controlplane

# What Can Go Wrong When You Trust Nobody?

## Threat Modelling Zero Trust

KubeCon | CloudNativeCon

Europe 2023

James Callaghan

Ric Featherstone

# About ControlPlane (Hall 5, Booth SU57)

A Cloud Native security consultancy established in 2017. Our diverse culture empowers and develops individuals with talent and integrity.

We regularly participate in the largest worldwide security conferences.

- Security specialists in cloud, Kubernetes, and containers
- Clients include government, financial services, and regulated industries
- 50 people across the UK, Northern Europe, and Australasia

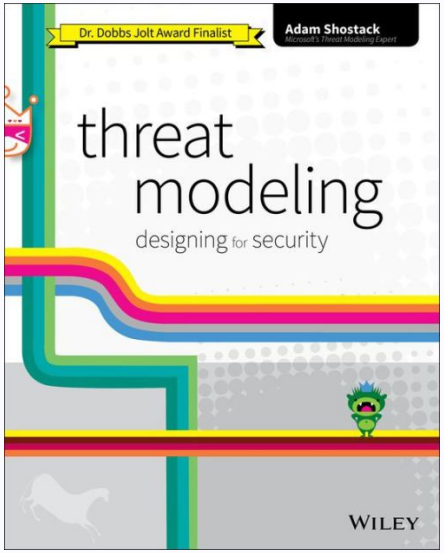
# Overview

How can we:

- derive a **Zero Trust philosophy** from a high-level Threat Model?
- build a **secure architecture** based on these principles?
- carry out a **detailed threat model** of our technical design?
- **understand the risks** posed to our system?
- **define additional controls** to mitigate our risks?
- **demonstrate** our control design using a **prototype**?

This talk will focus on **Zero Trust for workloads**

# What is Threat Modelling?




Threat Modelling is:

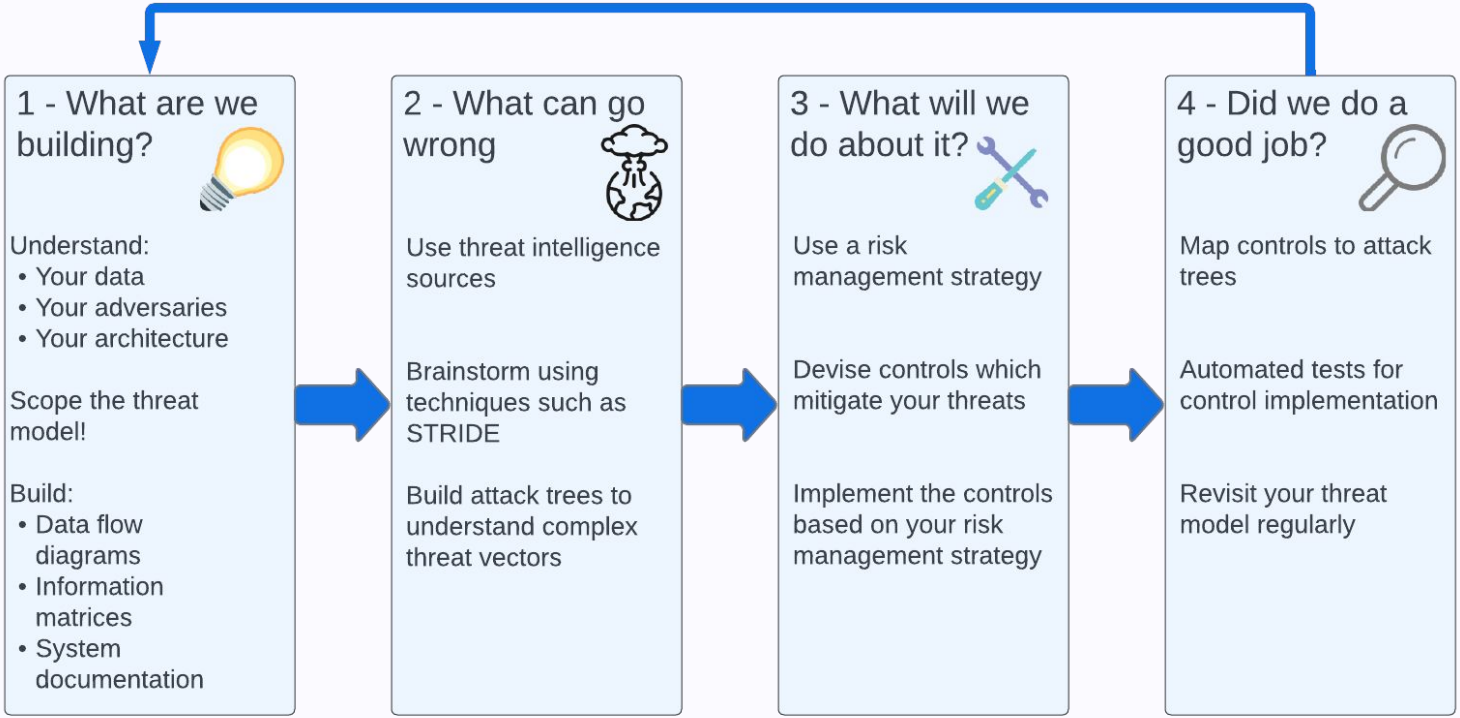
- Identifying and enumerating **threats and vulnerabilities**
- Devising **mitigations**
- Prioritising **residual risks**
- **Escalating** the most important risks

