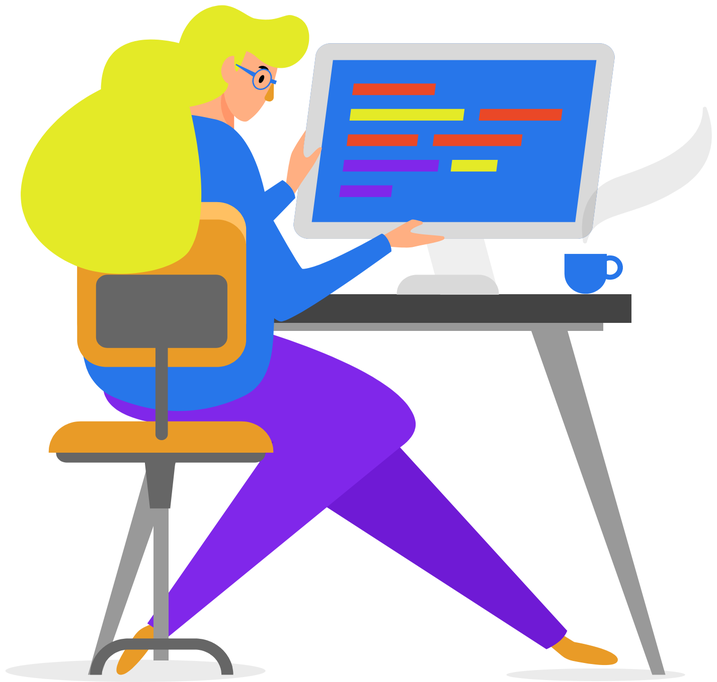
Agents have an **Identity Crisis** !

The state of Auth in MCP and Agentic AI

Swaraj Pande
Software Engineer (L2) @ Red Hat

Agenda



1. History
2. Frameworks
3. Gaps
4. Standards

Who this is for ?

- You are putting agents in front of real APIs
- You have (or will have) an MCP server on the internet
- You care what your (audit) logs say at 2 a.m. 😭

When no human was in the loop, who authorized what ?

OAuth was designed around a human sitting in front of a browser, clicking “Allow”.
Agentic systems break that assumption in three places at once:

1. The actor is often a **machine.**
2. The action is often **chosen by a model**, not a user click.
3. The call path is often multi-hop:

user ⟶ orchestrator ⟶ agent ⟶ MCP server ⟶ downstream SaaS.

Auth happened here !

Travelled all the way back till here !

Three identities, one Authorization header ! 💀

Badge	Answers	Typical credential
User	Who is this for?	OIDC ID token / 3LO access token
Agent	Which program is acting?	Agent ID, SPIFFE ID, client_id
Workload	Which process holds the key?	Service account, mTLS, SPIFFE SVID

A year ago, MCP had no login !

It started on your laptop. No login.
Trust = You started the process.

Then people put it on the internet.

What showed up overnight:

- API keys pasted in Slack
- Tokens stuffed into tool arguments - so the model could see, log and leak
- Every team invented a different schema

The fix was “log in like a website”.
However, that still left a hole: a token for the **notes** app could open **payroll**, because nobody checked which door the token was for.

Three Rules. This is the floor.

1. **Do not run your own login page**:
   a. Use the org's identity provider.
   b. The MCP server only checks the token.
2. **A token is for one door:** Notes app != Payroll app
3. **Do not forward the user's token:** Get a new, narrower one.

The MCP server is a **deputy**. Deputies need their own, smaller keys.

User token → MCP → same token → GitHub (example) ❌ WRONG
User token → MCP → new token (GitHub only) → GitHub ✅ RIGHT

MCP fixed the first hop. Not the rest.

Fixed

- Did not invent a custom login
- Token is for one server
- Do not pass the token through
- Ask for more permission later

Not fixed

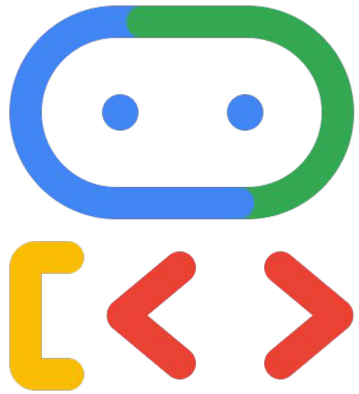
- Robots with no browser
- Identity through a chain of agents
- Who allowed the model’s choice

MCP is now a decent **user → one server** login.
It is not yet **agent authorization**.

Your agent framework is already an identity system. It just may not know it !

