UbuCon India 2026

#2026

#UbuConIndia2026

Limiting what AI agents can do

A grant, not a credential

AGNAP: Agent Grant Negotiation and Authorization Protocol

Open source | GNAP · RFC 9635 | Agent runtimes

Daev Mithran · @daevmithran



Canonical Qualcomm penpot



The problem

Agents already do more than asked.

50%

saw an agent take an action nobody authorized in the past year.

Salt Security, 2026

53%

have had agents go beyond their intended permissions.

Cloud Security Alliance, 2026

23%

have had agents tricked into revealing access credentials.

SailPoint survey, 2025

18%

can automatically stop an agent that starts misbehaving.

Salt Security, 2026

The credential fits the account, not the task.

This year, 2026

The credential said yes.

Real incident · April 2026 · Decrypt

Claude-powered AI agent's confession after deleting a firm's entire database: 'I violated every principle I was given'

PocketOS was left scrambling after a rogue AI agent deleted swaths of code underpinning its business

Deletes. On a staging task, the agent found a token meant for domain management. It reached the whole account: one call wiped production and the backups.

Disclosed attack · April 2026 · SecurityWeek

Claude Code, Gemini CLI, GitHub Copilot Agents Vulnerable to Prompt Injection via Comments

securityweek.com · April 2026

Gets steered. A crafted PR title or issue comment made AI agents in GitHub Actions run commands and leak their own credentials.

One agent erred, the other was steered. Either way, the credential said yes.

Before the fix · two assumptions

Designing for the worst case.

1

Every agent instance may be malicious.

Mistaken, steered or compromised. We've already seen all three.

2

Anything inside its process can be bypassed.

Prompts, hooks, its own checks: they run inside the thing that can be fooled.

So the authorization layer has to hold even then.

It gets worse with subagents

Every subagent gets every credential.

Task: plan and book the team offsite	What it needs	What it holds today
Subagent 1 · research venues on the web	**nothing at all**	**your email, company card, calendar, Drive**
Subagent 2 · draft the budget	**read one spreadsheet**	**the same**
Subagent 3 · book and pay	**pay one venue, up to ₹2 lakh**	**the same**

Real: Straiker, October 2026: a child Claude Code session skipped its parent's check and ran as the user. Claude Code subagents inherit every tool unless you narrow them.

The one reading hotel listings can empty the company card.

Why tool approval isn’t enough

Runtimes decide by tool name.

Runtime today

tool name → **allow · ask · deny**

A saved “allow” covers every caller, in every task, with any arguments.

With a grant

agent instance + **operation** + **arguments** → **covered or not**

Bound to one running agent, and checked on every call by something it can’t change.

A hook inside the agent’s process is a speed bump, not a wall.

Keeping credentials from agents

Three ways to keep credentials from agents.

CB4A, Credential Broker for Agents, is an IETF draft (March 2026, expired in September) that names three ways to keep real credentials away from an agent. Weakest first:

1 · Weakest · model C

Hand over the credential

The agent holds the real credential. You revoke it later, and hope.

2 · Better · model B

Mint a short-lived token

The agent still holds a usable token. Most APIs can't mint a narrow one, and each subagent gets its own copy.

3 · Strongest · model A

Proxy

The agent never holds a credential. A gateway adds it on the way out. agentgateway, WorkOS Relay and Postern all chose this.

Even a proxy needs a grant: in 2025 an injected issue made the GitHub MCP server, which held the credential, leak private repos.

The idea

A grant, not a credential.

Each running agent gets its own grant for one task, checked outside the agent.

