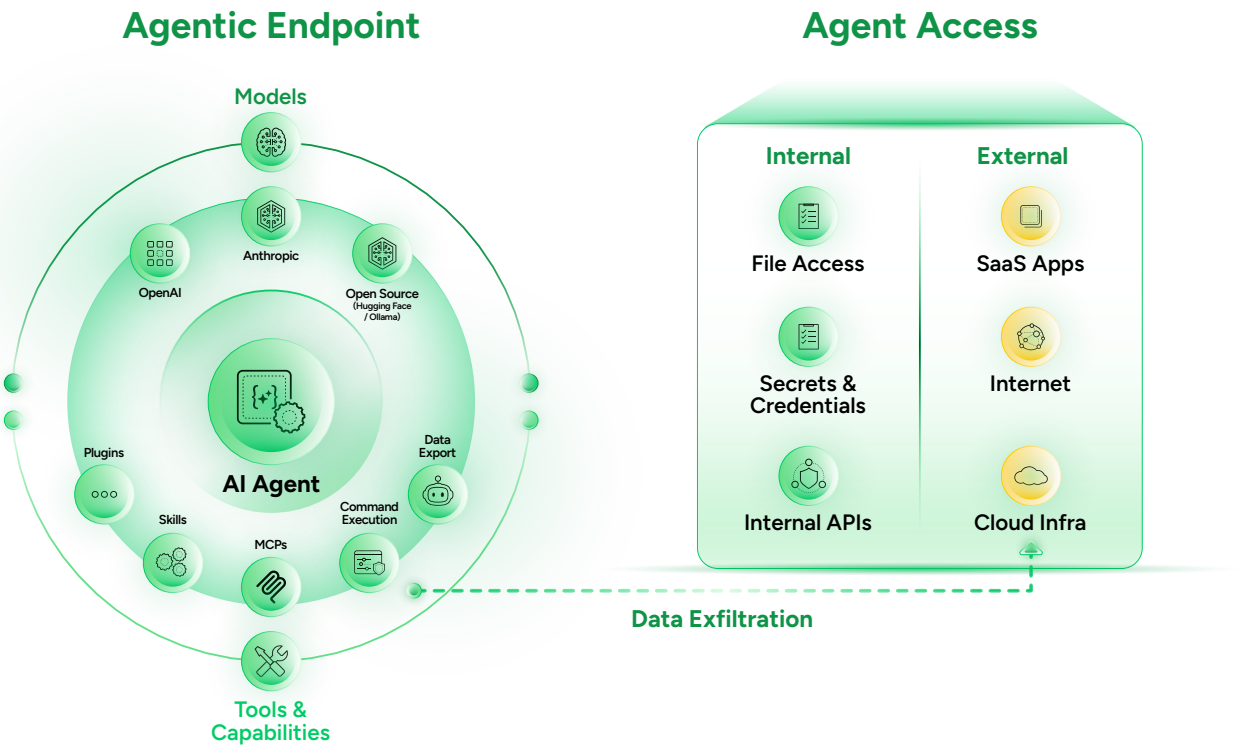


Figure 1. The Agentic Endpoint's Access

Agentic components are the software and connections that give an agent capabilities. Depending on the environment, they may include agent platforms, models, extensions, skills, plugins, MCP servers and tools, packages, or dependencies.

Security teams need to know which supported components exist on managed endpoints, where they came from, which versions are present, and whether their risk has changed.

Agentic actions are what an agent attempts to do with those capabilities and the authority available to it. They may include shell execution, file access, tool invocation, network requests, credential use, or changes to applications and systems.

Security teams need to know which agent initiated the action, what it attempted, whether policy permits it, and whether the action can be stopped before the host executes it.

The two dimensions can intersect. A compromised MCP server may give an agent a dangerous capability that later results in an unauthorized command. A vulnerable extension may expose functionality that malicious content can exploit. In other cases, a compromised component isn't involved. A trusted agent using trusted tools can still take a harmful action after receiving a malicious instruction or operating with excessive authority.

The distinction matters because neither component securit nor runtime monitoring covers the full problem.

Why Existing Endpoint Controls See Only Part of It

EDR and XDR remain critical to endpoint defense. They monitor processes, scripts, file activity, network behavior, exploitation, privilege abuse, and other signals needed to detect and investigate attacks.

Agentic activity introduces a different decision layer.

By the time an EDR platform observes a destructive command, the agent may already have interpreted the user's request, selected a tool, decided on the command, and passed it to the operating system. EDR can analyze the resulting behavior. It doesn't necessarily know which agent session produced the decision, what information influenced it, which agent capability enabled it, or whether policy should have prohibited the agent from taking that action in the first place.

Other endpoint controls address different pieces of the problem. Device management can inventory and govern applications. Application controls can restrict execution. Security teams may maintain allowlists for approved software. Software supply chain controls can evaluate packages and dependencies before deployment.

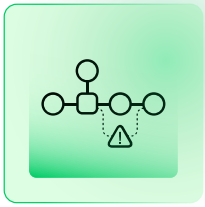
Each remains valuable. None was designed specifically to maintain continuous context around both the changing software that extends agent capabilities and the autonomous actions agents attempt through the endpoint.

Agentic Endpoint Security fills that gap. It adds agent-specific visibility and policy enforcement alongside existing endpoint controls rather than replacing them.

Trust Can Change After Approval

Agentic software also exposes a weakness in traditional approval models. Security teams commonly evaluate an application, extension, package, or other item before allowing it into the environment. Approval establishes trust at one moment.

Software continues to change.



