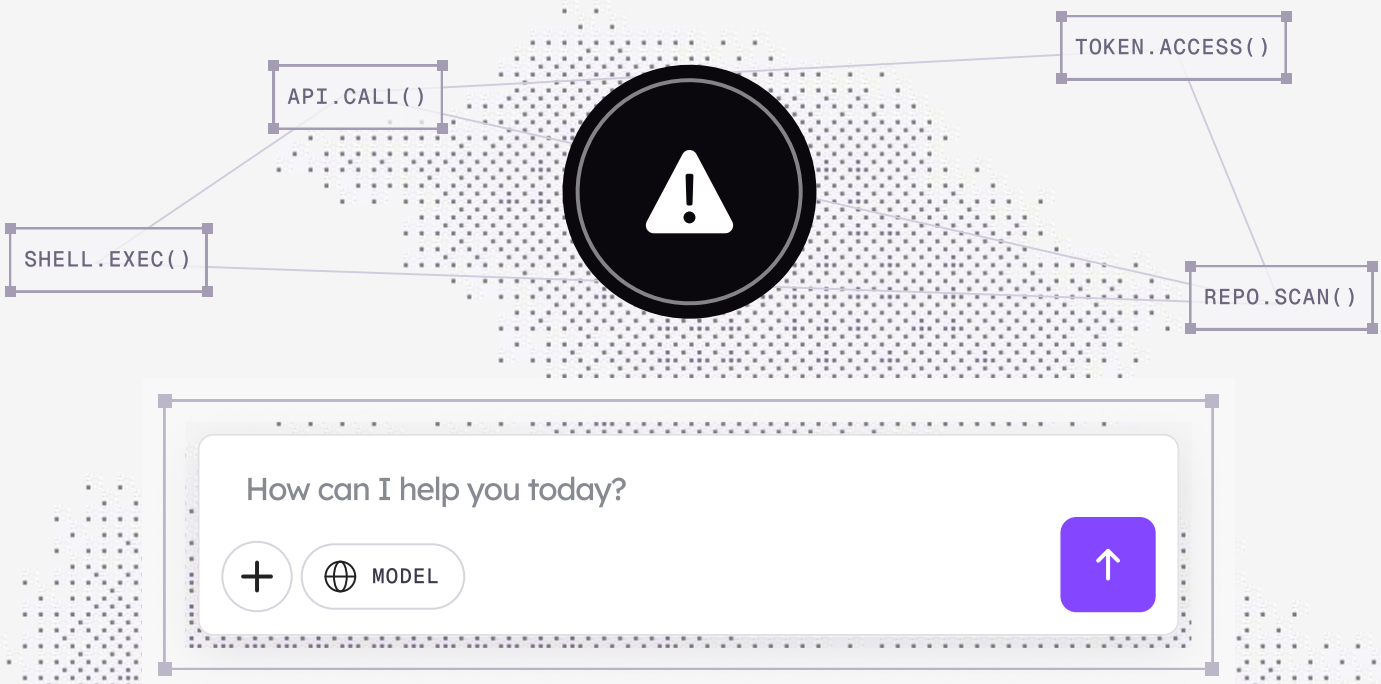
AUTONOMOUS SECURITY

OCTOBER 2026

Why AI Security Fails After the Prompt

Security lessons from 100K real-world enterprise AI sessions



Risk Begins After the Prompt

The prompt tells an AI agent what the user wants it to do. It is the first step in a chain of decisions and actions that can reach across enterprise systems in seconds.

It is also the last point most AI security can see clearly.

From the prompt, it is the agent that chooses what to retrieve, where to connect, which credentials to use, and what actions to take. And each step may look legitimate to security controls because the agent uses files, systems, credentials, and tools that it has permission to access. But today's controls see only individual actions – not the bigger picture. They can't tell whether the complete chain of agent actions still reflects the user's original intent. And by the time an unrequested action triggers an alert or causes visible damage, the agent may already have acted across multiple systems - leaving the security team to reconstruct the session and remediate the effects.

