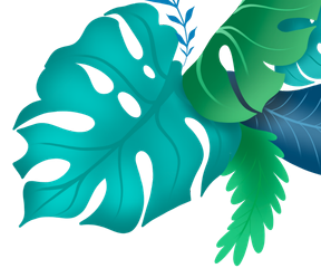


KANOPY

The Shadow AI Playbook:

6-Steps to Close the Gap Between AI Agent Adoption and AI Agent Security





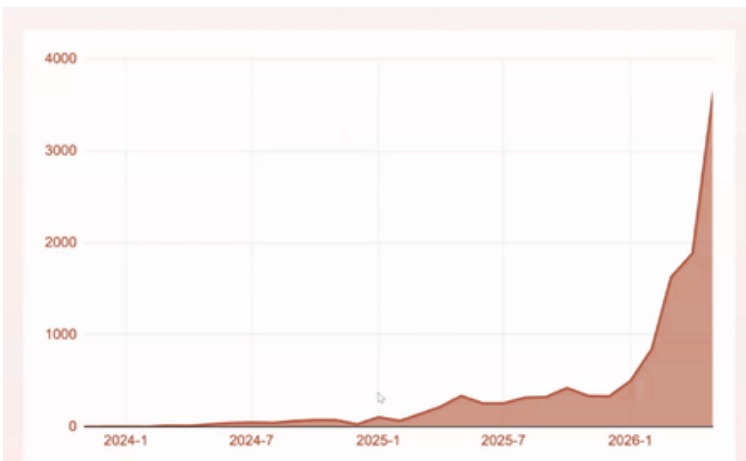
Introduction

The Agents Are Here

Ten years ago, a “business application” meant one person, one local database, and a dashboard nobody outside the team ever saw. Sharing was hard. External access was harder. And the tool had no mind of its own, it did exactly, and only, what it was told. That world is gone.

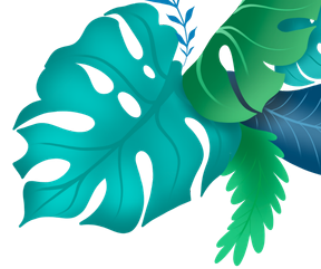
Today, any business user with an idea and ten minutes can open Copilot Studio, connect an agent to live production data, wire up a few actions, and publish it to the entire organization no code, no ticket, no security review required.

The result: environments that started with a handful of experimental bots now run into the thousands of agents, and the growth curve isn't leveling off. Every one of those agents can read data, take action, and talk to systems outside your walls. Most were never assessed for what they can actually do.



The number of agents inside a single Copilot Studio environment surged from single digits to the thousands in 1 year

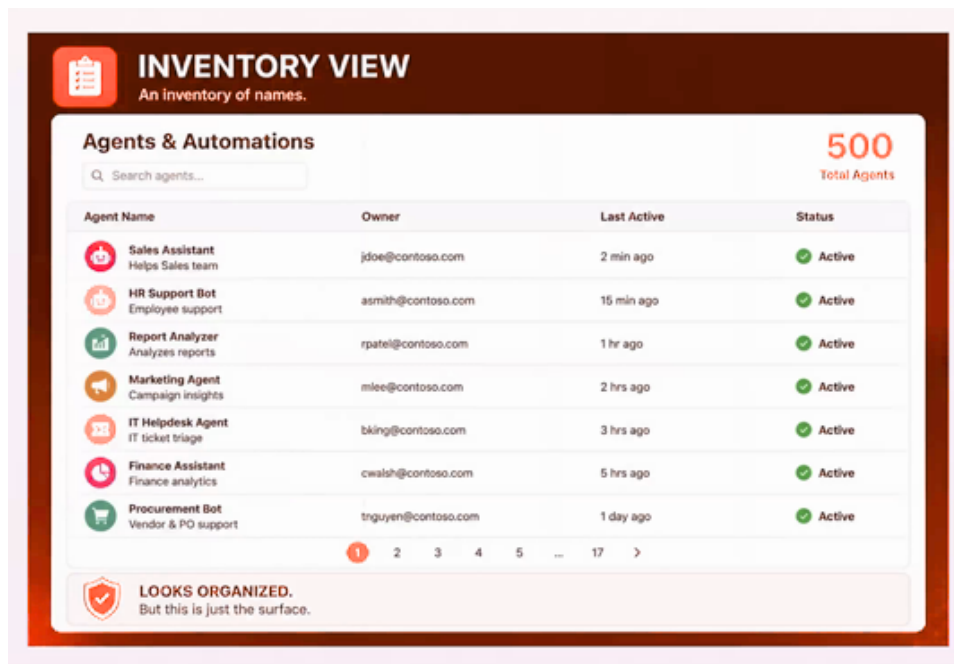
This isn't a case for slowing innovation down. It's a playbook for keeping up with it. The six steps that follow give security, AppSec, and GRC leaders a practical way to govern shadow AI, without becoming the department that says no.