Agentic Risk Takes More Than One Path

A trusted extension can become an agentic attack path

In ShadowPrompt, researchers found that a vulnerability in Anthropic's Claude Chrome extension could let a malicious website inject attacker-controlled instructions into the agent. The extension supplied the browser access and capabilities. Malicious content supplied the instruction. Anthropic patched the issue, and researchers advised users to run version 1.0.41 or later.

A legitimate tool can become malicious after users already trust it

The postmark-mcp package operated normally through its first 15 versions. Version 1.0.16 introduced a backdoor that secretly copied emails to an external address. Koi's risk engine detected the behavioral change in the new version, illustrating why approval at installation can't establish permanent trust.

A marketplace can distribute capabilities that agents use autonomously

Unit 42 found malicious skills in the OpenClaw ecosystem that delivered infostealers, established persistence, and exfiltrated cryptocurrency keys. The campaigns exploited the fact that skills extend what an agent can do and can turn a seemingly useful capability into an attack path.

The common thread: Agentic endpoint risk doesn't begin in one place. A component may become malicious, a legitimate component may contain a vulnerability, or malicious input may manipulate an otherwise trusted agent. The SOC needs visibility into both the capabilities agents acquire and the actions they attempt.

A new version can introduce a vulnerable dependency. A marketplace extension can change ownership. A previously legitimate package can become malicious. An MCP server can gain new functionality or access. An item security approved last month may no longer present the same risk today.

Version-level visibility lets security teams distinguish the trusted release from the risky one rather than treating every installation of an item alike. Continuous evaluation extends that principle further by reassessing software as new information, versions, and threats emerge.

Runtime protection addresses the other side of the problem. Even trusted software can participate in harmful activity when an agent is manipulated, overprivileged, or simply attempts an action the organization doesn't permit.

The SOC consequently needs security that follows the agentic endpoint beyond the initial trust decision: *what capabilities are present, how their risk changes, and what agents attempt to do with them.*

Closing this gap means replacing one-time approval with continuous evaluation:

- **AI-Native Total Visibility** into every AI agent, model, extension, and software artifact running across the enterprise, not just at install, but continuously.
- **Context-Aware Risk** that updates as software changes, rather than relying on a one-time installation check.
- **Proactive Governance & Control** over what enters the environment, blocking or routing risky components at the point of entry.
- **Real-Time Threat Prevention** that catches malicious behavior the moment it acts, not after the fact.

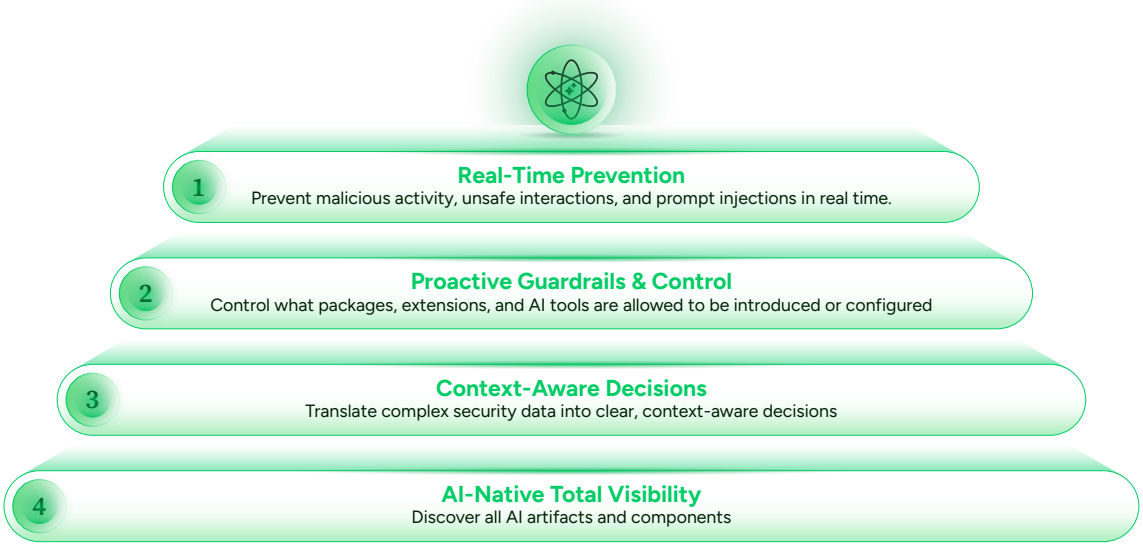


Figure 2. The 4 Pillars Needed for Agentic Endpoint Security

AI-Native Total Visibility

The SOC can't secure agentic activity that it can't identify or place in context. Traditional endpoint inventories answer familiar questions about applications, devices, and users. Agentic environments add another layer of detail, such as which agents are present, what extends their capabilities, where those components appear, and what activity is associated with them.

Cortex Agentic Endpoint Security (AES) provides continuous visibility into supported agentic software and components across managed endpoints, giving security teams a current view of what is present rather than relying on point-in-time discovery.

See the Software That Extends Agent Capabilities

Agentic functionality can arrive through many forms. Depending on the environment, Cortex AES can identify supported items such as browser and IDE extensions, packages, local models, MCP servers, agent platforms, skills, and related software.

Visibility into those components matters because they can materially change what an agent is able to access or do. An MCP server may expose new tools. An extension may add access to browser activity. A package may supply executable functionality. A model or agent platform may introduce a new runtime.

The security question extends beyond whether the software exists. Teams also need to know where it came from, which release is present, and how broadly it has spread.

