

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

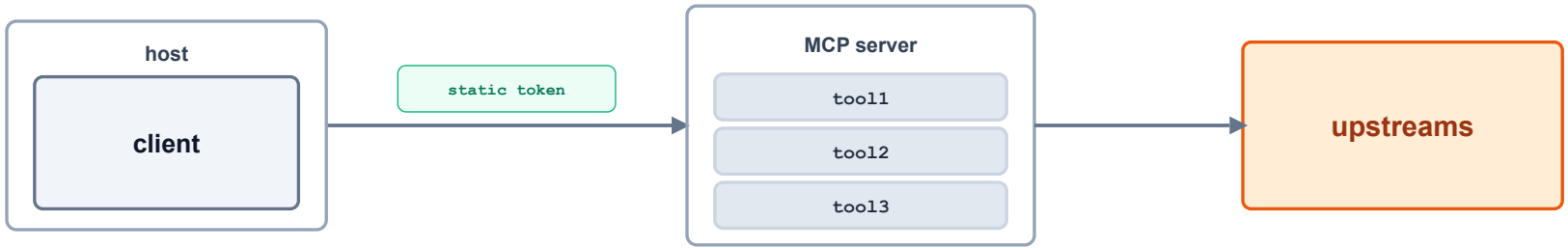


Who is the caller, and how do they prove it?

Authentication
Identity verification & credential proof

Static token

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

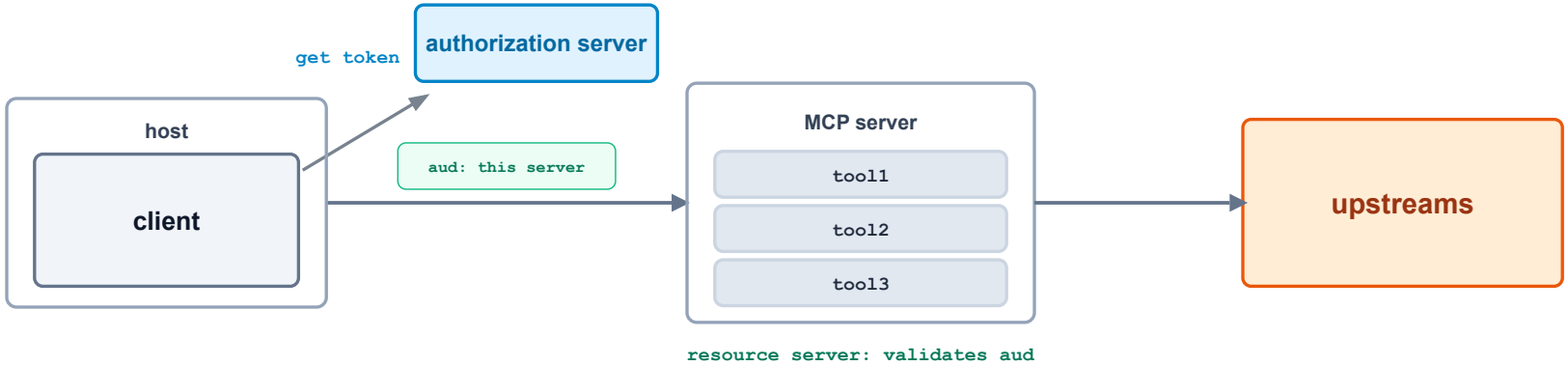


SECURITY CHARACTERISTIC

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

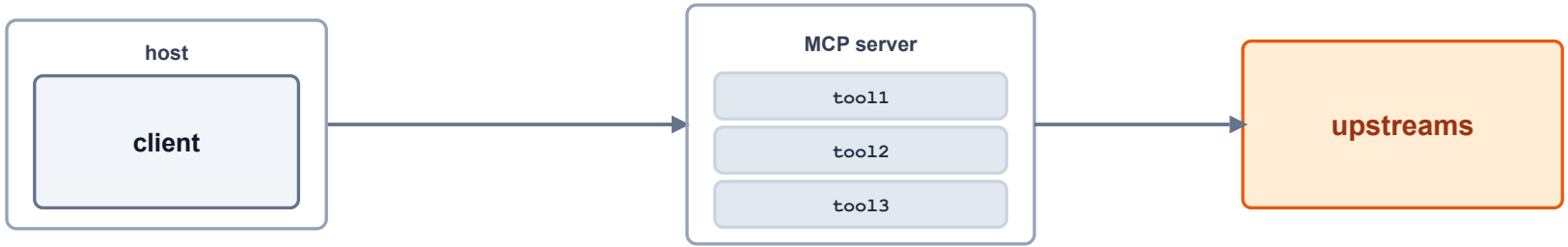


Who decides what they may do, and where is that enforced?

