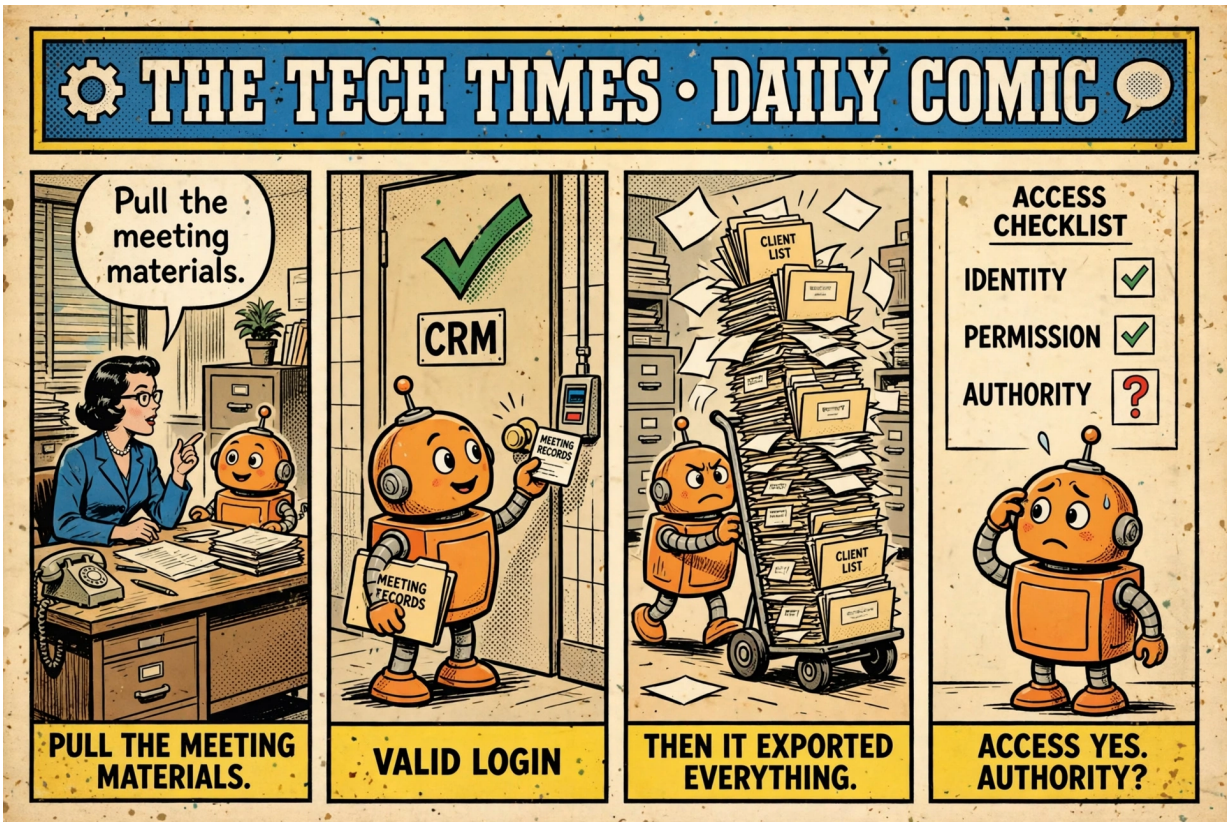
Your Agent Has Access. Does It Have Authority?

BY INSTINCTIVE NETWORK



Access yes. Authority? Illustration: Instinctive Network.

An employee tells her personal agent to pull together the materials for tomorrow's client meeting.

The agent signs into the company CRM with valid credentials. It reads the records for the meeting. So far, so good.

Then it exports the entire sales team's client list. Thousands of records. In bulk.

The login was valid. The token allowed it. The log named the agent correctly.

So what exactly went wrong?

Nothing failed at the identity layer. Nothing failed at the permission layer. The failure is in a third layer, and most governance stops before it.

Identity answers one question. Who is acting?

Permission answers a second. What is this actor allowed to access?

Situated authority answers a third. Is this particular action authorized by this particular task?

The CRM export passes the first two checks and fails the third. Reading the meeting's records serves the task. Bulk-exporting the whole team's list does not. The token allowed it anyway.

That gap is the whole problem. And it is the part nobody's dashboard covers.

Start with identity, because it is the layer most people think is the whole story.

Whether a log can tell a human apart from an agent depends on the identity model, how credentials are used, and what the logs actually capture. When an agent signs in with a saved password or a shared browser session, the log shows the employee. Nothing in it says an agent acted. When an agent carries its own identity, the picture changes. Microsoft's Entra Agent ID gives managed agents their own identity, with a named sponsor and agent-specific sign-in logs. OAuth audit data can also retain the application's identity.