Why Threat Model?

- Identify **security flaws early**
- **Save money and time** consuming redesigns
- Focus your **security requirements**
- Identify **complex risks and data flows** for critical assets

Everyone can (and should!) Threat Model - **not just security teams**

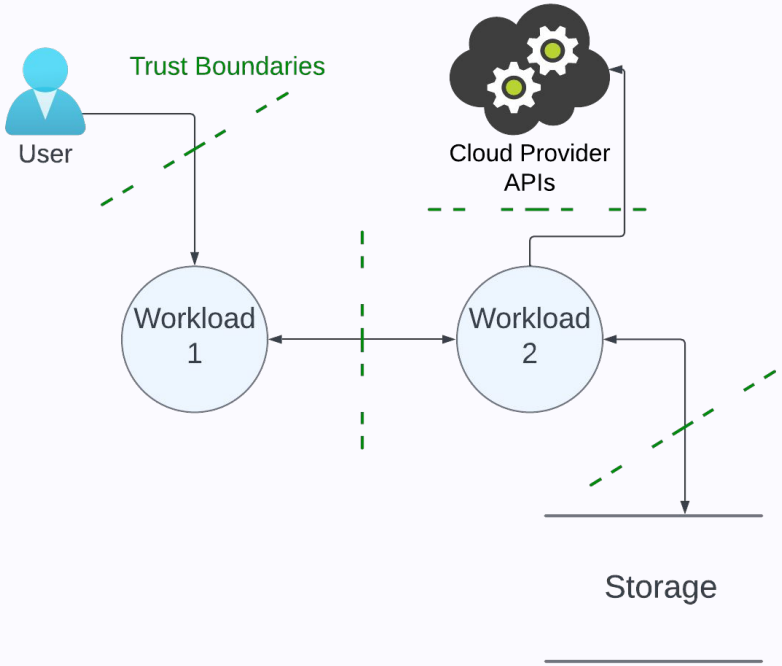
# Threat Modelling is Iterative



# **Making Sense of Zero Trust**

## Deriving Zero Trust Principles via Threat Modelling

# Sky-High-Level Threat Model




**S**poofing

**T**ampering

**R**epudiation

**I**nformation Disclosure

**D**enial of Service

**E**scalation of Privilege

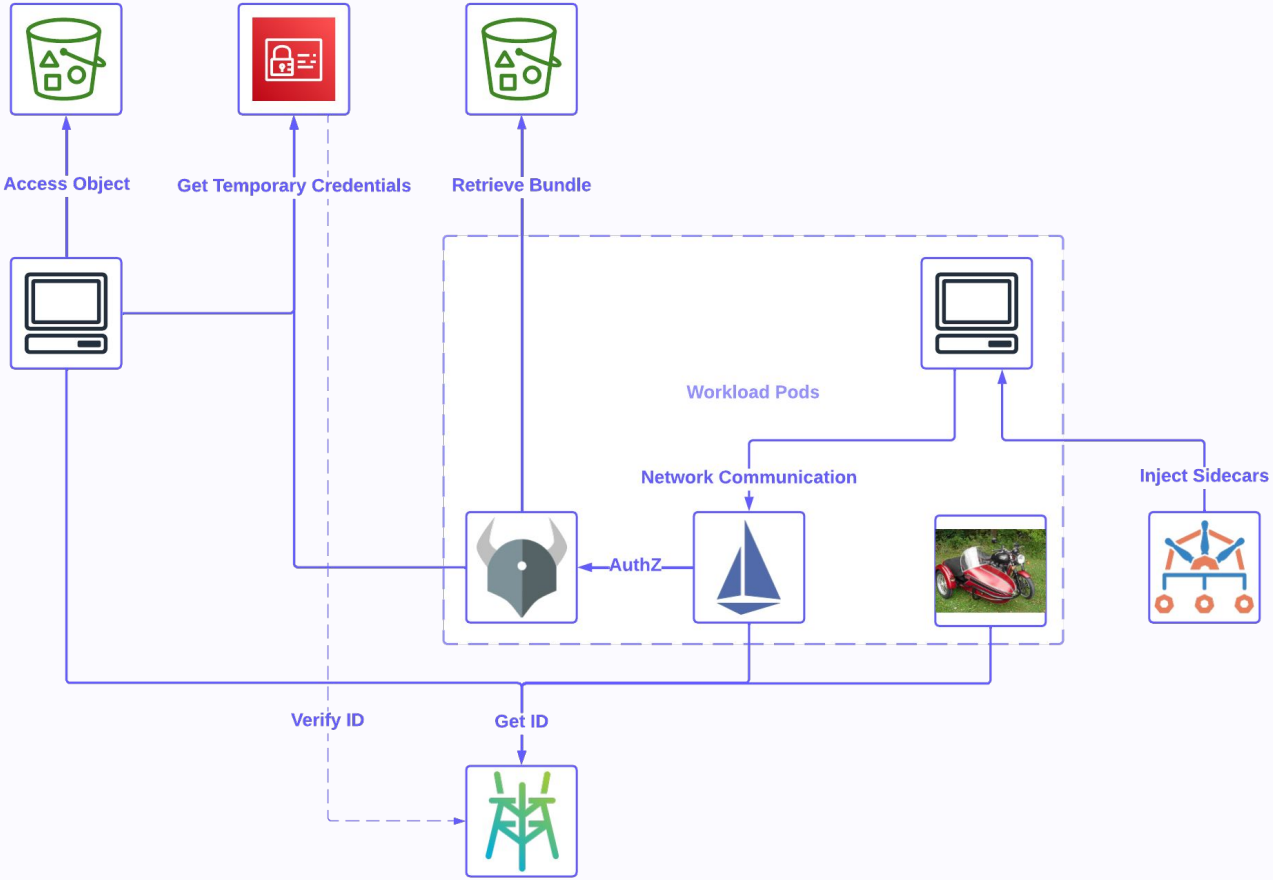
# Deriving our architectural principles

| STRIDE | Threat | Architectural Control |
|---|---|---|
| S | User Impersonation | User **AuthN** best practices<br>IdP controls, e.g. **MFA** |
| S | Workload spoofing (client or server side) | Cryptographically strong **workload identity**<br>**mTLS** |
| T | Alter information in transit | **mTLS** |
| T | Tamper with stored data | Strict **AuthZ** everywhere |
| R | Malicious action not attributable to an identity | **Audit logs** tied back to cryptographically strong identity |
| I | Exfiltrate data | **Egress control**<br>**Policy as versioned code** |
| D | Prevent workloads from communicating | **Highly available** workload identity mechanism |
| E | Compromised workload pivots | **Least Privilege** AuthZ policies |