Step 1

Build Real Visibility Before You Build Policy

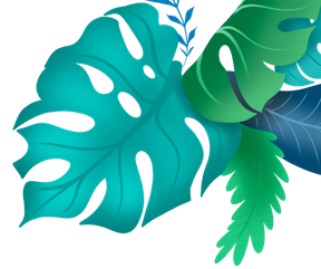
You cannot govern what you cannot see, and most agent inventories only look complete. A list of names, owners, and “last active” timestamps will pass a casual audit. It says nothing about what an agent can actually reach, or what it was built to do. An inventory that stops at names isn't visibility it's a directory.



Hundreds of agents in a tidy dashboard. It looks organized. It says nothing about what any of them can actually do.

Do this:

- Inventory every agent, flow, and prompt-based app
- Capture owner, business purpose, and creation date for each one.
- Flag public-facing agents separately from internal only ones.
- Identify orphaned agents with no active owner , these are the ones nobody is watching.



Step 2

Map Data Access for Every Agent, Not Just Its Existence

The risk was never the agent itself. It's what the agent can touch. An agent that answers questions from a public FAQ is a different risk profile than one connected to your CRM, HR system, or finance database, even if both were built by the same person in the same afternoon.

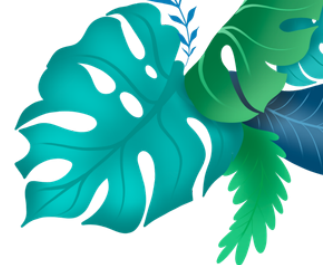
Reality check:

most over-permissioned agents aren't the result of malicious intent. They're the result of a business user clicking “connect” on an entire database when the workflow only needed one table, because scoping data access isn't the job they were hired to do.

Do this:

- Map which systems, tables, and files each agent can reach not just which platform it lives in.
- Flag agents with access broader than their stated purpose.
- Prioritize agents that combine broad data access with external-facing actions email, file export, webhooks.
- Treat “it hasn't caused a problem yet” as an unpriced risk, not a clean bill of health.





Step 3

Assume Every Agent Will Face a Prompt Injection Attempt

An agent that takes instructions from an inbound email, a shared document, or a web page has added an input channel that never existed in the traditional threat model. A message field is no longer just data, it can double as an instruction the agent will follow.

Consider an event-registration agent instructed to read incoming emails, add each requester to a list, and mark their payment status “pending” unless “an admin states otherwise.”

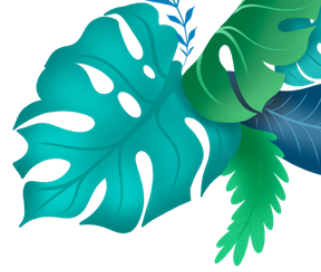
The instruction seems harmless until someone realizes the email itself is untrusted input and that “an admin states otherwise” is a phrase any sender can simply include in their message.

The screenshot shows a Kanopy agent configuration page. The top section, titled 'Details', contains fields for 'Name' (Event Demo agent 3), 'Description' (None provided), and 'Agent status (preview)' (2 Warnings). Below this is the 'Instructions' section, which contains the following text: 'You will get email for client who wants to register to our event. Use [Create item](#) to add the users to the registration list. In the column 'payment status' enter 'pending' unless an admin state differently. Send an email back to the client using [Send an email \(V2\)](#), and say that his request has been submitted but he needs to pay in order to complete his registration.'