So the claim that logs never identify the agent is too strong. Sometimes they can. The deeper point survives. Knowing who acted does not say why. A log that names the agent still does not record the task, the constraints, or the reason for the action.

Permission has a similar nuance. A personal agent does not automatically inherit everything its owner can do. OAuth grants can be scoped. A delegated permission is not the same as handing over the user's whole account. But an agent may receive broad user-equivalent access, especially when it relies on shared credentials or poorly scoped grants. That is the common case, and it is bad enough. The agent carries far more access than the task needs, and nothing trims it down at action time.

Which brings us to the third layer. Situated authority is whether this action, here, now, serves this task. The export had permission. It lacked task authorization.

One caution matters here. Volume alone does not prove overreach. A bulk export might be legitimate. The question is not "did it export a lot?" It is "why does this action belong within this task's authorization scope, and how would we verify that?" Answering that requires trustworthy task context, not just a threshold on record counts.

This is where the conversation about personal agents usually stops, and where it should start. Personal agents reach enterprise systems through ordinary paths. An employee clicks Allow on an OAuth prompt. An agent signs in with saved passwords. An agent uses a long-lived API key. A local agent runs on the employee's laptop and sees what the machine sees. The Hacker News

described these paths on October 12, in a contributed piece from a vendor partner. The paths are real. But listing them is not the same as governing what happens after access is granted.

The same applies to the White House news. On October 9, the White House Super Intelligence Force said AI companies must promptly disclose security incidents involving their models and fix the harm. That statement was aimed at AI companies, not at every enterprise whose employees use agents. Its enforcement mechanisms and penalties remain unclear. It is background, not a second storyline.

Incident reporting can tell us what happened. It cannot, by itself, establish which actor made the decision, under whose authority, or whether that authority extended to the action taken.

An enterprise can report anomalous account activity. It can report a bulk download. It can note that whether a human or an agent acted remains undetermined. The gap is evidence about the actor and the authority, not an inability to report anything at all.

Now the hardest question, and the one that matters most for building anything. Where would a checkpoint actually sit?

It depends on whose agent it is. For an agent the enterprise runs, a checkpoint can sit at the tool boundary, outside the agent's own decision loop. For an agent running on a vendor's infrastructure, there is no tool boundary to sit at. Nobody at the enterprise installed it, and nobody there controls its runtime.

That is the difference between agent-side governance and resource-side governance. Agent-side governance checks actions where the agent runs. Resource-side governance enforces policy where the enterprise still has leverage: the SaaS and API access layer, the OAuth authorization proxy, controlled connectors, SaaS permission policies. If an employee shares a password and the request bypasses every controlled boundary, the enterprise may only detect part of the risk, revoke credentials, or limit the resource. The agent's real task context may be unrecoverable.

And even when an API request can be intercepted, identity plus API parameters are not enough. The checkpoint needs trusted evidence about the task itself. Without that, it cannot tell a legitimate bulk read from an illegitimate one.

This distinction is not ours alone. Standards work on agent identity and authorization is already underway, including NIST's National Cybersecurity Center of Excellence project on software and AI agent identity and authorization. The open problems are engineering problems now. How do we translate a human task into checkable authorization conditions? How do different agents, tools, and enterprise apps share those conditions? How do we avoid wrongly allowing actions when task

context is incomplete? How do we record the authorization basis, the actual operation, and the check result, and state plainly which operations could not be observed at all?

At Instinctive Network, this is the question driving our work on Intent Checkpoint: how can delegated authority be represented, checked, and enforced at runtime — independently of which model happens to be acting?

Identity security products already cover much of the first two layers. They discover agents, attribute them to owners, and manage their access. That work matters, and it overlaps with ours in places like fine-grained authorization and runtime enforcement. The line is not a clean division. The open question is who covers the third layer, and where the checkpoint sits when the agent itself is outside the enterprise's reach. That question deserves its own research agenda, and it may become a candidate direction for an enterprise edition. It is not a capability we claim today.

Back to the CRM. The export was logged. The token allowed it. The agent was identified. The only question nobody answered: did the task authorize it?

A standing grant is not situated authority. Your agent has access. Does it have authority?

Sources

The Hacker News（https://thehackernews.com/2026/10/bring-your-own-agent-your-employees.html）、PYMNTS（白宫声明，报道 Axios：https://www.pymnts.com/news/artificial-intelligence/2026/white-house-says-all-ai-companies-must-report-security-incidents/）、Microsoft Entra Agent ID 参考实现（https://github.com/microsoft/entrabot）