# What are we Building?

Step 1 - Draw and annotate architecture diagrams

# Architecture Diagrams



# What are we Building?

## Step 2 - Build a Prototype

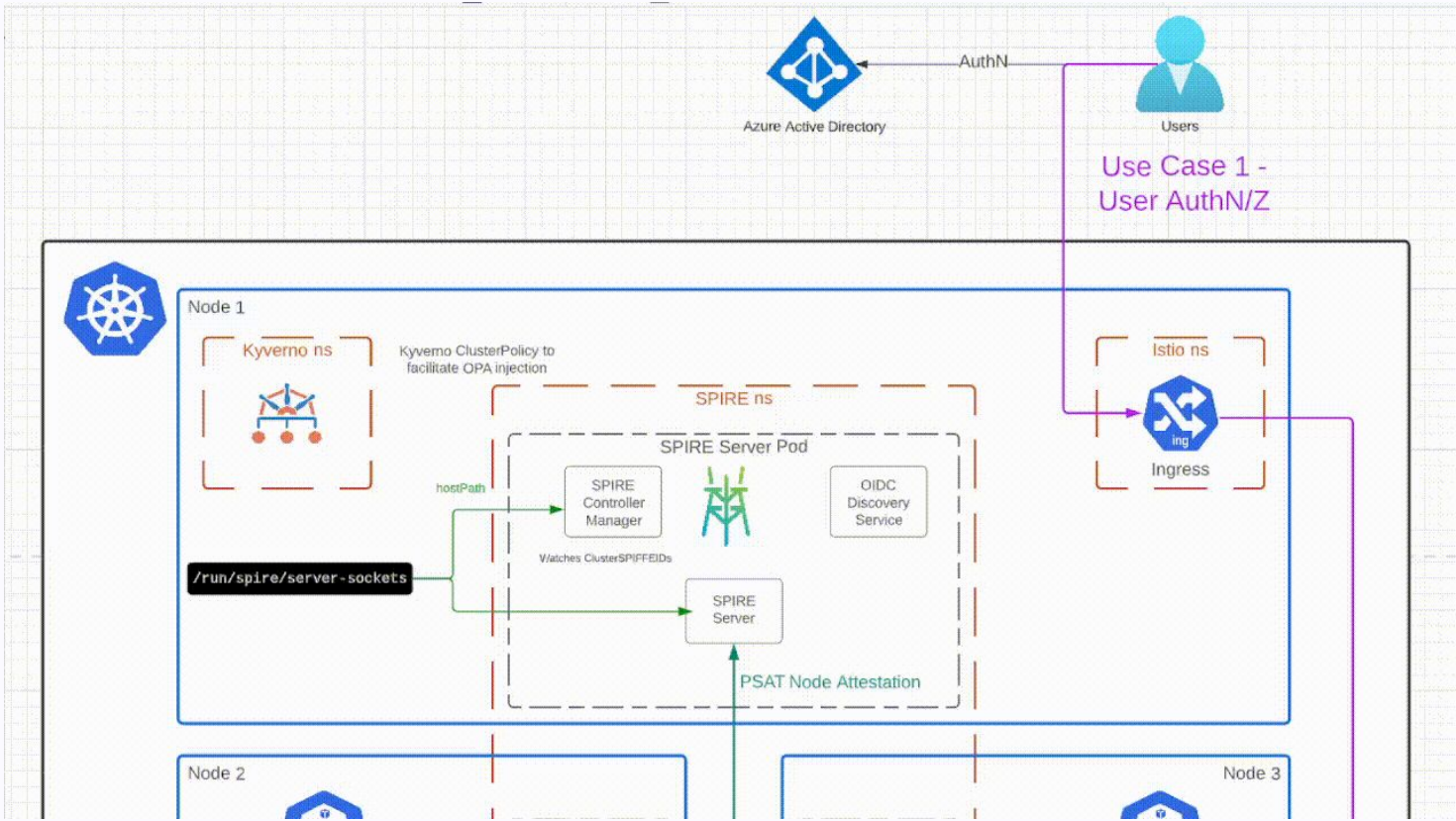
# Prototypes Help us Threat Model!