Authorization
Policy enforcement & permission boundaries

Only upstream authorizes

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

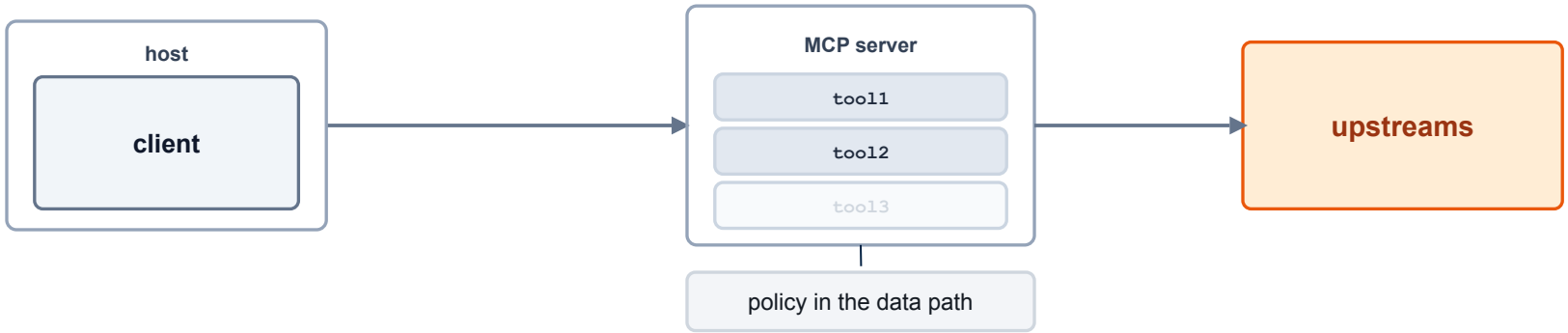


NOTE

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?



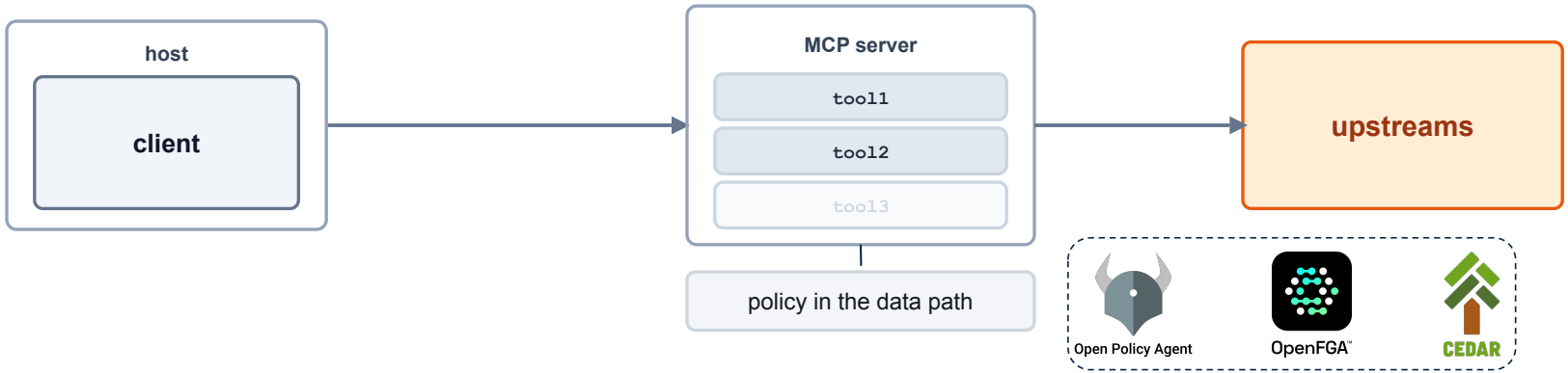
NOTE

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

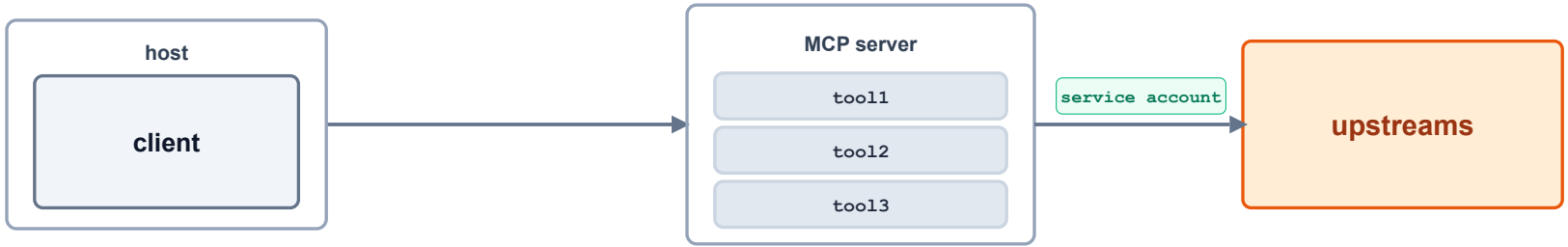


What identity reaches the thing behind the server?

Delegation & Identity
Upstream propagation & backend access

The server's own

QUESTION 03 What identity reaches the thing behind the server?

