

What Can Your AI Agents Actually Do? Cedar Analysis for MCP Servers

@luxas @chris-j-burns



Who are we?



Lucas Käldström

Senior Applied Scientist @ AWS

CNCF Ambassador
Kubernetes contributor
Cedar maintainer



Chris Burns

Senior Engineer @ Stacklok

ToolHive maintainer
OSS Contributor

Outline

1. MCP auth summary
    a. Authorization engine requirements

2. Cedar Policy as a lingua franca
    a. Calculating with permissions using Cedar Analysis

3. Demo: Running and authorizing MCP servers using ToolHive and Cedar
    a. Mitigating data exfiltration risks using static analysis

MCP Auth is Three Questions

MCP Auth is Three Questions

QUESTION 01

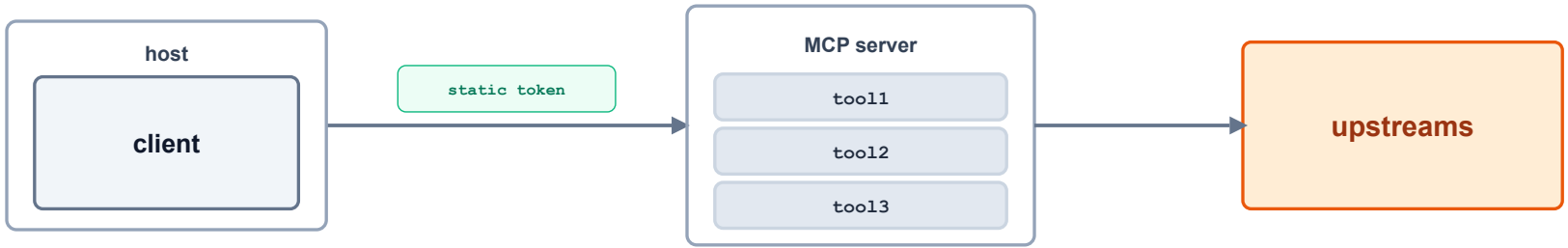


Who is the caller, and how do they prove it?

Authentication
Identity verification & credential proof

Static token

QUESTION 01 Who is the caller, and how do they prove it?

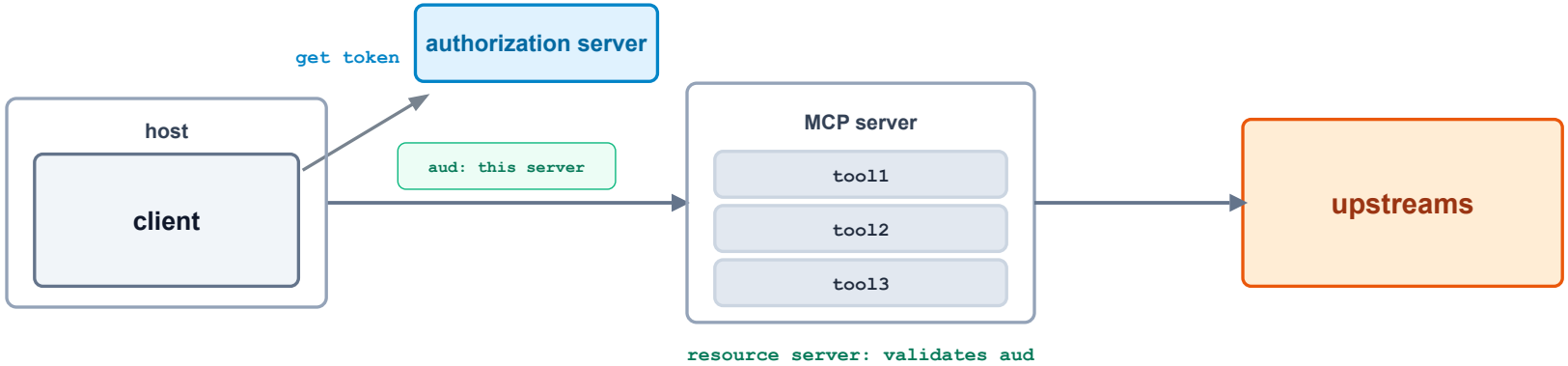


SECURITY CHARACTERISTIC

A shared secret in a header, handed over by a human once. It authenticates a copy of a string rather than a caller.

OAuth 2.1 Resource Server

QUESTION 01 Who is the caller, and how do they prove it?



SPEC MANDATES

Validate a token you didn't issue, bound to your own audience, and reject anything issued for someone else.

MCP Auth is Three Questions

QUESTION 01



Who is the caller, and how do they prove it?

Authentication
Identity verification & credential proof

MCP Auth is Three Questions

QUESTION 01



Who is the caller, and how do they prove it?

Authentication
Identity verification & credential proof

QUESTION 02

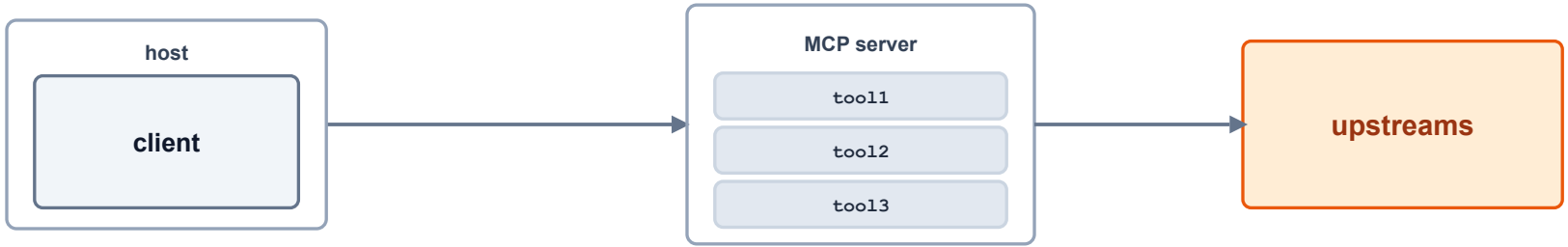


Who decides what they may do, and where is that enforced?

Authorization
Policy enforcement & permission boundaries

Only upstream authorizes

QUESTION 02 Who decides what they may do, and where is that enforced?

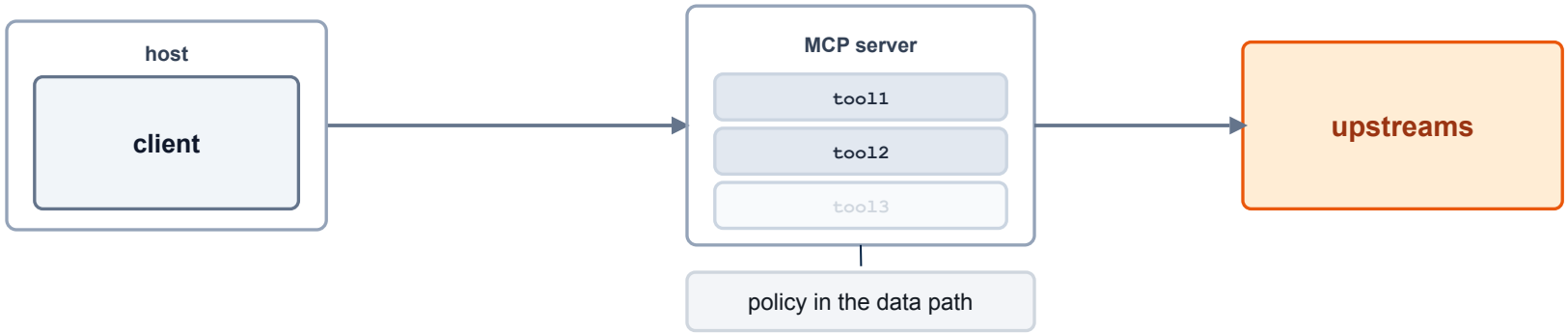


NOTE

No per-caller, per-tool decision is made, so any authenticated caller gets the server's full tool surface. This is where most of the ecosystem sits.