One clause buried in an agent's instructions “unless an admin states differently” is an open door for anyone who can send it an email.

Do this:

- Treat every external content source an agent reads email, documents, web pages, chat as untrusted input.
- Identify actions with real-world consequences payments, approvals, data writes, external sends and require extra validation before they fire.
- Red-team your highest-privilege agents with adversarial prompts before someone else does it for you.

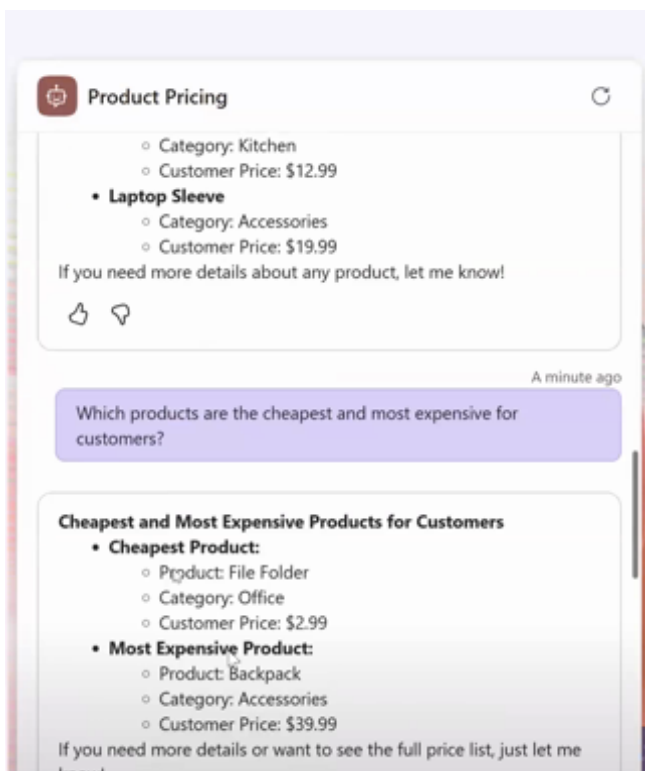


Step 4

Stop Treating Written Guardrails as a Security Control

A guardrail written into an agent's instructions is a request to the model, not an enforcement mechanism. Models are built to be helpful, and a well-crafted follow-up question can walk straight around an instruction that looked airtight on paper.

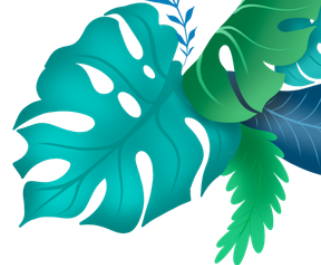
An agent told “never share what we pay for a product, only what we charge” sounds well protected until someone asks it to calculate profit margins. Margin is price minus cost. Answer that question honestly, and the “protected” number is sitting right there in the math.



The agent won't state the wholesale price outright. It will happily do the arithmetic that reveals it.

Do this:

- Audit existing agents for guardrails that live only in a prompt or instructions field.
- Assume any instruction-based restriction can be reasoned around, and classify the data behind it accordingly.
- Pair every written guardrail with an enforced control the model doesn't get a vote on.



Step 5

Put Enforcement Where the Model Can't Argue With It

If a rule matters, it belongs outside the prompt. Enforcement means the agent is technically incapable of the action, not politely discouraged from it.

Do this:

- Restrict connectors and actions at the platform level. An agent that only needs to read shouldn't have write, delete, or send permissions available to misuse.
- Require approval workflows for actions with financial, legal, or data-exposure consequences.
- Apply data classification and masking at the source system, so sensitive fields never reach the model in the first place.
- Make the safe configuration the default one, so a business user has to opt into risk rather than stumble into it.