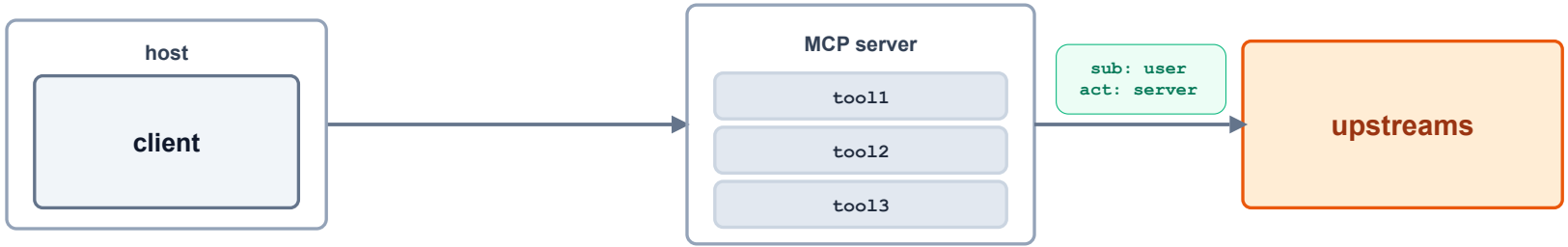


SECURITY CHARACTERISTIC

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

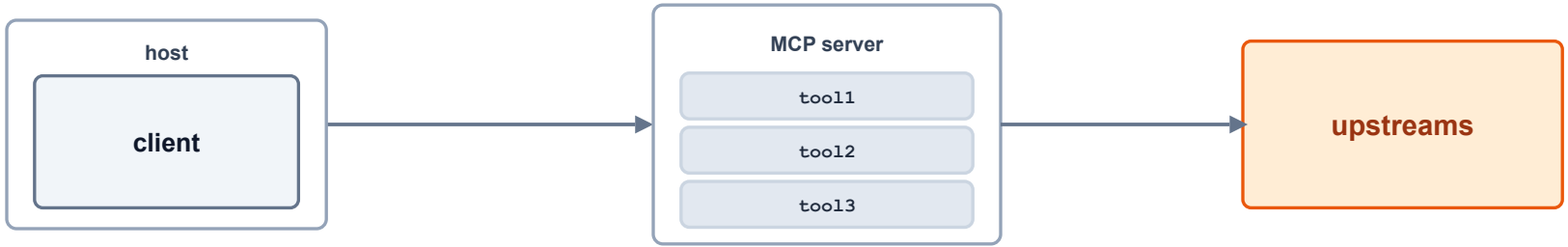
QUESTION 03 What identity reaches the thing behind the server?