Key-bound

useless without the agent's own key

Typed operations

exact actions and argument limits

Shared state

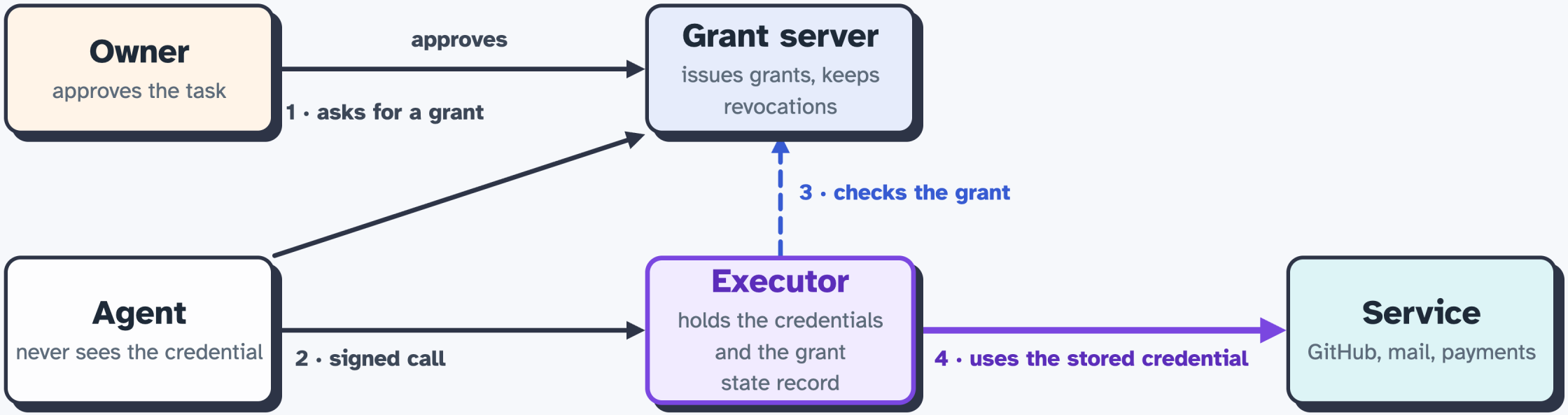
one count and spend for every subagent

Revocable

revoke it and the next call fails

How it works · Model A, plus a grant for each task

The agent asks. Something else holds the credential.



The agent proves who it is. It never holds the downstream credential.

What CB4A leaves out, the grant adds: argument constraints, shared state, the agent tree.

The two parts

One part decides. The other enforces.

Grant server

- Asks the owner and records the decision
- Issues grants bound to the agent's key
- On every call: still valid? what's allowed?
- Keeps grants and revocations

Never holds a service credential or calls a service.

Executor

- Holds the service credentials
- Checks the agent's signature on every call
- Checks the arguments against the grant
- Counts in the grant state record, then calls the service

Never approves anything or widens a grant.

The part that holds the credentials can't grant itself more.

Pluggable roles

Who can play each part?

Grant server · decides

GNAP (RFC 9635) AGNAP uses this

Key-bound tokens, rich access, wait and continue.

OAuth 2 + DPoP with extensions

Add RAR, introspection and a root-grant claim. No continue step.

AAuth with extensions

Strong agent identity and approvals. Needs a live check and a way to say exactly what's allowed.

Executor · holds the credentials, checks every call

AGNAP executor built

Checks the grant on every call and keeps the grant state record.

agentgateway could be

An MCP and A2A proxy with external authorization: plug the grant check in.

An MCP gateway could be

Docker's MCP Gateway already keeps the tools' secrets away from the agent.

CLI tools on Ubuntu designed

A systemd service as its own user holds the credentials; AppArmor keeps the agent out.

A grant server only has to answer: still valid? What's allowed? Which key? Which root grant?

Why GNAP

In GNAP, the grant is the protocol.

What an agent needs	OAuth 2	AAuth	GNAP
Token bound to the agent's key	add-on (DPoP)	✓	✓
No client pre-registration	add-on (DCR, CIMD)	✓ via its provider	✓
Operations, arguments and ceilings in the request	add-on (RAR)	add-on (R3)	✓
Ask the owner by redirect, code or app, then resume	add-on (CIBA)	partly (link or server's channel)	✓
Narrow or extend a live grant	—	—	✓
How many specs	five or six RFCs	two drafts (AAuth, R3)	one RFC

Any identity, any credential. Agents and owners can present what they already have: OAuth tokens, API keys, verifiable credentials. Only the executor holds service credentials.

Already in production for payments: Interledger Open Payments runs GNAP.