- **Prototype goals**:
  - Understand **how technologies integrate**
    - Helps us to understand **what can go wrong**
  - **Quick and easy** to spin up:
    - We use a **local kind cluster** with a **minimal set of AWS resources** to show the OIDC provider setup
- In **Production** we would expose the JWKS of our SPIRE server by making an **OIDC discovery service** available
- To avoid this setup, we ship OIDC discovery info to an **S3 bucket**

# Let’s have a quick look

…and make sure it works

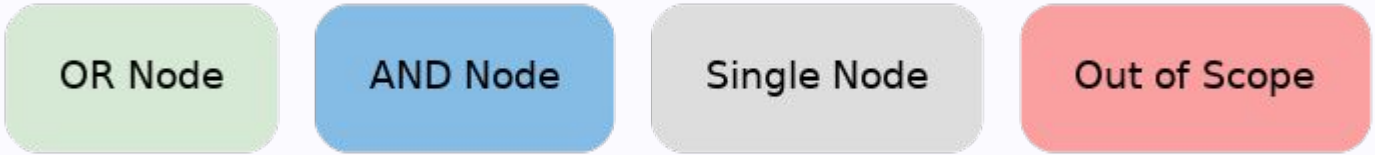
# Draw Data Flow Diagrams!



# **What can go Wrong?**
# Threat Modelling the Detail

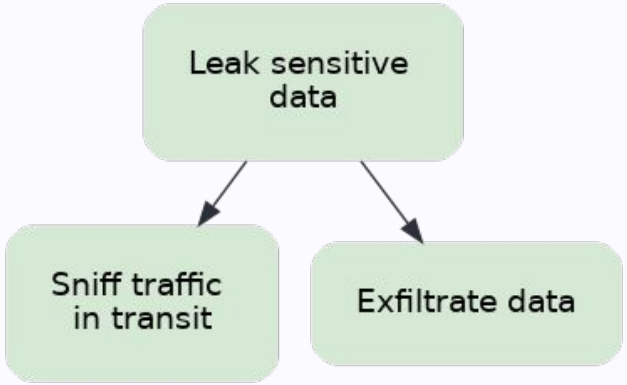
# Draw Attack Trees

- Useful way to brainstorm and document **what can go wrong**
- Kelly Shortridge shows us how to create **attack trees as code**: https://www.deciduous.app/
- From now, we use the following key:



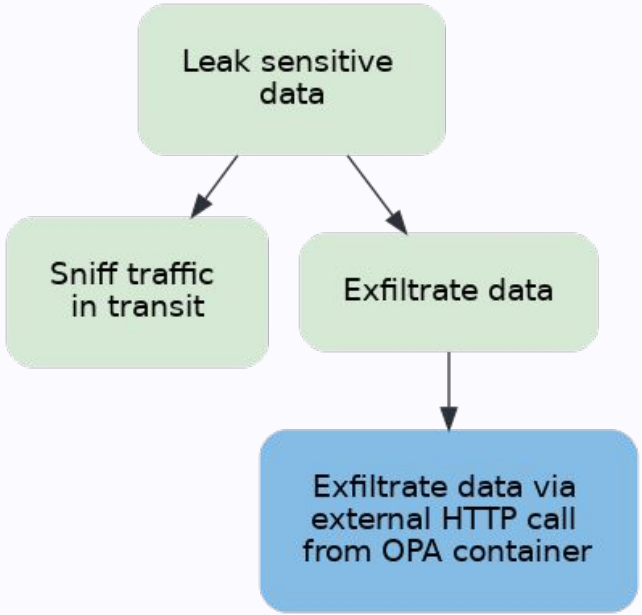
# Building an Example Tree

- Start with a **bad outcome**
  - Could be a compromise of **Confidentiality, Integrity or Availability**
- **Confidentiality example:**

