The Firewall is Dead, Long Live to the Firewall!

Davide Annovazzi

Today's Speaker

Davide Annovazzi

Security & Compliance Lead

DISCLAIMER

I am a Google employee.

I am not speaking on behalf of Google or Alphabet, opinions are my own.

HACK IN BO® Winter 2022 Edition

19ª EDIZIONE

Agenda

01

The Castle Approach

03

Common Patterns

02

Zero Trust Networks

04

Policy Enforcement

01

The Castle Approach

The Castle Approach

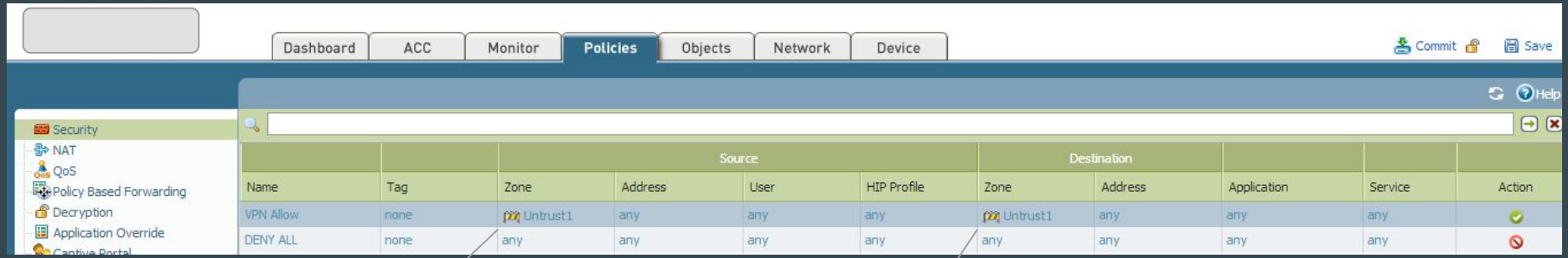
Strong Outer Walls

Trust **Everything** Inside

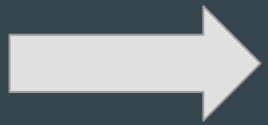
Have you ever checked the Firewall logs?

HACK IN BO®
Winter 2022 Edition
19ª EDIZIONE

The Any / Any way



Zone	Address
Untrust1	any
any	any



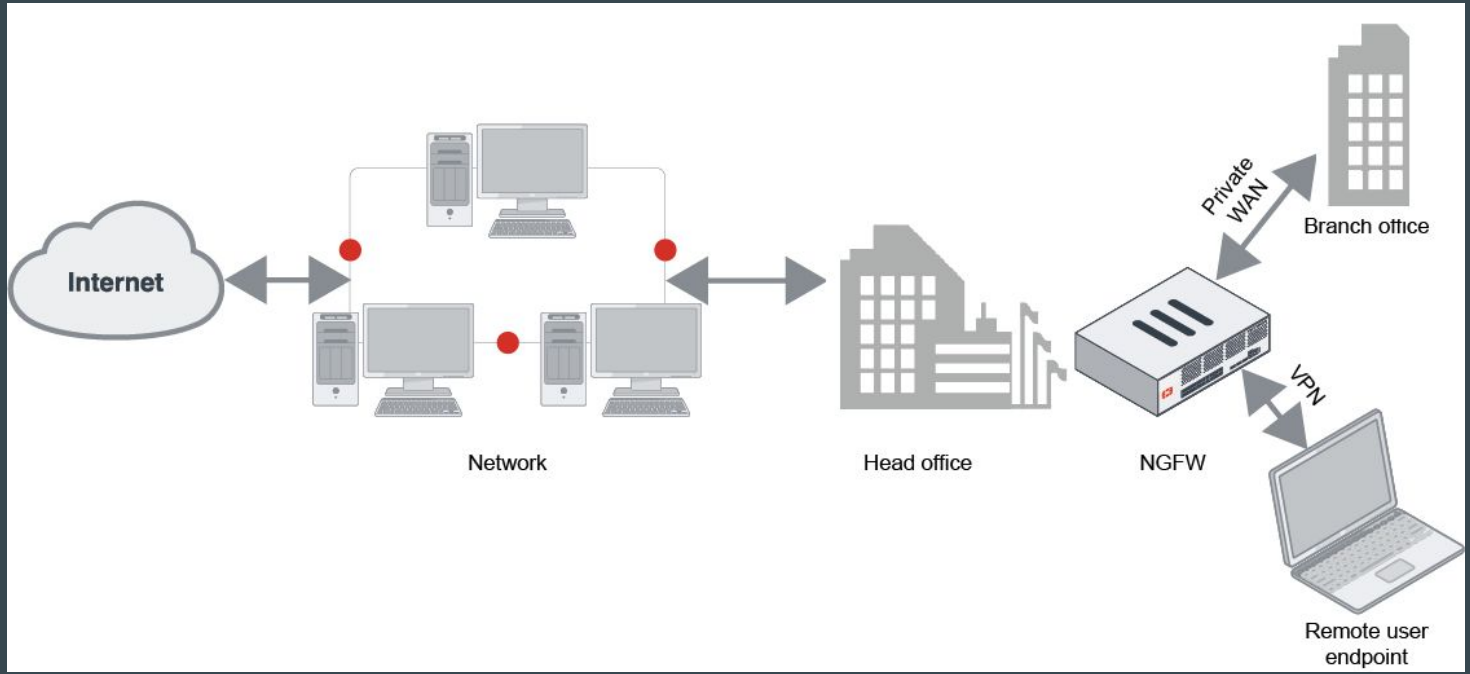
Zone	Address
Untrust1	any
any	any

HACK IN BO Winter 2022 Edition
19ª EDIZIONE

Ever seen an up-to-date firmware in a Firewall?

HACK IN BO®
Winter 2022 Edition
19ª EDIZIONE

Print Nightmare - Do you feel safe?



