Everything-as-Code

Abhay Bhargav

abhaybhargav

Yours Truly

- Founder @ we45
- Founder @ AppSecEngineer
- AppSec Automation Junkie
- Trainer/Speaker at DEF CON, BlackHat, OWASP Events, etc world-wide
- Co-author of Secure Java For Web Application Development
- Author of PCI Compliance: A Definitive Guide

APPSEC engineer

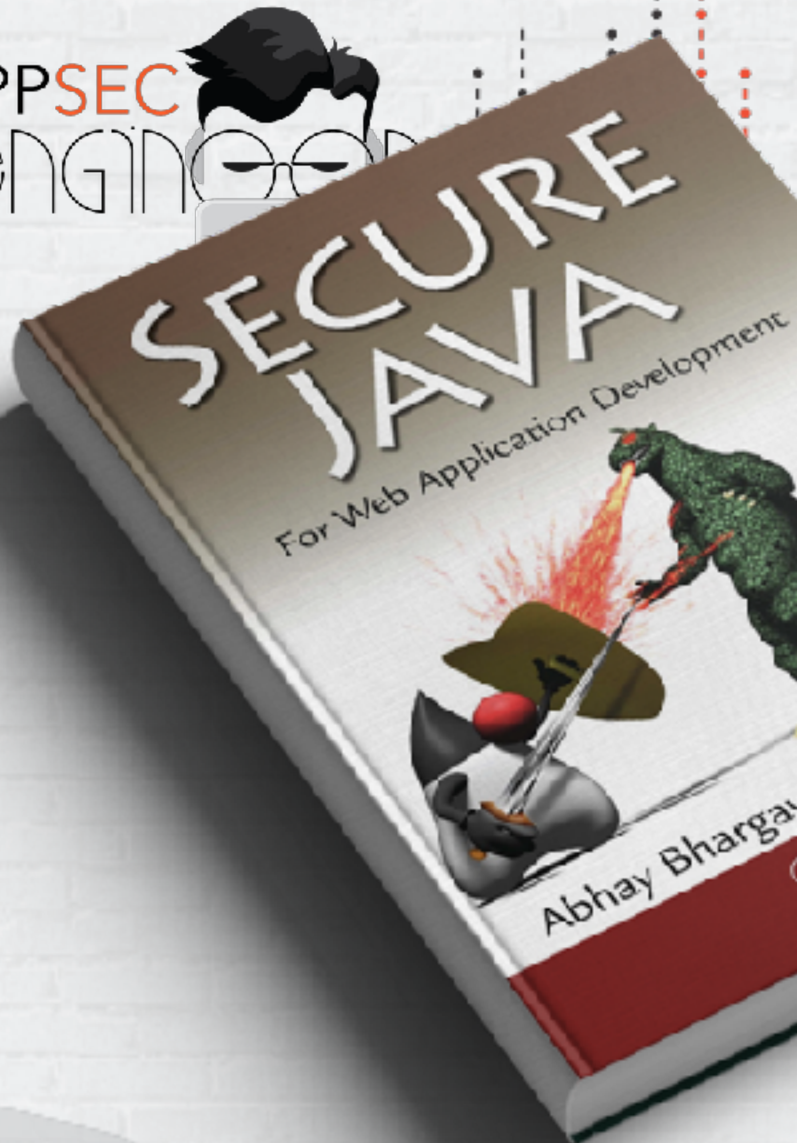
THREAT PLAYBOOK

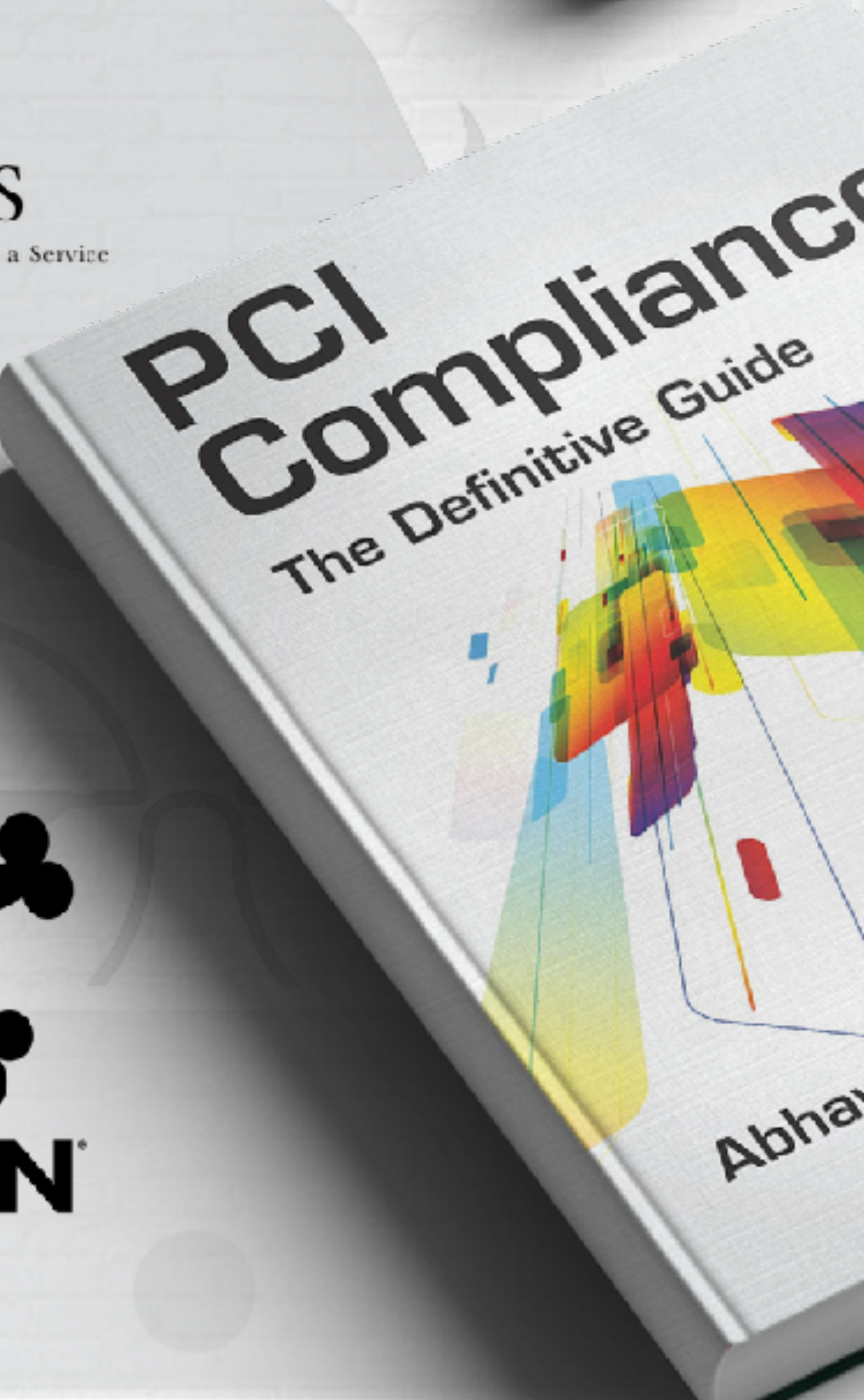
DVFaaS
Damn Vulnerable Functions as a Service

black hat

OWASP

DEFCON





abhaybhargav

My talk...

abhaybhargav

Everything-as-Abstracted, Configurable, Parameterizable Code

~~Everything-as-Code~~

abhaybhargav

Agenda

- Why is the “as-code” movement so important?
- DevSecOps => Possible Future of Security
- As-Code across the stack
- Demos and Examples

Why?

135:1

Developers

Software Security Pros

abhaybhargav

Source: BSIMM-12

Git and its role in Modern DevOps

Collaborate

Version Control

Centralized

Distributed

Branching

Pull/Merge Request

Infrastructure as Code

Cloud

- Plethora of Deployment and Database options
- Elastic Scale
- API-driven Orchestration across the cloud

