KTH ROYAL INSTITUTE
OF TECHNOLOGY

Trust Management For a Decentralized Service Exposure Marketplace

Ahmed Beder

Quick Intro

- Panagiotis Papadimitatos (**NSS Group**)
- Mohammad Khodaei (**NSS Group**)
- Aleksandra Obeso Duque (**Ericsson**)
- Antti Ylä-Jääski (**Aalto**)

Agenda

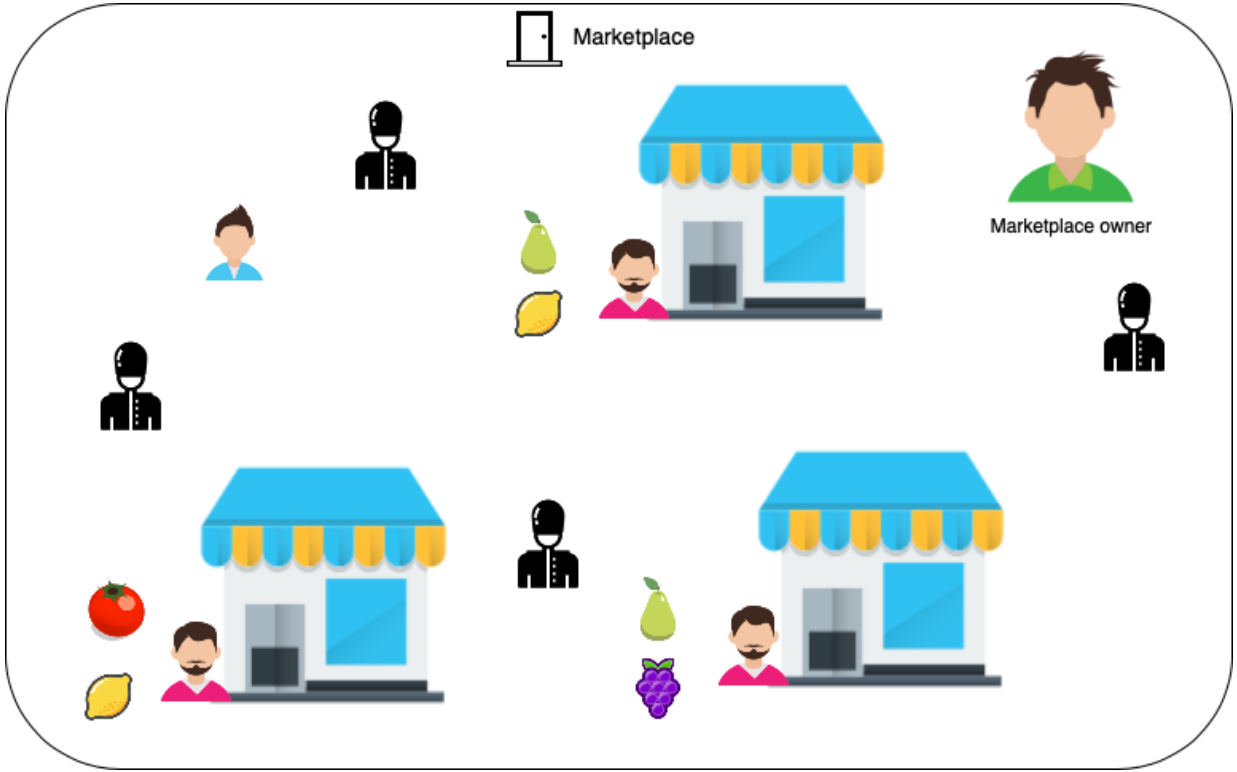
Background
Problem statement

Objectives
Implementation

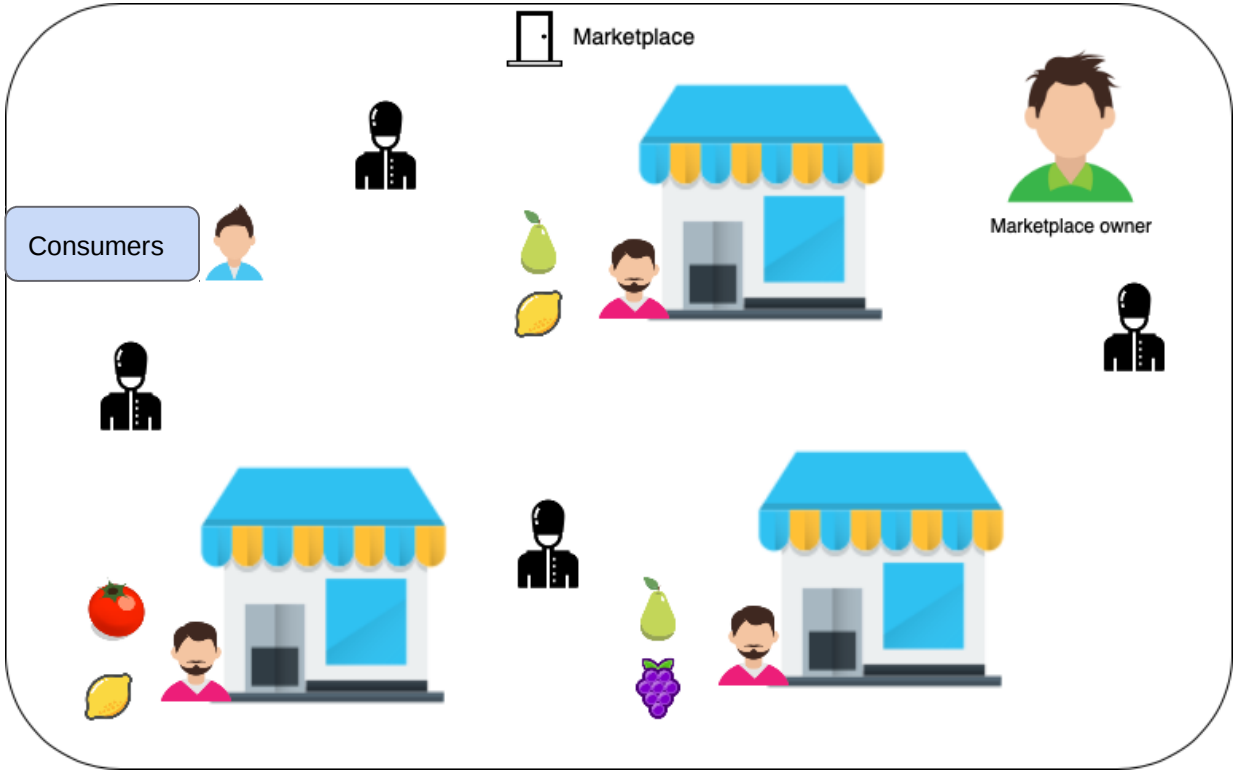
Results
Future work

Q&A

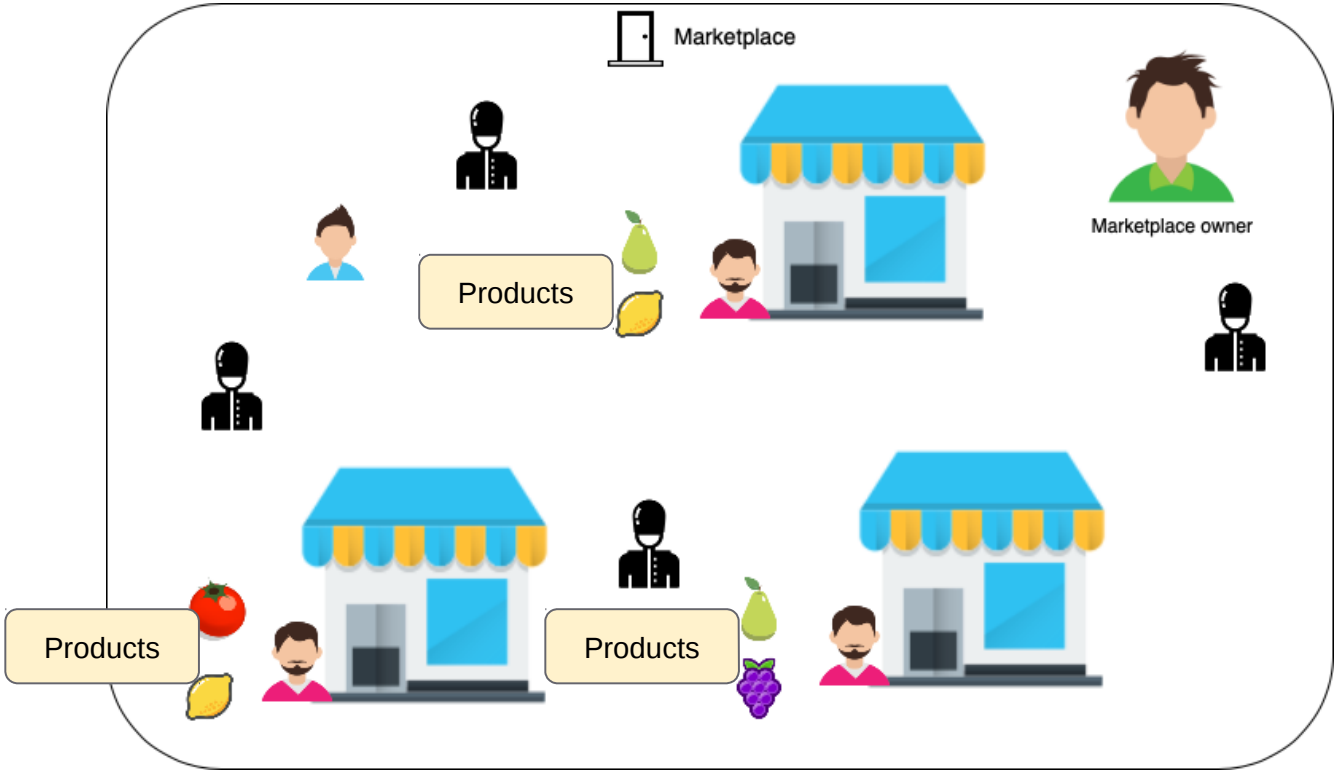