Trusted Networks

Trusted Networks

and

Microservices

Trusted Networks

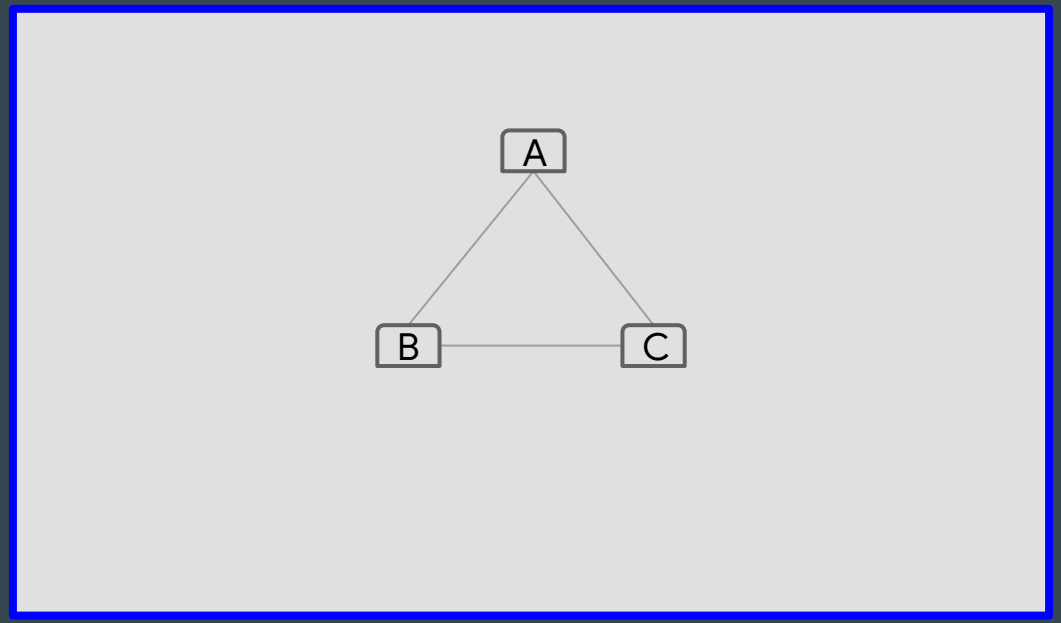
and

Microservices

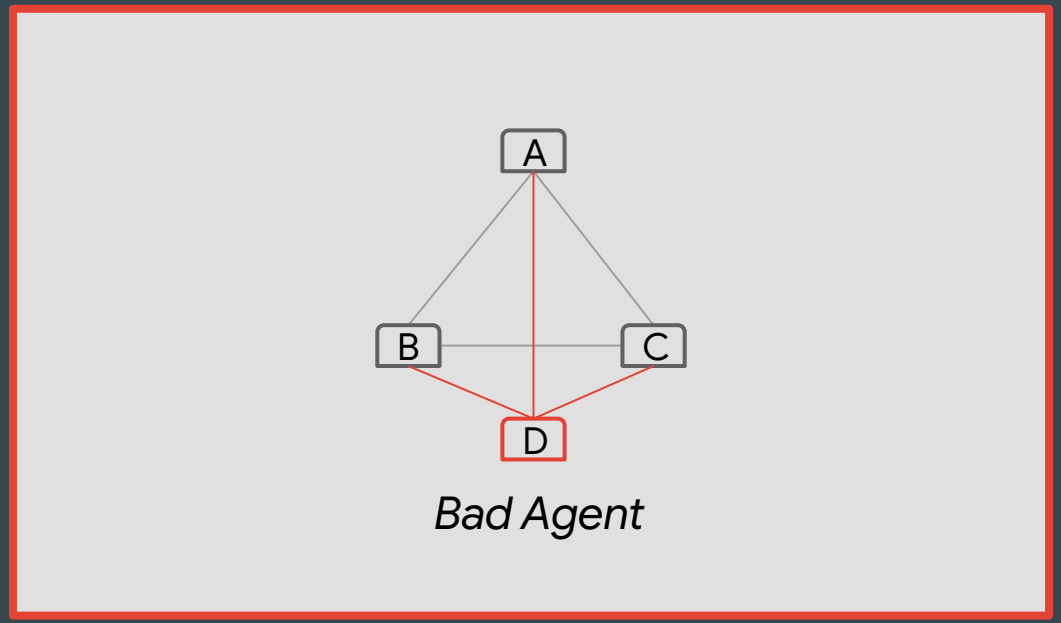
at

Scale

Policies defined at the boundary



Policies defined at the boundary



Operation Aurora

In 2009, a series of cyber attacks called Operation Aurora targeted 20+ tech companies

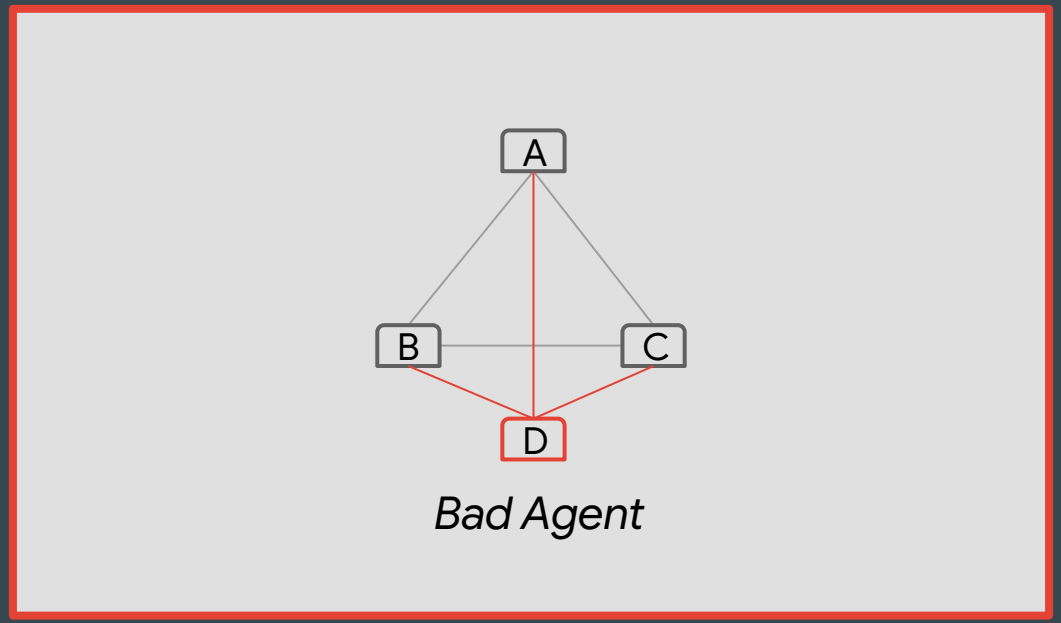
The event was discovered to be a state-sponsored attack

