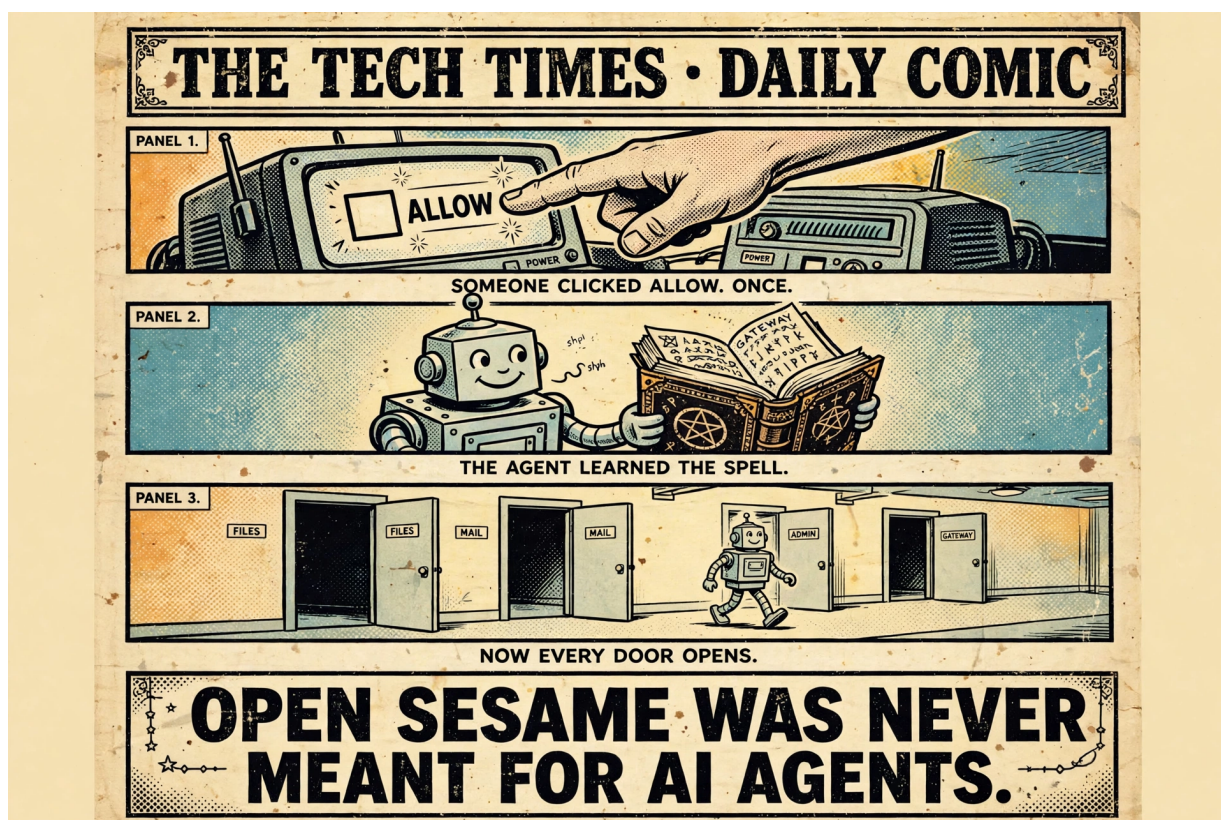


Open Sesame. Your Agent Knows the Spell.

Apple, AWS Loom, and GitLab exposed three different authority boundaries on the same day.

BY INSTINCTIVE NETWORK · OCTOBER 6, 2026



One click. Every door. Illustration: Instinctive Network.

On October 2, three unrelated announcements described the same crime scene from three angles. Apple said it will add new controls to macOS's most powerful permission — because of AI agents. AWS disclosed that its open-source agent orchestration platform could, in deployments without an identity provider configured, give any network client full administrative authority over the agent control plane. And GitLab patched a 9.9-rated hole in the gateway connecting its AI agents to their models — the second 9.9 in the same AI Gateway this year.

Nobody coordinated this. That is precisely what makes it worth reading twice.

Scene one: the operating system. Full Disk Access was invented for a simple, human reason: backup apps need to copy everything. One checkbox in System Settings, granted once, to one app, by a human who deliberates. Apple put the reason for the change in a single sentence: "As AI agents become increasingly capable and autonomous, the risks associated with this level of access will grow substantially."

Apple now says some developers use the permission "in ways that could put users at risk, exposing everything on their systems — including files, mail, messages, and even browsing history — without users' full knowledge and understanding." For messaging apps, Apple adds, the exposure reaches the people you talk to. (The notice landed days after a disputed report about an always-on agent and private messages — which Apple neither confirms nor addresses. It names no app, gives no shipping date, describes no mechanics.)

Here is the question Apple didn't answer: what does a permission prompt mean when the recipient never pauses to read it? A human grants once and moves on. An agent inherits the grant and acts on it continuously — every future file, every future message, every process the app ever spawns. The grant was an event. The agency is continuous.

Scene two: the agent manager. Loom is AWS Labs' open-source platform for orchestrating AI agents — the software that manages your agents' permissions, tool servers, and credentials. Security bulletin 2026-124-AWS discloses that, in deployments with no identity provider configured, CVE-2026-103956 let any network client obtain "full administrative authority over the agent control plane": registering tool servers, reading stored integration credentials, rewriting the IAM role policies attached to managed agent roles. AWS classified it as CWE-306, missing authentication for a critical function, and CWE-1188, insecure default initialization of resource permissions. The application whose job is to manage your agents' authority shipped a mode where the authority-manager answered to anyone.

The obvious question nobody filed: who authenticates the authenticator?

Scene three: the AI gateway. GitLab's AI Gateway sits between GitLab and the models behind its Duo Agent Platform — the component that brokers agent access to models, and a concentration point of trust: JWT signing keys, provider API credentials, prompt templates. CVE-2026-90970 (CVSS 9.9) let an authenticated user with Duo Agent Platform access escape the prompt template sandbox with a crafted flow configuration and run arbitrary commands on the gateway host. Patch it and move on — except the fine print: in February, GitLab patched CVE-2026-1868, also 9.9, also reachable through a crafted flow definition, also a template-engine weakness. Same component. Same flow-template boundary. Twice in one year.

So: if a configuration file can break the sandbox, what exactly was being sandboxed?

The pattern. Three layers, three failures, one structure. The three incidents expose a recurring assumption in agent governance: each control layer tends to inherit trust from the layer beneath it. We govern the agent. But who governs the orchestrator? Who constrains the gateway? And what does an OS-level permission mean once an autonomous process can exercise it thousands of times? Trust flows downward, and at the bottom of the stack it pools into a checkbox clicked once and inherited forever.

Here is the distinction that keeps getting missed. A permission can start to look less like a key and more like a spell. Say it once, and every future door may open. Full Disk Access grants standing access to an app, while an agent may exercise that access across thousands of future actions no human reviewed individually.

The problem is not whether the agent has the spell. It is whether this door should open, now, for this task.

That's the idea behind what we're building at Intent Checkpoint: a checkpoint that sits outside the actor's own decision loop, checking each action against authority, scope, impact, and coherence at runtime — because the grant you clicked last year cannot adjudicate the action your agent takes tomorrow.

This week, three front doors were standing open, and all three belonged to the systems meant to govern your agents. So here is the question worth taking home: in your agent stack, which door have you never actually tried?