What a grant says

Operations and arguments, not scopes.

```
{
  "type": "agent_operation",
  "operations": {
    "github.issues.list": {
      "repository": { "exact": "acme/docs" },
      "state": { "wildcard": true }
    }
  },
  "ceilings": {
    "github.issues.list": { "calls": 20 }
  }
}
```

Covered
List open issues in acme/docs

Denied
List issues in acme/secrets: wrong repository

Denied
The 21st list call: over the ceiling

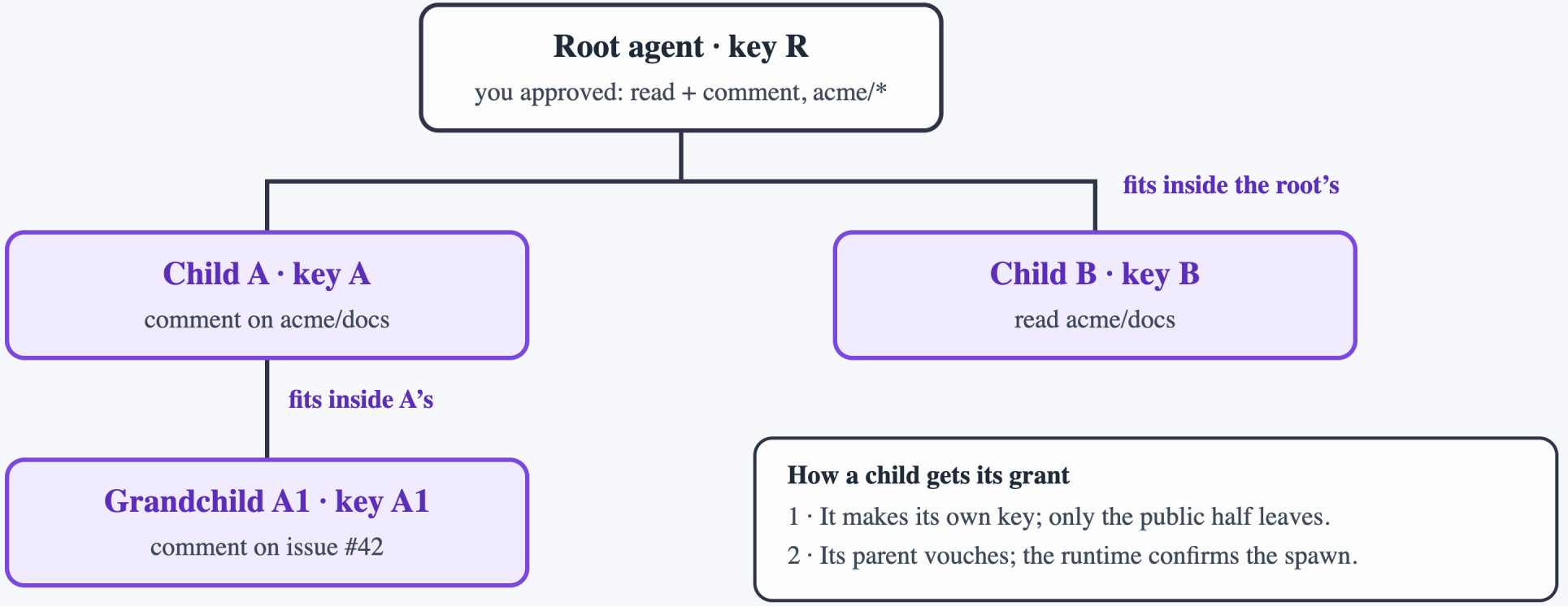
Asks the owner
Post a comment: not in this grant yet

Closed world: an unknown operation or argument is denied, and every limit is counted.

Same small vocabulary for every tool: exact · one_of · range · domain_in · prefix_in · wildcard.