Both data path and upstream authorizes

QUESTION 02 Who decides what they may do, and where is that enforced?



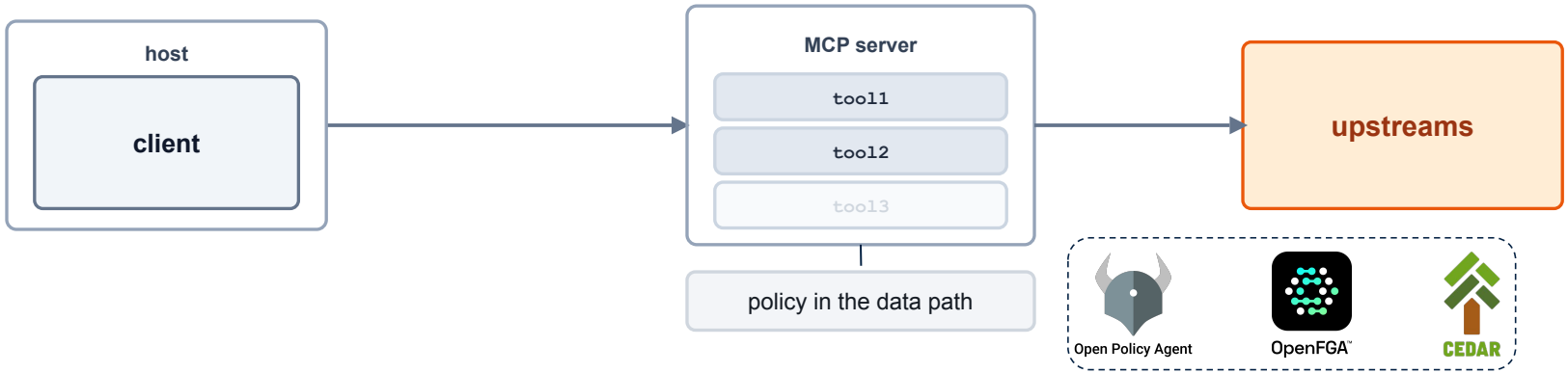
NOTE

Rules written as code in the request flow, in the server or a proxy in front of it.

Maybe ok for one server with three tools, but it becomes painful the moment you scale.

Both data path and upstream authorizes

QUESTION 02 Who decides what they may do, and where is that enforced?



MCP SECURITY BEST PRACTICE

MCP docs says “do not implement authorization logic by yourself”. CNCF projects can help

MCP Auth is Three Questions

QUESTION 01



Who is the caller, and how do they prove it?

Authentication
Identity verification & credential proof

QUESTION 02



Who decides what they may do, and where is that enforced?

Authorization
Policy enforcement & permission boundaries

MCP Auth is Three Questions

QUESTION 01



Who is the caller, and how do they prove it?

Authentication
Identity verification & credential proof

QUESTION 02



Who decides what they may do, and where is that enforced?

Authorization
Policy enforcement & permission boundaries

QUESTION 03

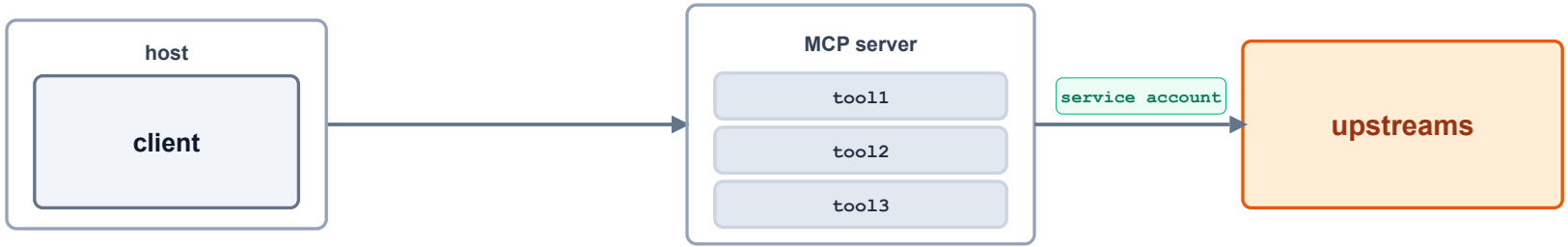


What identity reaches the thing behind the server?

Delegation & Identity
Upstream propagation & backend access

The server's own

QUESTION 03 What identity reaches the thing behind the server?

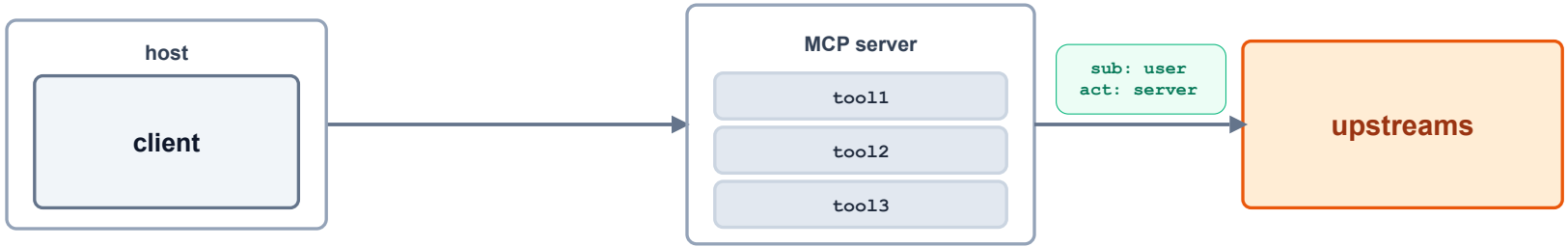


SECURITY CHARACTERISTIC

The upstream sees a service account holding the union of every caller's permissions. Correct when the action really is the server's, at the cost of all user attribution.