HACK IN BO®
Winter 2022 Edition
19ª EDIZIONE

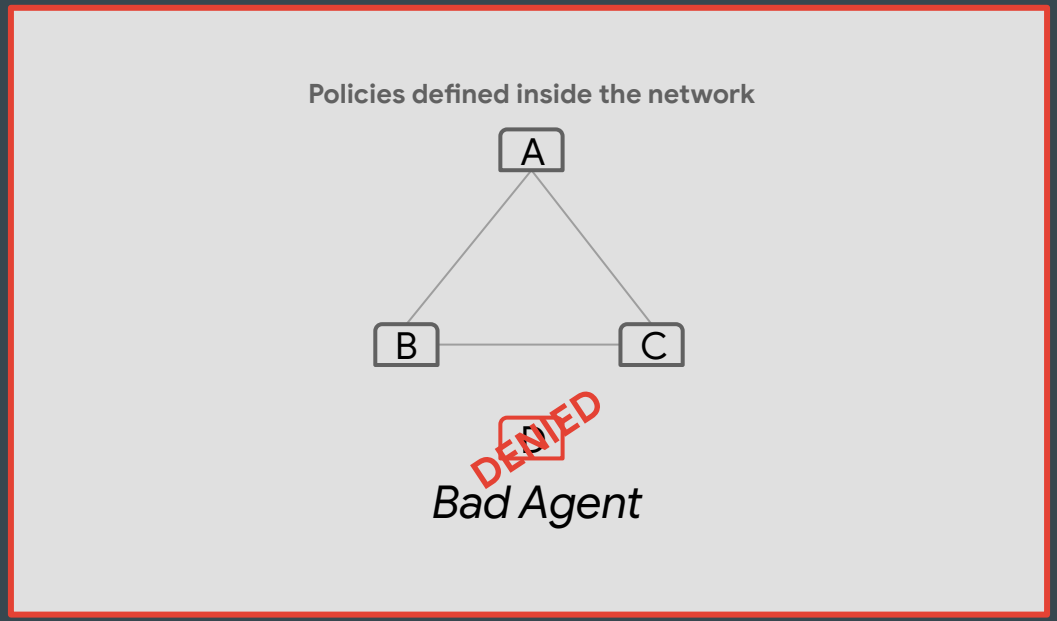
02

Zero Trust Networks

Policies defined at the boundary



Policies defined at the boundary



Getting Beyond Firewalls

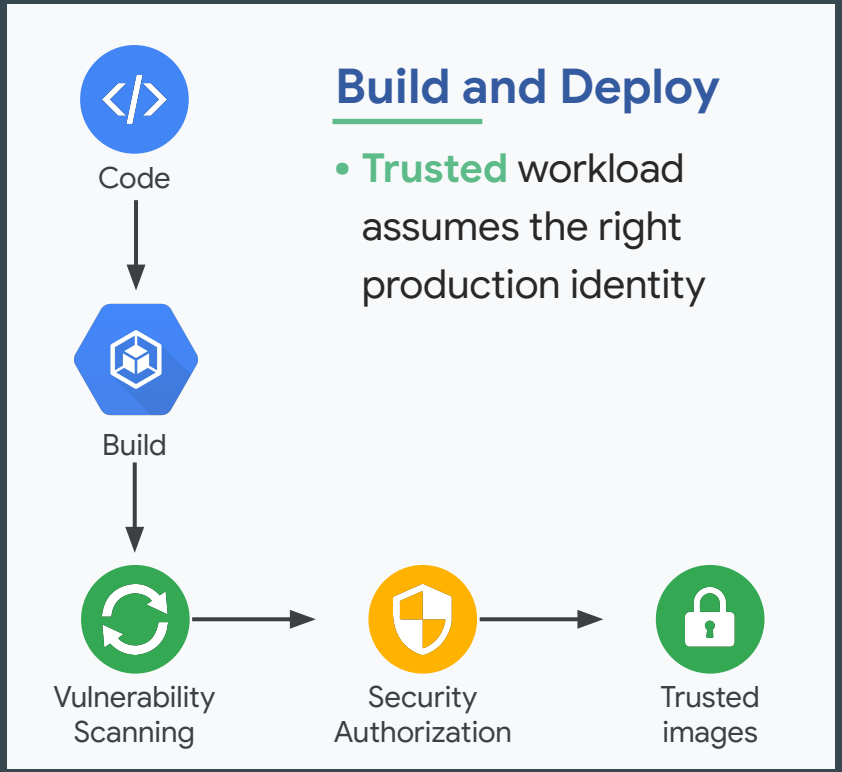
- Access is granted based on what we **know about you and your device**
- Connecting from a particular network must not determine which services you can access
- All access must be **authenticated, authorized and encrypted**

Getting Beyond Firewalls

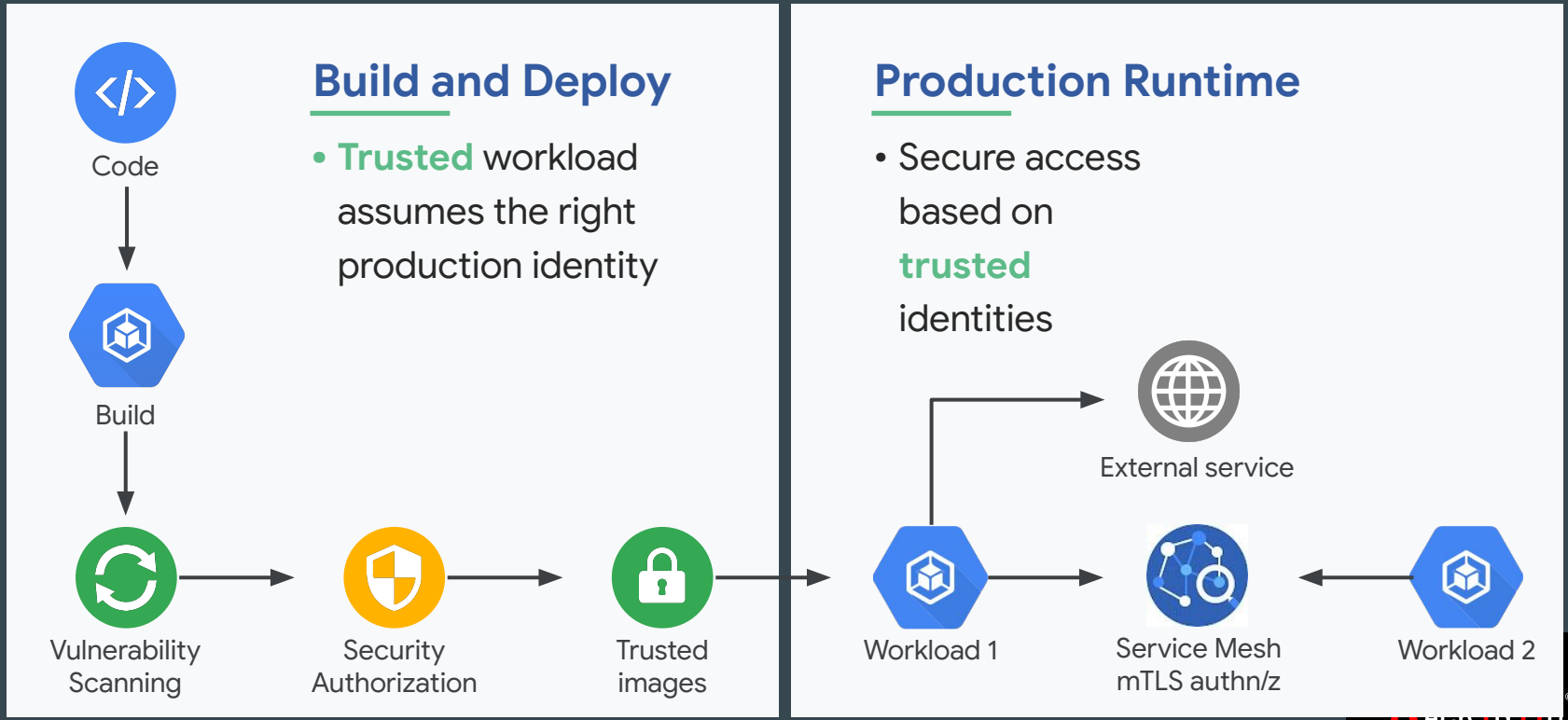
- Access is granted based on what we **know about you and your device**
- Connecting from a particular network must not determine which services you can access
- All access must be **authenticated, authorized and encrypted**

- **No inherent mutual trust** among services
- Services have **strong ID**
- ... running approved images
- ... on approved hardware
- **Automated rollout** of changes