For each discovered item, Cortex AES can surface attributes such as:

- Risk level
- Affected endpoints
- Marketplace or registry source
- Platform
- Version
- Publisher
- Installation method
- First and last seen

That inventory gives the SOC a way to move from an unfamiliar item to its organizational footprint.

Track Risk at the Version Level

Agentic software changes quickly. A trusted extension can receive a risky update. A new package release can introduce a vulnerable dependency. A previously acceptable component can change ownership or behavior.

Treating an item as permanently safe after approval leaves the SOC blind to those changes.

Cortex AES maintains version-level visibility so security teams can distinguish among releases of the same item. If one version becomes risky while another remains acceptable, analysts can identify which endpoints are affected rather than treating every installation alike.

That distinction becomes especially important during incident response. Instead of asking only, "Do we use this extension?" the SOC can ask, "Which users have the affected version, where is it installed, and how quickly can we contain it?"

Connect Inventory to Agent Activity

Inventory on its own doesn't explain what an agent is doing.

Cortex AES extends visibility into supported agent sessions and runtime activity so security teams can examine the context around agent behavior. Depending on the supported agent and action, that context can include the user account, model, extensions, skills, plugins, MCP servers, shell activity, file access, and blocked actions associated with the session.

The difference is operational.

A suspicious shell command no longer has to appear as an isolated endpoint event. The SOC can investigate it alongside the agent session that produced it and the capabilities available to that agent at the time.

The same principle applies to file access or tool use. Analysts gain more than a record of the resulting action. They gain agent-specific context that helps explain how the action originated.

What the SOC Can Answer

Continuous visibility gives security teams a foundation for investigation and response.

Instead of maintaining separate views of applications, packages, agent tools, and endpoint activity, the SOC can begin answering questions such as:

- Which supported agents and agentic components are present?
- Which users and endpoints have them?
- Which versions are running?
- Where did an item originate?
- Has its risk changed?
- Which agent session produced the activity?
- Which tools, extensions, or MCP servers were available to the agent?
- What action did the agent attempt?
- Where else does the same exposure exist?

Those relationships turn inventory into security context.

From Discovery to Prioritization

Visibility alone doesn't establish whether an item or action requires intervention. A widely deployed extension may be benign. A package installed on one endpoint may contain a critical security issue. A trusted agent may operate normally until new context, permissions, or software changes its risk.

Cortex AES uses the inventory established through continuous discovery as the foundation for ongoing risk evaluation. The next step is to determine which items and versions warrant attention, how their risk changes over time, and where the SOC should act first.

Context-Aware Risk

Discovery establishes what is present across managed endpoints. Security teams still need to determine which items deserve attention, whether a previously trusted component has changed, and which versions create exposure.

Cortex Agentic Endpoint Security (AES) continuously evaluates supported software rather than treating security review as a one-time event. Its risk engine analyzes individual items and releases as they appear, giving the SOC an updated assessment when software, publishers, dependencies, or threat intelligence change.

An Engine Built to Watch, Not Just Check

At the core of Cortex AES is an AI-powered risk engine that continuously evaluates the security posture of every software supply chain item on the endpoint. It's built to catch the full range of what can go wrong: malware and spyware, vulnerable dependencies, ownership transfers, and suspicious publisher behavior.

Rather than running a single scan, Cortex AES operates as an autonomous agent that orchestrates a set of specialized tools, including vulnerability scanners, secret detectors, sandbox environments, threat intelligence feeds, and AI-driven code comprehension. Together, these tools analyze both static structure and runtime behavior.

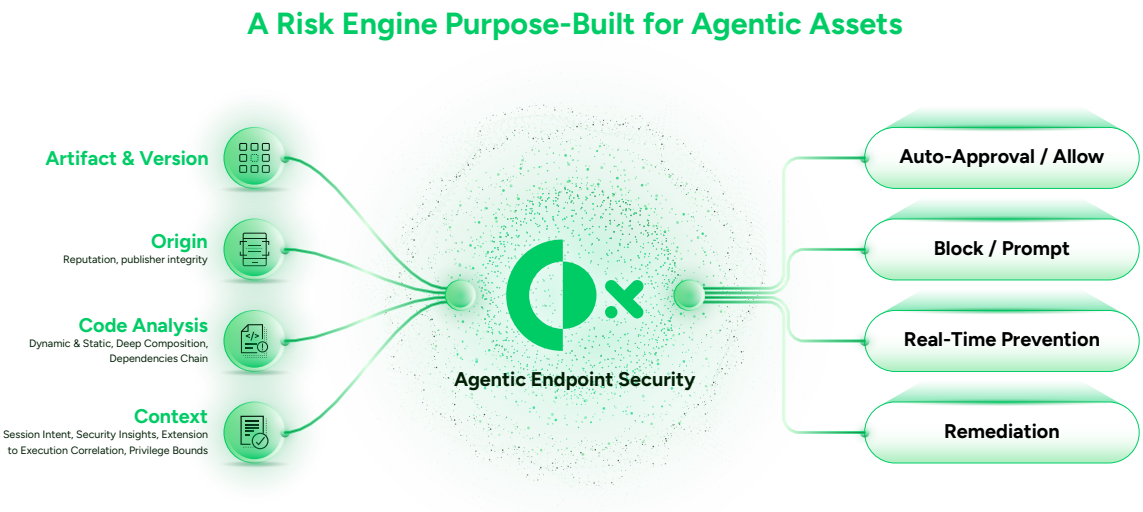


Figure 3. Cortex AES AI-powered Risk Engine

What the Engine Looks At

The AI Context Graph doesn't rely on a single method to assess risk. It orchestrates a set of specialized analysis tools, each surfacing a different signal, then combines them into one risk picture:

