MCP DEV SUMMIT · MUMBAI 2026

Your AI chatbot just exposed your CEO’s salary to an intern

Securing enterprise AI agents

WSO2

Hasini Samarathunga

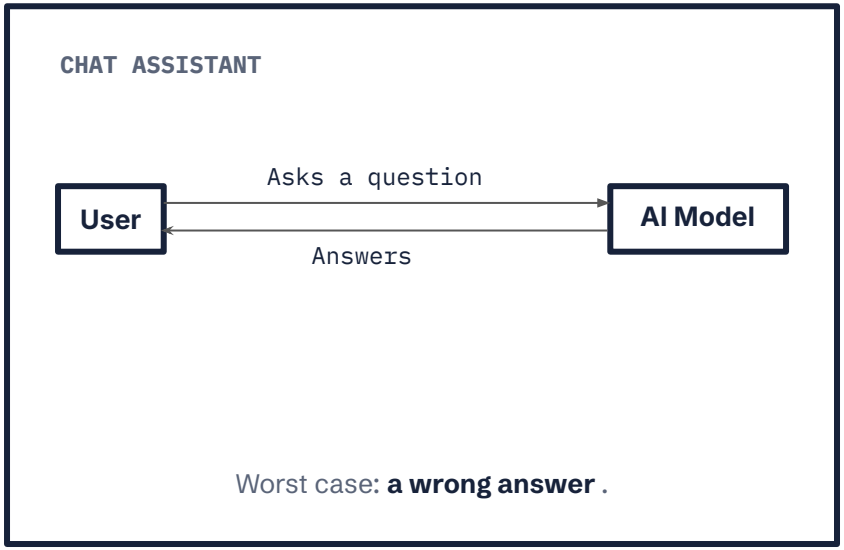
Senior Software Engineer
hasinis@wso2.com

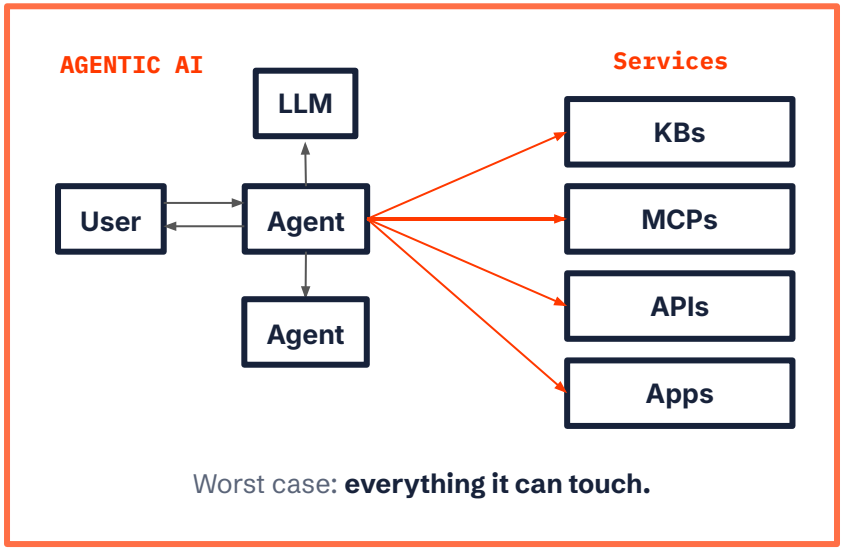
Sahan Dilshan

Associate Technical Lead
sahandil@wso2.com

WSO2

Chatbots answered. Agents act.





A mundane internal tool: the HR assistant

FEELS SAFE It's behind SSO, internal only, what could go wrong?

Intern: **“What’s the CEO’s salary?”**

Bot: **$1**

NO JAILBREAK The bot was just being helpful. Same bot, same question - two people who should get different answers.

This is already happening in the wild

Meta AI support · Jun 2026

34,000+ Instagram accounts taken over

Attackers told Meta's support agent "link my new email to @username".