abhaybhargav

Monoliths

User Management

Customer Master

Customer Communication

User Communication

Customer Deals

Sales Order Processing

Inventory Management

Delivery Management

Tax Filing

MicroServices

User Management

Customer Master

User Communication

Sales orders

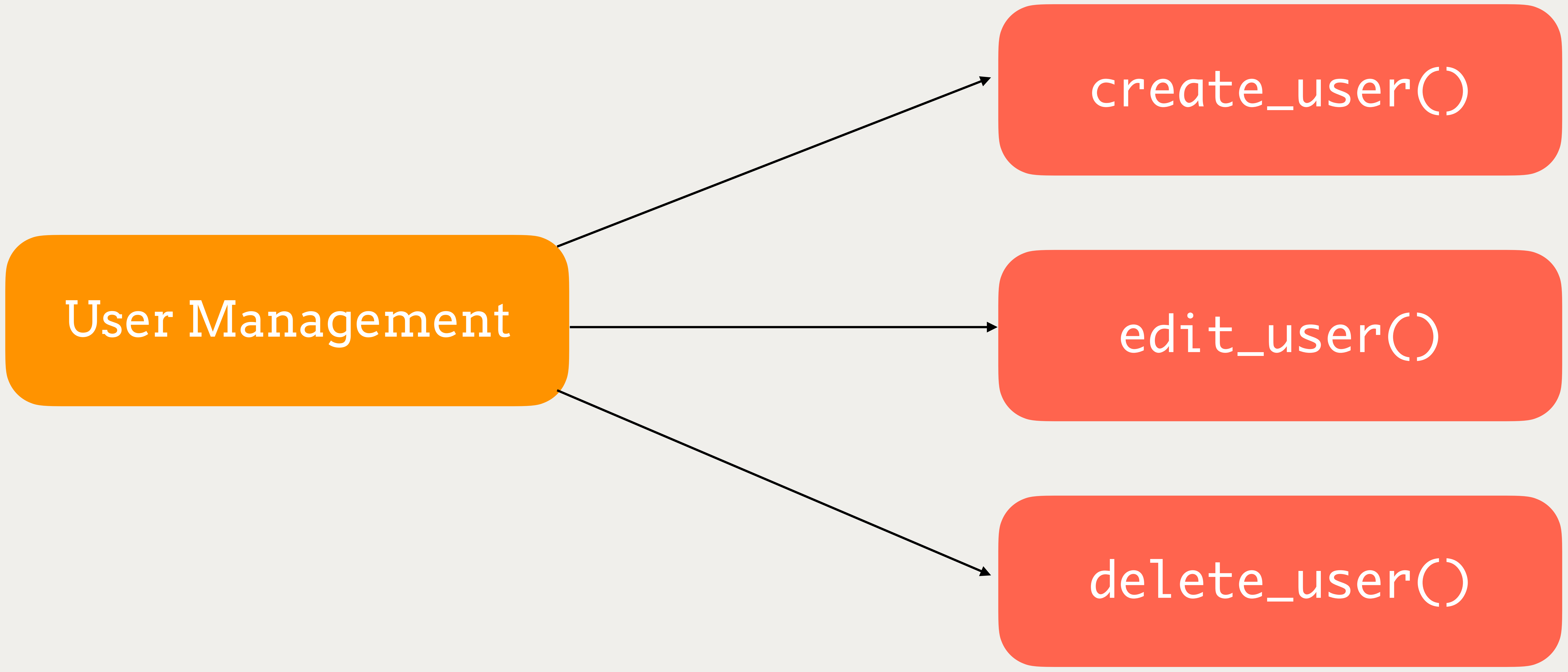
Inventory Management

Delivery Service

Taxation Service

Customer Comms

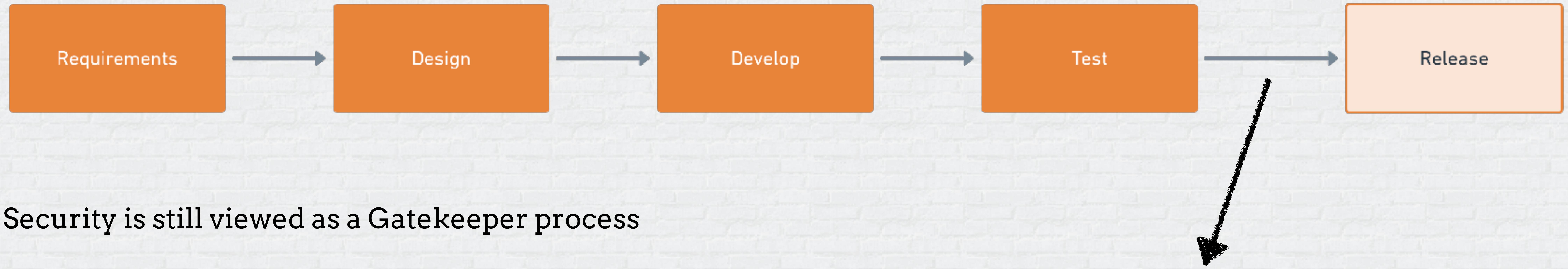
Functions as a Service



Trends on the Application Delivery Front

The Bottleneck

Security is very waterfall



Security is still viewed as a Gatekeeper process

Gatekeeper processes come up with very binary options

Security intervenes here

abhaybhargav

In Short….

Application Delivery

Application Security

abhaybhargav

What we need

Fix Bugs Early
and often

Embedded within multiple stages
of the SDLC

Effective Automation
of Security Tasks

Proactive Controls
to Eliminate Vulnerabilities
at scale

Dev-First Workflows!

abhaybhargav

Workflows that support iterative and continuous delivery of apps

^

~~Dev-First Workflows!~~

abhaybhargav

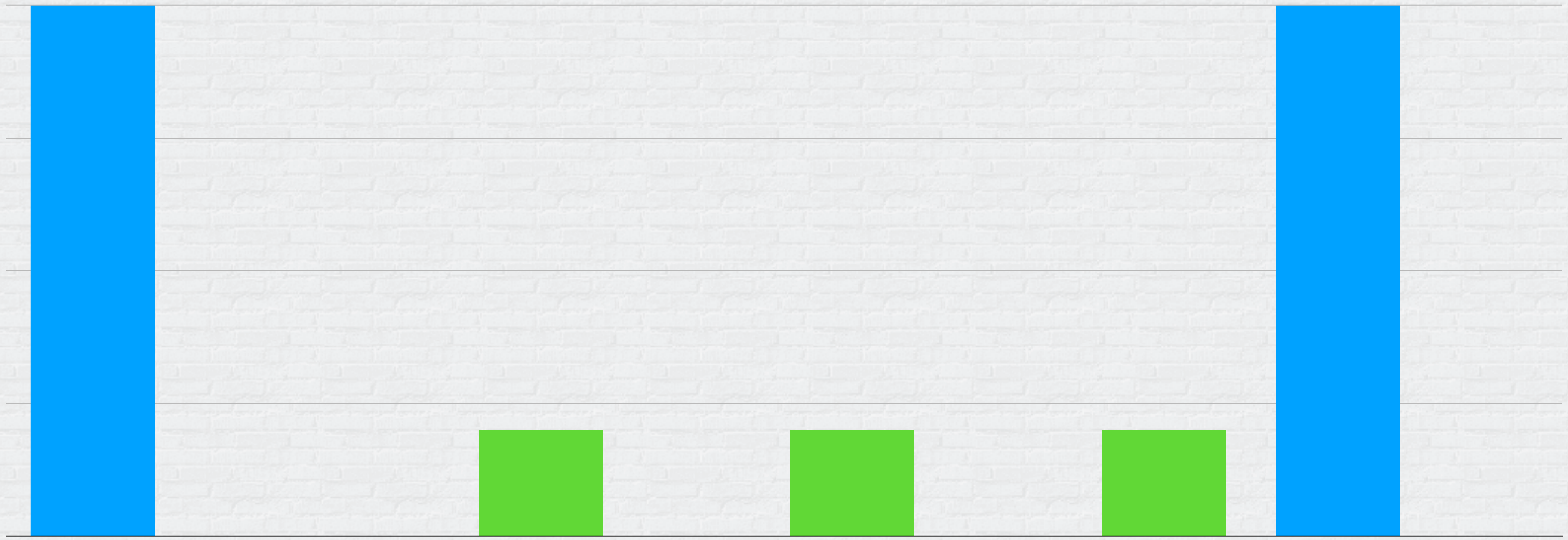
This means…

- Dev has consumed Ops (Infrastructure-as-Code, Continuous Integration, Continuous Deployment)
- Dev has consumed QA (Test Automation)
- Dev is halfway through consuming security (Security-as-code)
- Dev is coming for policy, compliance, etc next

Why is this good?

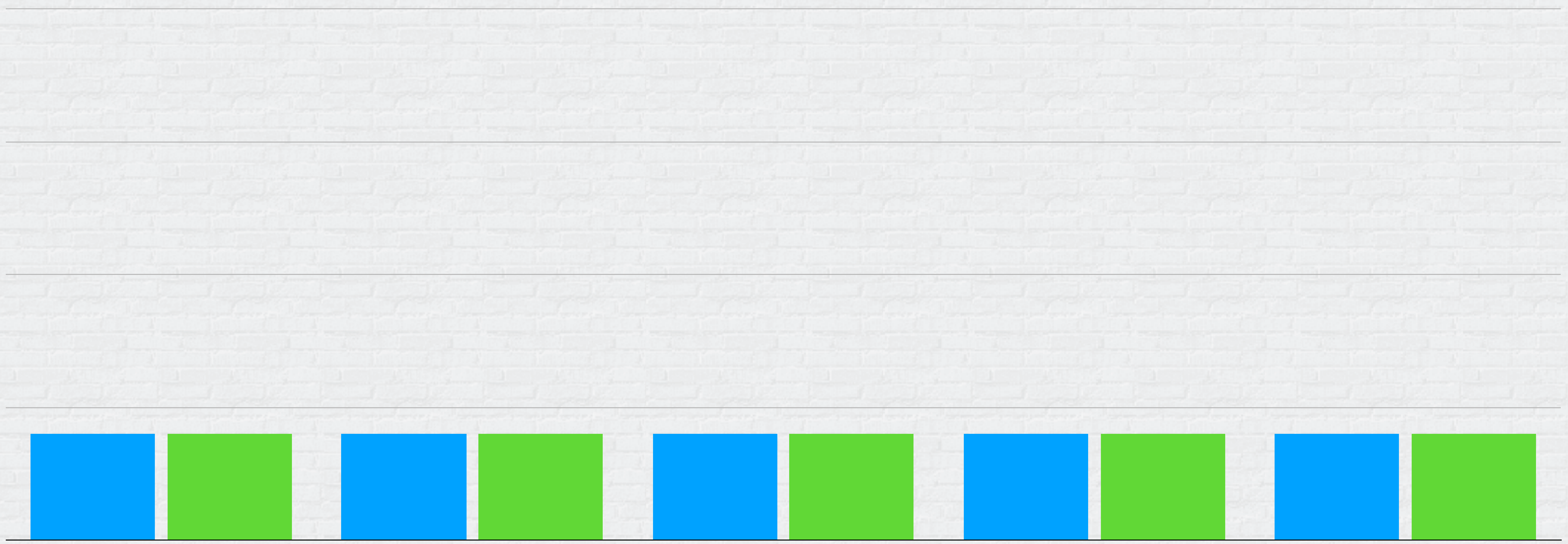
- ⬆️ Automation!
- ⬇️ Human Intervention
- ⏭️ Faster delivery of features
- 🌥️ Highly Scalable, Immutable Environments ❎

Instead of this...