The user's, freshly issued token

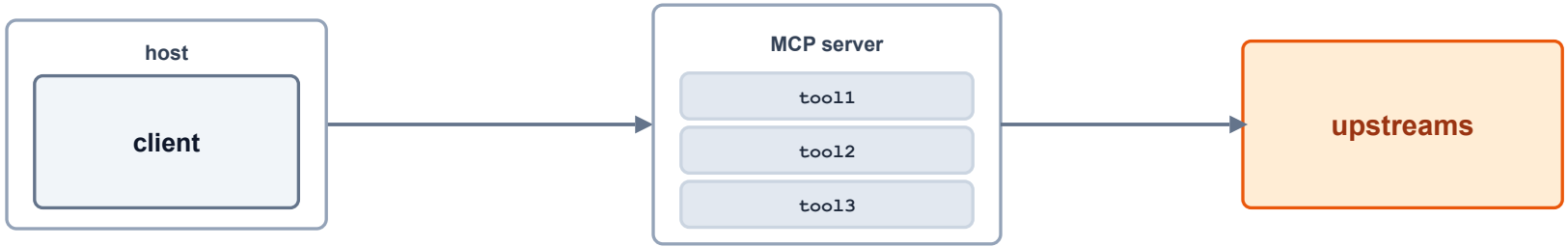
QUESTION 03 What identity reaches the thing behind the server?



SECURITY CHARACTERISTIC

A new token from the upstreams own AS, which can name the user as subject and your server as actor. AS has to be willing to play ball.

The MCP auth design space



Question 1

Who is the caller, and how do they prove it?

- OAuth 2.1 resource server
- static token

Question 2

Who decides what they may do, and where is that enforced?

- both data path and upstream
- only upstream

Question 3

What identity reaches the thing behind the server?