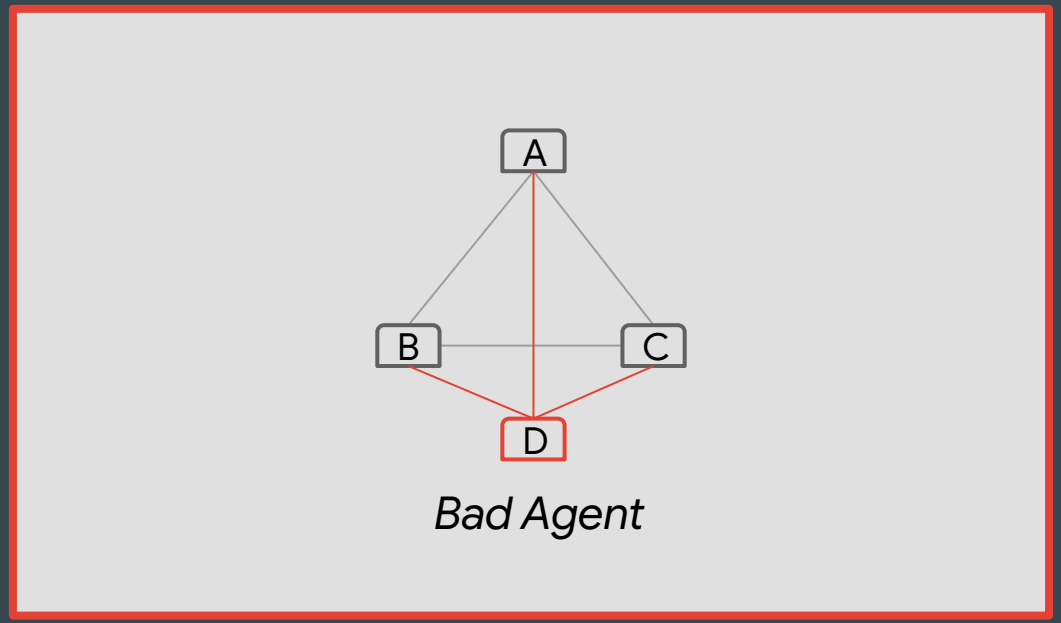
Containerized Workload Lifecycle



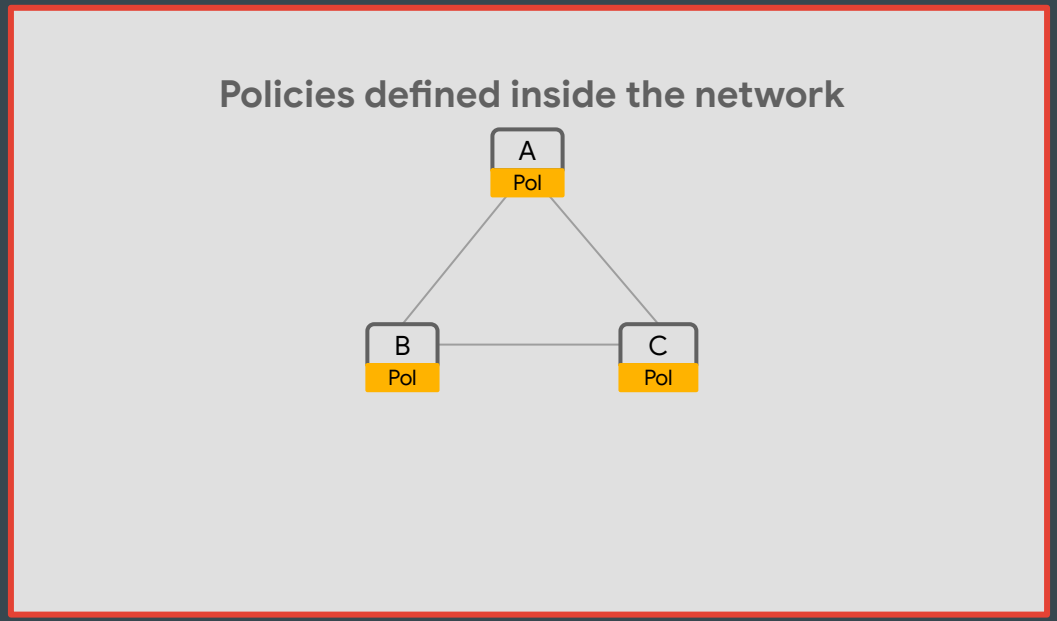
Containerized Workload Lifecycle



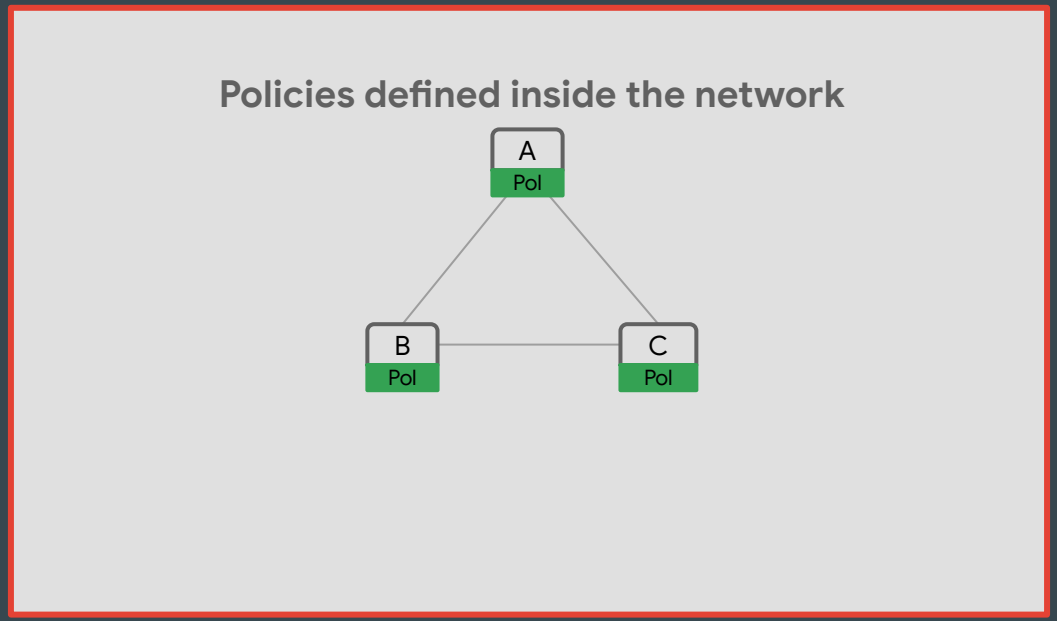
Policies defined at the boundary



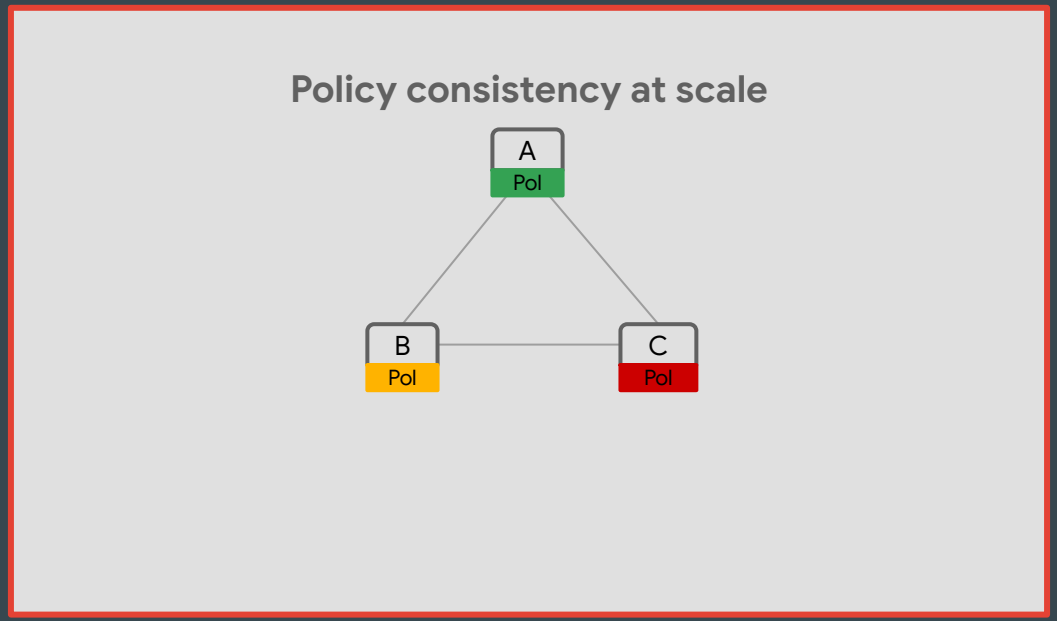
Policies defined at the boundary



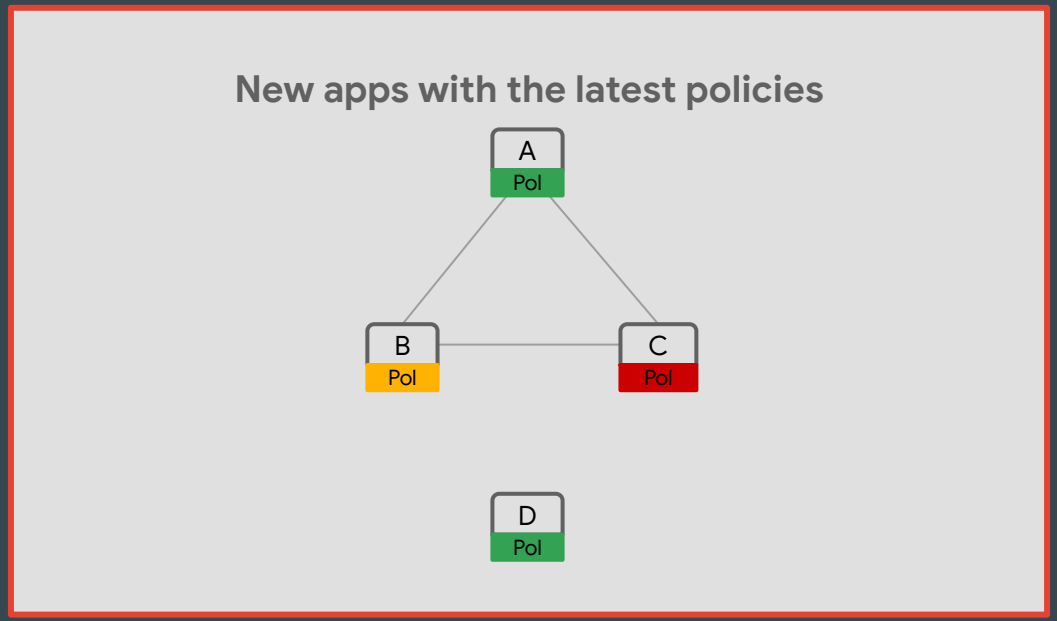