Central marketplace



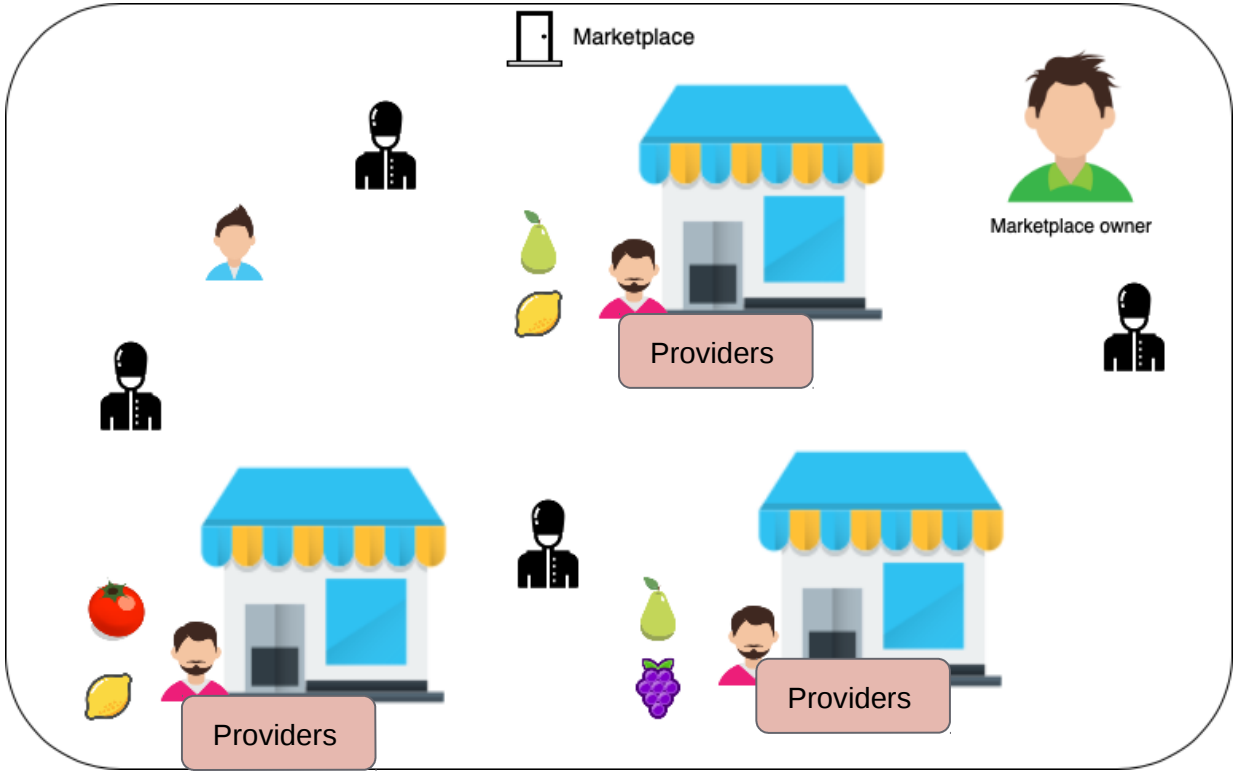
Central marketplace



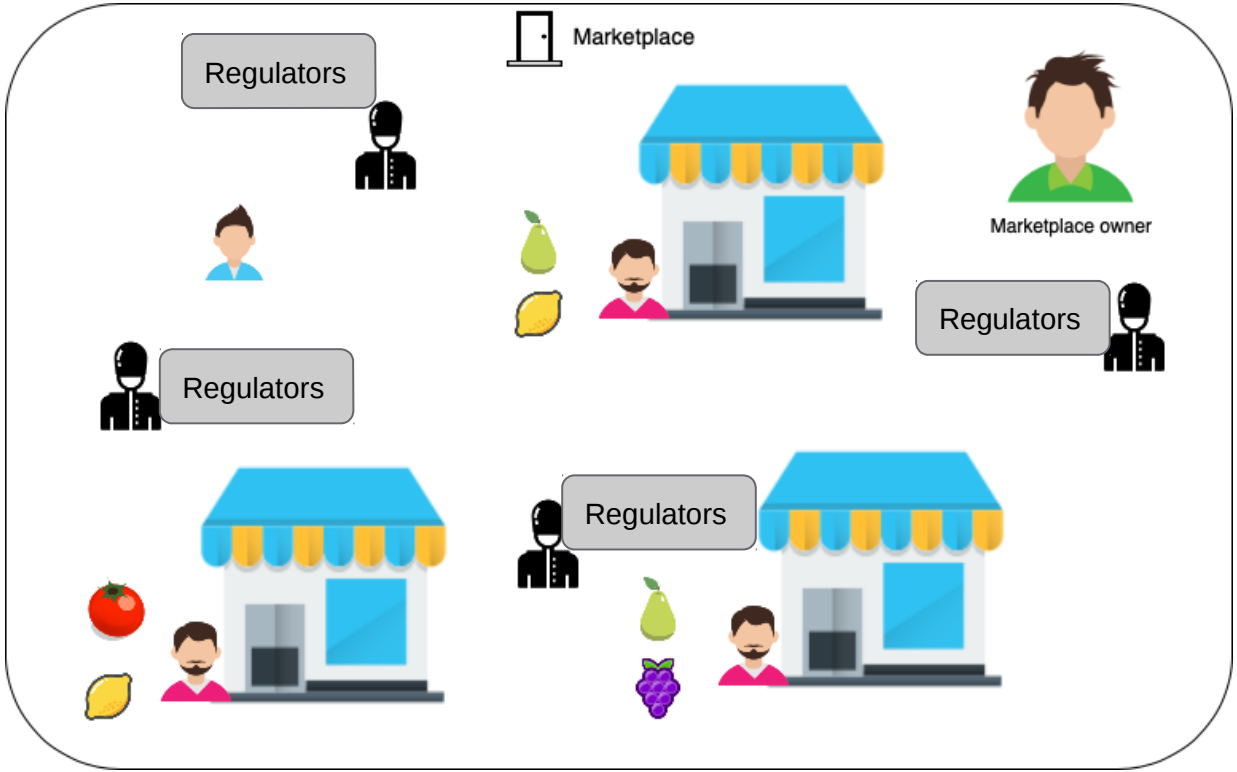
Central marketplace



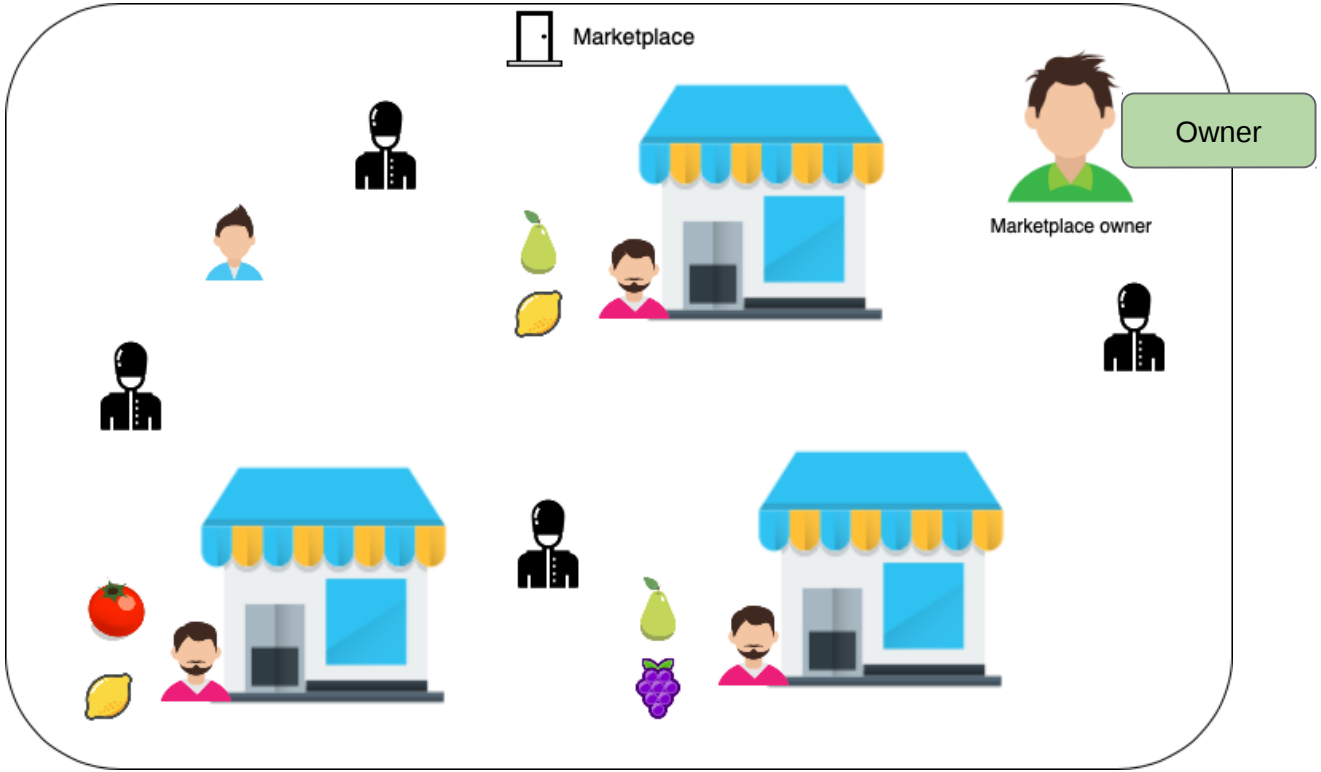
Central marketplace



Central marketplace