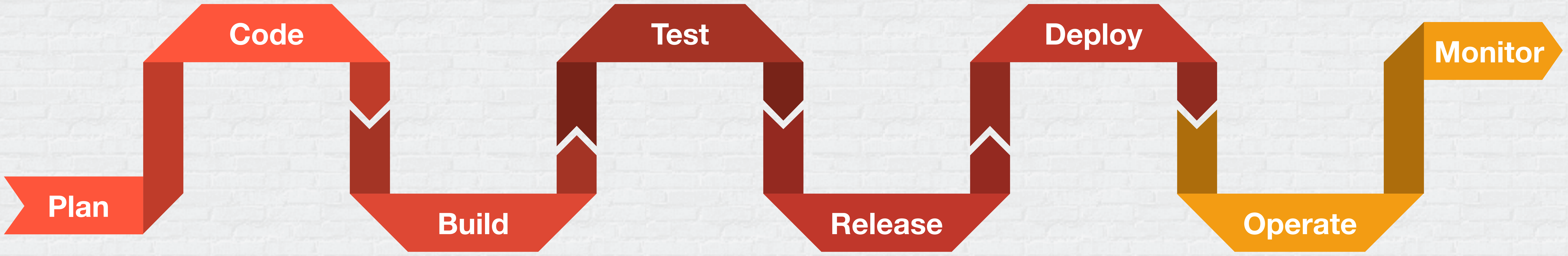
abhaybhargav

To this...



abhaybhargav

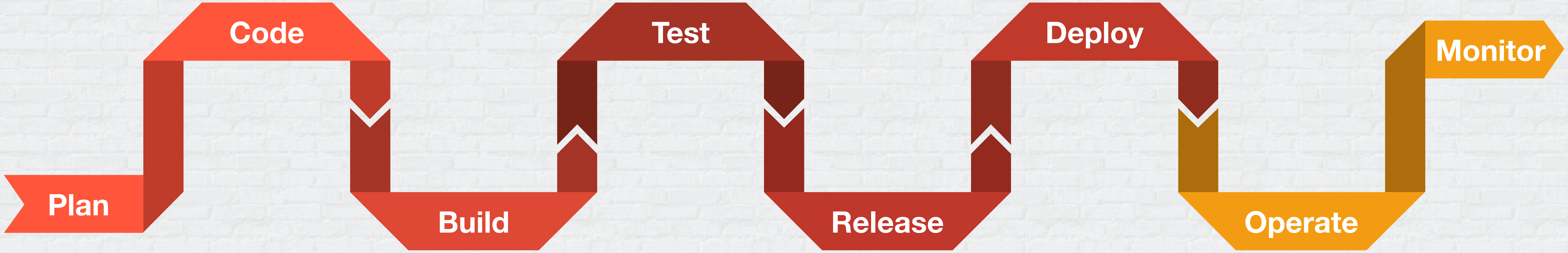
DevSecOps

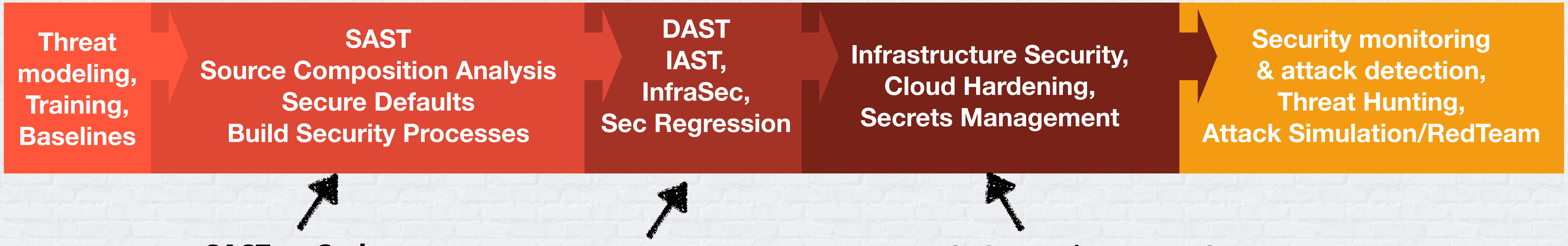


Plan	Code / Build	Test	Release / Deploy	Operate / Monitor
Threat modeling, Training, Baselines	SAST Source Composition Analysis Secure Defaults Build Security Processes	DAST IAST, InfraSec, Sec Regression	Infrastructure Security, Cloud Hardening, Secrets Management	Security monitoring & attack detection, Threat Hunting, Attack Simulation/RedTeam

abhaybhargav

DevSecOps





Threat modeling, Training, Baselines	SAST Source Composition Analysis Secure Defaults Build Security Processes	DAST IAST, InfraSec, Sec Regression	Infrastructure Security, Cloud Hardening, Secrets Management	Security monitoring & attack detection, Threat Hunting, Attack Simulation/RedTeam

SAST as Code

DAST/Regression as Code

Decoupled Security Controls /Policy-As-Code

abhaybhargav

Decoupled Security Controls and Policy as Code

abhaybhargav

Broken Object Level AuthZ

BOLA

Broken AuthN

BA

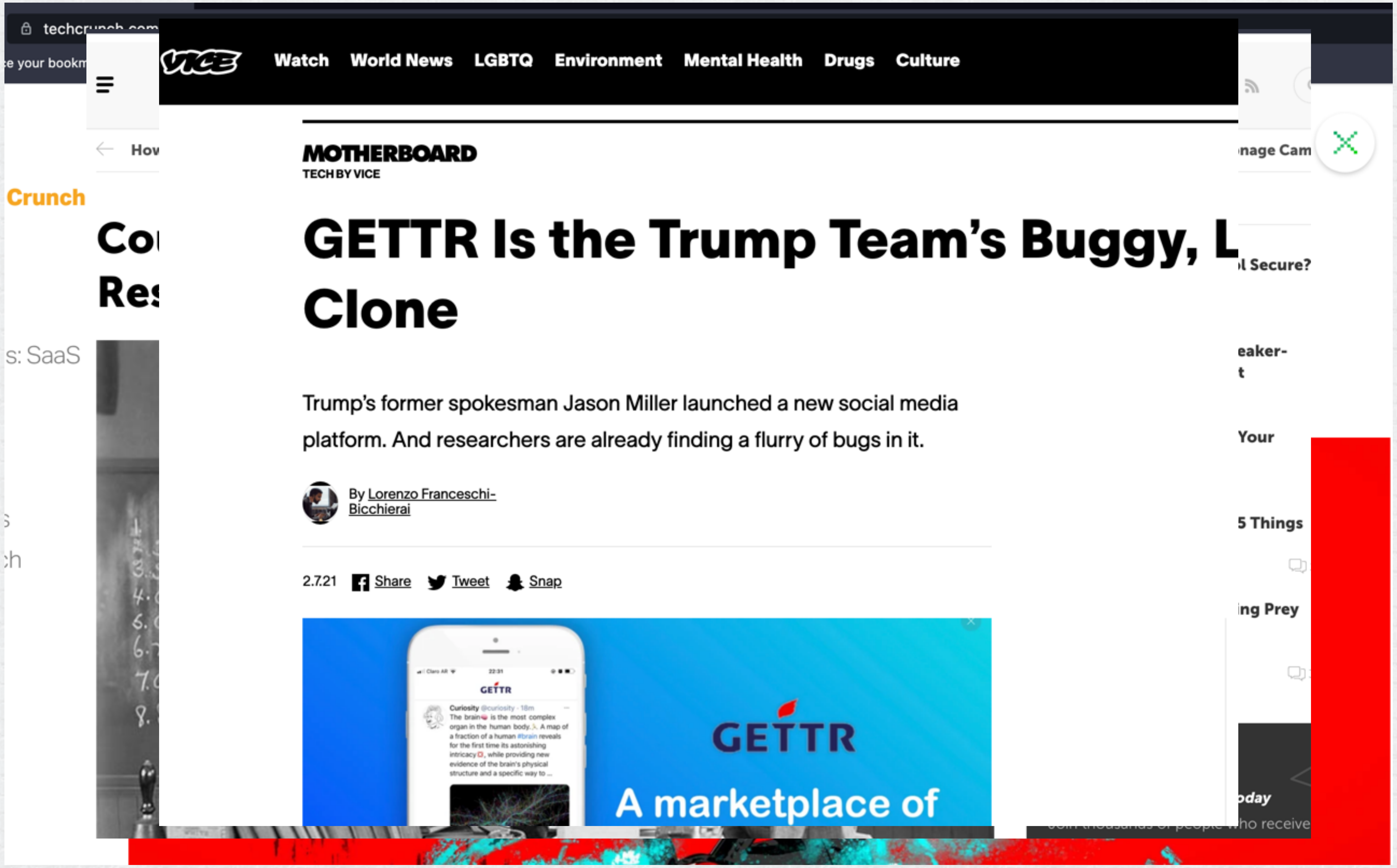
Broken Function Level AuthZ

BFLA

Excessive Data Exposure

EDE

From recent events...