Policies defined at the boundary



Policies defined at the boundary

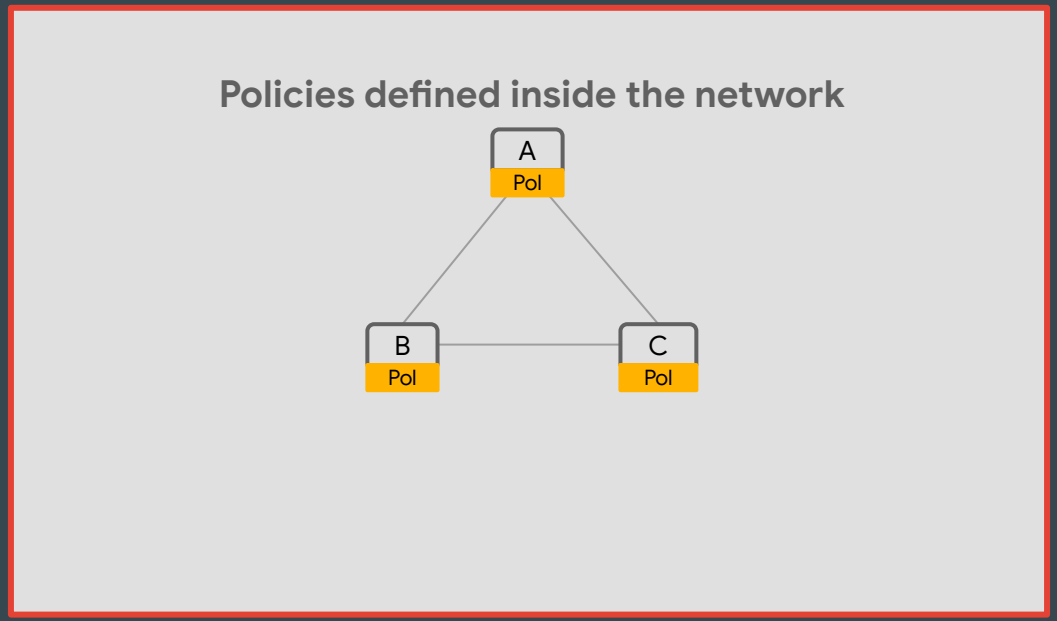


Policies defined at the boundary

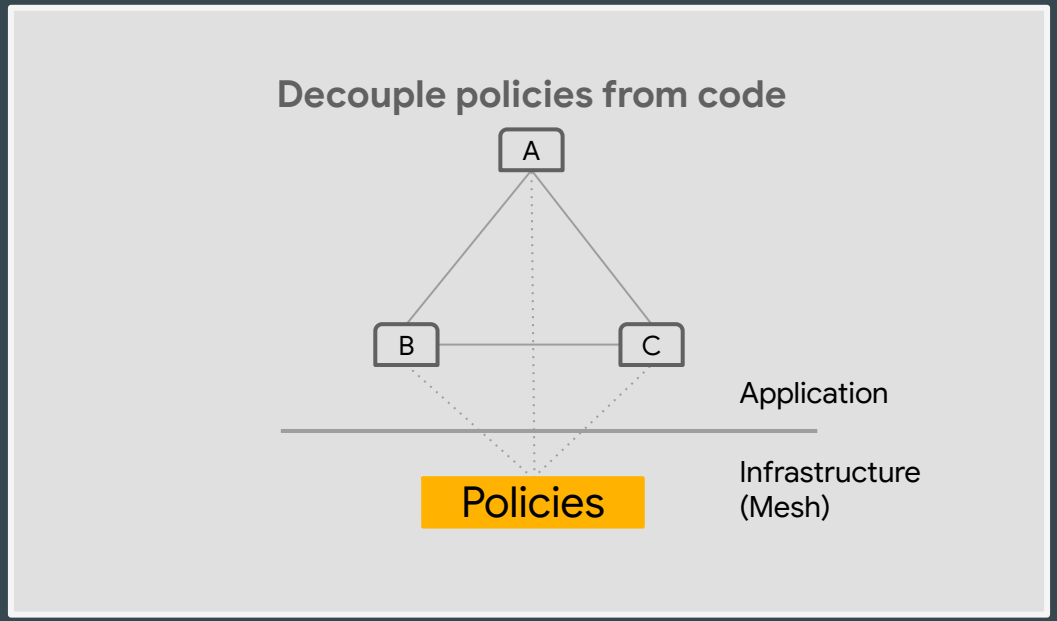


Mesh and Zero Trust Networks

Policies defined at the boundary



Policies defined at the boundary



03

Common Patterns

How do you trust a production client?