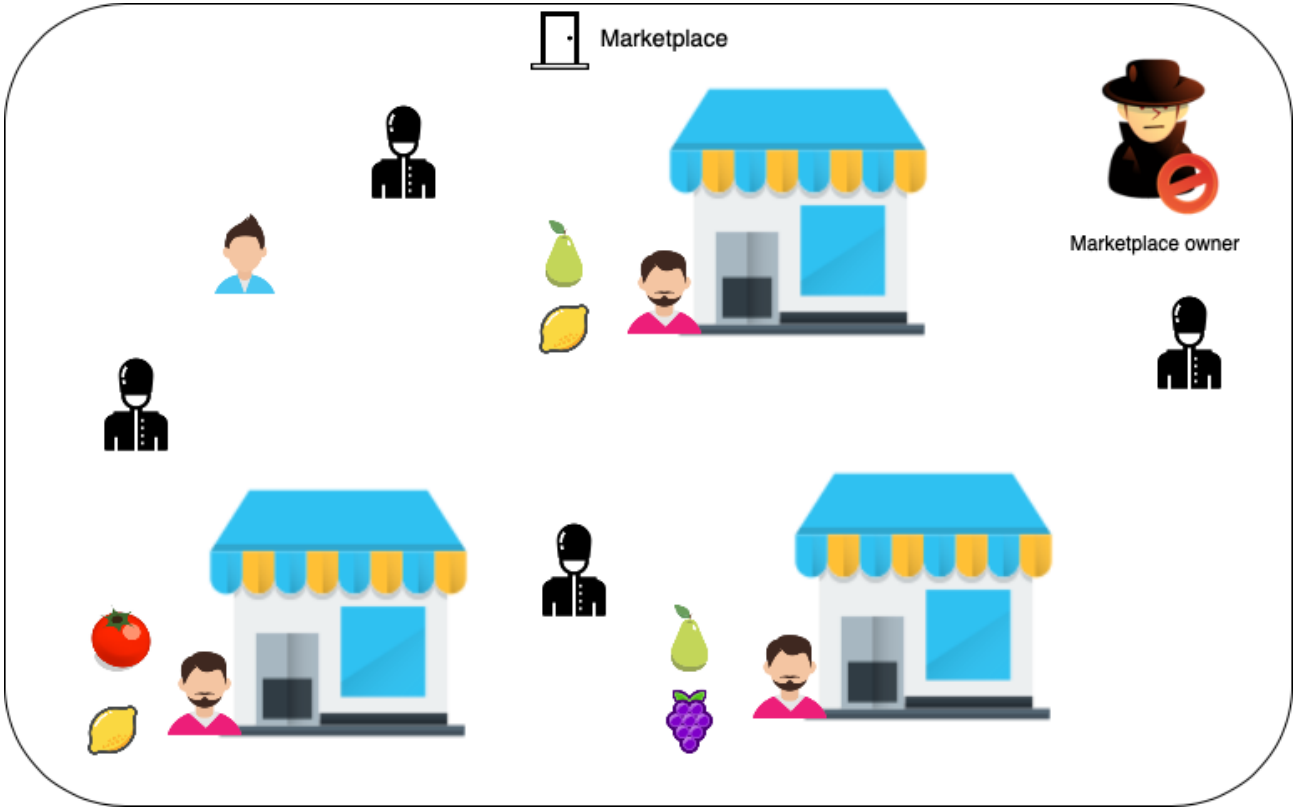
Central marketplace



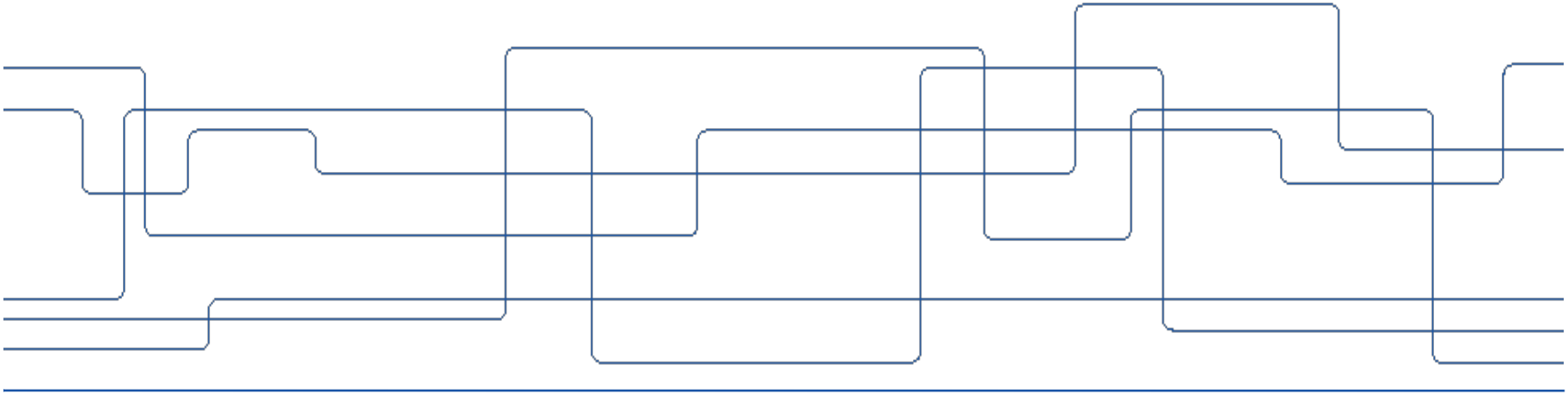
Central marketplace

Limitations

- Single Point of Failure
- Limited
  - Products
  - Providers



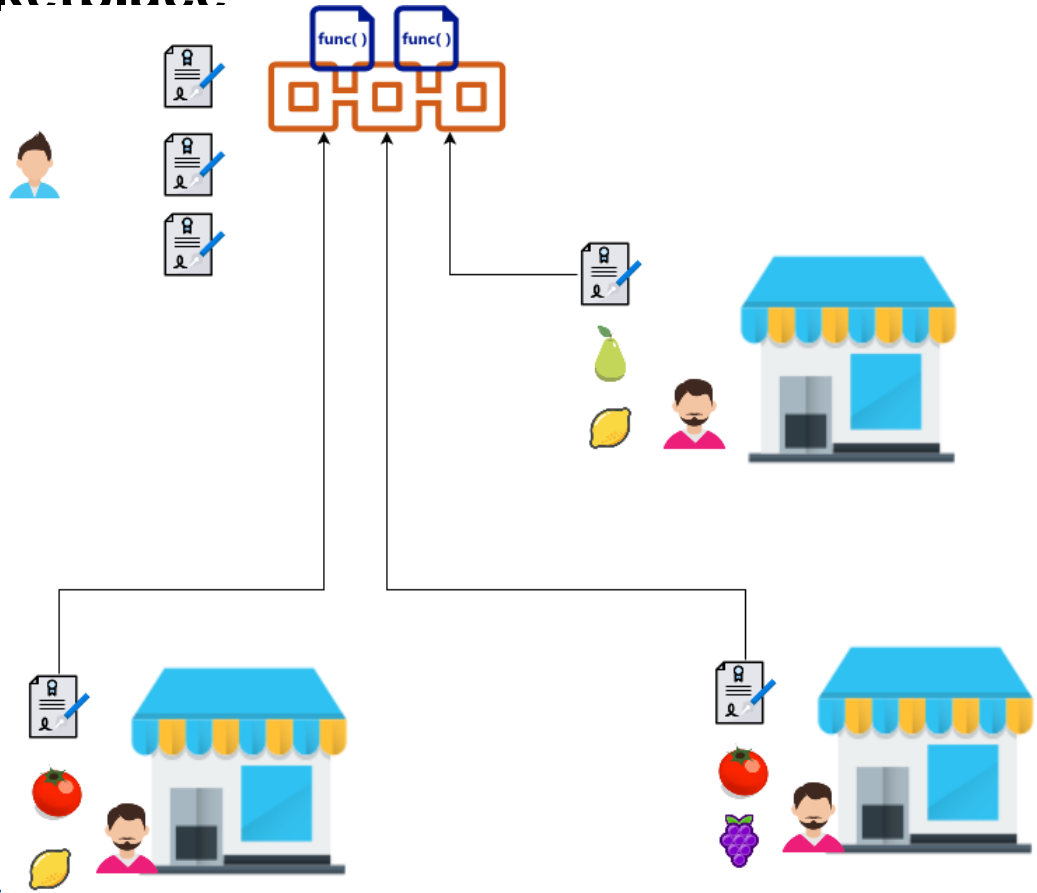
What can we do?