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
- token passthrough



Image by Rawpixel.com

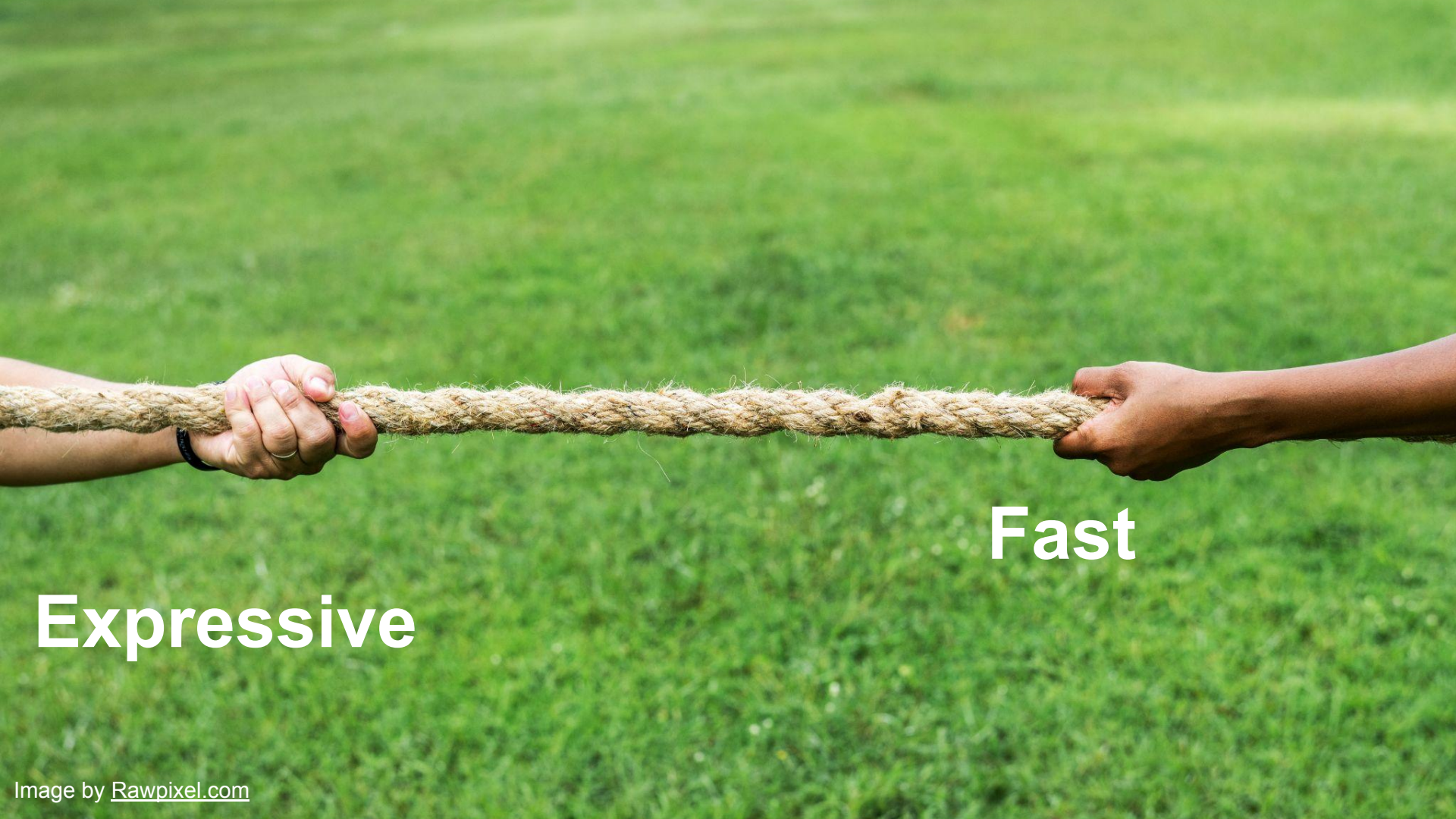


Image by Rawpixel.com



