Secure Systems Group, Aalto University

Metehan Koç, Jose Luis Martin Navarro, Tuomas Aura

Network Policy Verification in Kubernetes

Problem

Modern cloud-native environments increasingly rely on high-density, multi-CNI Kubernetes deployments where overlapping and conflicting network policies create significant operational complexity. Organizations struggle to:

- Determine accurate pod-to-pod reachability matrices across heterogeneous CNI configurations
- Identify and resolve ambiguities arising from overlapping or contradictory network policy rules
- Optimize network policy sets to reduce redundancy while maintaining the intended security posture

Challenges

1. Ambiguities of Kubernetes Network Policy Definitions
2. Custom Resource Definitions of different CNIs
3. Limitations of control plane analysis
4. Determining Endpoints
5. Default and optional fields in Network Policies

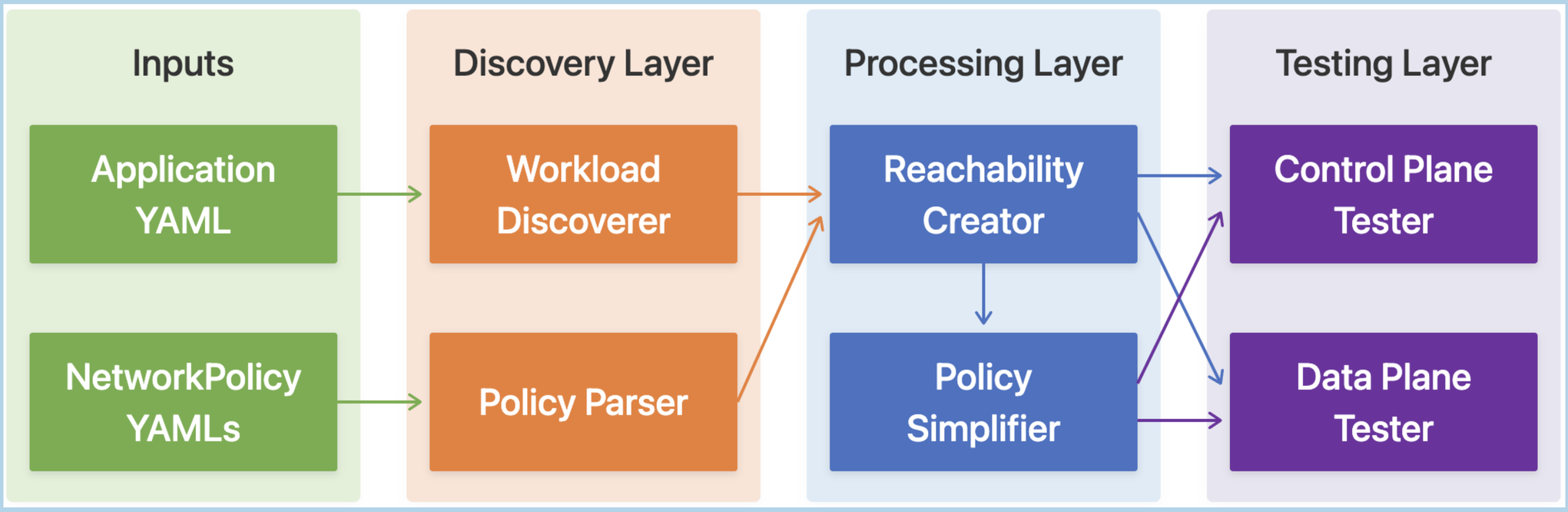
```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: tricky-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: myapp
  egress:
    - to:
        - podSelector:
            matchLabels:
              app: backend
      ports:
        - port: 8080
```

```yaml
apiVersion: cilium.io/v2
kind: CiliumClusterwideNetworkPolicy
metadata:
  name: cluster-wide-policy
spec:
  endpointSelector:
    matchLabels:
      app: myapp
  ingress:
    - fromEndpoints:
        - matchLabels:
            app: frontend
      toPorts:
        - ports:
            - port: "8080"
              protocol: TCP
```

Assumptions

1. YAML file shows the all containers and open ports of the application.
2. If it is not possible to determine connection between two entities, “maybe” will be used.
3. Rows of reachability matrix defined as workloads and columns are defined as endpoints.

Solution Architecture



Aalto University