Decentralized marketplace

Entities

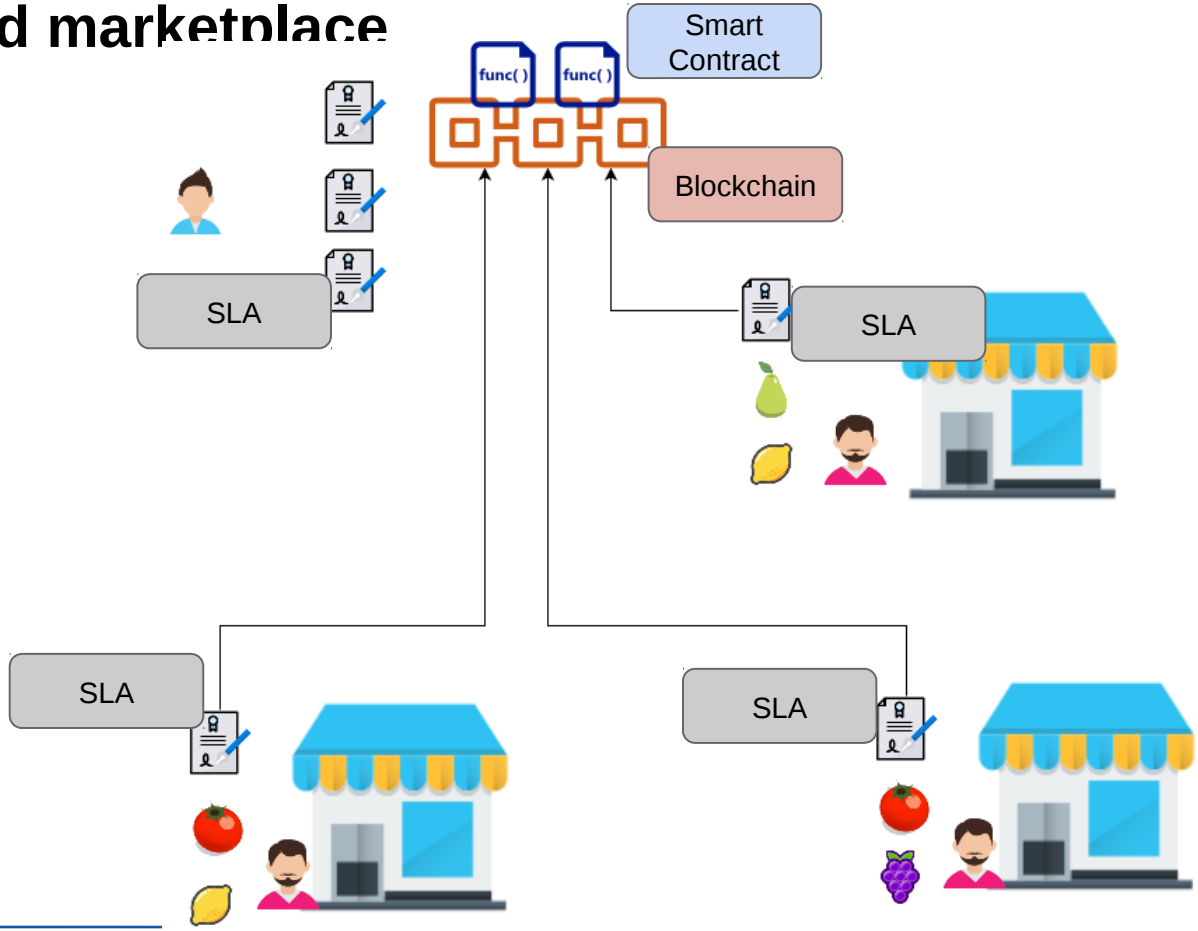
- Consumers ✔
- Providers ✔
- Products ✔
- ~~Regulators~~
- ~~Owner~~



Decentralized marketplace

Entities

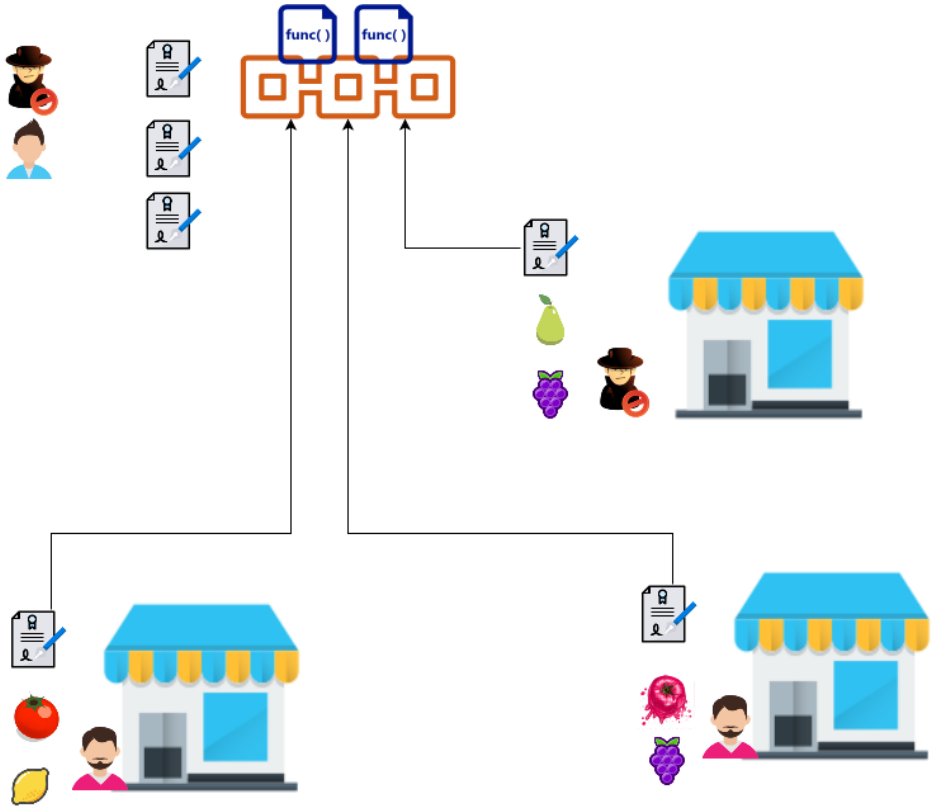
- Consumers ✔
- Providers ✔
- Products ✔
- ~~Regulators~~
- ~~Owner~~
- Blockchain & Smart Contracts
- Service Level Agreements



Decentralized marketplace

What can go Wrong?