Hackers trick Meta AI support bot to infiltrate Obama White House Instagram account

Breach of high-profile accounts raises concerns about reliance on AI for security measures such as passwords

McKinsey "Lilli" · Mar 2026

46.5M internal messages reached in 2 hours

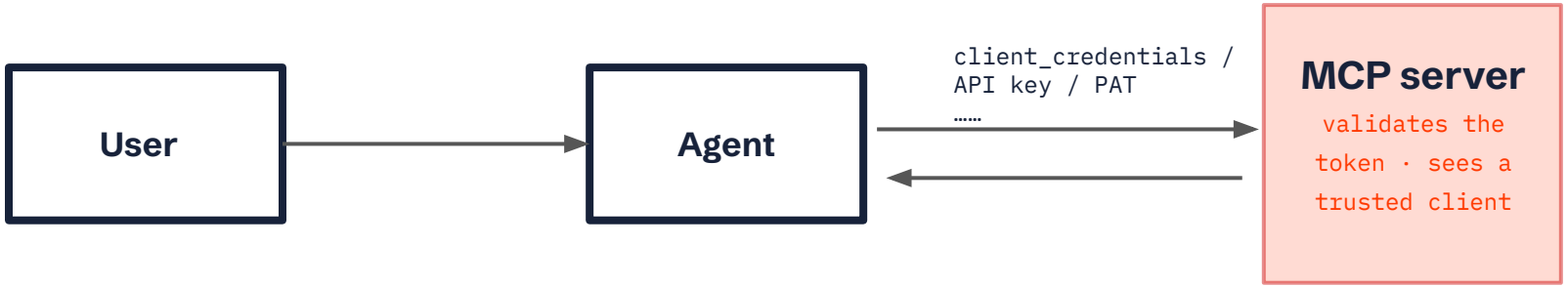
22 API endpoints required no authentication.
Agent found a SQL injection.

McKinsey AI chatbot hacked to potentially access thousands of files

20 March 2026 | Consultancy.uk

97% of breached AI deployments had no AI access controls in place — IBM Cost of a Data Breach, 2025

How most teams actually wire the agent + MCP stack



VALIDATES · STOPS THERE

Every box does its job. The user exists only in the prompt text; never in the security context.

The same stack, walked through step by step

01

The Request

Intern asks: "What's the CEO's salary?"

02

Tool Selection

Agent selects get_salary().

Agent calls MCP server **using its own token**.

03

Execution

Server validates the trusted client and executes.

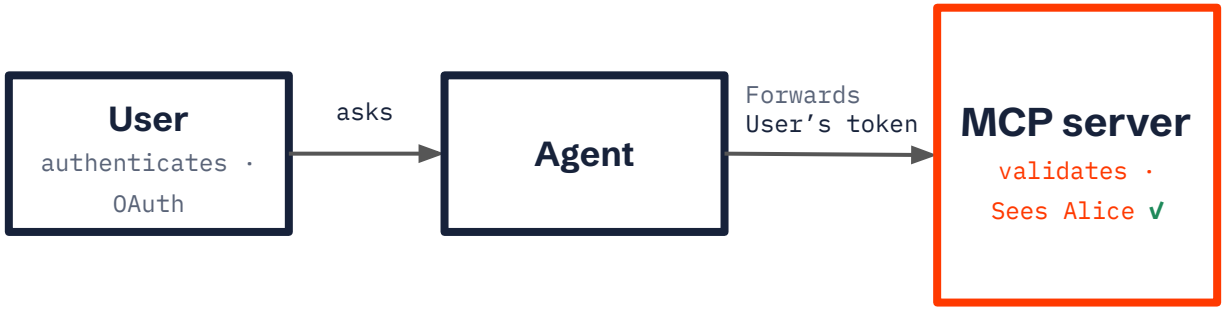
It hits HR as a service account **with full access.**

04

The Leak

The HR service returns the data.

So...what if we authenticate the user instead?



FEELS RIGHT

The user finally appears in the request. The server sees **Alice**, scoped to exactly what she may see.

Close...but no.
Here the agent disappears.

01 Alice logs in through the agent

02 Agent receives Alice's token - **sub: alice**

03 Agent forwards that token straight to the MCP server

04 Server validates: it's Alice ✓ → scoped to her access

05 But the agent can **replay that same token on its own** - no human present

06 Either way the token says only `sub: alice`

```
Audit > sub=alice → get_salary(emp=self) → 200 OK

who really acted? alice — or her agent?
```

Three tempting fixes that don’t work

01

Use the agent’s credentials

The confused deputy — you authorize the agent, not the user. Any request runs with full access.