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

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

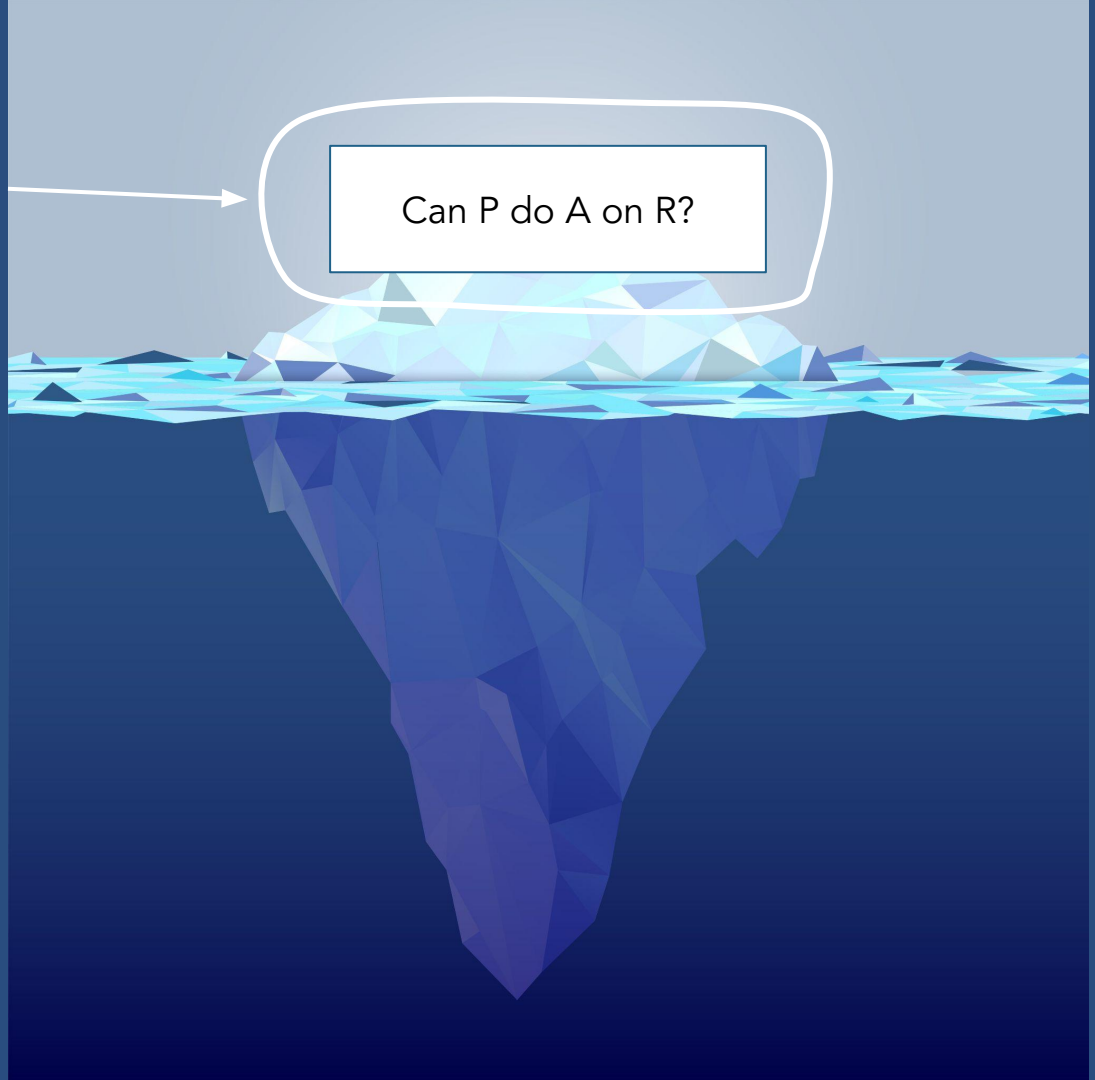
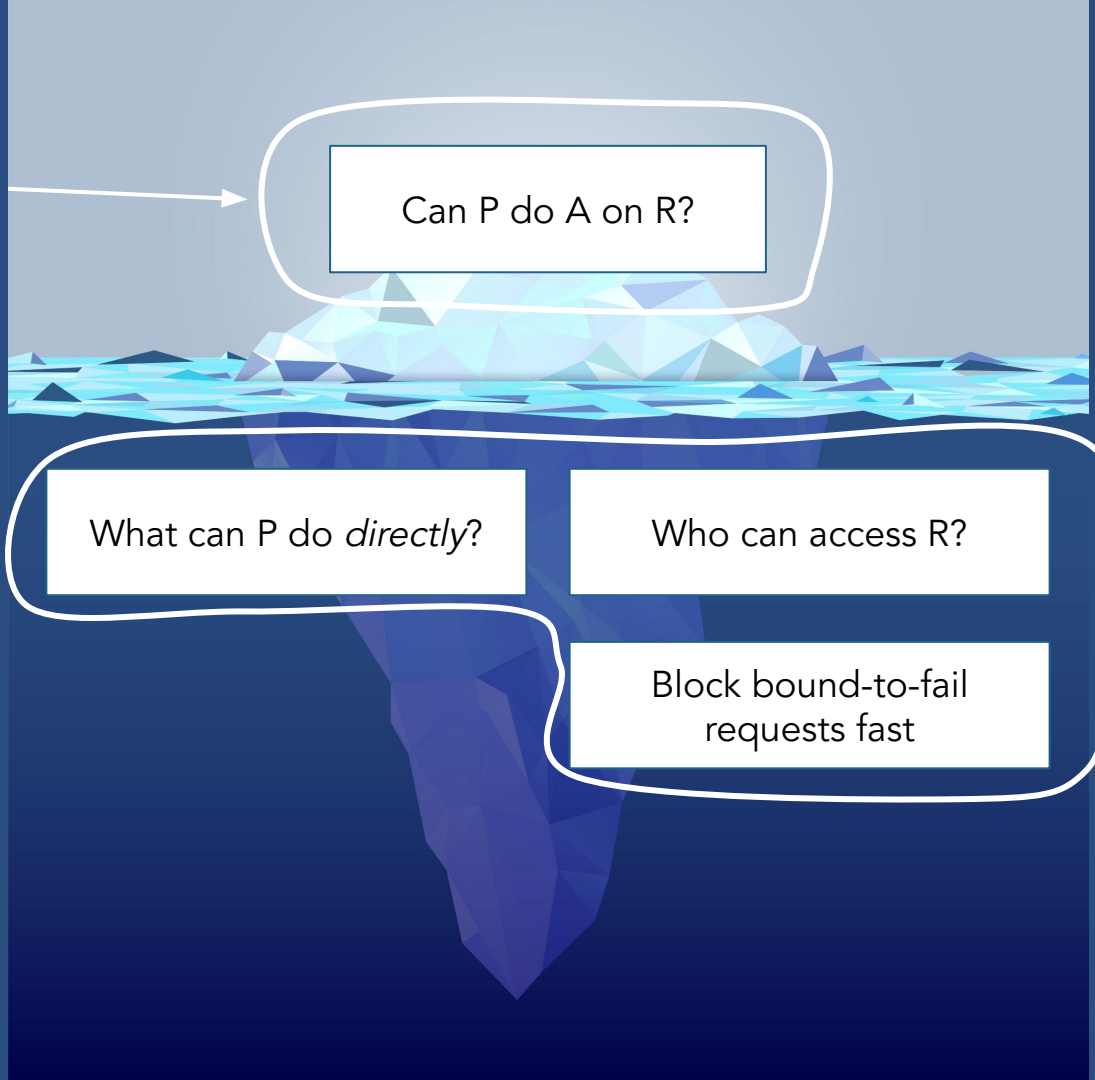