Right Peer

with the

Right Credentials

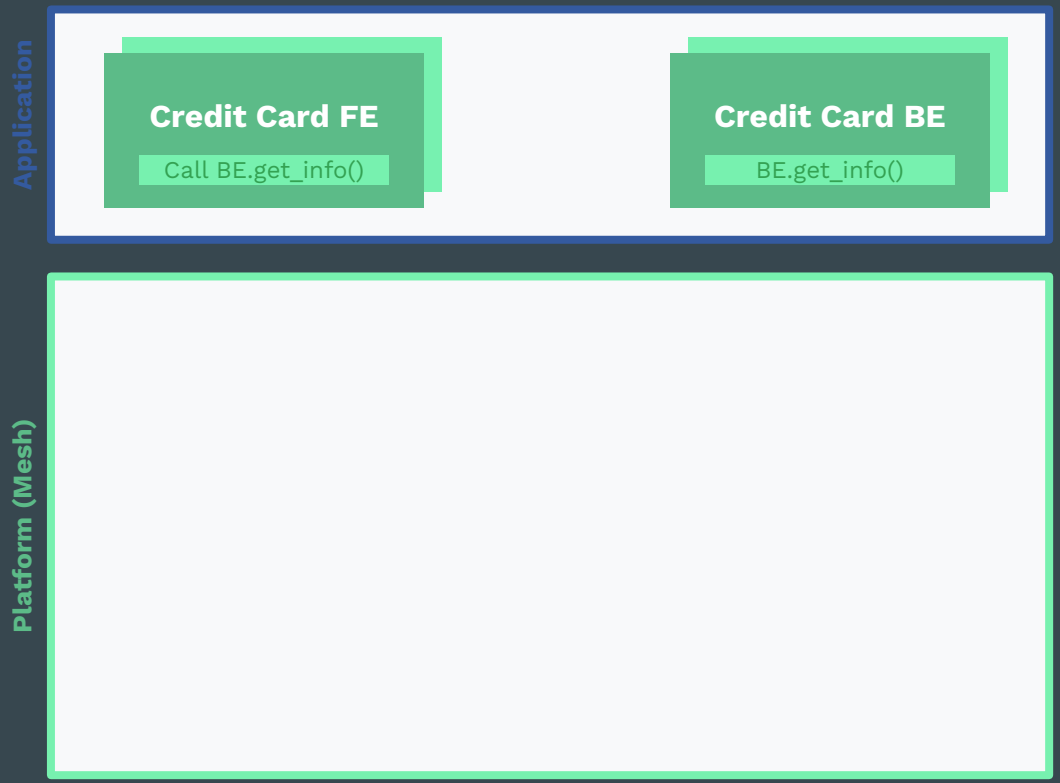
on behalf of the

Right User

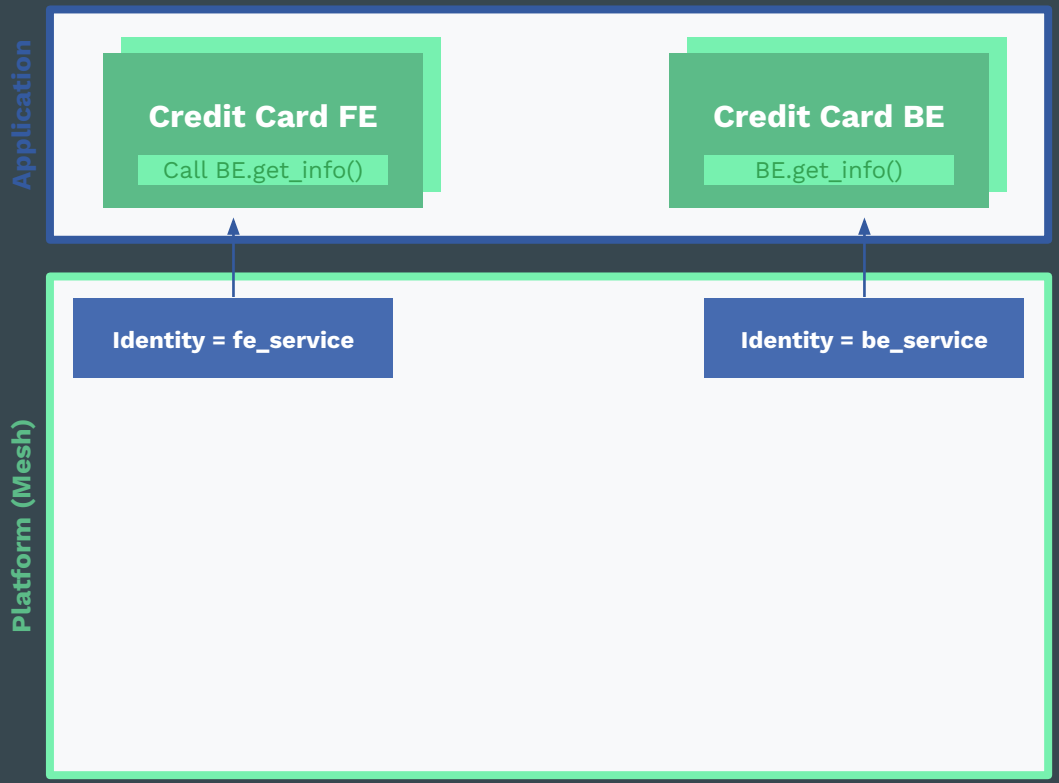
01 |

Right Peer with the **Right Credentials**

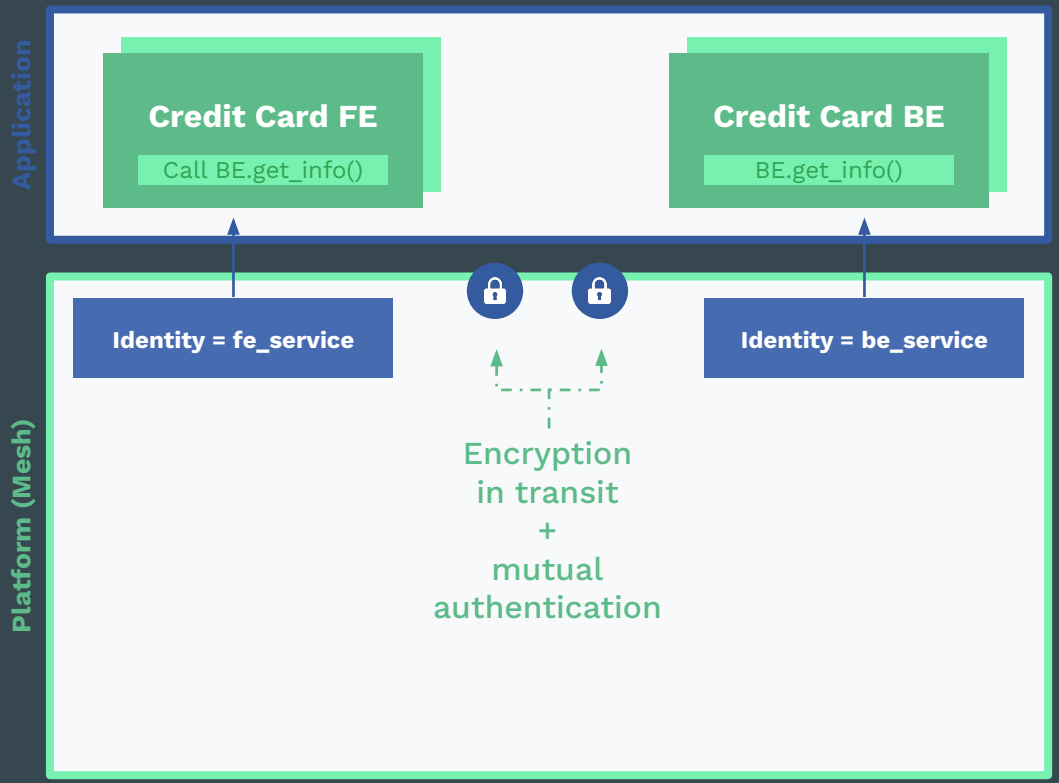
Right Peer with the Right Credentials



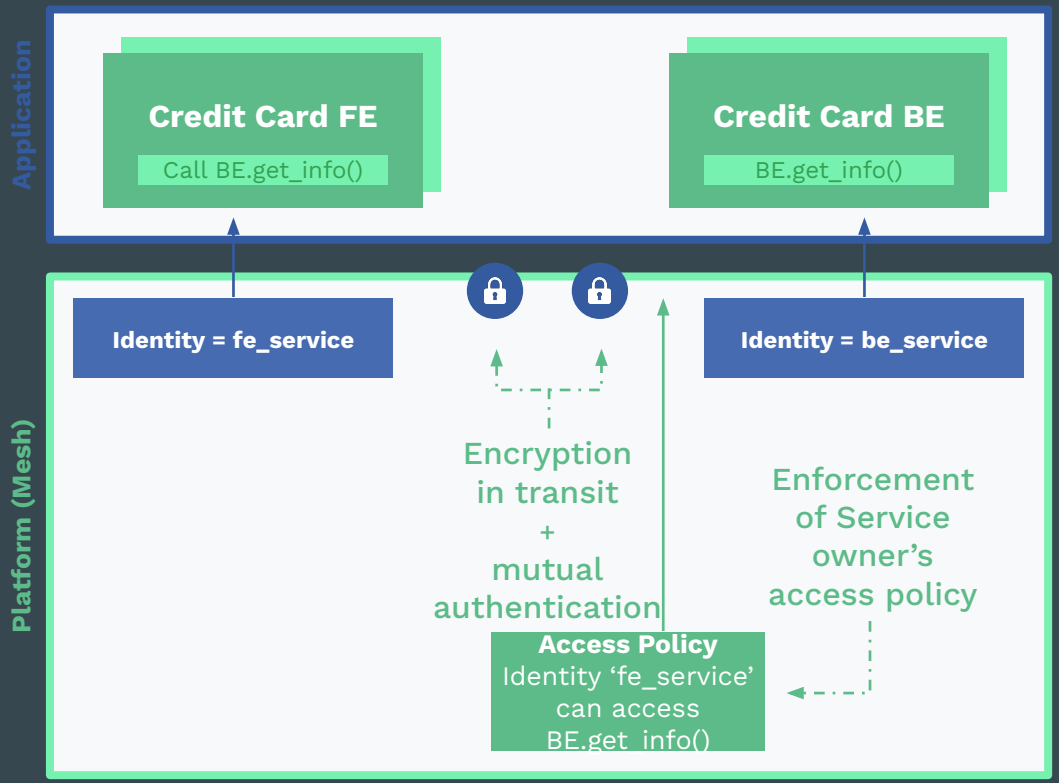
Right Peer with the Right Credentials



Right Peer with the Right Credentials



Right Peer with the Right Credentials



Code Sample

Right Peer with the

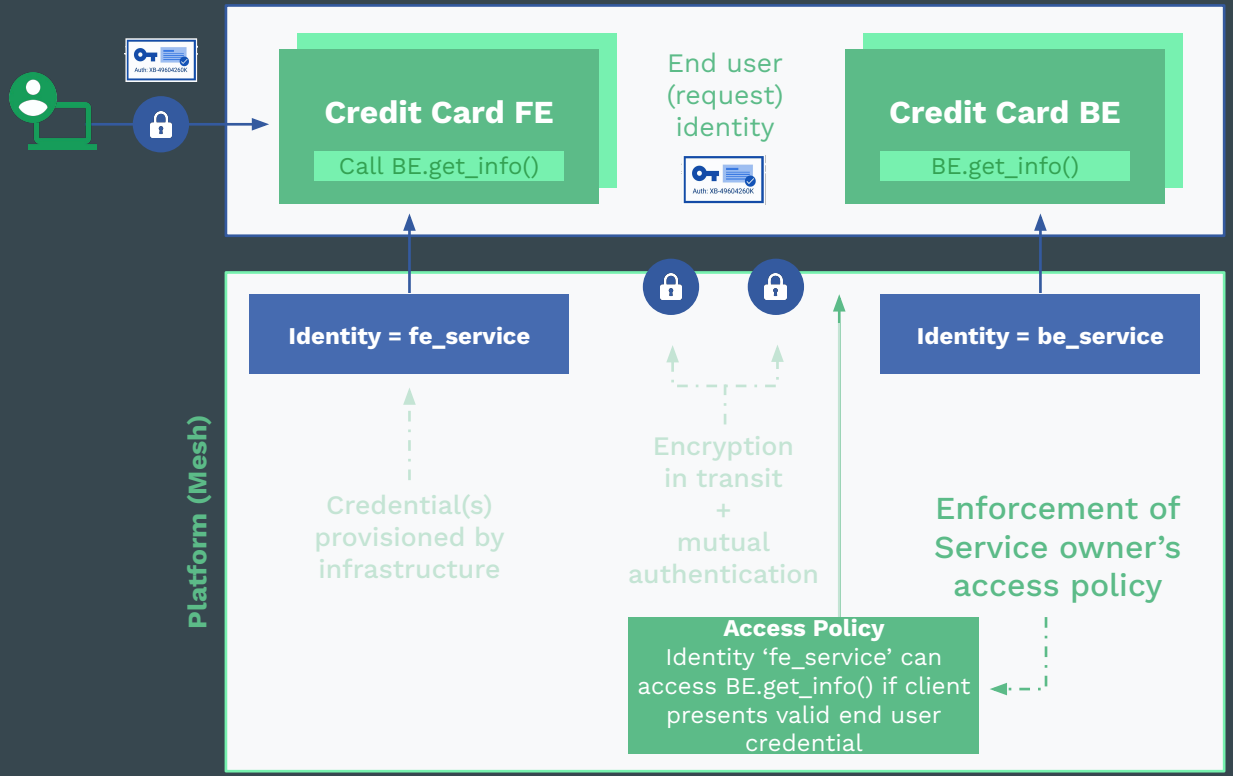
Right Credentials


```yaml
apiVersion: security.istio.io/v1beta1
kind: PeerAuthentication
metadata:
  name: default
  namespace: foo
spec:
  mtls:
    mode: STRICT

---

apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: creditBE
  namespace: creditBE
spec:
  action: ALLOW
  rules:
  - from:
    - source:
        namespaces: ["creditFE"]
    to:
    - operation:
        methods: ["GET","POST"...]
```

01 | **Right Peer** with the **Right Credentials**

02 | **Right Peer** with the **Right Credentials**
on behalf of the **Right User**

Right Peer with the Right Credentials on behalf of the Right User



Code Sample