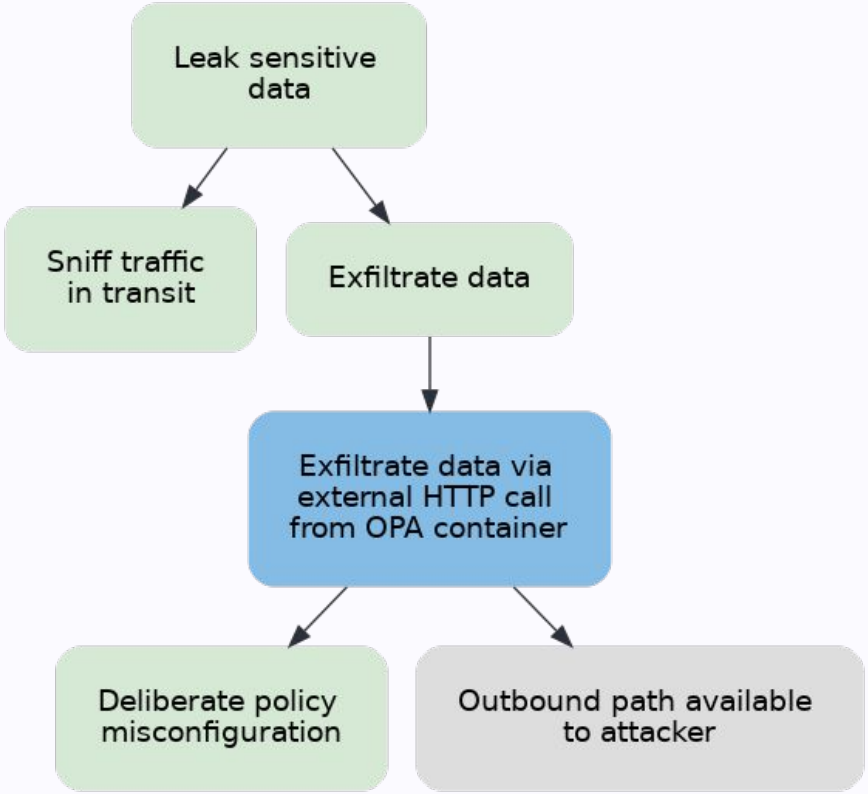


# Adding a level

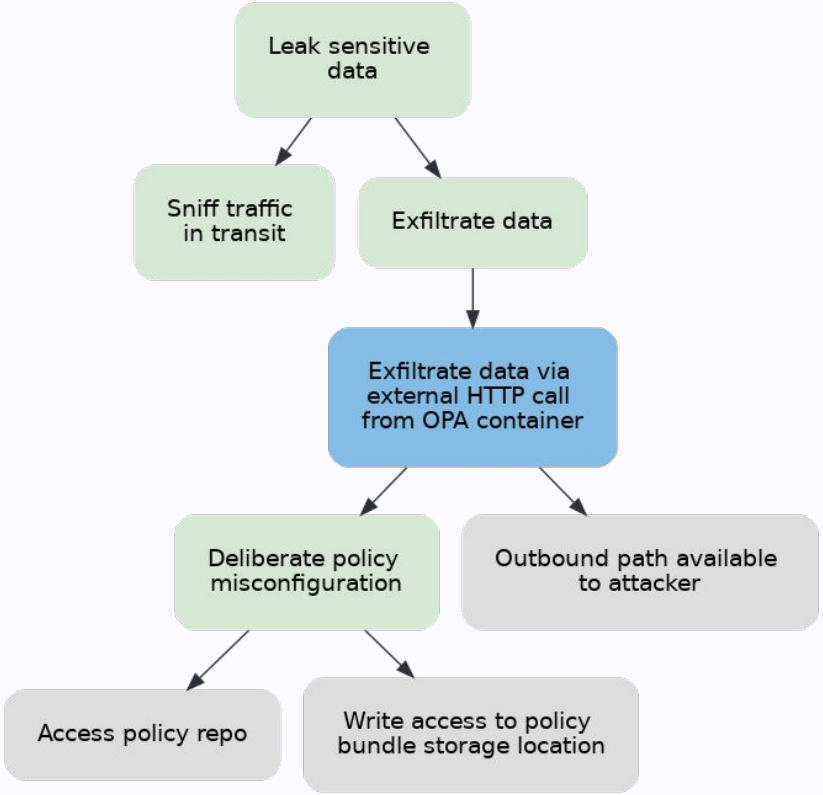
- For each node, brainstorm what an attacker would need to do to achieve that particular goal