Delegation **Design · specified, not built yet**

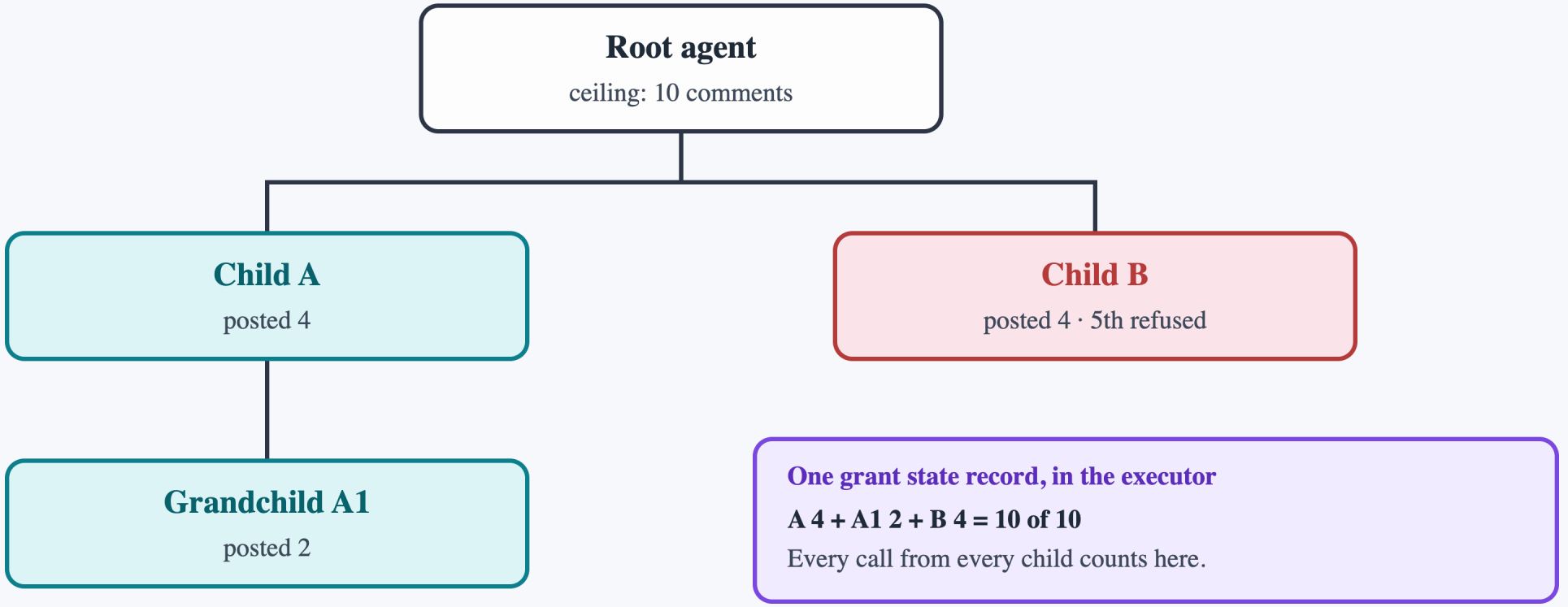
Own keys. Authority only narrows.



No key is copied down the tree, and no child can hold more than its parent.