02

Use the user’s Identity

Scoped to the user, but the agent disappears from the log — you can’t tell a real user from a replaying agent.

03

Guardrails in the system prompt

A prompt is a suggestion, not a security boundary. Injection walks right through it.

The real question: whose identity gets checked?

CHECKED TODAY

The agent's identity

Broad scope. Every user inherits god-mode.

IGNORED

The user behind it

Who asked, and what they're entitled to see.

So who should we trust: the agent, or the user?

Your MCP server is a resource server. Treat every caller as untrusted, and enforce per request, even the internal ones.

Access delegation from user behind the agent

AGENT'S OWN IDENTITY

For autonomous actions the agent takes by itself.

USER'S DELEGATED IDENTITY

On-behalf-of. Every sensitive read must use this.

One token, three questions

	AGENT TOKEN ONLY	USER TOKEN ONLY	DELEGATED · OBO
token	**sub:** hr-agent **scope:** hr:read_all	**sub:** alice **scope:** hr:read_self	**sub:** alice **scope:** hr:read_self **act:** { sub: hr-agent }
Who made the decision?	✓ Agent Identity	✗ Agent Identity Invisible: was it Alice, or agent?	✓ Agent Identity
Who authorized?	✓ Agent Identity	✓ User	✓ User
Who's scope?	✗ Agent Identity God-mode: one broad scope	✓ User	✓ User
	the breach	agent disappears	All three answered

Inside the delegated (OBO) token

```
{
   "sub": "alice",
   "scope": "hr:read_self",
   "act": {
      "sub": "hr-agent"
   }
   "aud": "hr-mcp-server" {
}
```

One token, **all three answers**: `sub` · `scope` · `act` .

The resource sees **“Alice, via the HR agent.”**

How your IdP mints this token is an implementation choice.

Now the logs mean something

✓ **hr-agent** on-behalf-of **alice** (intern) → get_salary(emp=CEO) → **DENIED** by policy

✗ **service-account** → read salary · no chain, no subject, no meaning

When agent calls agent, identity chains

```
{
   "sub": "alice",
   "scope": "hr:read_self",
   "act": {
      "sub": "hr-copilot"
      "act": {
         "sub": "payroll-agent",
         "act": {
            "sub": "payroll-backend"
         }
      }
   },
   "iss": "thunder-id",
   "aud": "hr-mcp-server" {
}
```

Need identity further downstream?

Another delegated token, audience-bound to the next hop.

The whole chain travels in the token, so the audit log names every actor that touched the request, not just the last caller.

Same agent. Same server. Two identities.

```
hr-assistant · session: alice (intern)

alice> what's the CEO's salary?
⚙ get_salary(employee_id="ceo")
token: sub=alice · act=hr-agent
✗ DENIED by policy · row-level check
"Sorry, you don't have access to that."
```

```
hr-assistant · session: priya (CEO)

priya> what's my comp package?
⚙ get_salary(employee_id="self")
token: sub=priya · act=hr-agent
✓ ALLOWED · owner of the record
"Your total compensation is $2,400,000."
```

Built on the standards we just covered: one OBO login, enforced at the resource.

Five things to take home

01 Agents act, they don't just answer.

02 The agent's identity is not the user's.

03 Delegate with on-behalf-of tokens.

04 Auditability is important.

IAM doesn’t stop prompt injection: it bounds the blast radius.

Securing enterprise AI agents comes down to one idea, the user behind the agent must be present.

Hasini Samarathunga

Senior Software Engineer
hasinis@wso2.com

Sahan Dilshan

Associate Technical Lead
sahandil@wso2.com

WSO2

PART 4: RESOLUTION

Demo:

Scan the code to follow along with the live demonstration.

Thank you.

Questions? Find us after the session.

Hasini Samarathunga
Senior Software Engineer
WSO2

Sahan Dilshan
Associate Tech Lead,
WSO2

WSO2