- **AI Code Analysis:** Reads and reasons about code the way a human analyst would, catching malicious intent in novel threats and obfuscation techniques that slip past traditional scanners.
- **Deep Composition Analysis:** Examines an item's structure directly, inspecting lifecycle hooks, dependencies, and any code that's been intentionally obscured.
- **AI-Native Security Analysis:** For agents, models, and MCP servers, evaluates what an artifact is designed to do, how it's expected to behave, what capabilities it exposes, and what permissions it holds, not just its source code.
- **Publisher Intelligence:** Looks at who built the item, including reputation history, account age, and any ties to known threat actors.
- **Dynamic Behavior Analysis:** Runs items in isolated sandboxes to observe what they actually do at runtime, including network activity and file access.
- **Marketplace and Threat Intelligence:** Flags suspicious signals like abnormal download patterns, inauthentic reviews, or known associations with active malicious campaigns.

How Findings Become a Risk Level

Each of these analyses produces individual findings, scored security indicators grouped into categories:

- **Malicious Behavior:** Confirmed or strongly linked malicious intent.
- **Exploitation & Vulnerabilities:** Technically exploitable weaknesses attackers can leverage.
- **Execution & Behavior:** Dangerous runtime capabilities frequently abused in attacks.
- **Data Exposure & Privacy:** Risk to secrets, sensitive data, or regulated information.
- **Publisher & Ownership Trust:** Signals that a publisher or source may not be stable or trustworthy.
- **Marketplace Signals:** Indicators of abandonment, manipulation, or instability in how an item is distributed.

Risk Per Item, Per Version

Here's what separates this from a one-time approval check: Cortex AES evaluates risk at the item and version level, not just once at install. Every time a new version of an item is published, Cortex AES automatically analyzes that specific version and assigns it its own independent risk assessment.

That distinction matters because risk isn't static. A new version can introduce malicious code, a vulnerable dependency, or risky new behavior. Ownership can transfer to an untrusted publisher. A build pipeline can be compromised. Software that was safe last month can become risky this month, and software flagged as risky can be fixed in a later release. Evaluating only at install time misses all of this.

By tracking risk per item and per version, Cortex AES lets security teams detect risky updates immediately, enforce policy based on the risk of a specific version rather than blanket rules for an entire item, and catch regressions when previously safe software becomes unsafe, all without over-blocking the tools your teams rely on daily.

Turn a New Risk Finding into Scope

Continuous assessment becomes operationally useful when analysts can connect a finding to exposure.

If Cortex AES identifies a newly risky release, the SOC can determine:

- Which version triggered the finding
- What security evidence changed the assessment
- How many managed endpoints have the affected version
- Which users or systems are exposed
- Where the item originated
- Whether other versions of the same item remain acceptable
- Whether policy or remediation already applies

The analyst no longer has to start with a threat report and manually determine whether the organization uses the affected software. Discovery and risk assessment establish that scope together.

From Risk to Control

Continuous assessment solves one part of the agentic endpoint problem. It identifies when software that gives an agent capability becomes unsafe or no longer meets the organization's requirements.

Risk assessment doesn't stop the item from entering the environment, remove an affected version, or determine whether an agent should be allowed to take a particular action.

Cortex AES uses its risk findings as input to guardrails, policies, and remediation. Security teams can then turn an updated assessment into an enforcement decision rather than leaving the SOC with another finding to investigate manually.

The security model moves from *know what is present to know when its risk changes and, next, control what happens when it does.*

Proactive Governance and Control

Continuous risk assessment tells the SOC when an item or version no longer meets the organization's security requirements. Governance determines what happens next.

Cortex AES's governance model is built on three complementary layers: guardrails, policies, and remediation, working together so enforcement is never a single static gate, but a continuous system that adapts as risk changes.

Apply Preconfigured Guardrails

Guardrails are out-of-the-box protections that deliver strong security with effectively zero setup. Enable them once, and they run seamlessly in the background, keeping your environment protected without adding friction to how your workforce works. Each guardrail clearly shows which marketplaces it applies to, so teams always understand its scope.

Guardrails include:

- **Malware Protection:** Blocks packages identified as malicious through Cortex AES threat intelligence.
- **Scan-First Protection:** Prevents installation of newly published supported software until the risk engine completes its analysis.
- **Version Update Cooldown:** Delays automatic updates for a defined period so newly released versions don't reach managed endpoints immediately.
- **Delayed Access:** Prevents installation of newly published software until it has existed for a defined minimum period and had time to establish a security record.
- **Auto-Remediate Delisted:** Removes supported items that a marketplace or source has delisted when configured criteria are met.
- **Sideloading Monitoring:** Identifies supported items introduced outside approved distribution channels.

Version Update Cooldown and Delayed Access address different moments. Update cooldown governs how quickly an existing item can move to a newly released version. Delayed access governs whether a newly published item can enter the environment at all.

Together, the guardrails reduce exposure during the period when a new item or release has the least security history.

Define Policies for Organizational Requirements

Where guardrails provide a strong default posture, policies let security teams define custom rules for what should happen when specific conditions are met. Policies automate actions such as alerting, removing, or blocking software based on criteria like risk level, security findings, publisher information, and other attributes, ensuring every extension, package, or agent in your environment adheres to your organization's specific security and governance requirements.

Policies give security teams:

- **Automation:** Monitoring and managing extension installation and usage without manual review of every item.
- **Customization:** Tailoring rules to your organization's specific security requirements rather than relying on generic, one-size-fits-all controls.
- **Compliance:** Helping ensure your environment consistently meets internal security standards and external regulations.