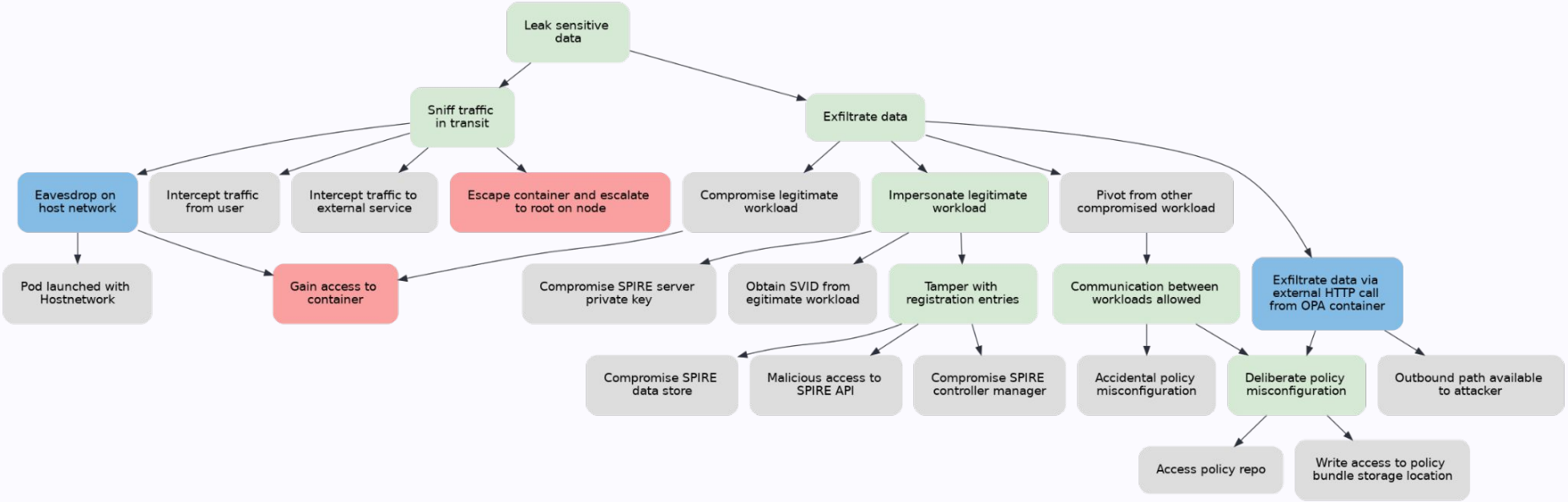
# One More Time!