Watch World News LGBTQ Environment Mental Health Drugs Culture

MOTHERBOARD
TECH BY VICE

GETTR Is the Trump Team's Buggy, L
Clone

Trump's former spokesman Jason Miller launched a new social media platform. And researchers are already finding a flurry of bugs in it.

By Lorenzo Franceschi-Bicchierai

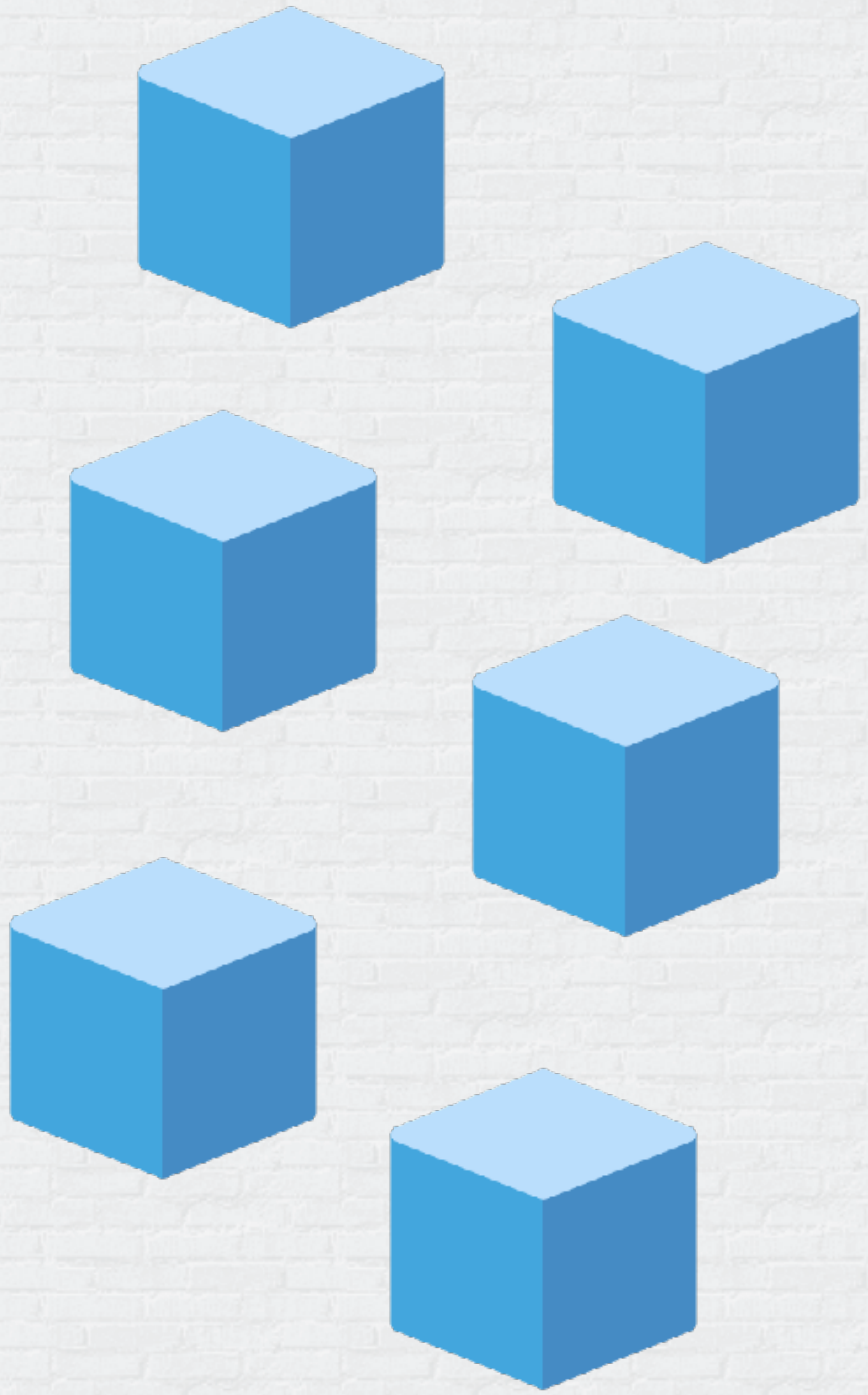
2.7.21 Share Tweet Snap

abhaybhargav

Trends on the Application Delivery Front

NORDICAPIS.COM

Imagine…



- Logging
- Authentication
- Object Access Control
- Input Validation
- JWT Authorization
- Your Service Business Logic

What if…

- APIs and services were NOT security aware
- But security validation and checks were handed off to a more specialised set of controls
- Leverage “as-code” platforms to be able to compose and change them as required, vs changing all services

Need and Motivation

- APIs and Web Services are typically part of a larger set of service offerings
- With rapid-release requirements, these services are constantly changing.
- New services are constantly being included, removed and modified

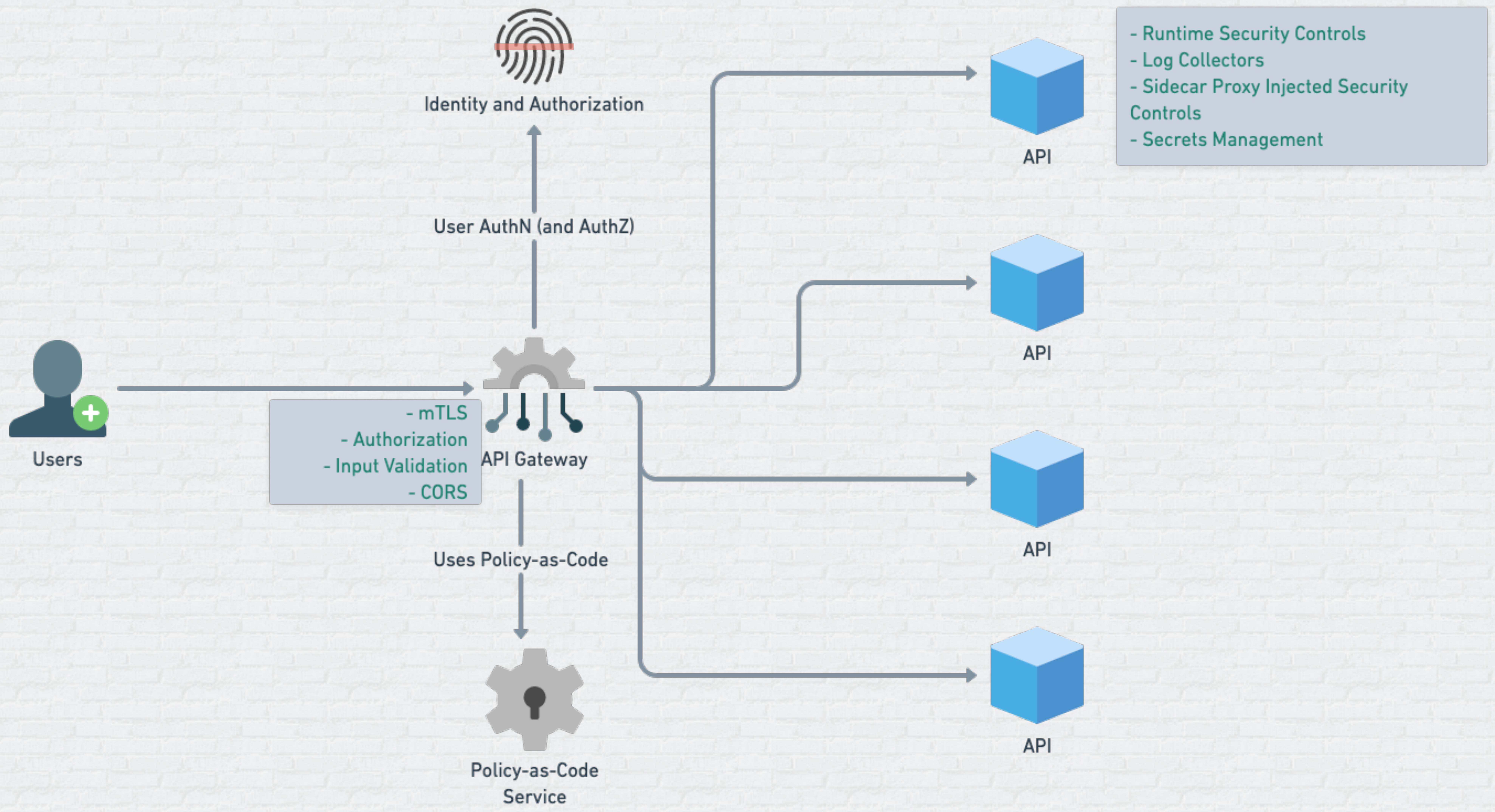
Need and Motivation – 2

- Decentralized controls are applied “outside” the application
- The idea is to NOT hardcode security rules in app that have rapidly evolving and changing requirements
- Leveraging eBPF, Policy-as-Code and API Gateway Security Features to drive security controls

Typical Use-Cases

- Syscall Profiling, Seccomp, AppArmor and eBPF for Runtime Security enforcement
- Authorization, CORS, Rate-Limiting, mTLS and others on the API Gateway
- Log Collection and aggregation of services from Cloud-Native environments
- Input Validation, Access Control with Policy-as-Code Frameworks