Together, guardrails and policies mean your governance posture is never based on a static status flag. Guardrails apply live threat intelligence automatically. Policies apply your organization's own logic against risk context that's constantly refreshed, so enforcement reflects what's true right now, not what was true at install.

Respond When Previously Trusted Software Changes

Continuous evaluation becomes most valuable when policy can respond to new information.

An item approved last month may become risky after a new vulnerability disclosure, publisher compromise, malicious update, or marketplace action. Cortex AES can identify the change and evaluate the affected item or version against current policy.

Security teams don't need to rediscover the exposure manually or wait for harmful behavior to reveal it.

The response depends on the finding and policy.

High-confidence conditions can support automatic remediation. For example, Cortex AES can remove an installed item identified as malicious during its next scheduled remediation cycle.

Other violations may require controlled action. Security teams can review the item, understand why it violated policy, and determine whether to block, remove, or otherwise address it. End-user notification and recorded enforcement actions provide an audit trail for what occurred and why.

Give the SOC an Actionable Governance View

Governance should reduce analyst effort rather than create another queue of disconnected findings.

Cortex AES gives security teams visibility into:

- Active policies and guardrails
- Current policy status
- Items in violation
- Affected endpoints
- Risk or policy conditions behind the violation
- Remediation already completed
- Remediation still requiring action

The SOC can move from a finding to scope and response without reconstructing the exposure from separate tools.

Separate Software Governance from Runtime Enforcement

Governance controls what software can enter, remain, or continue operating in the environment. It doesn't eliminate the need to evaluate what an agent attempts to do at runtime.

A trusted item can participate in unsafe activity. A manipulated agent can misuse an approved tool. A legitimate agent can attempt a command or file operation that violates organizational policy even when every installed component remains acceptable.

Cortex AES addresses that second problem through runtime enforcement.

The distinction is important. Governance controls which capabilities an agent can acquire and retain. Runtime protection controls what the agent is allowed to do with them.

Real-Time Threat Prevention

Governance can reduce risk before an agent acts by controlling which software and components are allowed into the environment. Runtime risk begins when an agent uses those capabilities.

A trusted agent can still take an unsafe action. Malicious content can manipulate its reasoning. An approved tool can expose more access than the task requires. An agent with a legitimate objective can select a destructive command.

Cortex Agentic Endpoint Security (AES) evaluates supported agent activity at the point where intent becomes action, giving security teams a way to enforce policy before the endpoint carries out the operation.

Enforce Policy at the Moment of Action

Traditional endpoint telemetry becomes most useful after software has translated a decision into operating-system activity. Cortex AES adds an earlier control point for supported agent workflows.

Runtime hooks observe actions such as shell execution, file access, and tool invocation as the agent attempts them. Cortex AES can then evaluate the action against policy and allow it, block it, or require user approval where supported.

The distinction matters. Security doesn't need to determine whether the agent is malicious. It can evaluate the action the agent is attempting.

A compromised agent and a trusted agent can attempt the same destructive command. A manipulated agent can try to access a sensitive file even though the underlying application remains legitimate. Policy can intervene based on the attempted behavior rather than assuming that trust in the agent extends to every decision it makes.

Stop Harmful Actions Before Host Execution

Consider an AI agent attempting to run a shell command.

EDR can observe the resulting process, command line, and downstream behavior if execution occurs. Cortex AES can add agent-specific context before that point, including which supported agent initiated the action and the session in which it occurred.

If policy prohibits the command, Cortex AES can block it before the host executes it.

The same model can apply to supported file operations and tool calls. Rather than waiting for an unauthorized action to become an incident, security teams can enforce boundaries while the agent is still attempting the operation.

Runtime enforcement addresses a different question from software risk assessment.

Software assessment asks:

> ***Can the organization trust this component?***

Runtime enforcement asks:

> ***Should the agent be allowed to take this action now?***

Both decisions matter because a trusted component doesn't guarantee safe behavior.

Contain the Consequences of Manipulated Agents

Prompt injection and poisoned context illustrate why runtime controls can't depend on determining whether an agent intended to cause harm.

An attacker may place malicious instructions in content the agent is expected to process, such as a webpage, document, email, or other external input. The agent may interpret those instructions as part of its task and attempt an action the user never requested.

Cortex AES doesn't need to establish malicious intent to enforce policy against the resulting supported endpoint action. Runtime hooks can evaluate what the agent attempts to do and block the prohibited operation before execution.

The same principle applies when risk originates from an MCP server, tool, skill, or another agent. A compromised component or manipulated interaction may influence the agent's decision, but the enforcement point remains the action the agent attempts through the endpoint.

Runtime protection consequently limits the impact of threats that bypass component-level controls or arise without a compromised component at all.

Four Enforcement Points

Cortex AES can apply controls at multiple points before and after software reaches the endpoint.

Source

The Inline Security Gateway evaluates supported downloads from public sources before they reach the endpoint. Security teams can block risky packages, extensions, and other covered software based on policy and available risk intelligence.

Figure 4. Inline Security Gateway

Installed State

The endpoint agent can enforce policy against supported software already present on the device. If an item later becomes malicious, violates policy, or requires remediation, Cortex AES can remove or otherwise address the installed copy.

Agent Runtime

Runtime hooks evaluate supported agent actions as they are attempted, including shell commands, file access, and tool calls. Policy can block or require approval for actions that exceed the organization's defined boundaries.

Binary Execution

On macOS, Santa extends policy enforcement to traditional binary execution. Binary controls complement agentic protections by governing executable activity that may result from or accompany agent workflows.