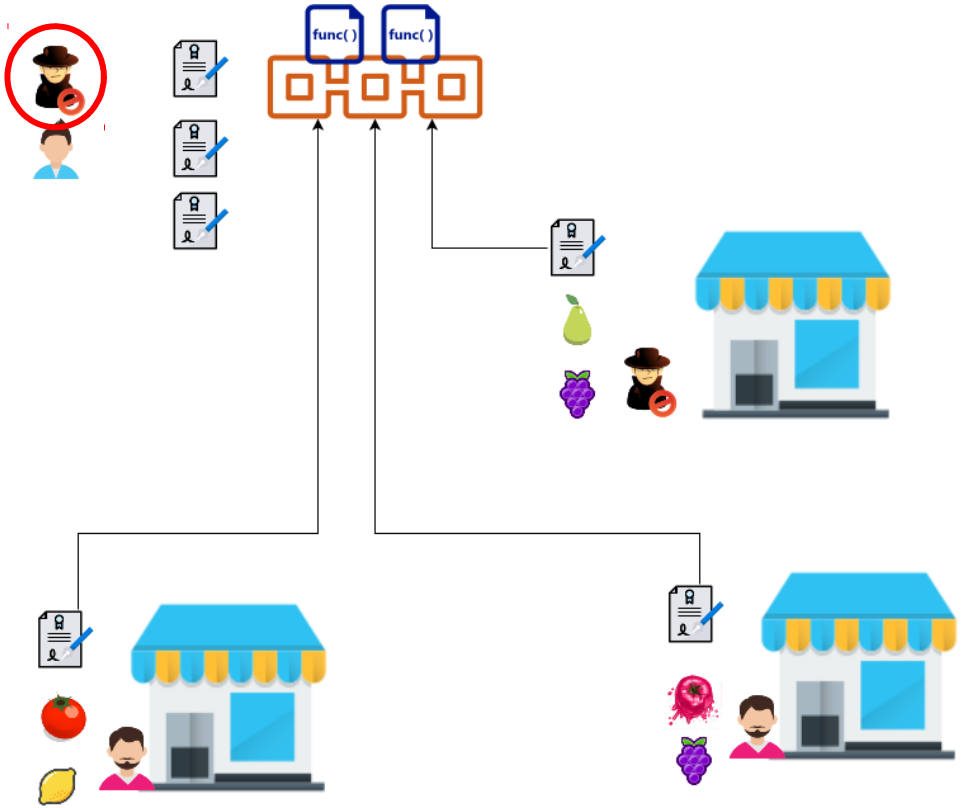
- Consumers
- Providers
- Products



Decentralized marketplace

What can go Wrong?

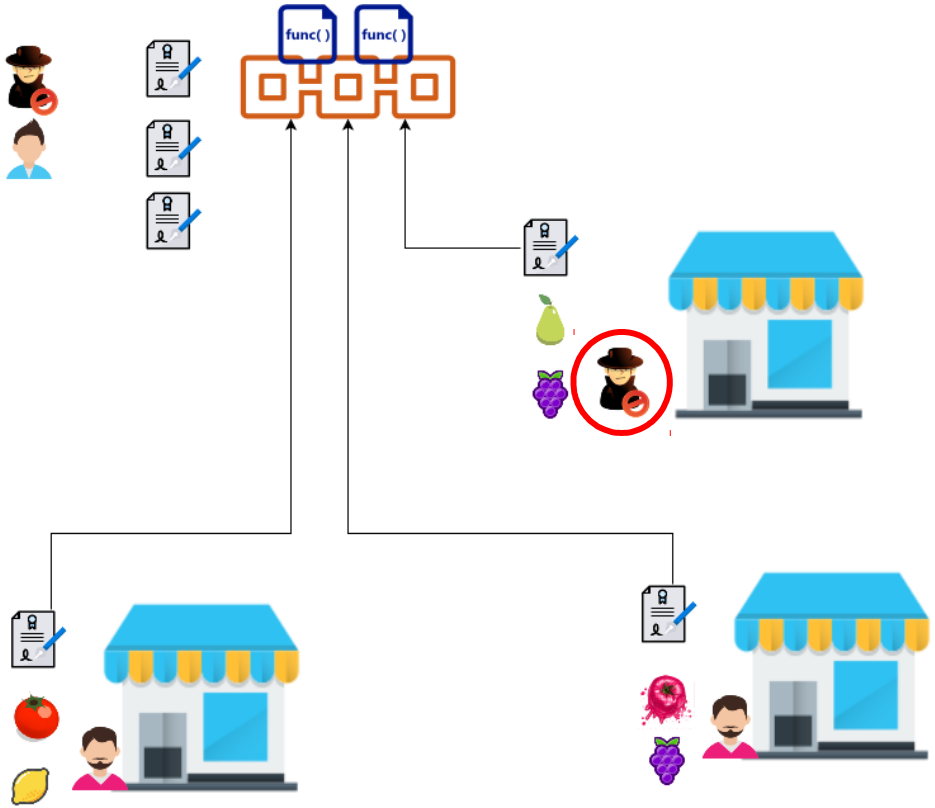
- **Consumers**
- Providers
- Products



Decentralized marketplace

What can go Wrong?