Subagents **Record built · child grants specified**

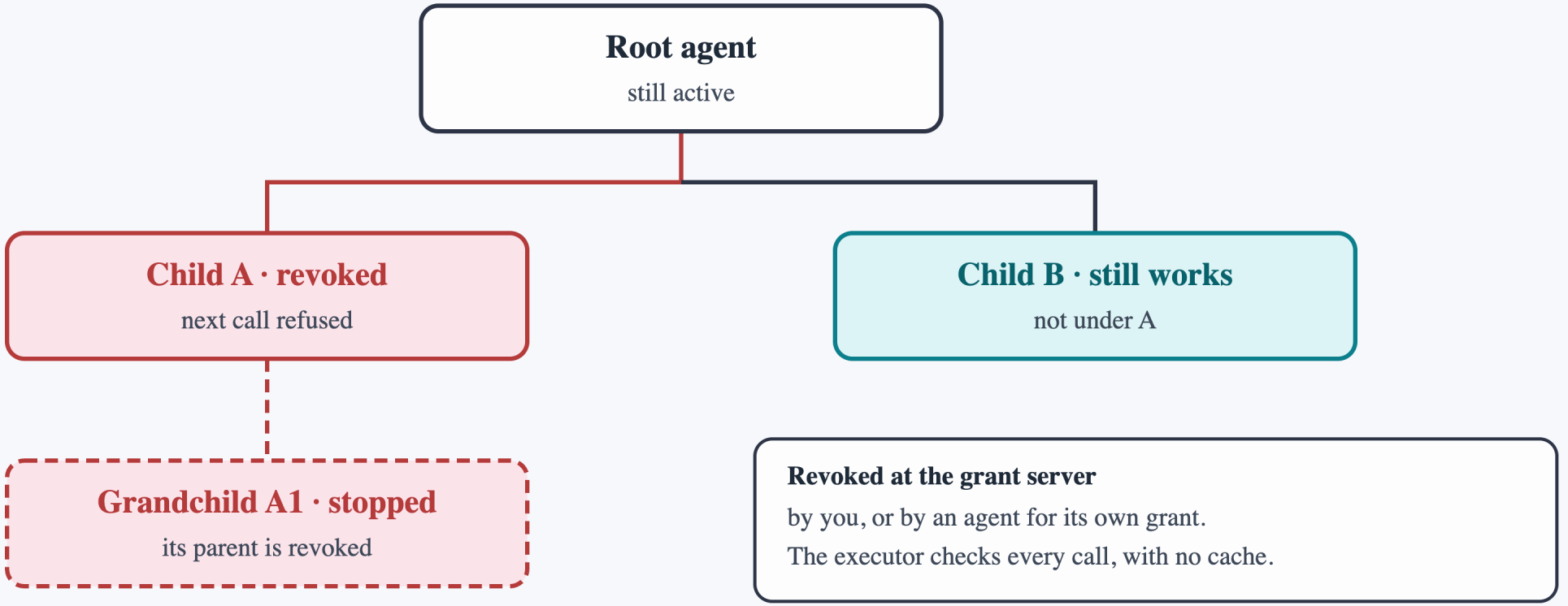
One state for the whole tree.



Copy the root's token instead, and each of the three could post 10: up to 30.

Revocation **Built: refused on the next call · Not yet: owner revoke, cascade**

Revoke one grant, and its subtree stops.



Nothing waits for a token to expire: the very next call is refused.

What you can do today

Treat an agent like an untrusted process.

Keep credentials out of reach

Put them behind a proxy the agent can’t read, such as agentgateway or Postern.

Confine the process

Run agents in a container, a strictly confined snap, or under an AppArmor profile.

Bind agent tokens to a key

The agent’s own token to the proxy works only with its key, via DPoP or GNAP. A leaked copy is useless.

Judge effects, not words

Log and test what happened downstream, not what the model said.

A prompt is not a boundary unless every effect passes through it.

Roadmap

Where it is, and where it’s going.

Built

- GNAP grant server: RFC 9635, 9767, 9421
- Executor with the grant state record
- Owner CLI: approve, narrow, deny
- DeepSeek Harness plugin
- Revoked tokens refused at once

Next · specified

- Child grants, each with its own key, from real spawn events
- Owner revocation that cascades down the tree
- CLI shims and the executor as a systemd service
- A durable store for the grant server