Image by Jaroslav Machacek

Open Policy Agent
OpenFGA
Cedar

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

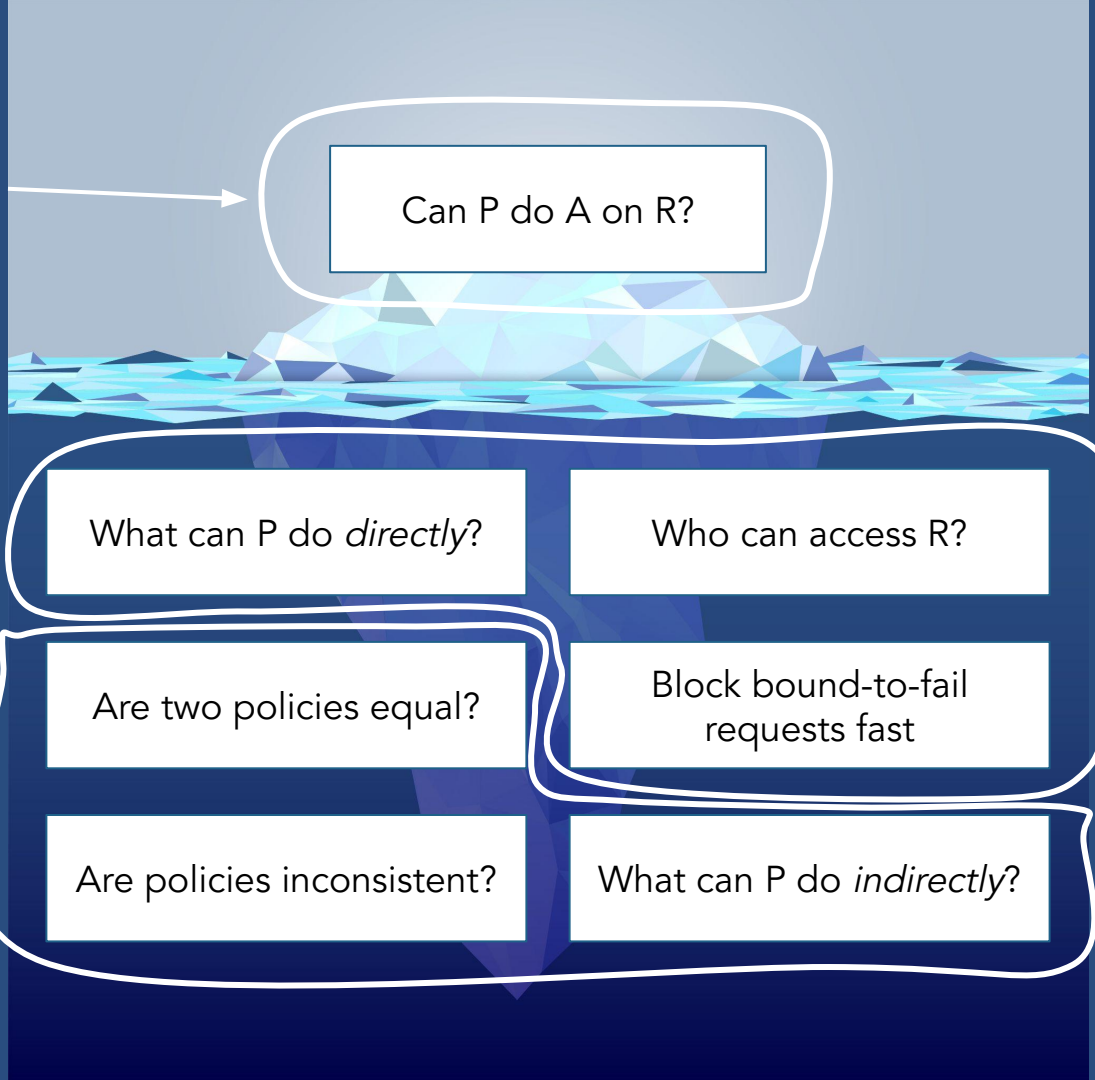


OPA PE
OpenFGA List*
Cedar TPE

Image by Jaroslav Machacek

Open Policy Agent
OpenFGA
Cedar

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



OPA PE
OpenFGA List*
Cedar TPE

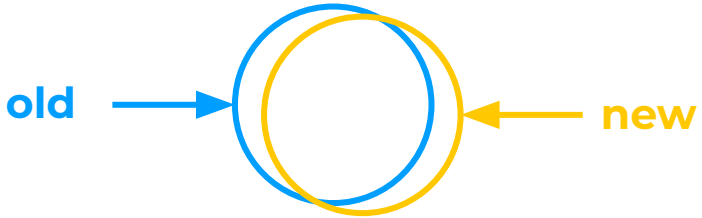
Cedar Analysis

KubeCon CloudNativeCon North America 2026

Image by Jaroslav Machacek

Cedar Analysis: Answer questions *about policies*

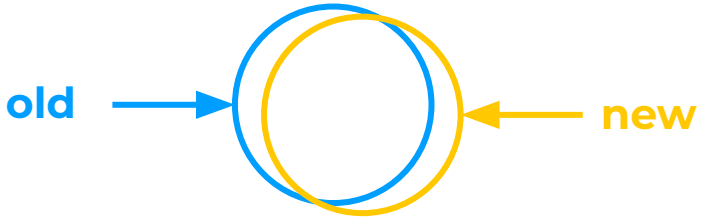
For refactor correctness:
“Are two policies equivalent?”



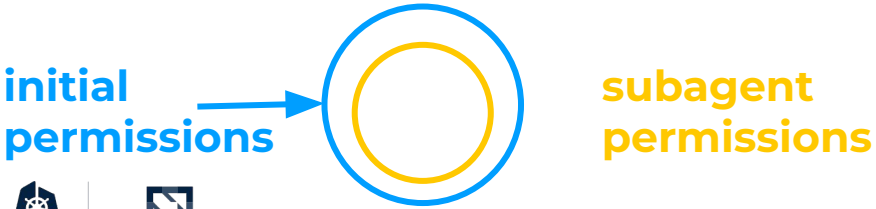
Powered by automated reasoning,
a mathematical logic called SMT

Cedar Analysis: Answer questions *about policies*

For refactor correctness:
“Are two policies equivalent?”



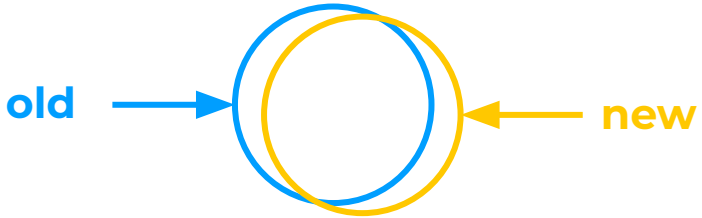
For privilege escalation prevention:
“Is policy A a subset of policy B?”



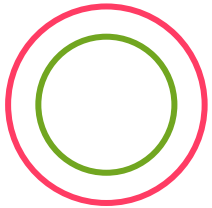
Powered by automated reasoning,
a mathematical logic called SMT

Cedar Analysis: Answer questions *about policies*

For refactor correctness:
“Are two policies equivalent?”

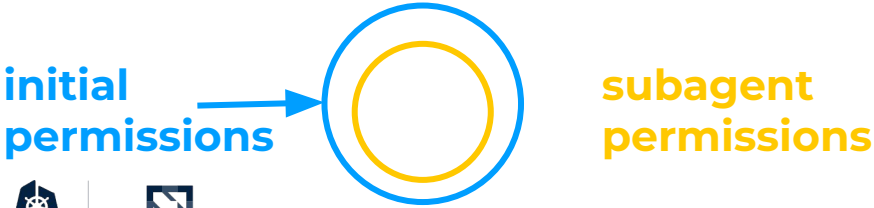


Find logical inconsistencies:



← Deny shadows allow

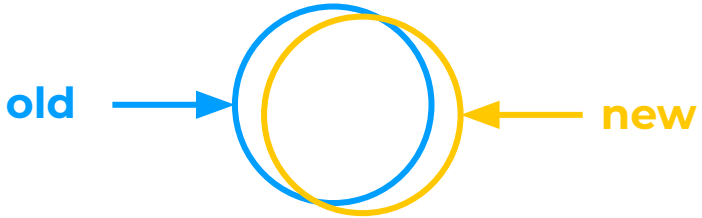
For privilege escalation prevention:
“Is policy A a subset of policy B?”



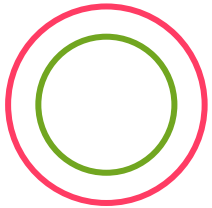
Powered by automated reasoning,
a mathematical logic called SMT

Cedar Analysis: Answer questions *about policies*

For refactor correctness:
“Are two policies equivalent?”