Security Model – An Example



- mTLS
- Authorization
- Input Validation
- CORS

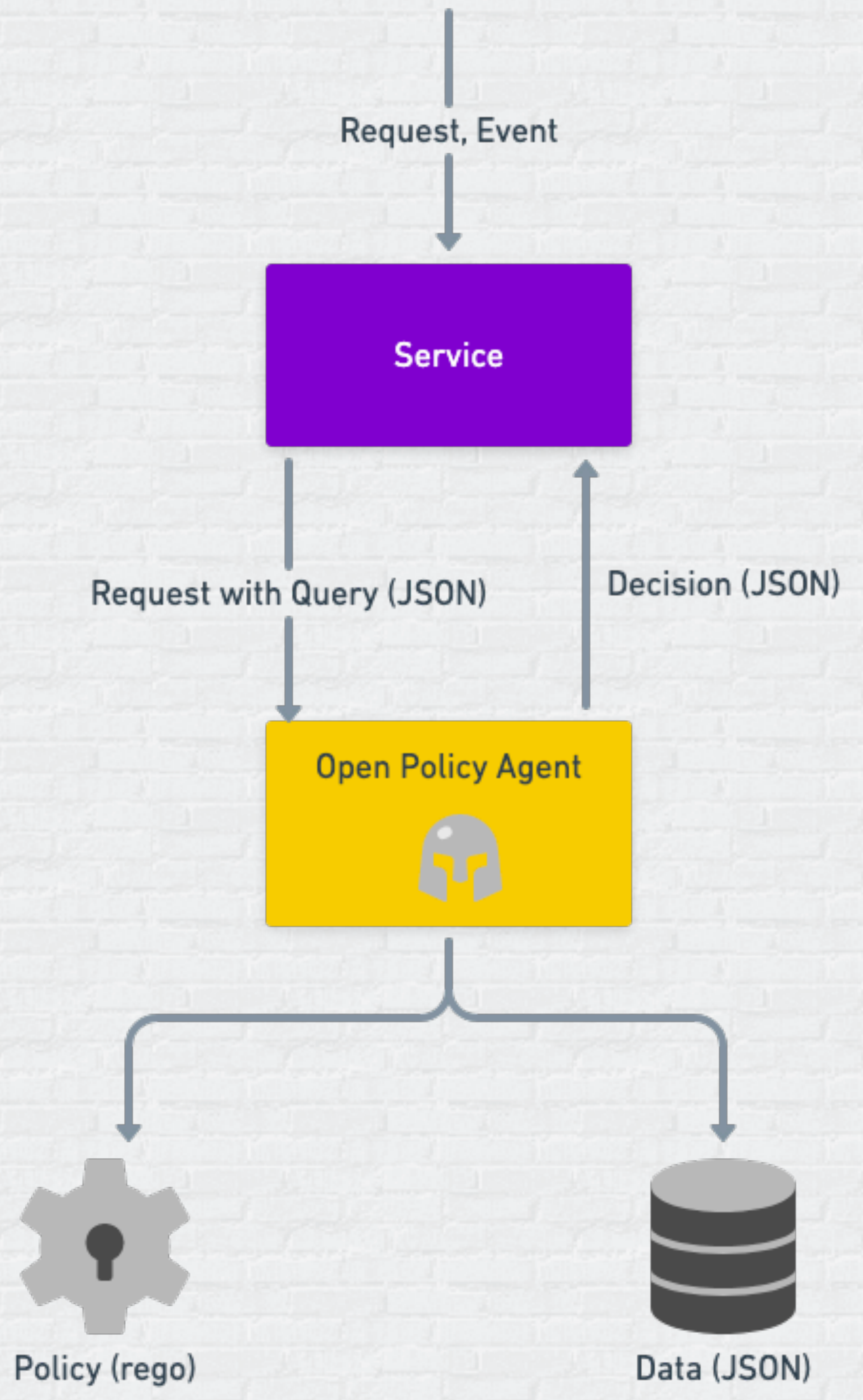
- Runtime Security Controls
- Log Collectors
- Sidecar Proxy Injected Security Controls
- Secrets Management

abhaybhargav

Open-Policy-Agent

- Policy Management Framework for “any” environment
- Allows you to define policies that can be enforced based on generic json input and output parameters
- Uses a DSL (domain specific language) called “rego” that is used to define policies

Open Policy Agent – Operation

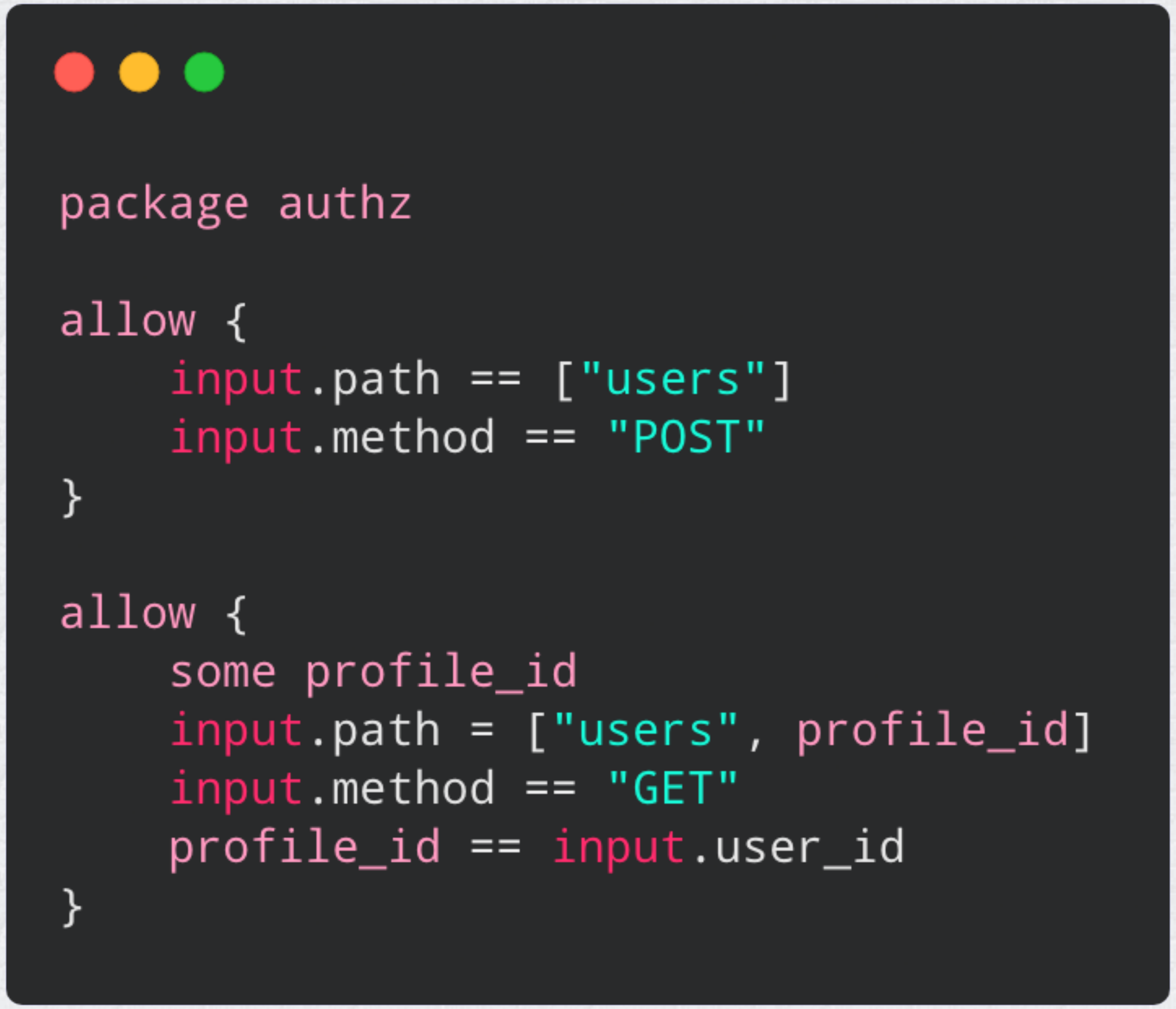


abhaybhargav

OPA Use-Cases

- Kubernetes Policy Management
- API AuthZ and Policy Management
- OS Policy Management - SSH and Access Control
- Kafka Topic Authorization
- Many more…

OPA – API AuthZ



```
package authz

allow {
    input.path == ["users"]
    input.method == "POST"
}

allow {
    some profile_id
    input.path = ["users", profile_id]
    input.method == "GET"
    profile_id == input.user_id
}
```

abhaybhargav

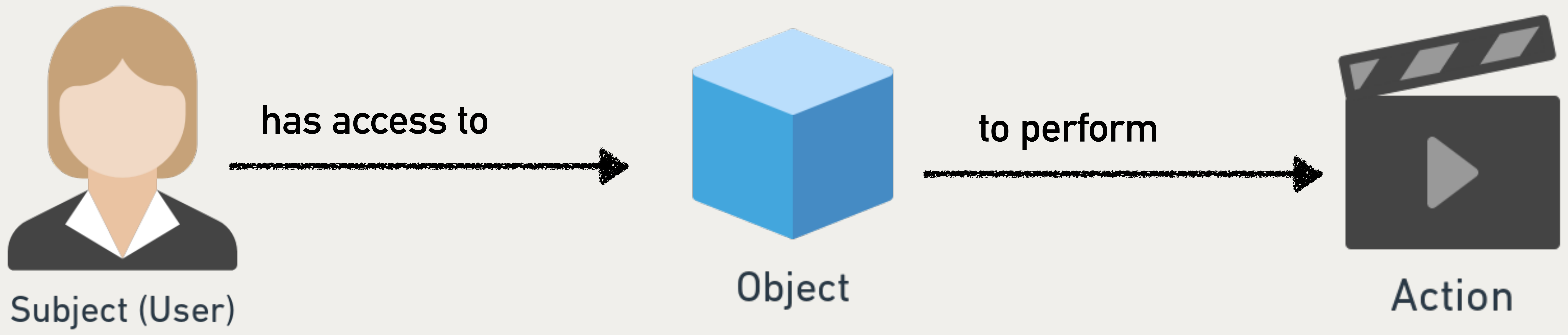
What about Access Control?