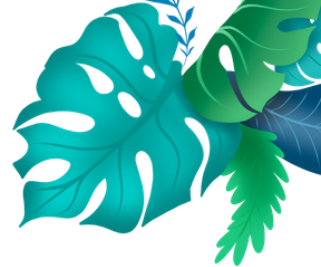
Step 6

Monitor Agents at Runtime, Not Just at Publish Time

An agent that looked safe in review can behave differently in production. Prompts evolve, data connections change, and users find creative ways to ask questions nobody anticipated. A one-time approval is a snapshot, and snapshots go stale the moment anything in the environment changes.

Do this:

- Establish a behavioral baseline for each high-risk agent, what data it normally touches, what actions it normally takes.
- Monitor for deviations from that baseline in real time, not in a weekly report.
- Treat an agent reaching outside its baseline as an incident, not a log entry.



Bottom Line

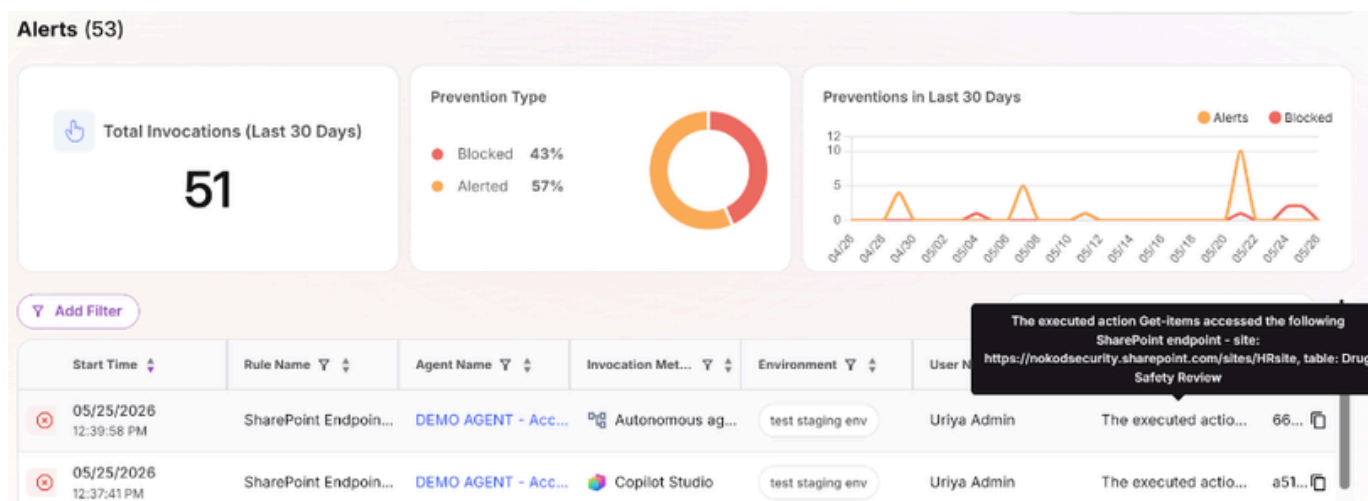
Innovation and Governance Aren't Opposites

Adaptive governance treats agent security as continuous, not a one-time sign-off. Business users should keep building, and security teams shouldn't need to slow that down, but make sure visibility, least-privilege data access, injection-resistant design, and enforced guardrails are already in place before the next thousand agents show up.

The agents are already out there, growing quietly inside the tools you approved. The only real question left is whether you can see them.”

See Every Agent. Know Every Permission.

Kanopy gives security teams continuous visibility into every agent, automation, and workflow the runtime guardrails to stop risky behavior before it becomes a breach.



Book a demo



About Kanopy Security

Kanopy is the leading AI security platform dedicated to securing the growing jungle of shadow AI built by business users. Kanopy provides complete visibility, risk detection, and automated remediation to secure this often overlooked attack surface.

By proactively detecting and remediating hidden threats while enforcing continuous, automated guardrails, Kanopy enables organizations to govern their digital transformation safely without slowing down productivity.

Trusted by Fortune500 companies, Kanopy turns hidden risks into secure, governed innovation.

kanopysecurity.com