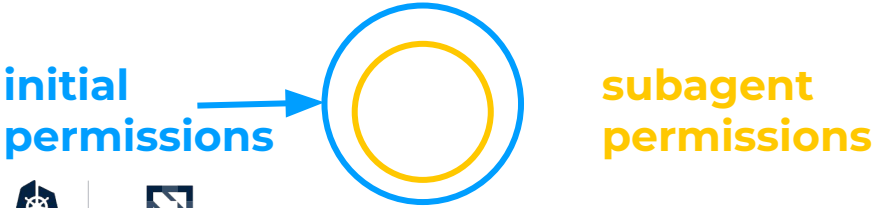


Find logical inconsistencies:



← **Deny shadows allow**

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

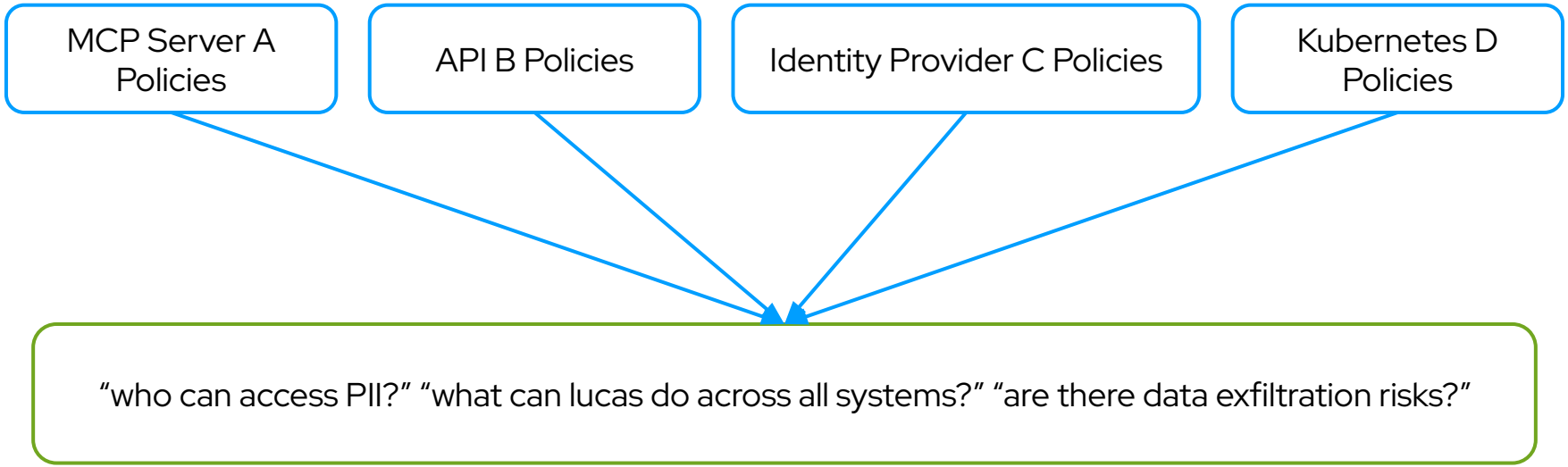


Simplify policies
by finding dead branches:

← **Never true**

Powered by automated reasoning,
a mathematical logic called SMT

Cedar as a lingua franca for access control?



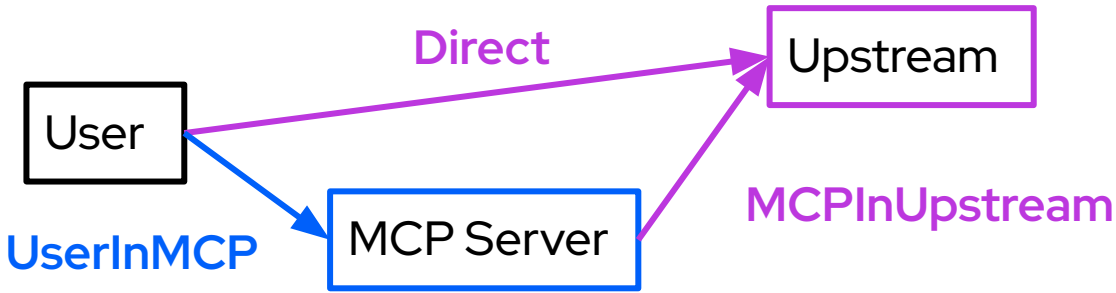
See the Typed Partial Evaluation blog post how Cedar can answer the first two questions.

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

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”>

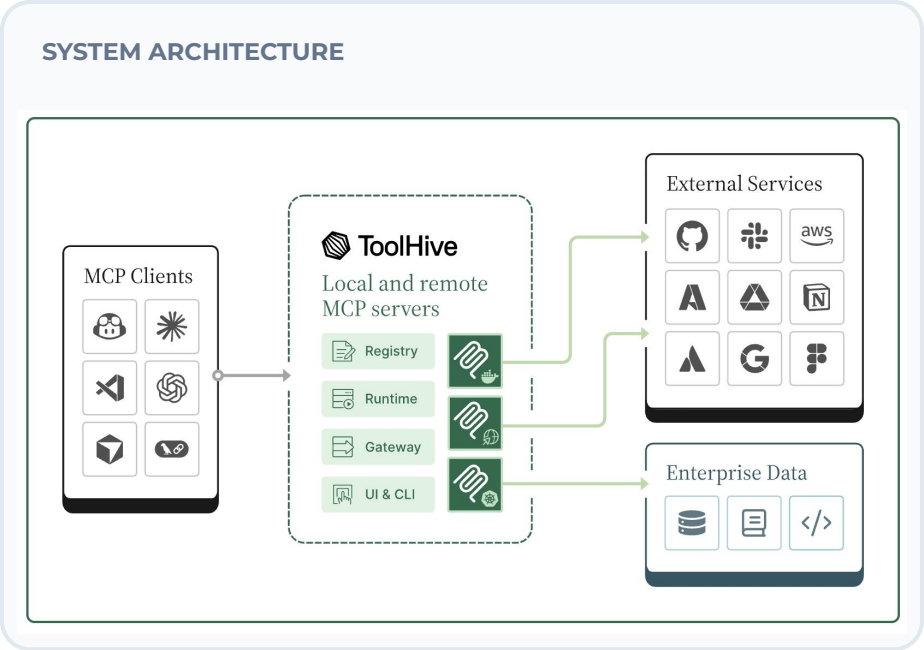
List Kubernetes API objects	+	Write to a GitHub issue	=	“Finding” of data exfiltration
List Kubernetes API objects	+	Send email	=	“Finding” of data exfiltration
Read internal wiki	+	Write to a GitHub issue	=	“Finding” of data exfiltration
Read internal wiki	+	Send email	=	“Finding” of data exfiltration