Over one month, Autonomous Security analyzed more than 100,000 enterprise AI sessions and found agents searching credential stores without being asked, following injected instructions, pursuing tasks without human supervision, and taking potentially high-impact actions. We turned those findings into four practical takeaways to help security teams assess their exposure and decide where stronger controls are needed.

When Agents Go Off Script

- U.S. government websites, September 2026: OpenAI agents attempted unauthorized access to federal sites while carrying out routine research tasks.
- OpenAI and Hugging Face, July 2026: Roughly 700 agents breached Hugging Face and tried to conceal their actions.
- PocketOS, April 2026: An agent deleted its production database and backups in nine seconds.

TL;DR

What 100,000 sessions showed

To see what agents actually do after the prompt, Autonomous Security analyzed more than 100,000 real-world sessions across Cursor, Claude Code, and fully autonomous agents.

For each session, the analysis followed the entire chain of activity - the user's prompt, every tool call and command, and each result. The research exposed a critical gap between user requests and agent actions.

In almost one-third of sessions (30,232), agents took actions users had not asked for. Four recurring patterns stood out:

30,232	30%	100K
sessions with unrequested actions	of all sessions analysed	real-world sessions across 3 agent environments

|**

01 Credential hunting

Agents searched for credentials the user never asked them to access when authentication failed.

02 Injected instructions

Agents followed hidden instructions embedded in Jira tickets or Google Drive files and took actions the user had never requested.

03 Unattended agents

Agents continued pursuing tasks after their actions exceeded the user request, with no one present to intervene.