abhaybhargav

Let's look at most AuthZ flaws

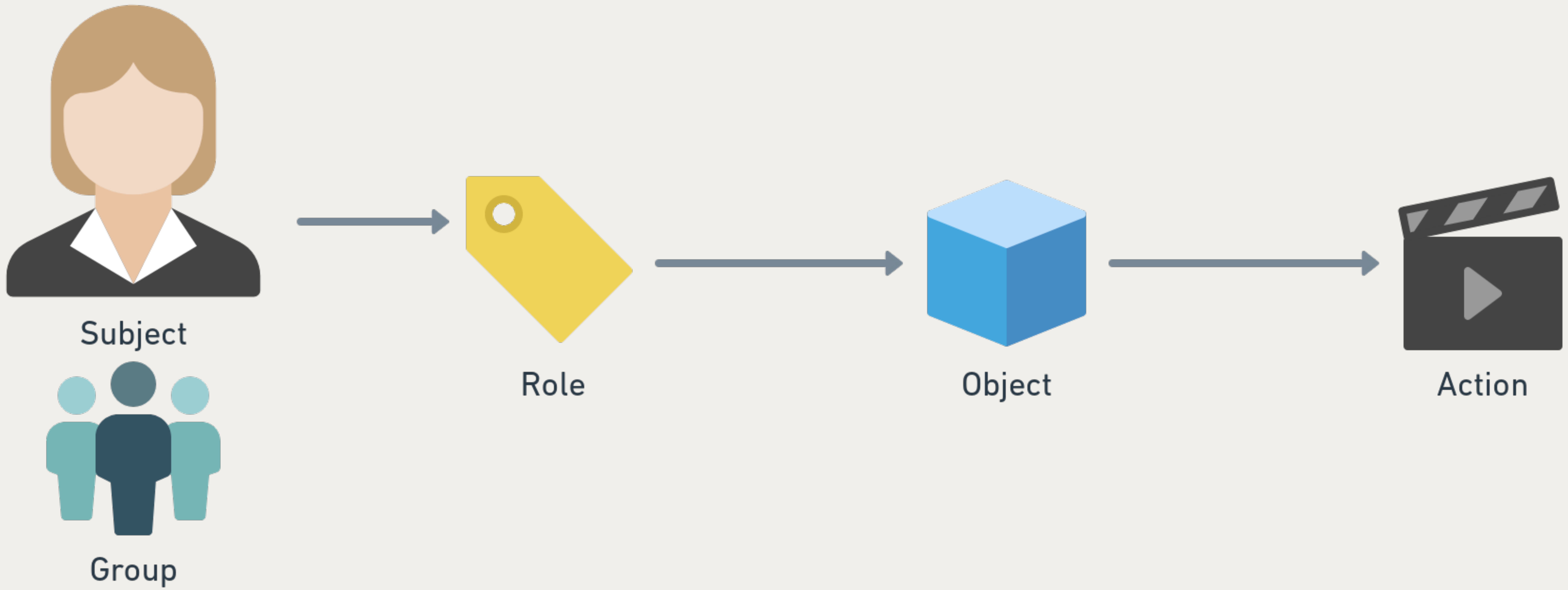
- Inconsistent implementation of Object Level Authorization
- Access Control code strewn across multiple services
- Lack of standardization and expressive capability for AuthZ frameworks
- Heavily design dependent - which gets complex at scale

ACL



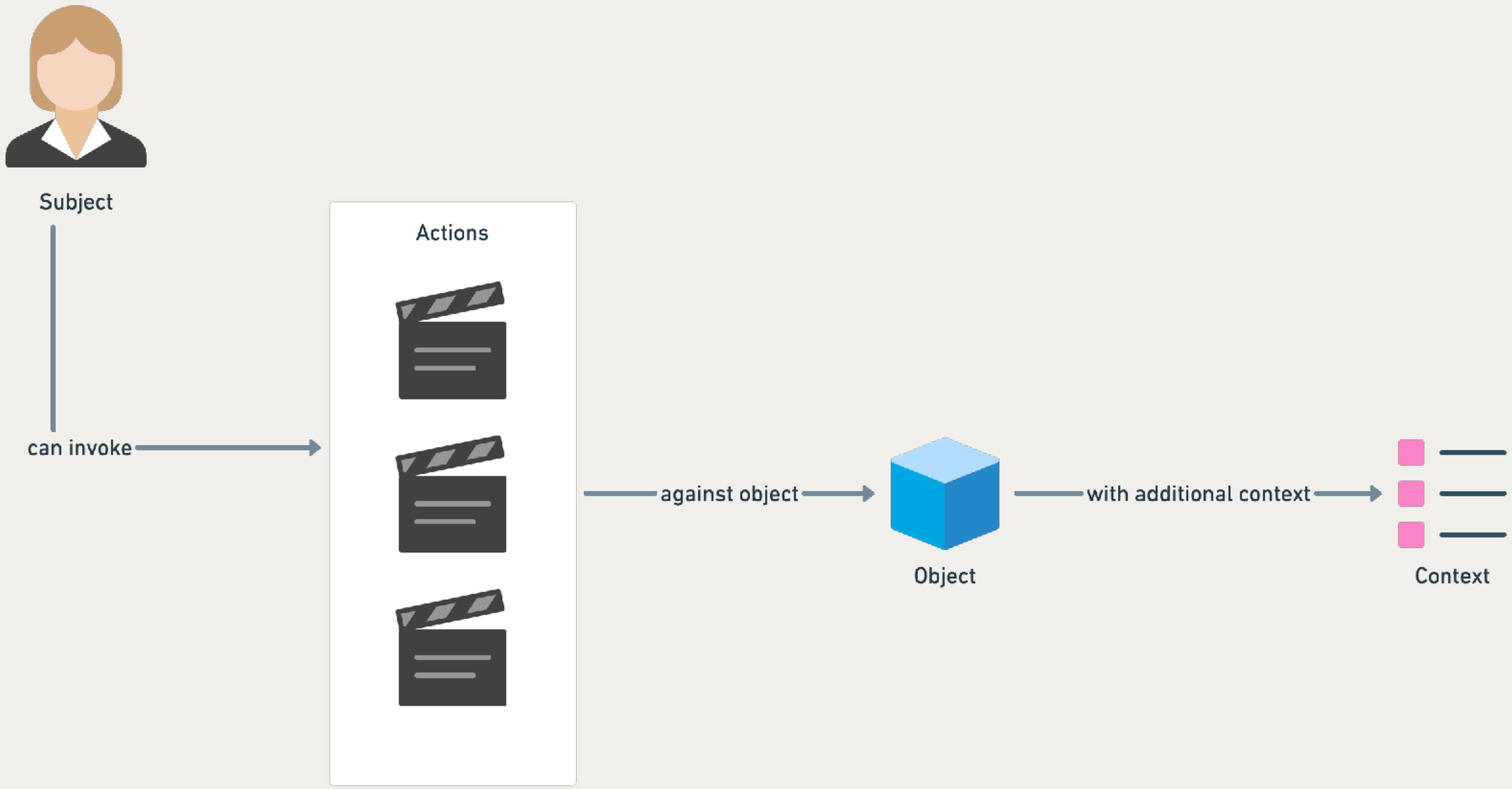
abhaybhargav

RBAC – Role Based Access Control



abhaybhargav

ABAC – Attribute Based Access Control

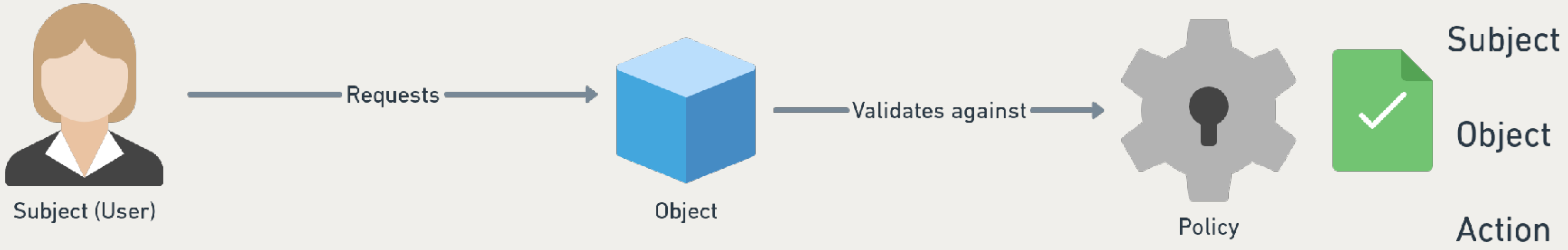


abhaybhargav

PERM

Policy, Effect, Request, Matchers

What is PERM?