# One More Time!



# Putting it all Together



# What will we do about it?

## Designing Controls

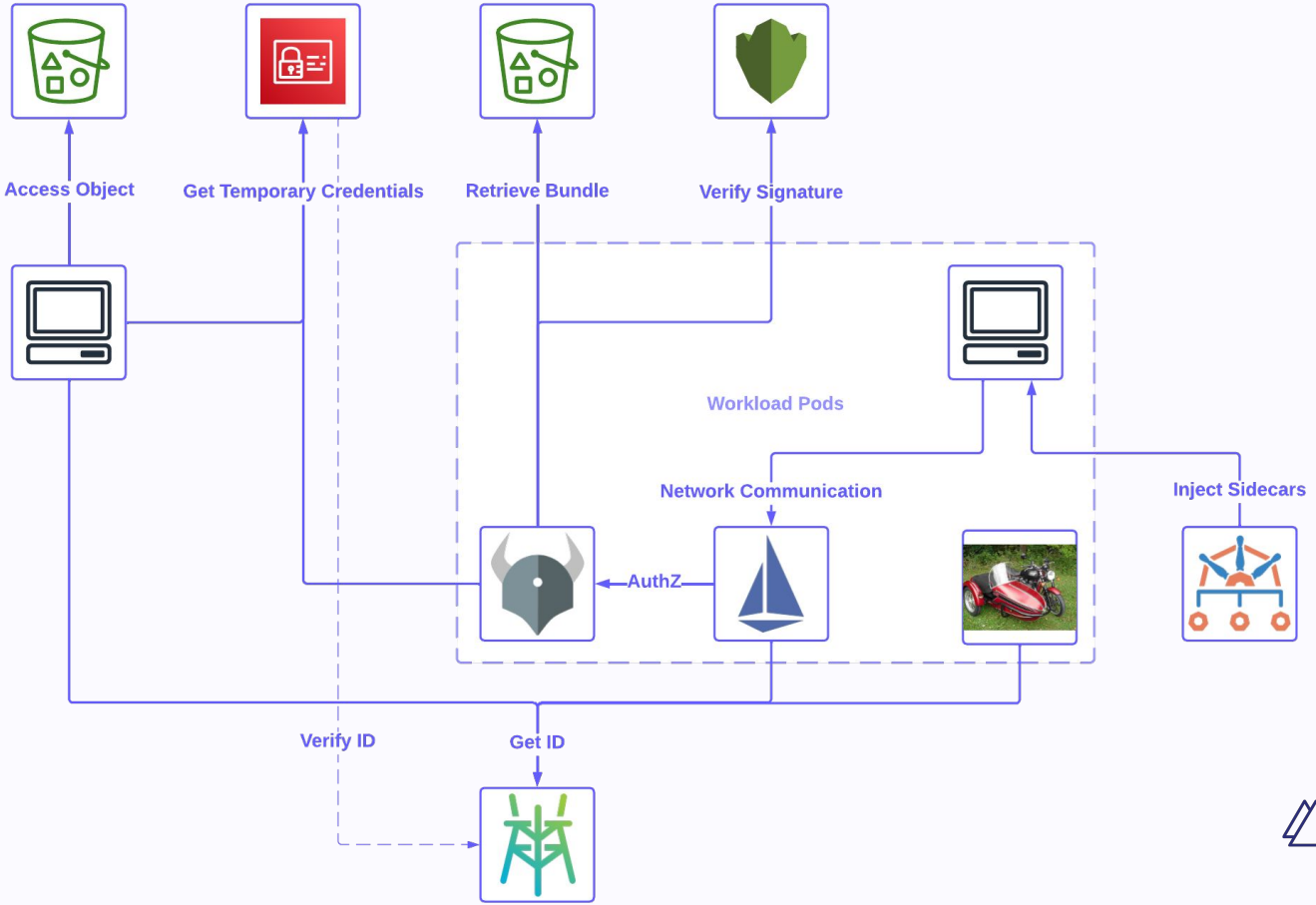
# Devise Mitigating Controls

| Confidentiality Threat | Control |
|---|---|
| T01 - Eavesdrop on Host Network | C01 - Validating Admission Control Policy<br>C02 - Seccomp and Apparmor profiles |
| T02 - Intercept traffic in transit between users/workloads/services | C03 - TLS enforced for ingress and access to cloud services<br>C04 - mTLS between workloads |
| T03 - Exfiltrate data | C05 - Egress control policies |
| T04 - Pivot from compromised workload | C06 - Network Policy<br>C07 - Cryptographically strong workload IDs from SPIRE<br>C08 - Istio External AuthZ with OPA Policy Engine |
| T05 - Accidental policy misconfiguration | C09 - Policy linting<br>C10 - Policy unit tests |
| T06 - Access policy repo | C11 - Least Privilege repo access control<br>C12 - Protected branches/enforced PRs with mandated review |
| T07 - Overwrite policy bundle | C13 - Cloud Provider RBAC least privilege & audit<br>C14 - Policy bundle signing and verification |
| etc. | |

# Building Custom Controls

- Some controls will need **further architectural work** to be precisely defined
  - e.g. '**compromise SPIRE data store**' threat
    - requires data storage **design decision**
    - **Lower-level attack tree** can then be created
- We have looked at one **example attack path**:
  - **malicious internal actor** exploits **misconfigured IAM** policy to **overwrite policy bundle**
- We currently have a mitigating control for this:
  - C14 - **Policy bundle signing and verification**
- However, we still need to design the control implementation...

# Iterating the Architecture



# OPA Custom Bundle Signing

- What's in the **signature**?
  - https://www.openpolicyagent.org/docs/latest/management-bundles/#signing
- Where do you even **start**?
  - **OPA contrib repo**
  - https://github.com/open-policy-agent/contrib/tree/main/custom_bundle_signing

# Kick the tyres!

...but not in production!

https://github.com/controlplaneio/threat-modelling-zero-trust-talk

# Summary

- Even well designed systems should be threat modelled:
  - Threat landscapes change
  - Technologies evolve
- Adopting a Zero Trust mindset is becoming increasingly crucial
- Cryptographically strong workload identity is key
  - SPIRE
- Existing integrations can be used
  - e.g. Istio External AuthZ
- Custom controls can be designed if risk decisions require them
  - e.g. OPA custom bundle signing

# Did we do a good job?

You tell us!

Thank you for listening!!