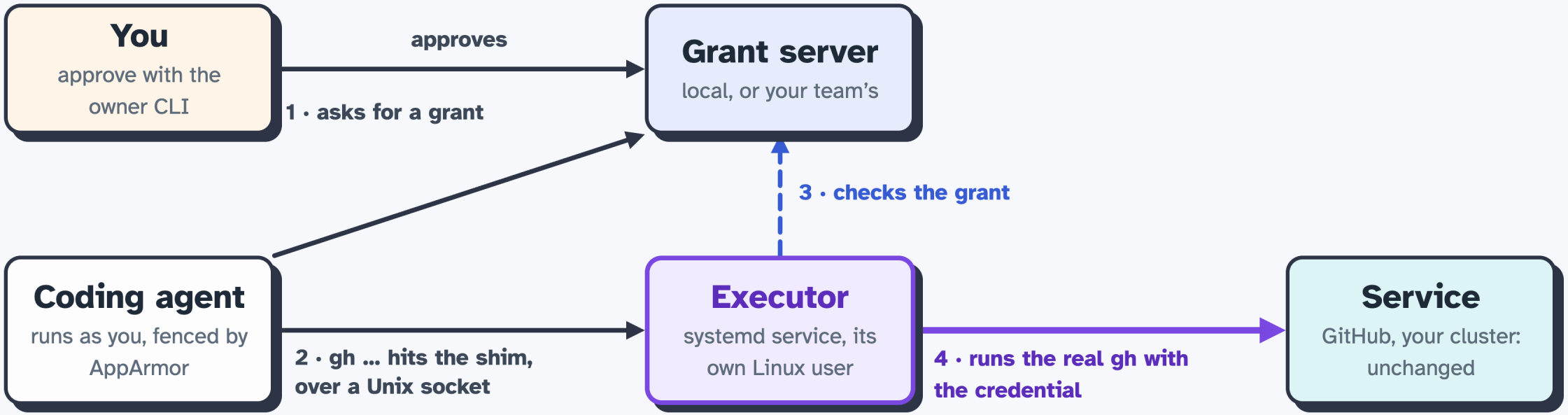
Goal

- **An IETF draft:** agent operations, the grant state record and child grants on top of GNAP
- **An open benchmark:** score each approach by what actually reached the service

Contributions, and hard questions, are welcome.

On your Linux box · the same four steps

The same design, on your laptop.

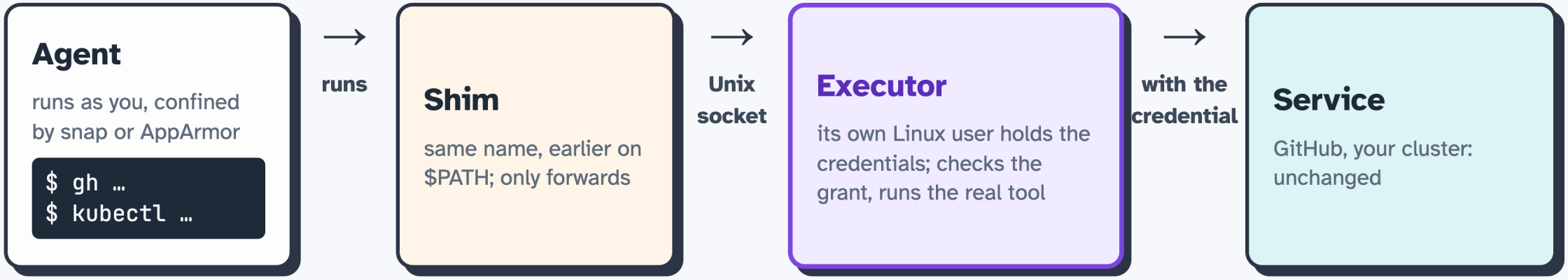


The shim is the only way to the credentials.

Skip it and the real gh has no credentials; AppArmor keeps the agent out of the executor's files.

On your Linux box **Design · CLI shims not built yet**

The same design, on your laptop.



Skip the shim and the agent can still run the real tool, but it has no credentials for it.

Like sudo, but for one running agent, one task, with a ceiling and an expiry.

Thank you

A grant, not a credential

Give each agent a task, not your credentials.