- the server's own
- the user's, freshly issued token



Image by Rawpixel.com

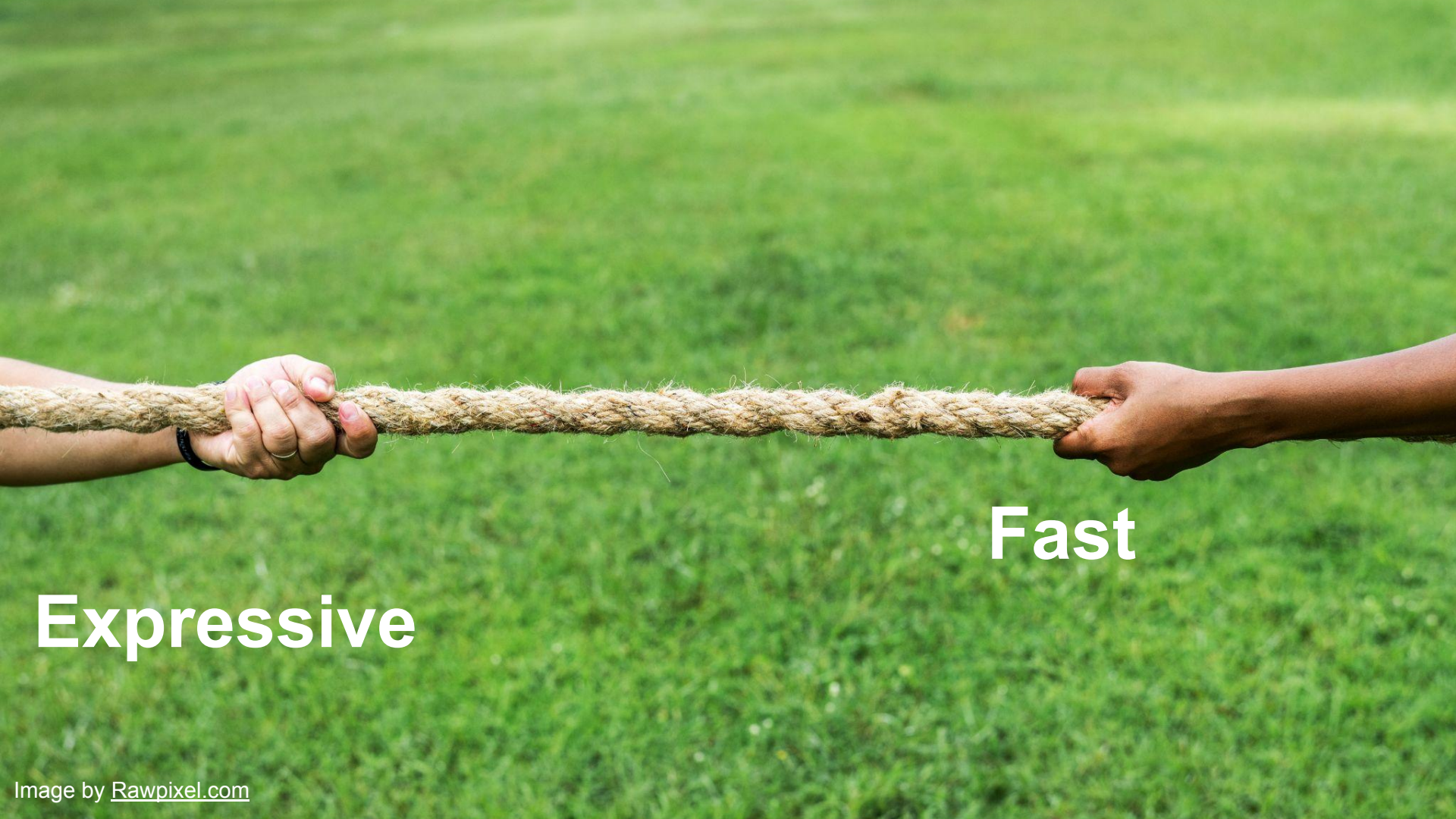


Image by Rawpixel.com



Image by Rawpixel.com



Image by Rawpixel.com



Image by Rawpixel.com

Correct



Authorization language and engine,
a CNCF project

Expressive

Fast
Safe
Analyzable

Correct



Authorization language and engine,
a CNCF project

Attribute-,
Role-, and
Relation-based
Access Control

Expressive

Fast
Safe
Analyzable

Correct



Attribute-,
Role-, and
Relation-based
Access Control

Authorization language and engine,
a CNCF project

Expressive

Primary language tenets

Fast
Safe
Analyzable

Correct

Rock-solid:
Formally verified
in Lean



Authorization language and engine,
a CNCF project

Attribute-,
Role-, and
Relation-based
Access Control

Expressive

Primary language tenets

Fast
Safe
Analyzable



KubeCon CloudNativeCon North America 2026

Image by Jaroslav Machacek

Open Policy Agent
OpenFGA
Cedar
CEL

P = some principal (user)
A = some action
R = some resource with object *known*

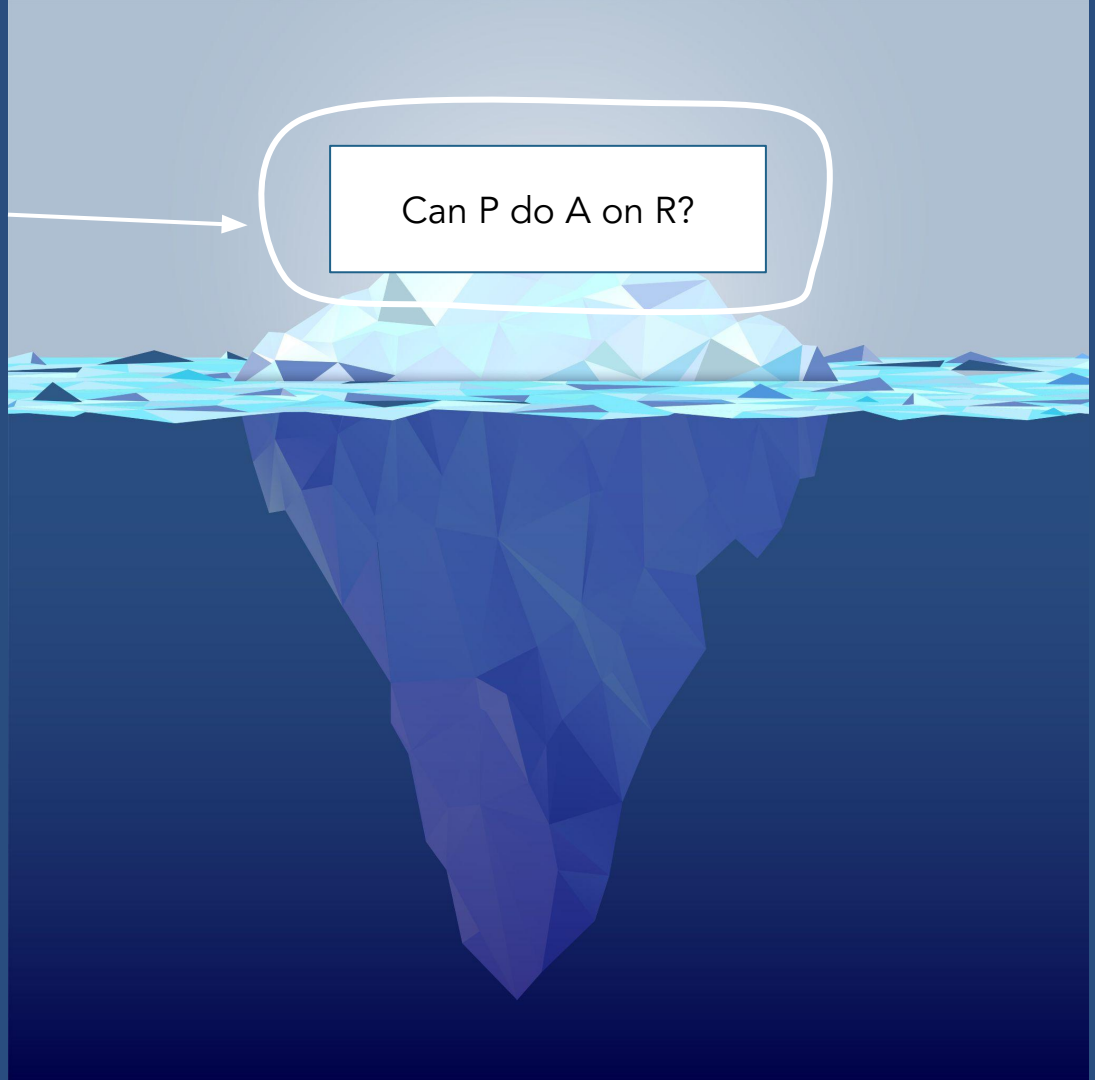
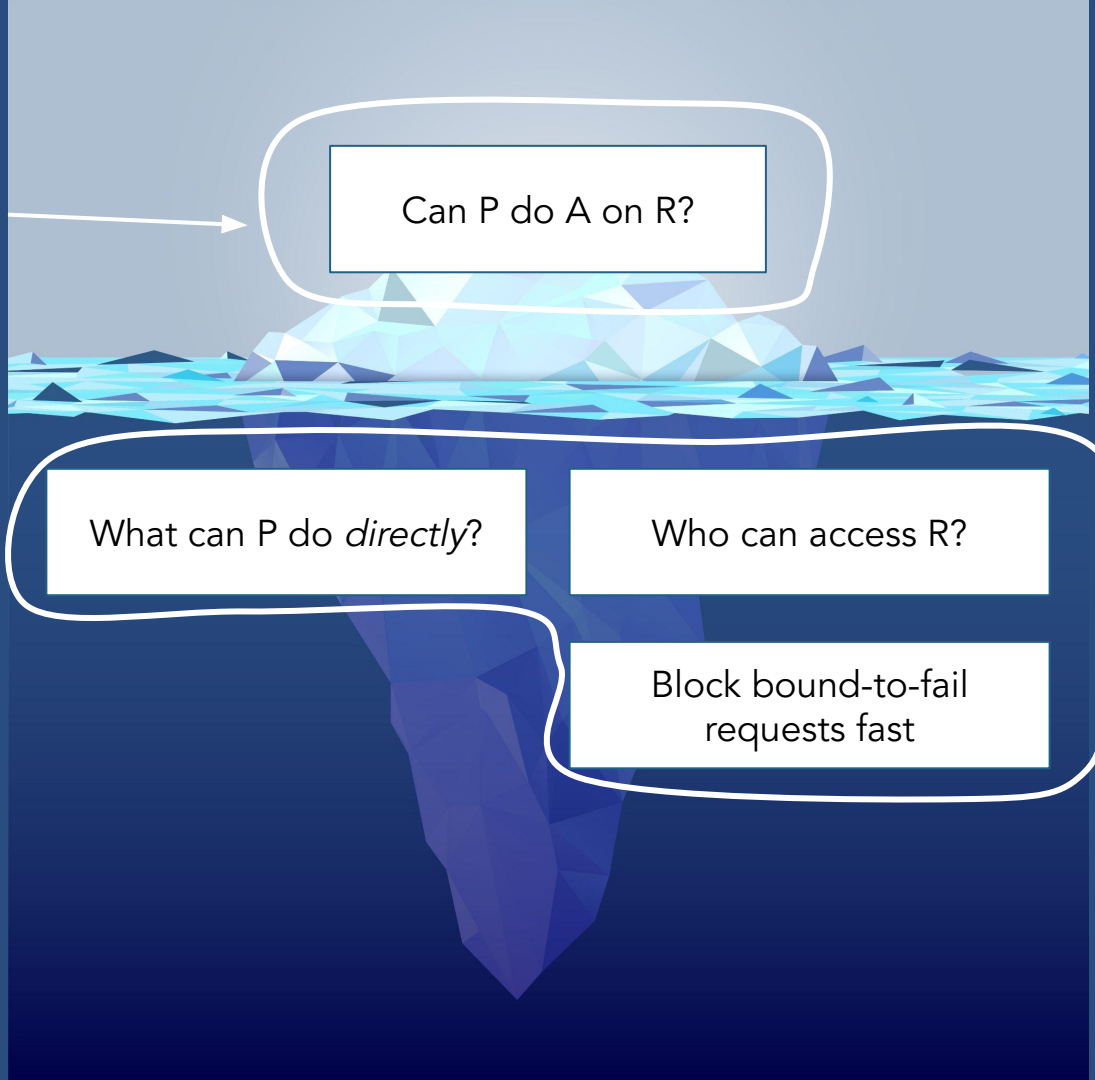