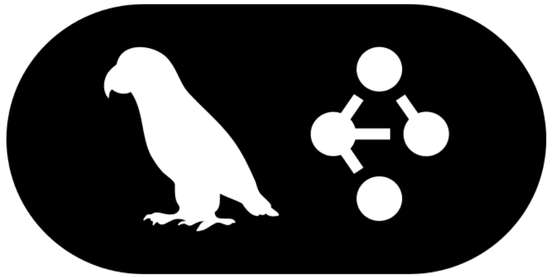
Google ADK: the agent is a person in IAM

- Each agent gets its **own ID,** tied to its lifetime
- You write permissions on **that ID**, not on a shared bot user.
- A **vault** holds tokens. Tools never see secrets.
- Human path: pause ⟶ browser Allow ⟶ resume
- Robot path: the **agent** is the caller. Nobody clicks allow.



LangGraph: who hit the API?

- Two layers:
  - Who is calling the graph?
  - Which conversations may they see?



- Checks the caller on every request
  - Isolate **threads** so Alice cannot read Bob's chat
  - Without that: every user looks like **the developer's** API key

- Strong at **user isolation of chats**.

- Silent on how tools get tokens unless you add it.

- Robot path: the **agent** is the caller. Nobody clicks allow.

LangChain: bind the token to this user and this bot!

- First use: login in the browser
- Mid-run: **pause**, show Allow, resume
- Stored token is scoped to **this person + this assistant**
- Agent A does not inherit Agent B’s GitHub, Slack, … and other apps

The Gaps: The hard problems are not “add login”. You already did that !

Gap 1: nobody is at the browser

MCP login assumes a human can **click Allow**.
Production is full of callers that cannot:

- A cron job that runs as 2 a.m.
- CI that opens PRs
- One agent calling another over HTTP etc.

What teams do instead (all bad):

- Bake **a person's** refresh token into the job
- Share a static API key
- Mint one god-mode token
- Pretend a robot is a user in the directory etc.

There is no standard answer for "this agent, this task, this long, no human".

Gap 2: the token forgets the story

User (Alice)
└─ Orchestrator
  ├─ Research agent → docs
  └─ Finance agent → ledger → ERP

At every hop you must choose:

- Forward the user's token? **Unsafe.**
- Mint a new one? **For whom? Which API?**
- Remember who acted? Usually nobody writes that down.

By the time ERP sees the call, it often looks like a robot admin.
Alice is gone. The finance agent is gone.

Gap 3: When no human was in the loop, who authorized what?

What was the consent ?

- Was it given **earlier**, for a **vaguer** job, to a **different** program OR
- given by a **developer** who pasted a key OR
- **never given ?!?!**

A model choosing something is not a user clicking Allow.
If you treat them as the same event, your audit logs are nothing but fiction.

The order of the Pain that you will have to bear ! 🤕

1. Tokens in prompts, traces and debug dumps
2. One token reused on **many** servers
3. Mid-run the bot needs more permissions - and you lose the graph
4. n server calls, n tokens, n refresh races
5. A sub-agent spends the **parent's** keys
6. Logs name the HTTP caller, not the **tool the model picked**
7. Alice left the company. Which vaults still have her Slack token ?

What to ship and standardize ?

Identity: log **user**, **agent** and **machine** on every line.

Tokens:

- MCP server does not run login. One token, **one door**. Never forward.
- New token for downstream (GitHub / ERP etc.). Smaller than the last one.
- Vault holds secrets. The model never sees them.

Consent:

- Human: start small, ask again for more.
- No Human: **policy + approval + short-lived task tokens**.
  Not Alice's leftover login.

Audit: User, agent, which API, which tool, yes/no
More logs are always better !

RFC cheat sheet (for the backend crowd)

RFC / draft	Role in this talk
OAuth 2.1	Baseline. PKCE required. No implicit.
RFC 7636 PKCE	Bind the code to the client that started the flow.
RFC 8414	Authorization Server Metadata
RFC 9728	Protected Resource Metadata — how clients find the AS
RFC 8707	Resource indicators — aud binding
RFC 6750	Bearer challenges; insufficient_scope
RFC 8693	Token exchange; act claim
RFC 7523	JWT bearer grant (EMA's second hop)
RFC 7591 DCR	Legacy client registration
CIMD	Client ID = HTTPS metadata URL
SEP-835	Step-up / incremental consent
SEP-990 EMA	Enterprise IdP as policy brain

Thank you!

Connect

 @swarajpande05

 /swarajpande5

 @swarajpande5

QnA