The four enforcement points cover different stages of the problem. Source and installed-state controls govern software acquisition and persistence. Runtime controls govern autonomous agent behavior. Binary execution controls extend protection when activity crosses into traditional executable form.

What the SOC Can See and Act On

When Cortex AES intervenes, the SOC needs enough context to understand what occurred and determine whether further investigation is required.

Depending on the enforcement point and supported activity, security teams can see information such as:

- What action or item triggered enforcement
- Which endpoint and user were involved
- Which agent or session initiated the activity
- Which policy or guardrail applied
- Whether Cortex AES blocked, held, removed, or allowed the activity
- Whether analyst review or additional remediation remains necessary

Blocked activity shouldn't end the investigation automatically. A destructive command prevented at runtime may still indicate prompt injection, a compromised component, stolen credentials, or broader malicious activity.

Cortex AES provides the agentic context and enforcement decision. The SOC can then use that information to determine whether the event represents an isolated policy violation or part of a larger attack.

Extend Prevention into Investigation

Real-time prevention closes the gap between identifying risk and allowing it to become endpoint activity. It doesn't replace endpoint detection and response.

Cortex AES and EDR protect different decision points. AES can govern supported agentic software and actions before execution. EDR and XDR continue to detect malicious behavior across the endpoint, correlate events, support investigation, and contain activity that has already reached the operating system or extends beyond the agentic context.

Together, the controls give the SOC both sides of the problem—the ability to stop a harmful agent action when policy can intervene and the broader detection and response capabilities needed when an attack progresses beyond that point.

From Agentic Event to SOC Investigation

Blocking an unsafe agent action solves the immediate problem. The SOC still needs to determine why it happened, how far the exposure extends, and whether the event signals a broader attack.

Agentic incidents can begin with a compromised component, malicious input, excessive permissions, or misuse of a legitimate tool. The resulting endpoint action may look familiar to an analyst, but the path that produced it can involve an agent, model, extension, MCP server, tool, and user session.

Cortex Agentic Endpoint Security (AES) adds that agentic context to investigation.

Start with the Trigger

An investigation may begin when Cortex AES identifies a risky item, detects a policy violation, blocks a runtime action, or surfaces software whose risk has changed.

The first question is no longer simply what happened on the endpoint. Analysts need to establish what triggered the event and what role the agent played.

A newly critical package creates a different investigation path from a blocked shell command. A vulnerable MCP server requires different scoping from an agent attempting unauthorized file access.

Identifying the trigger gives the SOC a starting point for determining whether it's dealing with software exposure, unsafe autonomous behavior, or both.

Reconstruct the Agentic Context

Traditional endpoint investigation reconstructs activity from processes, files, users, network connections, and other telemetry. Agentic incidents add another set of relationships.

For supported activity, Cortex AES can provide context around the agent session, including the user account, model, extensions, skills, plugins, MCP servers, shell activity, file access, and blocked actions associated with the interaction.

Those relationships help analysts answer questions that endpoint telemetry may not resolve:

- Which agent initiated the activity?
- Which user or account was operating the agent?
- What capabilities were available during the session?
- Which tool, extension, skill, or MCP server was involved?
- What action did the agent attempt?
- Did Cortex AES block the action or allow it to proceed?
- Did the event involve software already known to be risky?

The idea is to understand the agentic path that produced it.

Scope the Exposure

Once the SOC identifies the relevant component or behavior, it needs to know where else the same risk exists.

Cortex AES combines continuous discovery with item- and version-level assessment so analysts can determine which managed endpoints have the same affected software or release.

That distinction can materially reduce the scope of an incident.

If version 3.2.1 of an extension becomes malicious while version 3.2.0 remains unaffected, analysts can identify the endpoints running the compromised release rather than treating every installation as exposed.

The same approach helps answer broader questions:

- How many endpoints contain the affected item?
- Which users have it?
- Are multiple versions present?
- Did the item arrive through the same marketplace or source?
- Has policy already blocked or remediated some installations?
- Does related agent activity appear elsewhere?

Scoping turns a single finding into an organizational assessment.

Determine What Requires Response

Not every AES event represents an active compromise.

A blocked package may never have reached the endpoint. A runtime policy violation may reflect unsafe agent behavior without malicious intent. A newly risky extension may require removal even though analysts find no evidence that anyone exploited it.

The SOC can use the available context to distinguish among different response needs.

A software-risk finding may call for removal, policy enforcement, and a hunt for affected versions.

A blocked autonomous action may warrant examination of the agent session, input, available tools, and user activity to determine whether prompt injection, compromised context, or misuse caused the attempt.

Evidence that malicious behavior reached the operating system may require broader endpoint investigation and containment.

The blocked event becomes the beginning of analysis rather than the end of it.

Coordinate Across Security Functions

Agentic endpoint security also crosses organizational boundaries.

Endpoint teams may manage agent deployment and device-level policy. Application or developer security teams may own exceptions involving packages, extensions, or development tooling. AI governance teams may define acceptable agent use and permission requirements.

The SOC becomes central when agentic risk requires investigation or response.

Analysts may need to:

- Triage the event
- Determine whether compromise occurred
- Scope affected users and endpoints
- Hunt for related behavior
- Contain malicious activity
- Coordinate component removal or policy changes
- Escalate broader identity, data, network, or cloud exposure

Cortex AES supplies the agentic context needed for those decisions. Broader detection and response platforms provide the telemetry and response capabilities required when the investigation extends beyond the agent.

Carry Agentic Context into Broader Response

An agentic attack rarely respects a product boundary.