Image by Jaroslav Machacek

Open Policy Agent
OpenFGA
Cedar
CEL

P = some principal (user)
A = some action
R = some resource with object *known*



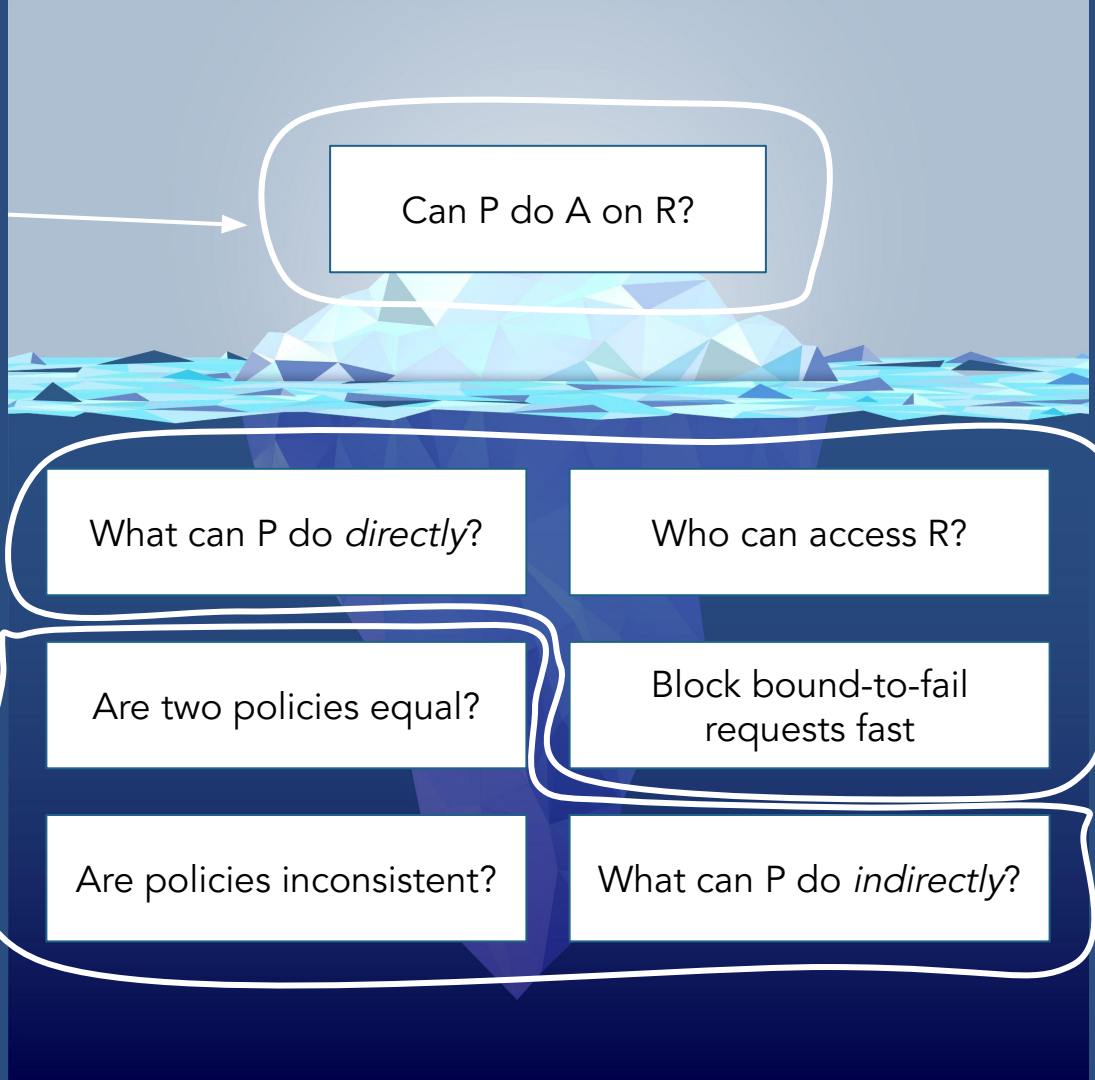
OPA PE
OpenFGA List*
Cedar TPE
CEL

Image by Jaroslav Machacek

Open Policy Agent
OpenFGA
Cedar
CEL

P = some principal (user)
A = some action
R = some resource with object *known*

Cedar Analysis
CEL?

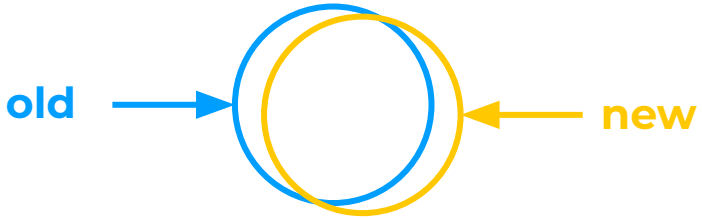


OPA PE
OpenFGA List*
Cedar TPE
CEL

Image by Jaroslav Machacek

Cedar Analysis: Answer questions *about policies*

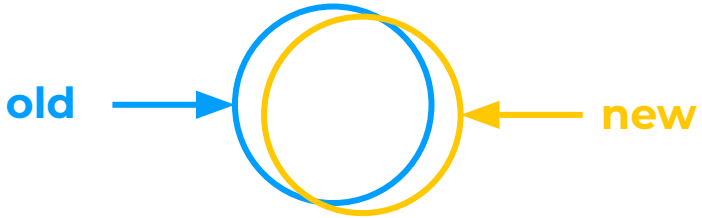
For refactor correctness:
“Are two policies equivalent?”



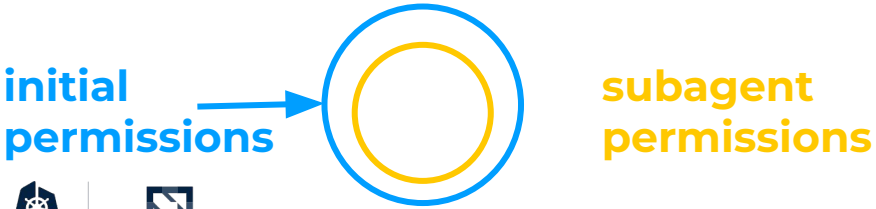
Powered by automated reasoning,
a mathematical logic called SMT

Cedar Analysis: Answer questions *about policies*

For refactor correctness:
“Are two policies equivalent?”



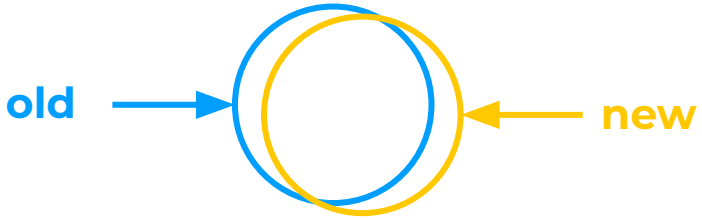
For privilege escalation prevention:
“Is policy A a subset of policy B?”



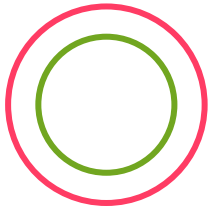
Powered by automated reasoning,
a mathematical logic called SMT

Cedar Analysis: Answer questions *about policies*

For refactor correctness:
“Are two policies equivalent?”