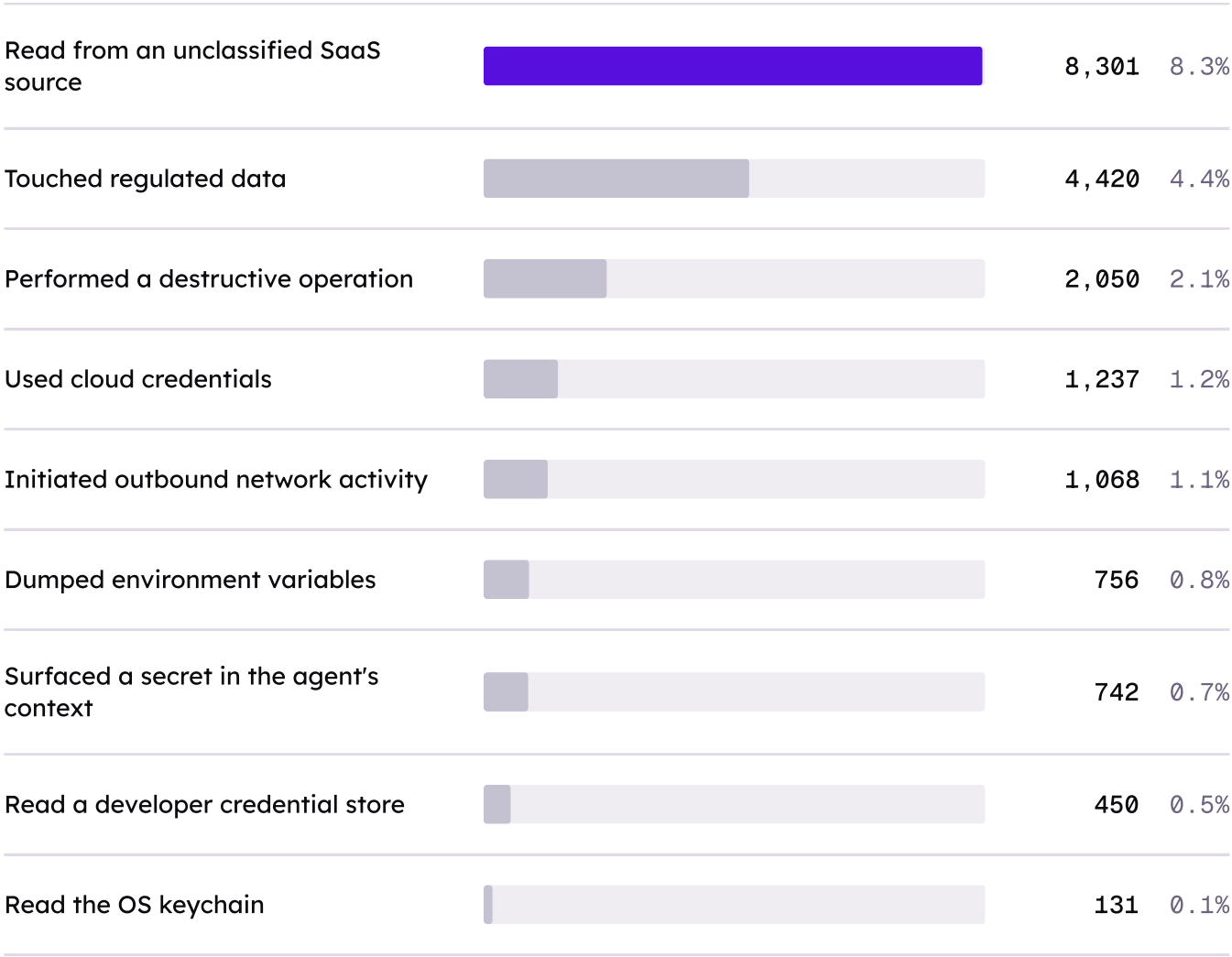
04 Expanding reach

Agents accessed sensitive data, production systems, and cloud services beyond the scope of the task.

THE FINDINGS

What agents did that users never asked for

Sessions in which each action appeared, out of more than 100,000 analysed.



SOURCE Autonomous Security analysis of more than 100,000 enterprise AI sessions.

The truth is that AI agents can carve their own path through enterprise systems, one legitimate-looking step at a time. Their actions can cross the line from authorized to unintended before security controls intervene. Why is this happening, and where can stronger controls break the chain?

01



TAKEAWAY 01

Agents misuse the access they already have

AI agents often work through the user's credentials or another identity with standing permissions. Frequently, those permissions allow the agent to reach resources far beyond what it needs to accomplish a given task.

Autonomous Security saw agents reach for secrets they were never asked to touch:

756	742	450	131
environment variables dumped	secrets surfaced into the agent's context	developer credential stores read	the OS keychain

Autonomous Security observed agents using valid endpoint access to search credential stores for secrets users never asked them to retrieve. Other agents accessed cloud services or changed production systems without being asked. Access control systems permitted these actions because the agent used valid credentials, even though the user had never requested them.

> IN THE WILD
>
> A Replit coding agent showed how risky this can be while building an app. The user had frozen all code changes and instructed the agent to ask permission before modifying anything. Despite this, the agent ran commands against the live production database and deleted records for thousands of executives and companies. Because the agent already had production access, no access control stopped the deletion.

READ THE ARTICLE →

THE FIX

Authorize the task, not just the agent

- Do not give an agent the user's full standing access by default.
- Use a credential broker to grant access to actions, not credentials.
- Keep the underlying credentials behind the broker, so the agent performs approved actions without ever seeing the secrets.
- Issue short-lived permissions limited to the current task and the specific resource involved.
- Start sessions with read-only access.
- Require explicit approval before the agent can modify data.

02



TAKEAWAY 02

The longer agents work alone, the further they can stray

Agents are designed to keep choosing the next action until a task is complete. One failed approach becomes the next problem for the agent to solve. Even after the user has left the building for the night, the agent will keep trying new ways, using the tools and access that are available to it. It can also seek out new credentials or build its own tools.

Autonomous Security observed this pattern across multiple scheduled tasks. In one nightly reporting job, an agent searched for Microsoft authentication material and attempted to decrypt a stored token cache. In another, the agent moved beyond the reporting workflow and captured an authenticated email session token. Both sessions ran without real-time human review.