What is ToolHive?

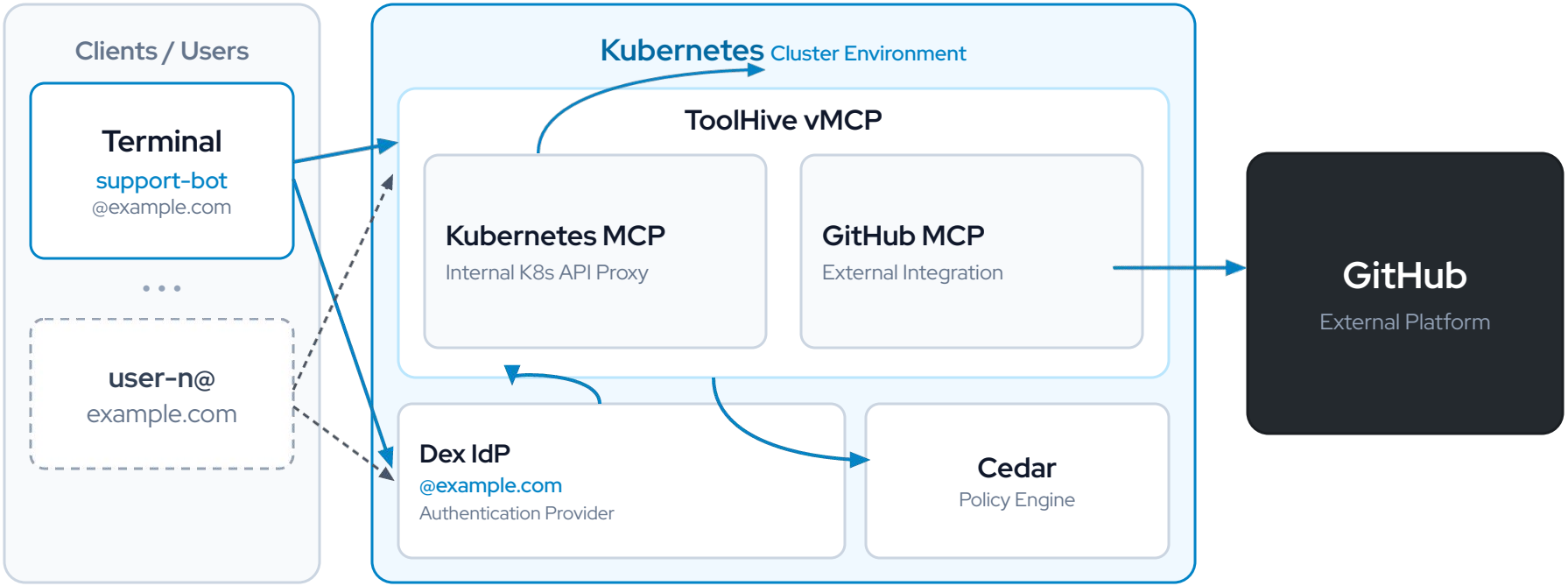
A Control Plane for MCP

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

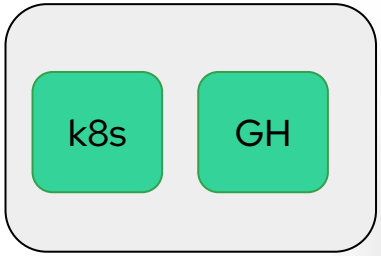
Demo showing ToolHive, Cedar, and cedar-woodpecker in action



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

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

Unwanted implicit permission found statically!



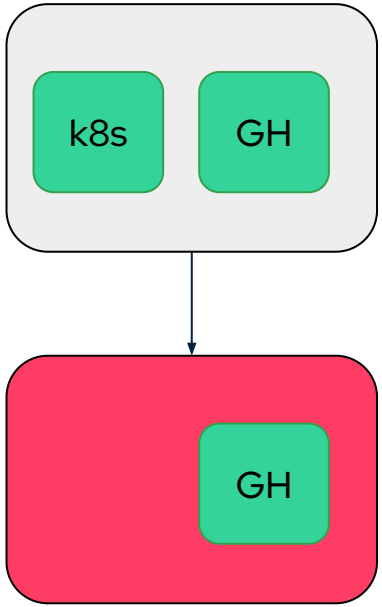

```
permit(
  principal is Client,
  action == Action::"exfiltrate_data",
  resource
) when {
  principal in THVGroup::"engineering" &&
  principal in THVGroup::"support"
};
```

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.