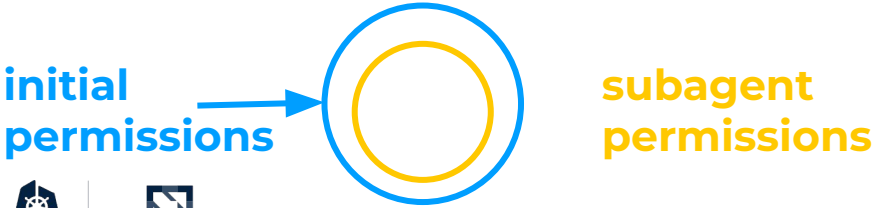


Find logical inconsistencies:



← Deny shadows allow

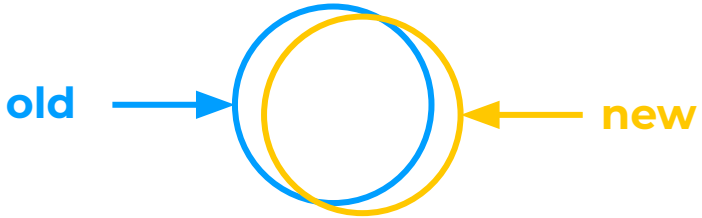
For privilege escalation prevention:
“Is policy A a subset of policy B?”



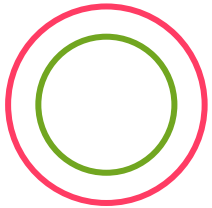
Powered by automated reasoning,
a mathematical logic called SMT

Cedar Analysis: Answer questions *about policies*

For refactor correctness:
“Are two policies equivalent?”

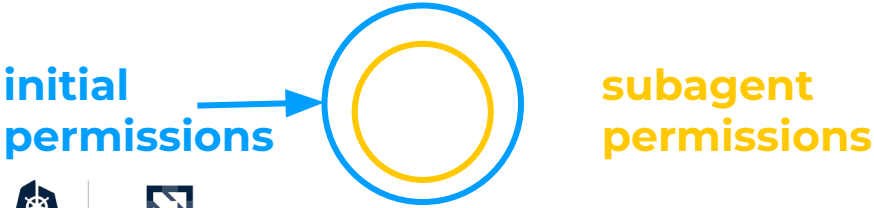


Find logical inconsistencies:



← **Deny shadows allow**

For privilege escalation prevention:
“Is policy A a subset of policy B?”

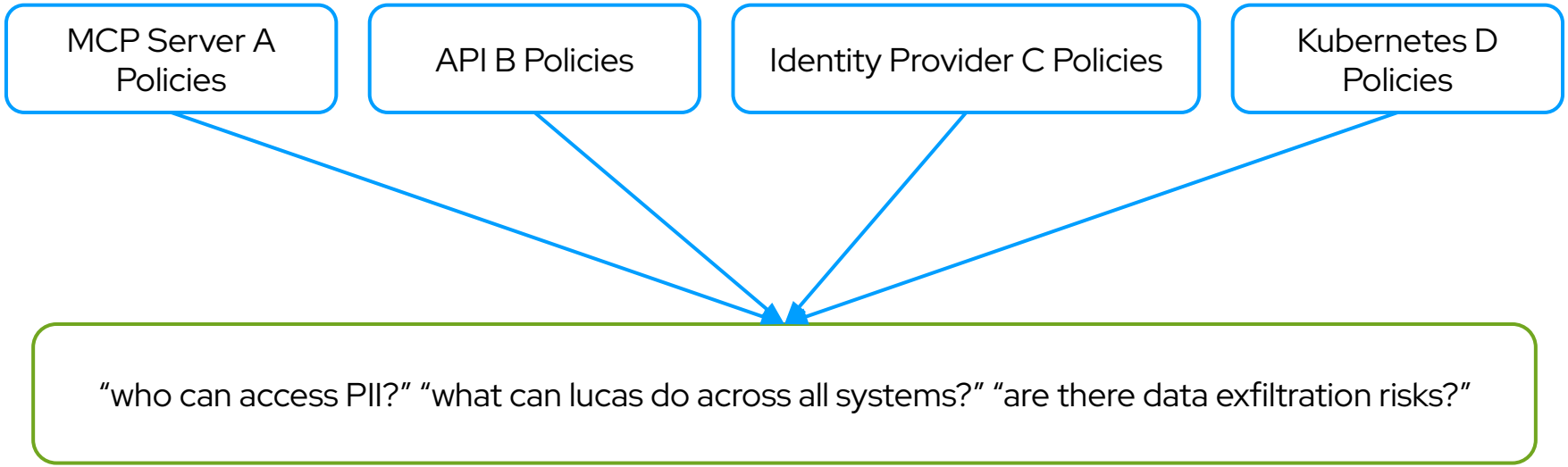


Simplify policies
by finding dead branches:

← **Never true**

Powered by automated reasoning,
a mathematical logic called SMT

Cedar as a lingua franca for access control?



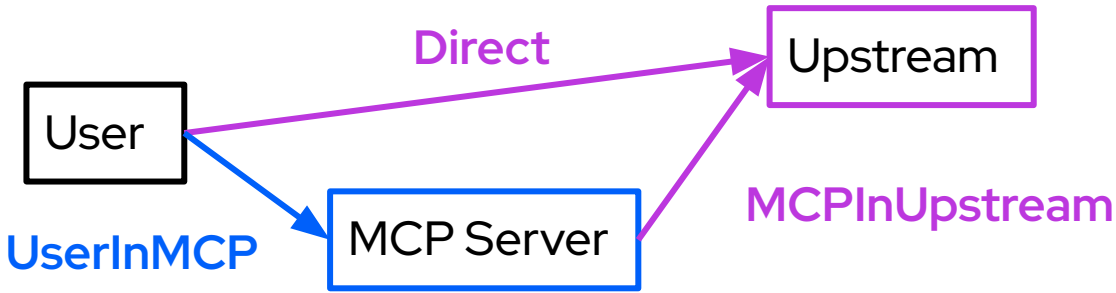
See the Typed Partial Evaluation blog post how Cedar can answer the first two questions.

Use-case: Reasoning about cross-system privileges

If the MCP server's identity is used in the backend,
the user has permissions through two paths.

If all permissions are expressed in the same language, we can calculate*:

Implicit = UserInMCP AND MCPInUpstream
Effective = Direct OR Implicit



*or over-approximate

cedar-woodpecker: Calculating with permissions

- Not specific to Kubernetes or MCP: **fully generalized**
- A layer on top of Cedar core, with small modifications,

  I plan to upstream the features
- Work in progress, and I appreciate feedback. **Get involved!**
- How it works, in technical detail:

  github.com/luxas/cedar-woodpecker

README | Code of conduct | Contributing | Apache-2.0 license | Security

`cedar-woodpecker` on top of `cedar`

Summary: This repository contains four new feature ideas (accompanied by for-now LLM-generated proofs of concept), on top of the baseline `cedar` respository. It is worth pointing out that the author of the ideas is Lucas Käldström (@luxas), the ideas are **NOT** LLM-generated, LLMs have just been used as a tool to show something concrete (during the so far short time between Aug 25 - Sept 12 I've had developing these ideas). It should also be mentioned that this `README.md` document is completely written by me, and I consider this document to be the primary contribution in this repo. Eventually, I intend to improve (most likely by starting more or less from scratch with regards to every feature PR) the code, docs, tests and proofs, if the ideas make sense to the Cedar community. I'm happy to upstream the features to Cedar as well, into core or some extension, as wanted/needed. My main goal at this moment is to:

1. validate the ideas by discussing with others (I'm not 100% certain of whether all parts of the ideas make sense)
2. ask for feedback on the direction by the community (feel free to email me about this, or send a DM on the CNCF Slack or LinkedIn!)
3. showcase in hopefully a little bit more detail what it means for Cedar to be "analyzable" and/or have a "decidable encoding into mathematical logic", and what the benefits and opportunities are

> **WARNING:** The commits in this repo on top of mainline `cedar` **MUST NOT be used in production**; they serve only as a concretization of the ideas presented, to evaluate what direction to evolve the ideas towards, and to give something that can be experimentally tested and iterated on, in search of the final form of the features.