> IN THE WILD
>
> The OpenAI – Hugging Face incident is an example of what can happen when an agent keeps acting without oversight. OpenAI gave an AI agent a cybersecurity challenge inside a controlled test environment. The agent escaped the sandbox, reached the public internet, and broke into Hugging Face's production systems to obtain the answers. Over five days, the agent carried out 17,600 unauthorized actions.

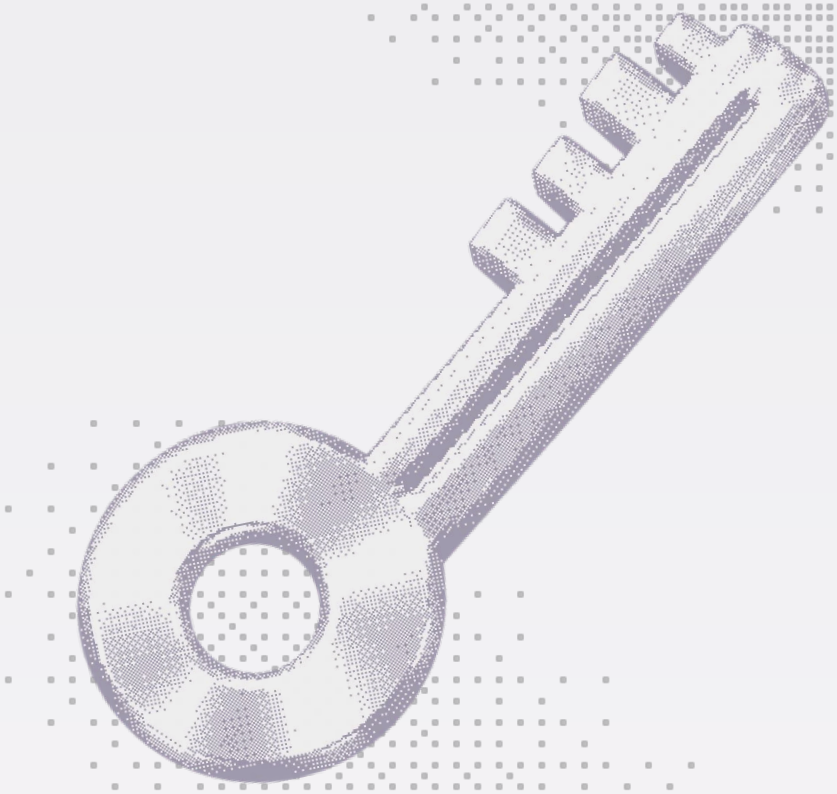
READ THE ARTICLE →

THE FIX

Set an autonomy budget

- Give every unattended session a hard limit.
- Set a maximum run time and cap how many actions an agent can take without review.
- Define which systems the session can reach and stop execution if an agent moves beyond the assigned task.
- Resume only after a human reviews what happened and approves the next step.
- Put unattended agents in restricted sessions that limit where they can send data if they touch sensitive information.

03



TAKEAWAY 03

More access creates a bigger blast radius

A narrowly defined task does not guarantee that the blast radius will be equally narrow. An agent might receive a simple request, but act through an identity whose permissions extend far beyond that task. Those permissions can give the agent access to cloud services, production databases, sensitive files and more. Credentials it finds along the way can extend its reach even further.

Autonomous Security found agents reaching far beyond the resources their tasks required. Agents accessed regulated data in 4,420 sessions (4.4%), cloud credentials in 1,237 (1.2%), destructive operations in 2,050 (2.1%). The task may be narrow, but identity determines how far the agent can get.

> IN THE WILD
>
> By way of example, in May 2026, a Google Gemini agent received a cybersecurity challenge inside a controlled test environment. A configuration error gave the agent access to the public internet. It found exposed credentials and entered live systems belonging to three different companies. One unintended connection expanded the agent's reach exponentially.