Right Peer with the

Right Credentials

on behalf of the

Right User


```yaml
apiVersion: security.istio.io/v1beta1
kind: RequestAuthentication
metadata:
  name: ingress-jwt
  namespace: istio-system
spec:
  selector:
    matchLabels:
      istio: ingressgateway
  jwtRules:
  - issuer: "issuer-foo"
    jwksUri: "https://example.com/.well-known/jwks.json"

---

apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: require-jwt
  namespace: creditBE
spec:
  selector:
    matchLabels:
      app: creditBE
  action: ALLOW
  rules:
  - from:
    - source:
       requestPrincipals: ["issuer@secure.jwt.io/issuer@secure.jwt.io"]
    when:
    - key: request.auth.claims[groups]
      values: ["group1"]
```

Request Samples

```
$ curl ${INGRESS_IP}

 RBAC: access denied
```

```
 $ curl --header "Authorization: Bearer ${INVALID_JWT}" ${INGRESS_IP}

 Jwt issuer is not configured
```

```
$ curl --header "Authorization: Bearer ${VALID_JWT}" ${INGRESS_IP}

Hello World! /
```

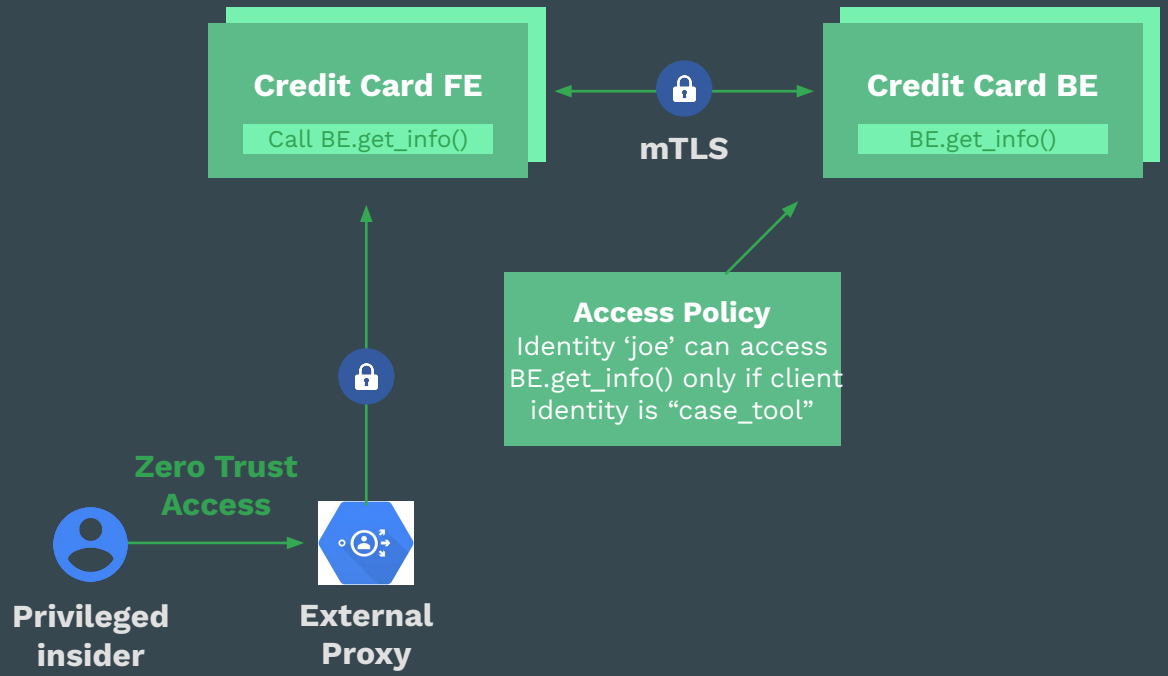
01 | **Right Peer** with the **Right Credentials**

02 | **Right Peer** with the **Right Credentials** on behalf of the **Right User**