A compromised agent may execute a process, access credentials, communicate with an external service, or create activity elsewhere in the enterprise. Once that happens, the SOC needs to correlate the agentic event with the rest of the attack.

Cortex AES establishes what happened at the agentic decision point. Cortex XDR® can extend the investigation across endpoint activity and other available security telemetry, helping analysts determine what occurred before and after the agent action and whether additional containment is required.

The combination gives the SOC a more complete investigative sequence: identify the agentic trigger, reconstruct the context, scope the exposure, determine the impact, and respond across the broader environment.

One Platform, One Agent with Cortex XDR

Cortex Agentic Endpoint Security (AES) adds visibility and control at the agentic decision layer. Cortex XDR extends protection across the broader endpoint and attack chain.

The distinction matters because agentic activity doesn't remain confined to the agent. A manipulated agent may launch a process, access credentials, modify files, communicate with an external service, or trigger activity elsewhere in the environment. AES can help the SOC understand and govern the agentic action. XDR provides the broader behavioral detection, investigation, and response capabilities needed when activity progresses beyond that point.

Protect Different Decision Layers

Cortex AES and Cortex XDR address different parts of the same security problem.

Cortex AES focuses on supported agentic software, components, sessions, and actions. It can help security teams determine what agentic capabilities are present, whether their risk has changed, what a supported agent is attempting to do, and whether policy should allow the action.

Cortex XDR analyzes endpoint and other security telemetry for signs of malicious behavior, including process activity, exploitation, persistence, credential abuse, lateral movement, and other attack techniques.

A destructive command illustrates the difference.

Cortex AES can evaluate the command as a supported agent attempts it and block the action before host execution when policy requires it. If execution occurs or related malicious behavior appears elsewhere, Cortex XDR can detect the resulting activity, correlate it with other events, and support investigation and containment.

The two products don't compete for the same role. AES governs the agentic decision point. XDR detects and responds to the broader consequences of an attack.

Deploy Alongside the Existing Endpoint Stack

Organizations can adopt Cortex AES without replacing their broader endpoint security strategy.

Cortex XDR customers can enable AES capabilities within the Cortex endpoint footprint, reducing the need for another independently managed endpoint deployment.

Organizations using another XDR or endpoint detection platform can deploy Cortex AES alongside the existing stack to add agentic endpoint visibility, software governance, and supported runtime controls while retaining their current detection and response platform.

Both approaches preserve the same architectural principle. Agentic endpoint security complements EDR and XDR rather than replacing them.

Extend Agentic Endpoint Security Into the SOC

When Cortex AES and Cortex XDR operate together, agentic security becomes part of the SOC workflow rather than an isolated governance function.

The SOC can use Cortex AES to identify risky agentic software, understand supported agent activity, and enforce policy at agentic control points. Cortex XDR can extend the investigation across endpoint telemetry and other integrated data sources, helping analysts determine whether a blocked or suspicious action connects to additional malicious behavior.

The combined model supports a broader response sequence:

- Identify the agentic trigger
- Understand the agent and session context
- Determine what reached the endpoint
- Correlate related activity
- Scope affected systems and users
- Contain broader compromise
- Feed lessons back into agentic policy

Agentic Endpoint Security closes a new control gap. Cortex XDR carries the resulting intelligence into the broader detection and response process.