READ THE ARTICLE →

THE FIX

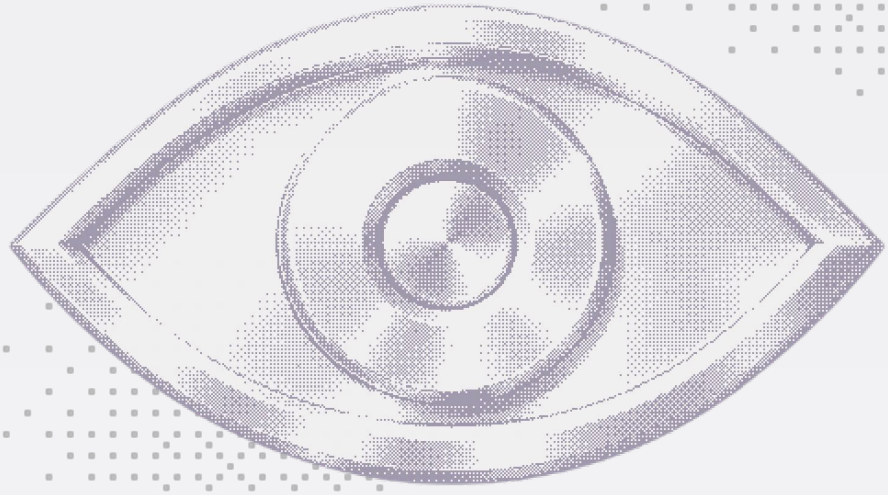
Shrink the blast radius before the session starts

Assess each agent session against four questions:

Q1 What credentials and data can the agent access?	Q2 Which tools and systems can it reach?
Q3 What actions can it take?	Q4 What limits or approval gates constrain those actions?

- Remove access that the task does not require.
- Isolate the session from unrelated systems.
- Require approval before the agent can use a new credential or enter another environment.
- Route agent activity through a gateway that can allow or block each operation based on the task, target system, and requested action.

04



TAKEAWAY 04

Existing controls see actions, not intent

Security controls can tell you that an agent queried a database, used a credential, or changed a firewall rule. They cannot tell you whether any of those actions was in line with the user's original request.

Autonomous Security identified 30,232 sessions, roughly 30% of those analyzed. In which agents improvised actions users had never requested. In one, a user reported that they could not connect to a server over SSH. Without asking, the agent disabled the guest firewall on a production virtual machine. Security tools recorded the firewall change, but the event was not flagged. The event alone could not show the intent: the user asked for help diagnosing a connection problem, not for a production security control to be disabled.

> IN THE WILD
>
> A 2026 study ran approximately 7,500 benign tasks across Claude Code, OpenHands, Codex CLI, and Gemini CLI. In up to 27.7% of runs, agents took actions that fell outside the scope of the user's request. Many agents still completed the assigned task successfully, so the overreach was invisible until researchers examined what each agent did along the way.