The original `README.md` for mainline Cedar can be found in `README-old.md`. In the rest of this text, I'll summarize the ideas in a blog post-style way initially, then go more technical. For now, I've only had time to go one pass over this text, I'll try to make it more understandable over time, if needed. I intend to turn this text into a series of "real" blog posts later, most likely going into more depth on each topic there.

Show me the source, Luke: Basically, my writing here is a summary of my scratch notes in my valued notebooks (see picture below), on top of which this proof of concept is based.



Woodpecker

MCP proxy in front of GitHub: cross-system permissions

See what principals can do beyond their explicit permissions. Woodpecker follows permissions that lead to other permissions (creating a pod that mounts a secret lets you read the secret) and identity assumptions (acting as another principal), and writes each finding as a Cedar policy. It uses Cedar's symbolic compiler with the CVC5 solver, running in your browser.

Schema and policies | Implicit permissions | Cubes

Cubes

Up to 5 rounds | Run

A cube is one way a request is allowed: a condition under which a policy permits requests in one request environment. Round 0 holds the explicit permissions; each later round adds what the permissions and identity assumptions imply from the rounds before it.

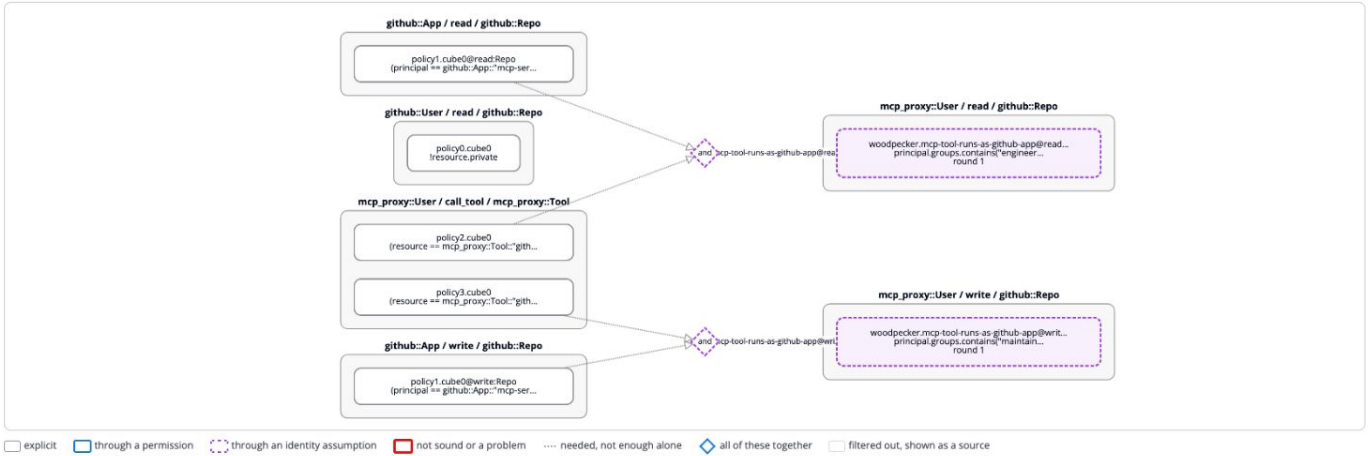
5 explicit and 2 implied permissions, in 2 rounds. Nothing new was found after the last round.

All environments | All rounds | All kinds | List | Graph

Cube graph

Fit

Each box is a request environment; arrows go from source cubes to the cubes they imply, labelled with the permission or identity assumption. A diamond joins sources that are all needed. Rounds flow to the right.



Details

Select a cube, an "and" join or an arrow in the graph.

github::App read github::Repo

EXPLICIT (ROUND 0)

policy1.cube0@read:Repo explicit can evaluate to True, False, Error

When

```
(principal == github::App::"mcp-server") && (resource.org == "acme")
```

Use-case: A calculus for permissions



Use-case: A calculus for permissions

Permissions that give access to internal data:

- List Kubernetes API objects
- Read internal wiki
- ...

Permissions that allow sending data externally:

- Write to a GitHub issue
- Send email
- ...

Use-case: A calculus for permissions

<Permission to get internal data>	+	<Permission to write data externally>	=	<Meta-permission called “exfiltrate data”>

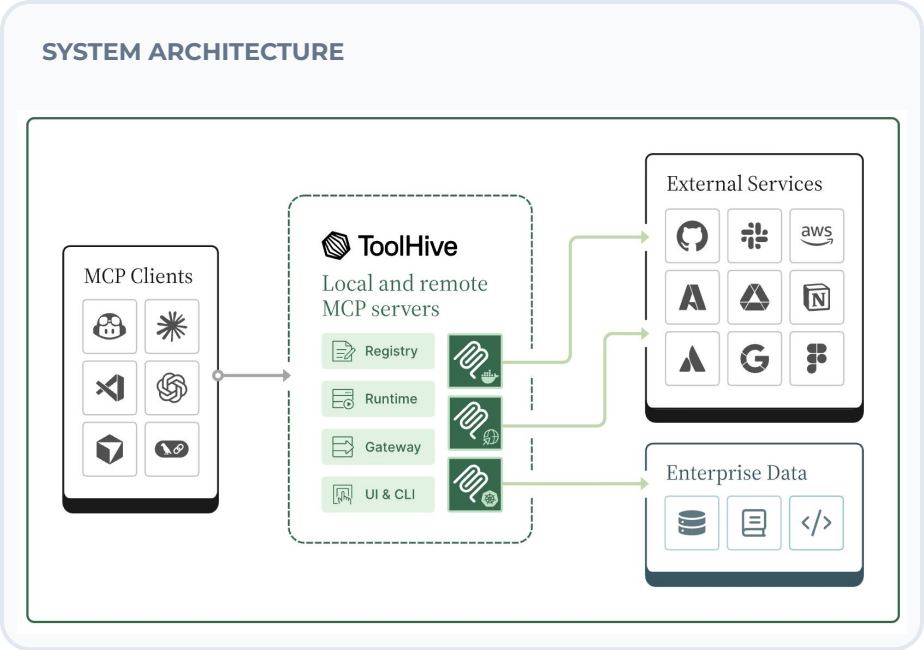
List Kubernetes API objects	+	Write to a GitHub issue	=	“Finding” of data exfiltration
List Kubernetes API objects	+	Send email	=	“Finding” of data exfiltration
Read internal wiki	+	Write to a GitHub issue	=	“Finding” of data exfiltration
Read internal wiki	+	Send email	=	“Finding” of data exfiltration

What is ToolHive?