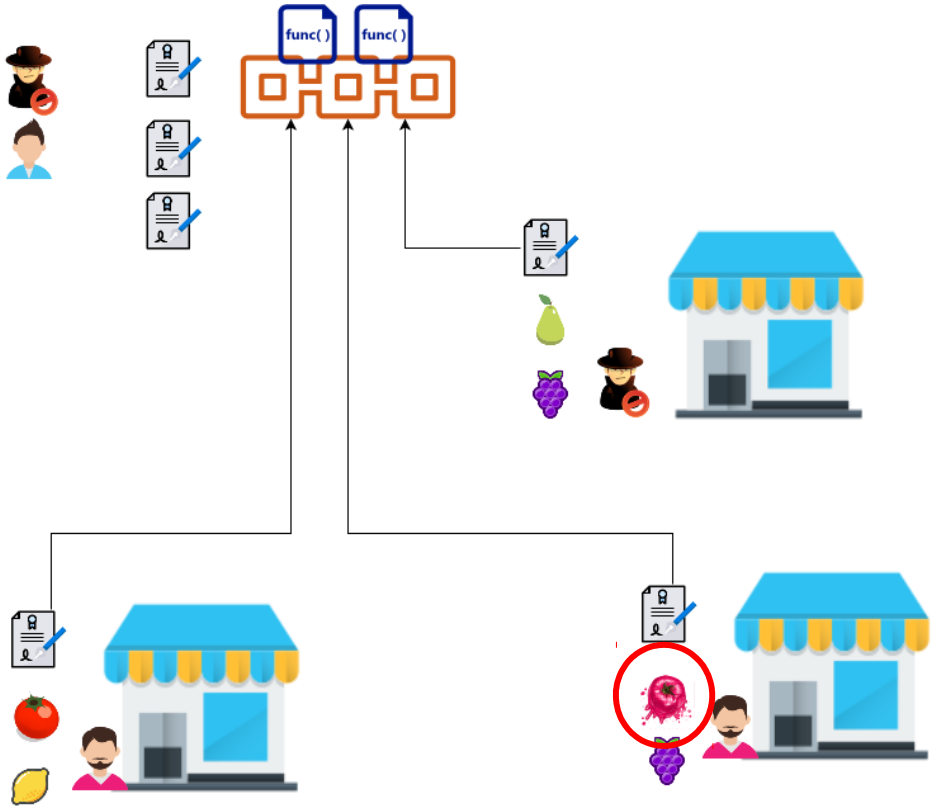
- Consumers
- **Providers**
- Products



Decentralized marketplace

What can go Wrong?

- Consumers
- Providers
- **Products**



The Problem

Managing Trust

Due to lack of centralized **mediator** and **enforcers**, entities cannot **trust** each other.

The Problem

Managing Trust

Due to lack of centralized **mediator** and **enforcers**, entities cannot **trust** each other.

Digital Trust

An assessment of a digital Entity in regards to qualities as **Competence**, **Security** and **Reliability**

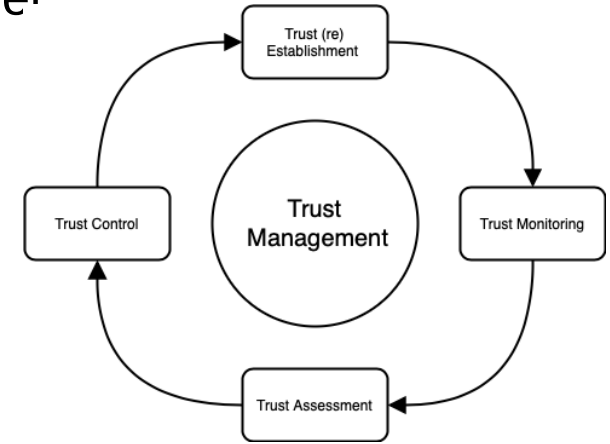
The Problem

Managing Trust

Due to lack of centralized **mediator** and **enforcers**, entities cannot **trust** each other

Digital Trust

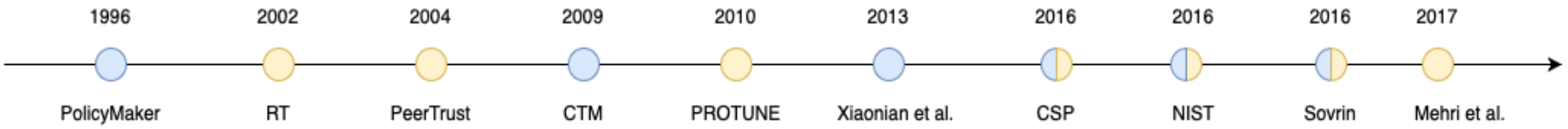
An assessment of a digital Entity in regards to qualities as **Competence**, **Security** and **Reliability**



Challenge

Enable blockchain to be the **trust anchor** for our decentralized marketplace

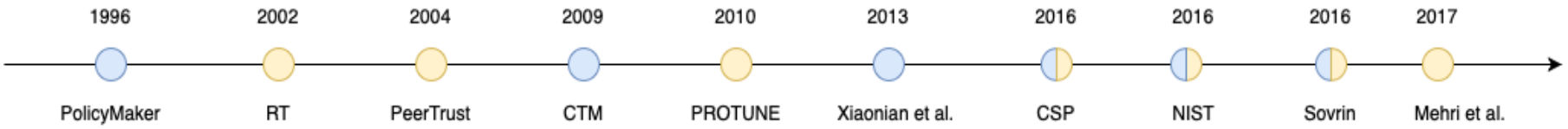
Previous solutions



Reputation-based trust management

Rule/SLA-based trust management

Previous solutions

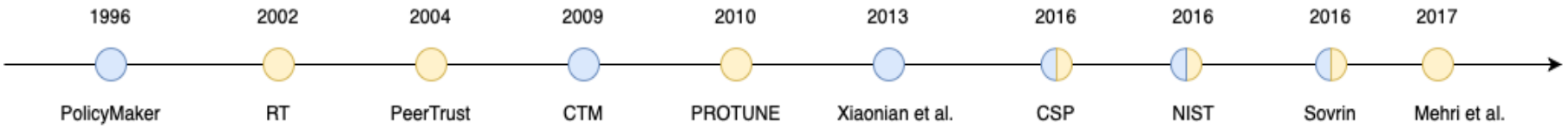


Reputation-based trust management

Rule/SLA-based trust management

relies on the ratings or feedback of peers in a decentralized system to evaluate trustworthiness of members in the system