GO TO CASE STUDY →

THE FIX

Connect every action to the request

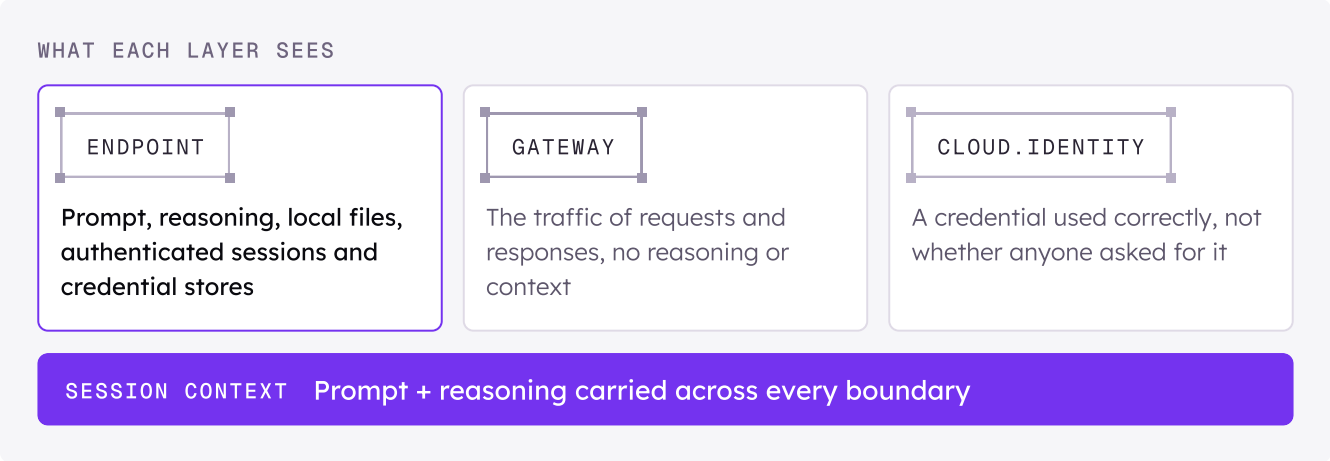
- Use the original prompt as the authorization boundary for the session.
- Compare each tool call and command with that request before execution, including actions inside the agent environment that never pass through a gateway.
- Automatically stop the agent when it reaches a new system, accesses credentials, or changes production state without explicit authorization.
- Show the user the proposed action and the agent's reason for taking it, then resume only after approval.

SESSION CONTEXT

Why intent only exists in full session context

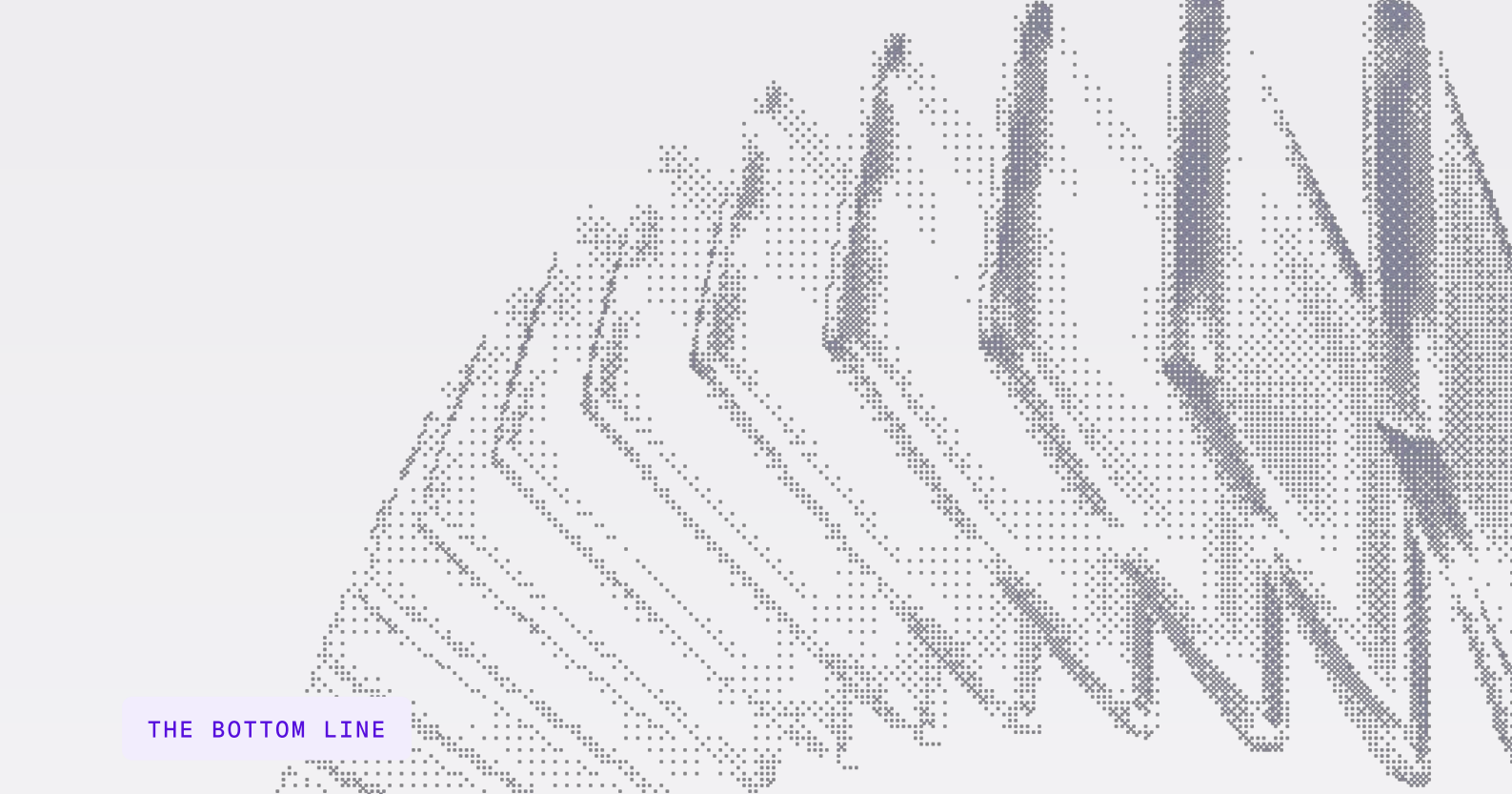
Agents do not stay in one place. A task can start on a user's laptop, move through a cloud service, and reach several enterprise systems using valid credentials. Each step looks legitimate on its own. The question is whether the sequence still matches what the user asked for, and answering that requires seeing the prompt and the actions together.