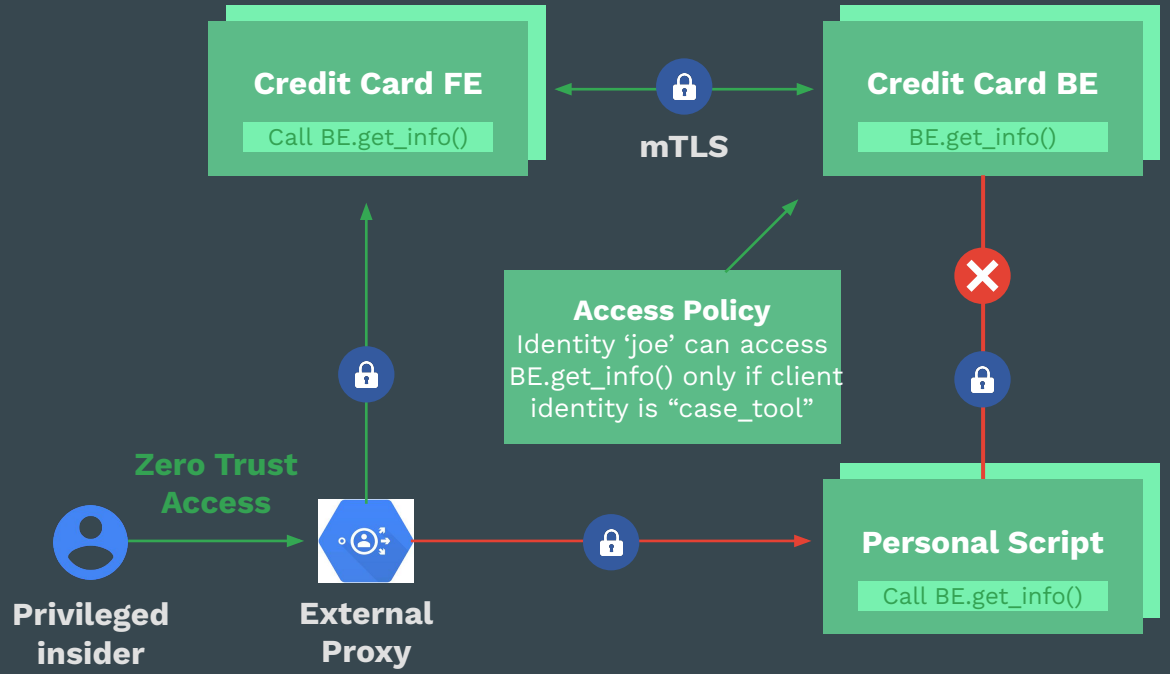
03 | **Right Peer** with the **Right Credentials** on behalf of a **Strongly Authenticated Insider**

Right Peer with the Right Credentials on behalf of a Strongly Authenticated Insider

Right Peer with the Right Credentials on behalf of a Strongly Authenticated Insider



Right Peer with the Right Credentials on behalf of a Strongly Authenticated Insider



Code Sample

Right Peer with the **Right Credentials** on behalf of a **Strongly Authenticated** Insider

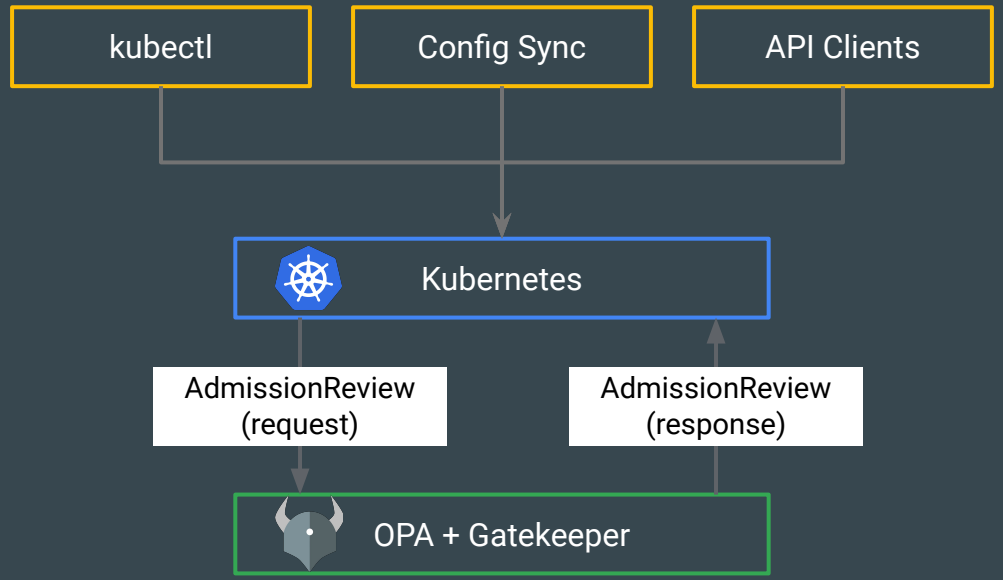

```yaml
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: BE
  namespace: BE
spec:
  action: ALLOW
  rules:
  - from:
    - source:
        namespaces: ["FE"]
    to:
    - operation:
        methods: ["GET","POST"...]
---
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: require-jwt
  namespace: app
spec:
  selector:
    matchLabels:
      app: app
  action: ALLOW
  rules:
  - from:
    - source:
       requestPrincipals: ["issuer@secure.jwt.io/issuer@secure.jwt.io"]
    when:
    - key: request.auth.claims[groups]
      values: ["group1"]
```

04

Policy Enforcement

Gatekeeper

- Kubernetes Admission Controller that extends OPA
- Actively enforce custom rules against all API clients
- Passively audit all K8s objects



Policy structure

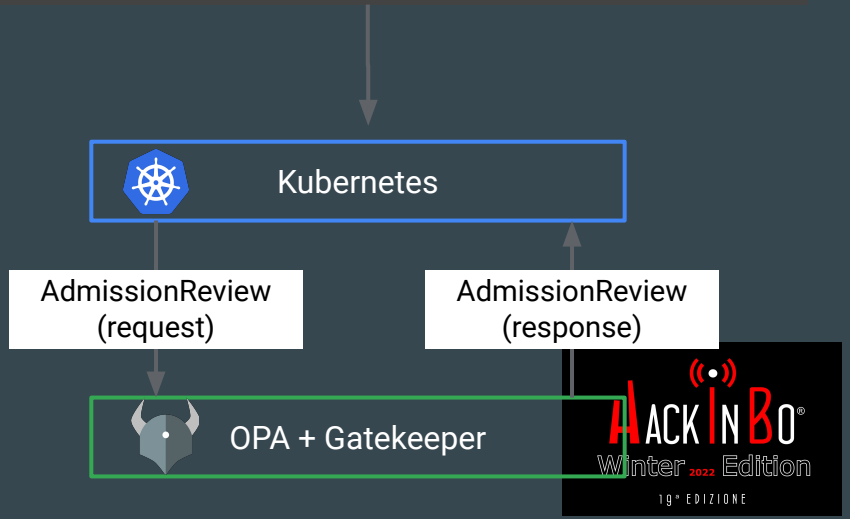
Policies are written using Rego and packaged as parameterized objects.

```
apiVersion: templates.gatekeeper.sh/v1beta1
kind: ConstraintTemplate
metadata:
  name: destinationruletlsenabled
spec:
  crd:
    spec:
      names:
        kind: DestinationRuleTLSEnabled
  targets:
    - target: admission.k8s.gatekeeper.sh
      rego: |
        package asm.guardrails.destinationruletlsenabled

        # spec.trafficPolicy.tls.mode == DISABLE
        violation[{"msg": msg}] {
          d := input.review.object
          tlsdisable := { "tls": {"mode": "DISABLE"}}

          ktpl := "trafficPolicy"
          tpl := d.spec[ktpl][_]
          not tpl != tlsdisable["tls"]

          msg := sprintf("%v %v.%v mode == DISABLE",
            [d.kind, d.metadata.name, d.metadata.namespace])
        }
```

Denying & auditing K8s configuration

With Gatekeeper configured, incoming objects can be **denied admission** into the cluster or **audited** if they violate governance.

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: my-destination-rule
  namespace: default
spec:
  host: myservice
  trafficPolicy:
    tls:
      mode: DISABLE
```



(Zero) Trust but Verify

Summary

- Treat all networks as **untrusted**
- **Decouple** policies from Applications
- **Right Peer** with the **Right Credentials** on behalf of the **Right User**
- Enforce and **verify**

Thank you!

Davide Annovazzi