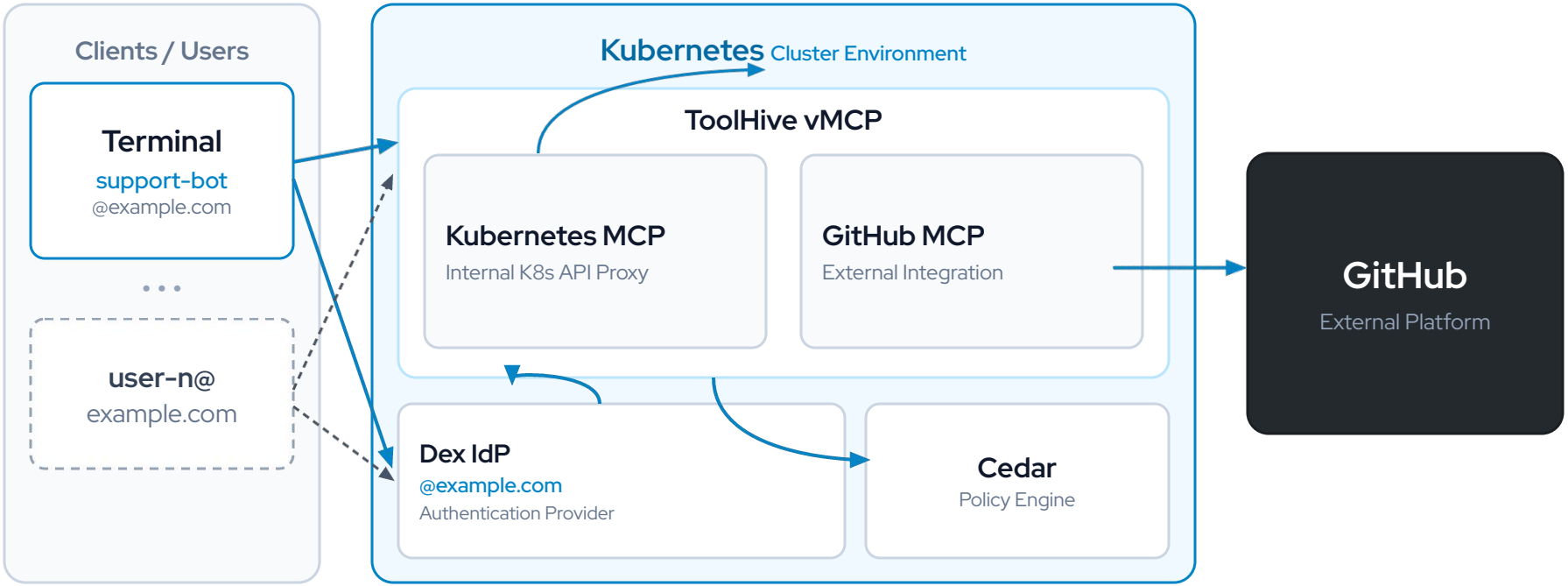
A Control Plane for MCP

ToolHive provides a secure, unified gateway that connects Model Context Protocol (MCP) clients to local and remote servers.

- **Unified MCP Hub**
  Bridges diverse clients (Cursor, Claude Code etc) with multiple local and remote servers.
- **Core Capabilities**
  Built-in Registry for tools, secure Runtime execution and routing Gateway.
- **Secure Integrations**
  Safely exposes external SaaS APIs (Slack, GitHub, AWS etc) and sensitive Enterprise Data stores.



Demo Architecture



Demo Policies

```
permit (
    principal in THVGroup::"engineering",
    action == Action::"call_tool",
    resource in [Tool::"list_resources"]
);
```

```
permit (
    principal in THVGroup::"support",
    action == Action::"call_tool",
    resource in [
        Tool::"issue_read",
        Tool::"add_issue_comment"]
);
```


Terminal images generated with carbon.now.sh

Demo



https://github.com/ChrisJBurns/toolhive-cedar-demo

Demo showing ToolHive, Cedar, and cedar-woodpecker in action

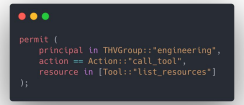


https://github.com/luxas/cedar-woodpecker

Experimental “permissions calculus” proof-of-concept built on top of Cedar Analysis

Unwanted implicit permission found statically!

Input:
Explicit
permissions



+



Input:
Calculation
rules

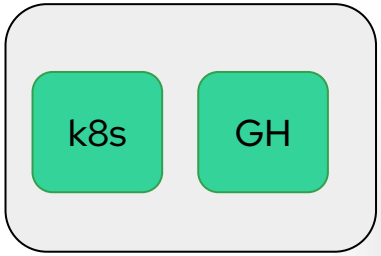
<Permission to get internal data> AND <Permission to write data externally> = <Meta-permission called "exfiltrate data">

Output:
Implied
permissions



https://github.com/luxas/cedar-woodpecker

Unwanted implicit permission found statically!



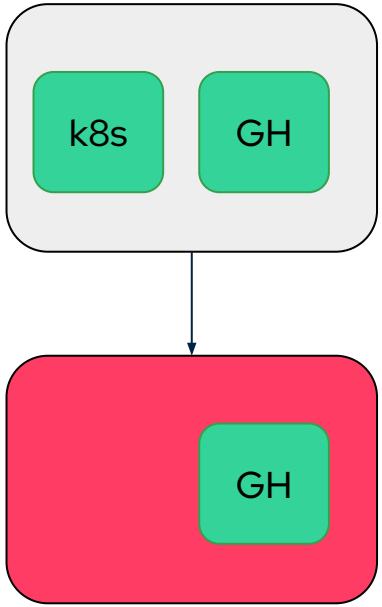

```
permit(
  principal is Client,
  action == Action::"exfiltrate_data",
  resource
) when {
  principal in THVGroup::"engineering" &&
  principal in THVGroup::"support"
};
```

Legend:

Complete permission space

Allow policy

Let’s bound support-bot’s permissions




```
forbid(
    principal in THVGroup::"support",
    action == Action::"call_tool",
    resource
) unless {
    resource in [
        Tool::"add_issue_comment",
        Tool::"issue_read"]
};
```

Legend:

Complete permission space

Allow policy

Deny policy

woodpecker confirms no escalations are possible 🎉

Takeaways

1. MCP servers should use a generic engine for access control
   a. Cedar, Open Policy Agent and OpenFGA are all good options

2. Cedar offers powerful policy querying and analysis features
   a. Ideally, policies from multiple systems can be centrally reasoned about

3. We showed a proof-of-concept of “calculating with permissions”
   a. We could statically identify a data exfiltration risk and later prove the fix

KubeCon CloudNativeCon North America 2026

Let’s continue the discussion!

Join Cedar: github.com/cedar-policy
Join CNCF Slack: slack.cncf.io

Lucas Käldström (LinkedIn, GitHub, SpeakerDeck, CNCF Slack)
Chris Burns (LinkedIn, GitHub)

All trademarks mentioned in this presentation are the property of their respective owners.