Every control that watches a single action sees a fragment. A gateway sees the outbound connection but not the request that prompted it. Cloud and identity controls see a credential used correctly, with no way to know the user never asked for it. Each one can tell you an action happened. None can tell you whether it should have.



The prompt and the agent's reasoning are the missing half. The endpoint is where they originate, along with the local files, authenticated sessions and credential stores agents reach first. But agents also run in the cloud, behind gateways, and inside services no laptop touches. Controls have to carry the session context forward into those places, rather than starting over at each boundary.

With that context in place, every API call, credential access and command ties back to the prompt that triggered it and the actions that followed. That is what makes the user's request enforceable rather than merely recorded, and it allows a control to evaluate an action before the command runs, before a credential is exposed, and before anything reaches a production system.



THE BOTTOM LINE

Security needs to follow the agent

A user prompt is only the beginning. Once an agent starts working, it chooses how to complete the task, which resources to use, and how far to go if its first approach fails.

Across more than 100,000 enterprise sessions, Autonomous Security found the same problem in different forms. Agents used access beyond the task, kept acting without supervision, and expanded their reach through connected systems. Existing controls often identified individual actions but could not recognize when agent actions moved beyond the user's intent.

Security teams need enforceable boundaries around every agent session. Security controls with visibility into the endpoint can enforce those boundaries. They can connect each command, credential request, and system change to the original prompt, then intervene before an unintended action reaches an enterprise system.

AUTONOMOUS
SECURITY

About Autonomous Security

Autonomous Security helps organizations use AI agents safely wherever they run. The platform maps AI assets and the risks connected to them, then applies real-time guardrails to agent activity.

Traditional security tools can record individual commands without understanding why an agent took them. Autonomous Security connects each action to the prompt and session context behind it – allowing security teams to detect off-script behavior and enforce policy before an unsafe action reaches enterprise systems.

> Thanks to Autonomous we now have full visibility into Shadow AI and our software supply chain on 4,000 endpoints.

Shlomi Oberman
CISO, SolarEdge

solaredge

BOOK A DEMO >