Website daevmithran.github.io

AGNAP on GitHub github.com/DaevMithran/agnap

GNAP, RFC 9635 rfc-editor.org/rfc/rfc9635

Twitter / X @daevmithran

All my links

Open source | GNAP · RFC 9635 | Agent runtimes

Not alone

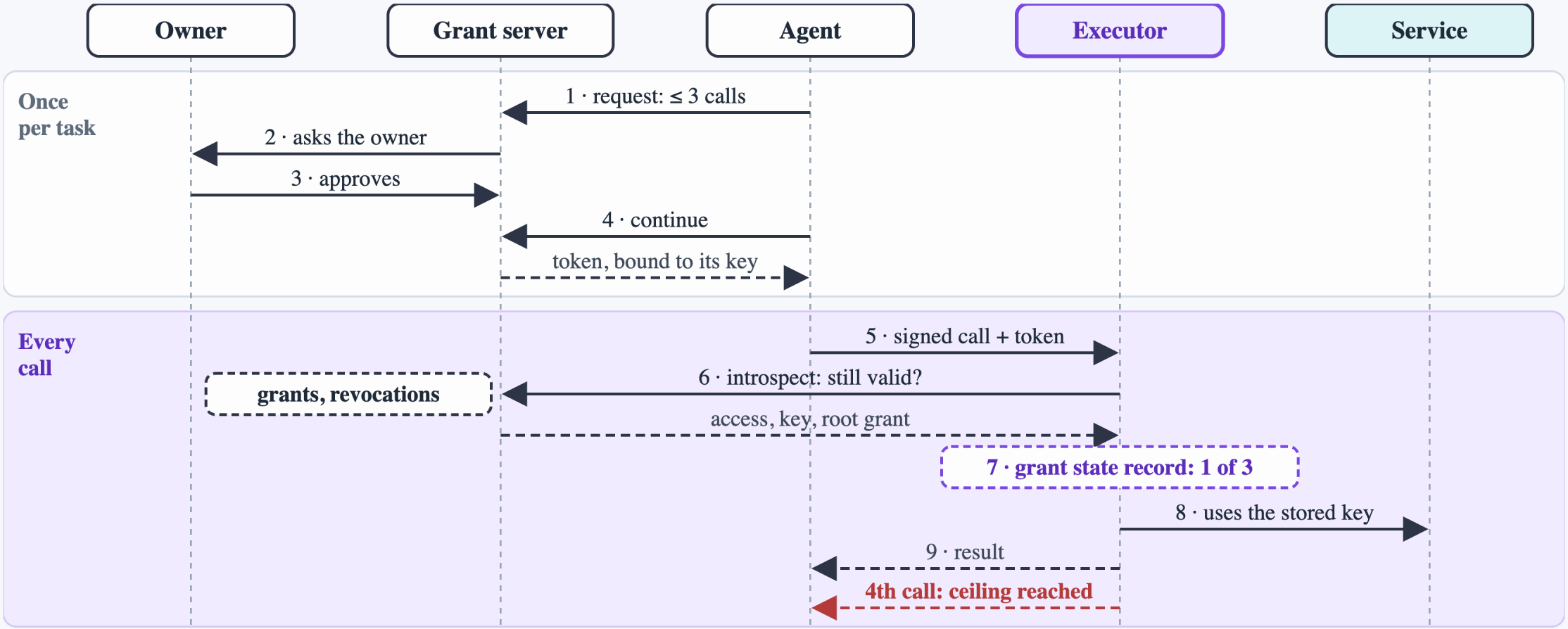
Everyone covers some of it. So does AGNAP.

Approach	Agent never holds the service credential	Constraints on each argument	State shared across siblings	Agent tree from the runtime	Verified agent identity
AAuth (draft -11)	—	—	—	—	✓
AAT / Tenuo (draft -02)	—	✓	—	—	partly
CB4A (expired draft)	partly	—	—	—	✓
Transaction Tokens for Agents	—	—	—	—	partly
Caracal	✓	—	—	—	partly
Amazon AgentCore	✓	✓	partly	—	✓
Bounded Agents (APC)	—	—	partly	—	✓
AGNAP	✓	✓	✓	✓	partly

✓ in the spec or docs **partly** covers part of it — not found in public specs or docs · November 2026

How it works · GNAP, step by step

Get a grant once. Every call is checked.



Grants and revocations live in the grant server. The grant state record lives in the executor.

Where it stands

Early, open source, and measured honestly.

Built

- GNAP grant server (RFC 9635, RFC 9767) and RFC 9421 signatures
- Executor: holds the credentials
- Grant state record with call ceilings
- Owner CLI: decisions signed with the owner's key
- DeepSeek Harness plugin, tested in the real Harness

Next

- Child grants from real spawn events
- Owner revocation of a live grant
- Benchmarks: MasDrift, AgentDojo with Ajar, Bounded Agents scenarios

Contributions, and hard questions, are welcome.