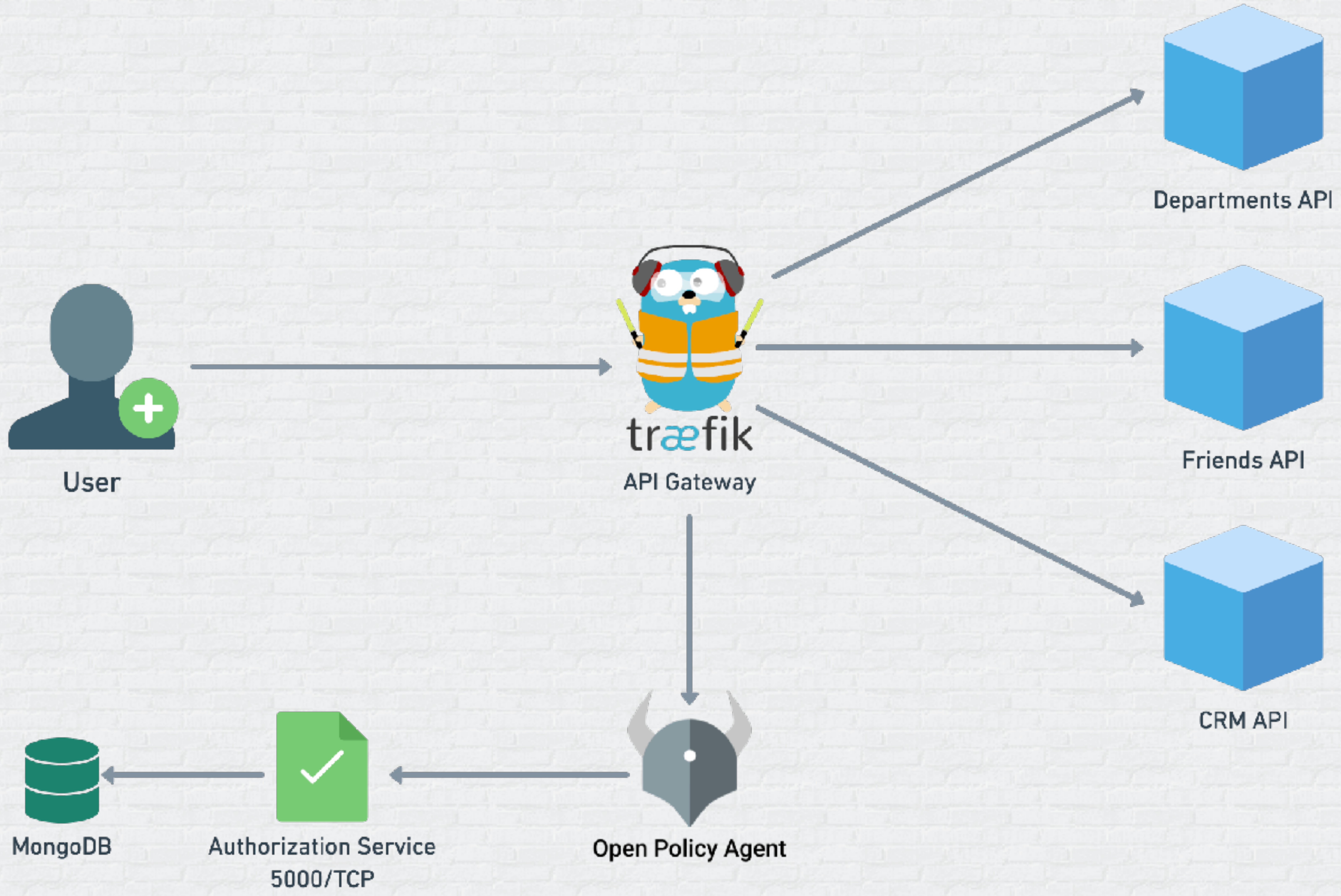


Request Attributes must MATCH Policy Attributes

Casbin

- Authorization libraries and framework for multiple Access Control models
- Uses a DSL based on the PERM model to be able to define access control functionality that can integrate with access control data
- All you need to do is pass the library with a Subject-Object-Action definition and Casbin’s APIs handle the validation

Lab: OPA, Traefik and Decentralized security Controls



abhaybhargav

Other applications of Policy-as-Code

- Managing Kubernetes Clusters
- Threat Hunting with Audit Logs
- Cloud Admission Controls

SAST as Code

abhaybhargav

SAST Test Approaches

- Good ol' Regular Expressions
- Abstract Syntax Trees
- Semantic Grep or QL

Regular Expressions

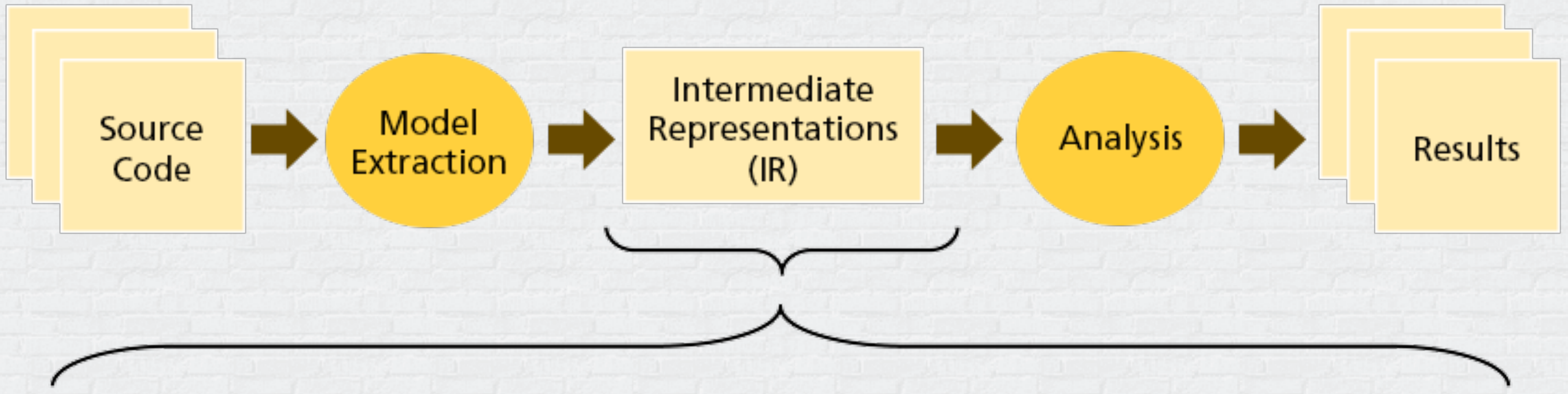
- Regular Expressions are useful in identifying patterns.
- However, they can be inaccurate, because they don't really look understand the code in context
- Heavily dependent on the quality of Regexes written as rules

Errors

```
Code Comments:

  # Don’t use this!! jwt.decode(something, secret,
  verify=False)
```

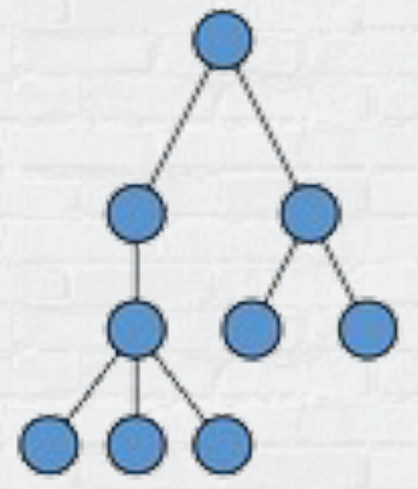
SAST with AST



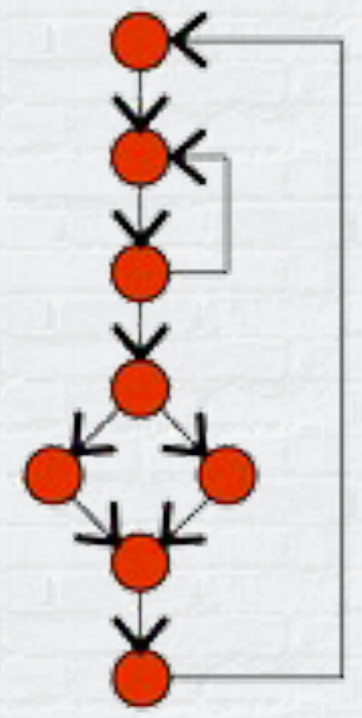
Names Database/Symbol Table

Name	*Kind*	*Location*
copy_item	function	item.c:25
item_cache	variable	item.c:10
color	parameter	pallette.c:23
header.h	file	shapes.c

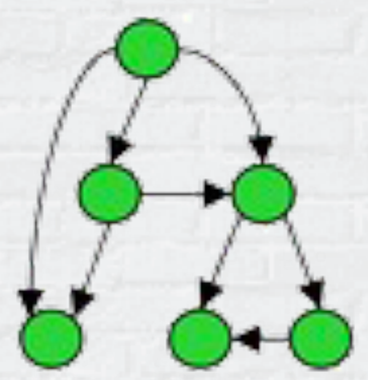
Abstract Syntax Tree (AST)