Build Toward a Unified SOC

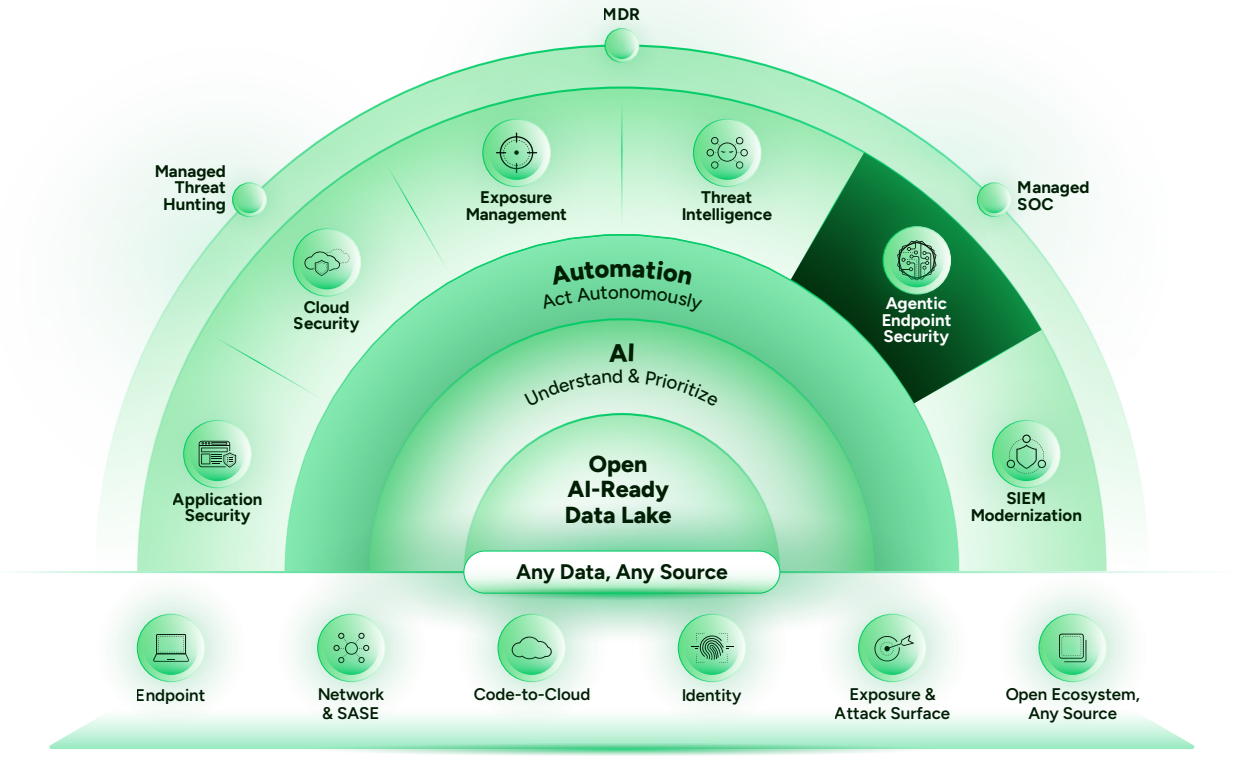


Figure 5. A Unified SOC with Cortex

For organizations already standardizing on Cortex, the shared endpoint footprint can also reduce operational friction as the security program expands.

Cortex AES adds agentic endpoint visibility and control without requiring the SOC to treat agent security as an entirely separate operational discipline. Cortex XDR provides the surrounding detection and investigation layer, while Cortex XSIAM® can extend those capabilities into broader security operations and automation.

The value lies less in consolidating products for its own sake than in preserving context across the security lifecycle. The same agentic event that begins as a policy decision can become an endpoint detection, an investigation, and a coordinated response without losing the context that explains how the activity started.

Extend Cortex AES with Unit 42 Expertise

Some organizations will want to operate Cortex Agentic Endpoint Security (AES) directly. Others may need additional expertise as agentic endpoint security matures and new attack techniques emerge.

Unit 42 Managed AES extends Cortex AES with dedicated security expertise, including proactive threat hunting, investigation and response, AI-related cyber hygiene, and guidance around agentic security governance.

The service can help security teams interpret unfamiliar agentic activity, investigate suspicious behavior, and respond when an AES finding connects to broader malicious activity. For SOCs building experience with agentic threats, Unit 42 provides an option to augment internal operations without changing the underlying Cortex AES security model.

Securing the Agentic Endpoint

AI agents have changed what ordinary endpoints can do.

A laptop or workstation no longer serves only as the place where a user runs applications and accesses data. When an AI agent can use the endpoint's files, credentials, browser sessions, tools, applications, and execution environment, the agent can make decisions and take actions through authority the organization already granted to the user.

That shift creates a new attack surface for the SOC.

Existing endpoint controls remain essential. They can detect processes, scripts, file activity, network behavior, exploitation, credential abuse, and the other signals required to identify and respond to attacks. Agentic activity adds security questions those controls were not designed to answer on their own.

Security teams now need to know what agentic software and components are present, whether their risk has changed, what capabilities they give an agent, what the agent is attempting to do, and whether security should allow the action before it reaches the host.

Cortex Agentic Endpoint Security brings those questions into the SOC.

Continuous discovery gives teams visibility into supported agents, extensions, packages, MCP servers, models, skills, and other agentic software across managed endpoints. Continuous risk assessment identifies when an item or version becomes unsafe. Guardrails, policies, and remediation govern which capabilities can enter and remain in the environment. Runtime controls enforce boundaries when supported agents attempt actions such as shell execution, file access, or tool invocation.

Together, those capabilities give the SOC a way to address both sides of agentic endpoint risk—what gives an agent its capabilities and what the agent does with them.

The operational test is straightforward.

Can the SOC identify the agentic software and components present on managed endpoints? Can it detect when a previously trusted version changes risk? Can analysts reconstruct the context behind an agent action and determine the scope of exposure? Can security stop a prohibited action before execution and investigate what caused the attempt?

Cortex AES is designed to make those questions answerable.

Combined with Cortex XDR, that agentic context can extend into broader endpoint detection, investigation, and response when malicious activity moves beyond the agent. The result is not a replacement for endpoint security. It's an extension of it for an environment in which autonomous software can now act through the endpoint.

The endpoint did not change. What can act through it did. Securing that difference is the role of Agentic Endpoint Security.

Cortex AES

Cortex Agentic Endpoint Security (AES) brings security to AI agents now acting through the endpoint. Continuous discovery brings hidden agentic activity into view and puts risk in context. Cortex AES turns that visibility into control, enforcing guardrails in the execution path to stop malicious or unauthorized actions before they reach underlying systems. Bring the agentic endpoint into view—and under control—at www.paloaltonetworks.com/cortex/agentic-endpoint-security.

About Palo Alto Networks

Palo Alto Networks (NASDAQ: PANW), the global AI cybersecurity leader, protects our digital way of life with a comprehensive portfolio of cybersecurity solutions and platforms across Network, Cloud, Security Operations, AI, and Identity. Trusted by more than 75,000 customers and powered by Unit 42® threat intelligence, our AI-driven platforms eliminate complexity, empowering enterprises to modernize with confidence and securing the speed of innovation. Explore the future of security at www.paloaltonetworks.com.

3000 Tannery Way
Santa Clara, CA 95054

Main: +1.408.753.4000
Sales: +1.866.320.4788
Support: +1.866.898.9087

www.paloaltonetworks.com

© 2026 Palo Alto Networks, Inc. A list of our trademarks in the United States and other jurisdictions can be found at https://www.paloaltonetworks.com/company/trademarks.html. All other marks mentioned herein may be trademarks of their respective companies.

cortex_wp_AgenticEndpointSecurity_092426