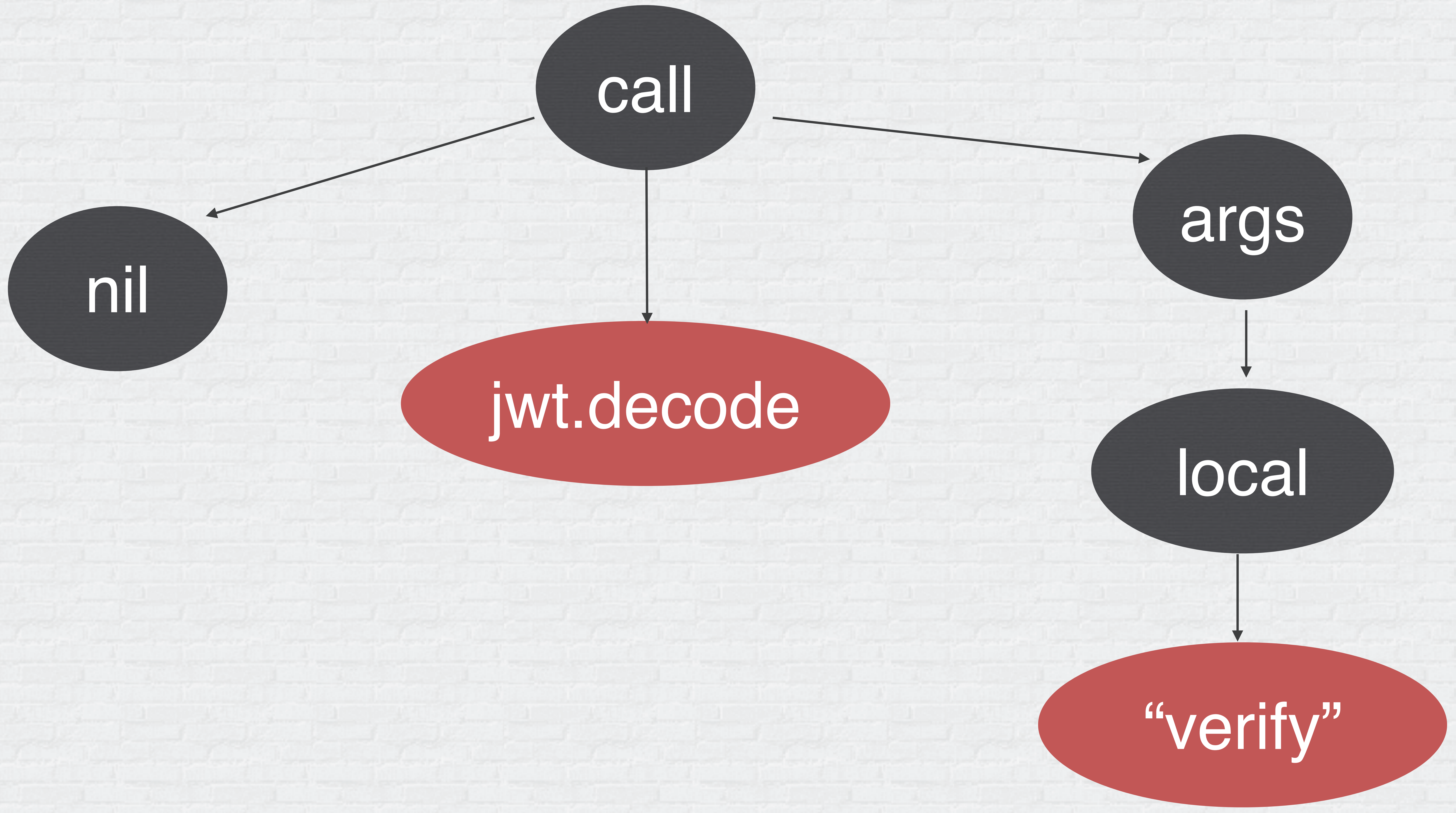
Control Flow Graph (CFG)



Call Graph



AST example with Python



SAST – AST Benefits for DevSecOps

- New rules can be written into SAST or Linter/Code Quality tool
- Very fast, especially if using as a Linter/Code Quality tool, rather than a full-featured SAST Tool
- Can be embedded into the IDE for immediate feedback loops to the developer

Good Rules for SAST

- Every check should do ONE THING ONLY!
- False Positives abound when complexity increases
- Extending SAST with Custom Checks is a good idea
  - IF you know what you are doing
- Getting Engineering teams to extend SAST should be the ultimate objective

abhaybhargav

Custom SAST Rules

- Custom SAST rules become necessary as you are scaling up in SAST Maturity
- Custom SAST rules help identify specific cases that make sense to your applications, in terms of security
- Increases Depth of your overall SAST Process
- Leveraging AST is better for SAST, as it makes it more accurate

abhaybhargav

Lab – Custom SAST: Bandit Python

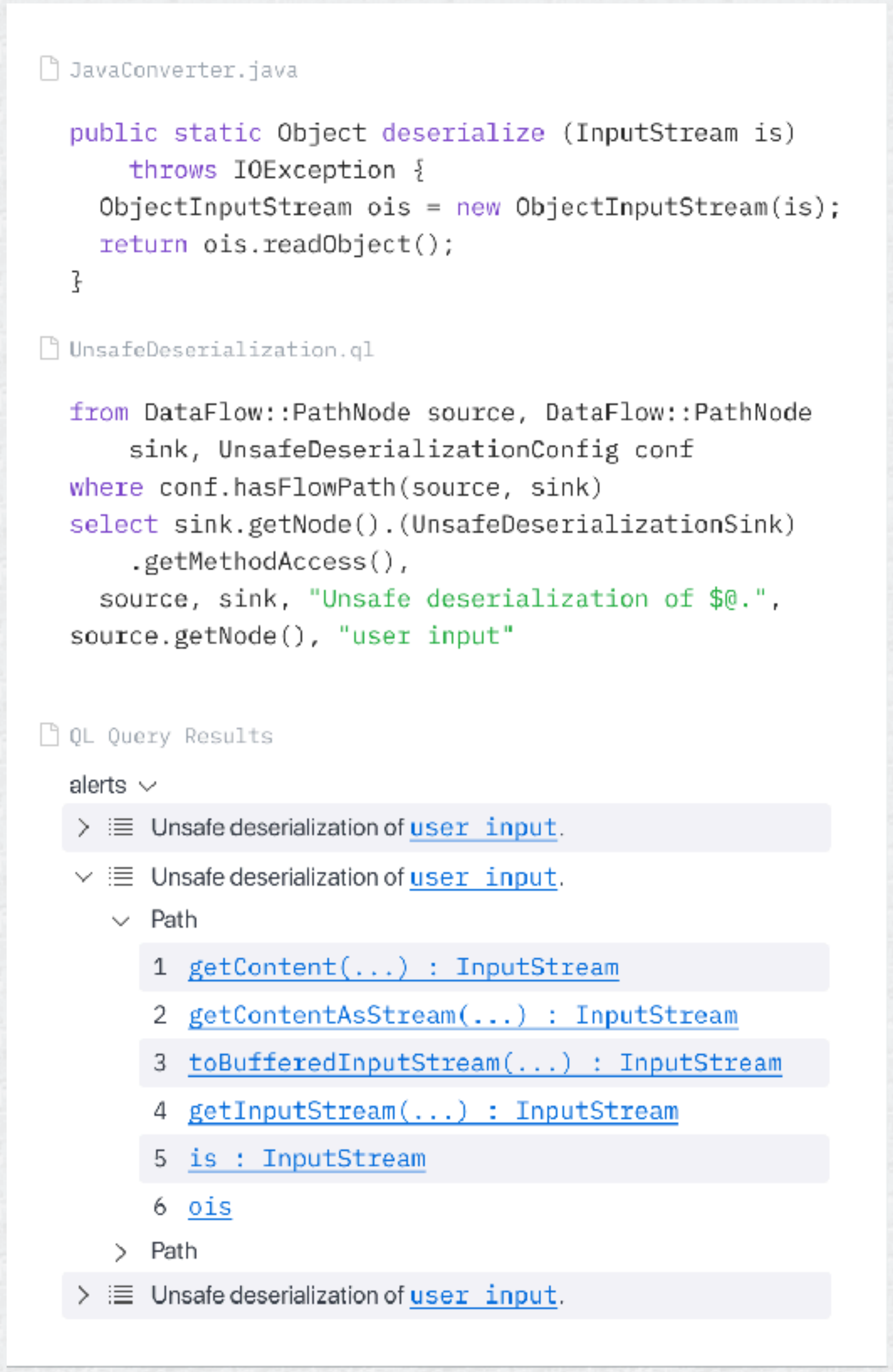
```
@test.checks(‘Call')
@test.test_id('B350')
def unsafe_jwt_verify(context):
    if (context.call_function_name_qual == 'jwt.decode'):
        if context.get_call_arg_value('verify') == 'False':
            return bandit.Issue(
                severity = bandit.HIGH,
                confidence = bandit.HIGH,
                text = 'JSON Web Token decode() method does not verify the HMAC/Key. Attacker can use this to spoof Authentication Tokens'
            )
```

abhaybhargav

Semantic Grep and QL

- Combines the power of Regular Expressions or a full-feature Query Language with the context of Abstract Syntax Trees
- Faster
- More Accurate
- Easier to customise
- Current Landscape:
    - Semgrep
    - CodeQL

CodeQL



JavaConverter.java

```
public static Object deserialize (InputStream is)
    throws IOException {
  ObjectInputStream ois = new ObjectInputStream(is);
  return ois.readObject();
}
```

UnsafeDeserialization.ql

```
from DataFlow::PathNode source, DataFlow::PathNode
    sink, UnsafeDeserializationConfig conf
where conf.hasFlowPath(source, sink)
select sink.getNode().(UnsafeDeserializationSink)
    .getMethodAccess(),
  source, sink, "Unsafe deserialization of $@.",
source.getNode(), "user input"
```

QL Query Results

```
alerts
> Unsafe deserialization of user input.
v Unsafe deserialization of user input.
    v Path
        1 getContent(...) : InputStream
        2 getContentAsStream(...) : InputStream
        3 toBufferedInputStream(...) : InputStream
        4 getInputStream(...) : InputStream
        5 is : InputStream
        6 ois
    > Path
> Unsafe deserialization of user input.
```

abhaybhargav

securitylab.github.com

Semgrep

- Tool for offline static analysis
- Borrows simplicity from Grep, but with the context of an Abstract Syntax Tree Parse engine built in
- Polyglot support
- Existing Database of rules

Demo

abhaybhargav

Notable Areas of As-Code

- Runtime Security Defence/Detection => eBPF
- Threat-Modeling-as-Code => ThreatPlaybook
- Security Orchestration, Automation and Response (SOAR)
- Natural Language Test Automation for DAST