Previous solutions

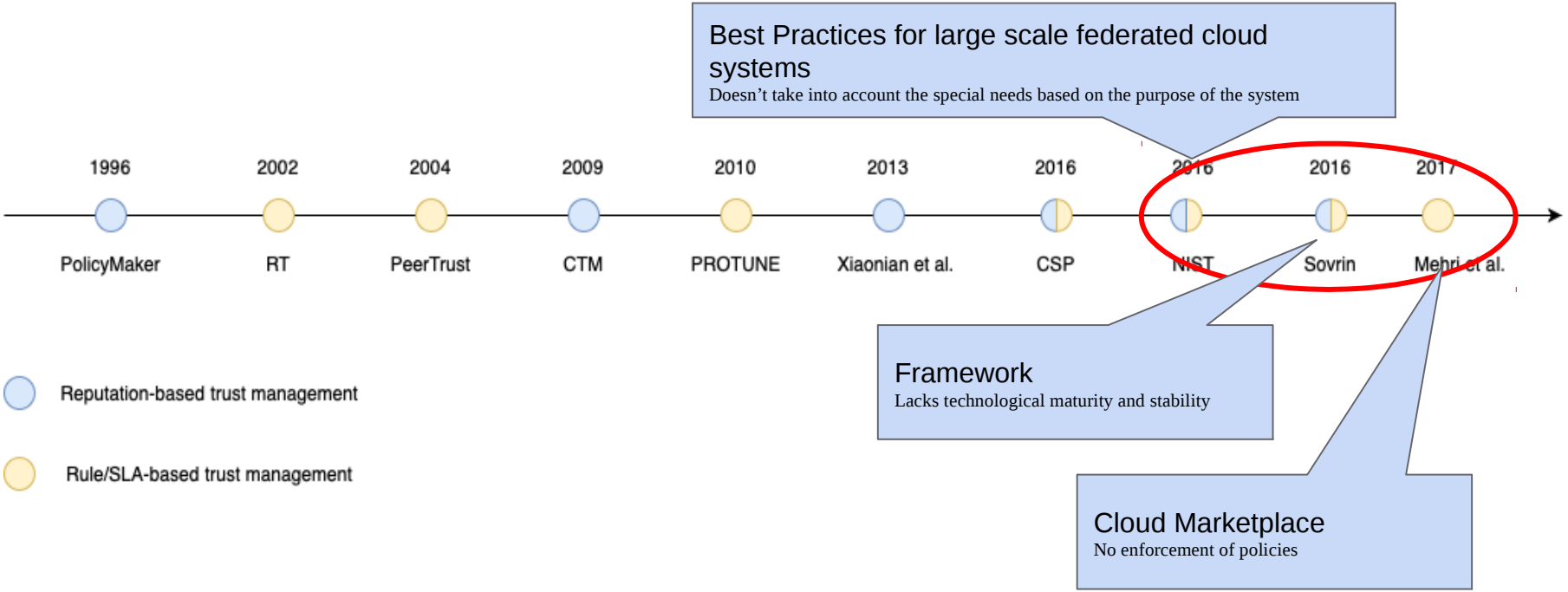


Reputation-based trust management

Rule/SLA-based trust management

relies on policies and credentials to decide what a member in the system can do

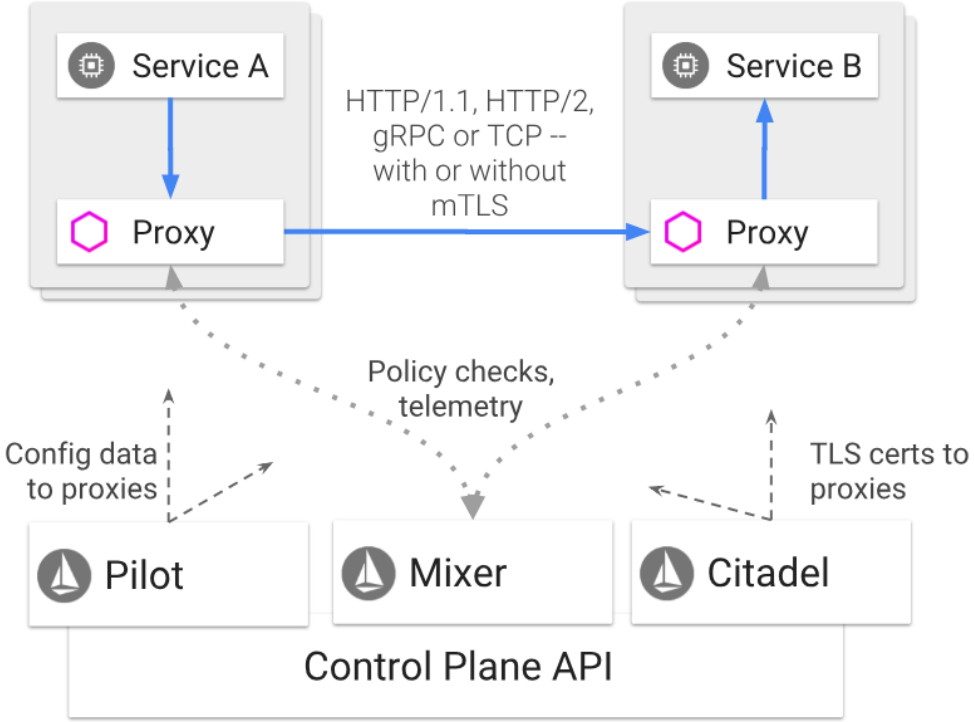
Previous solutions



Solution

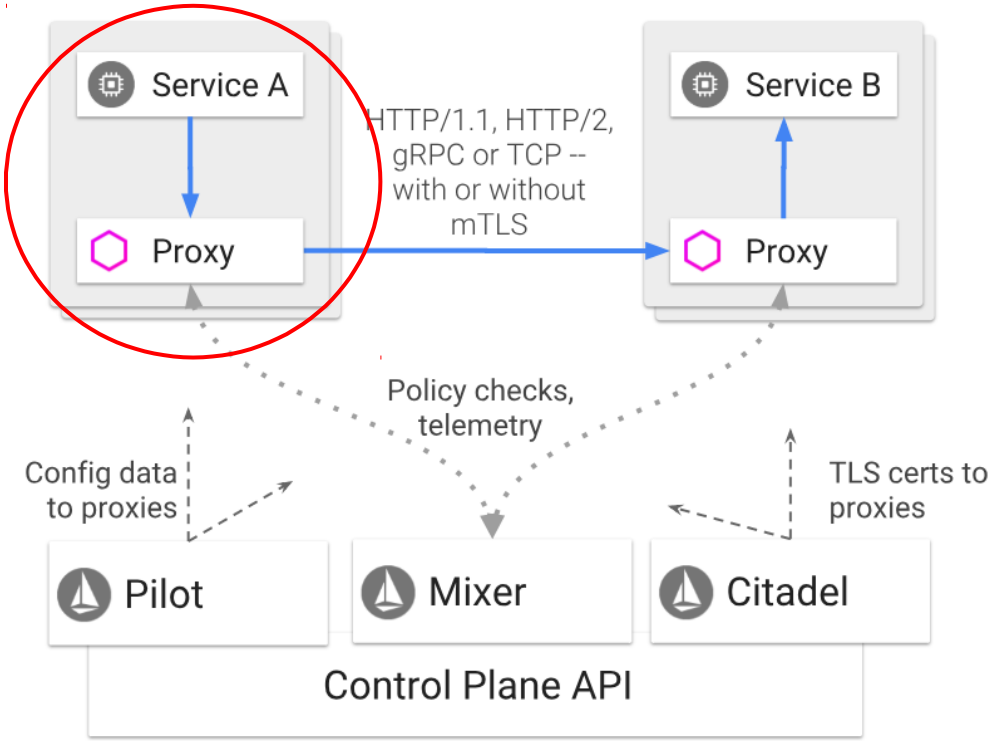
Leveraging **Service Mesh** and **Smart Contracts** to provide a **trusted operation** of the service exposure marketplace

Typical Service Mesh Architecture



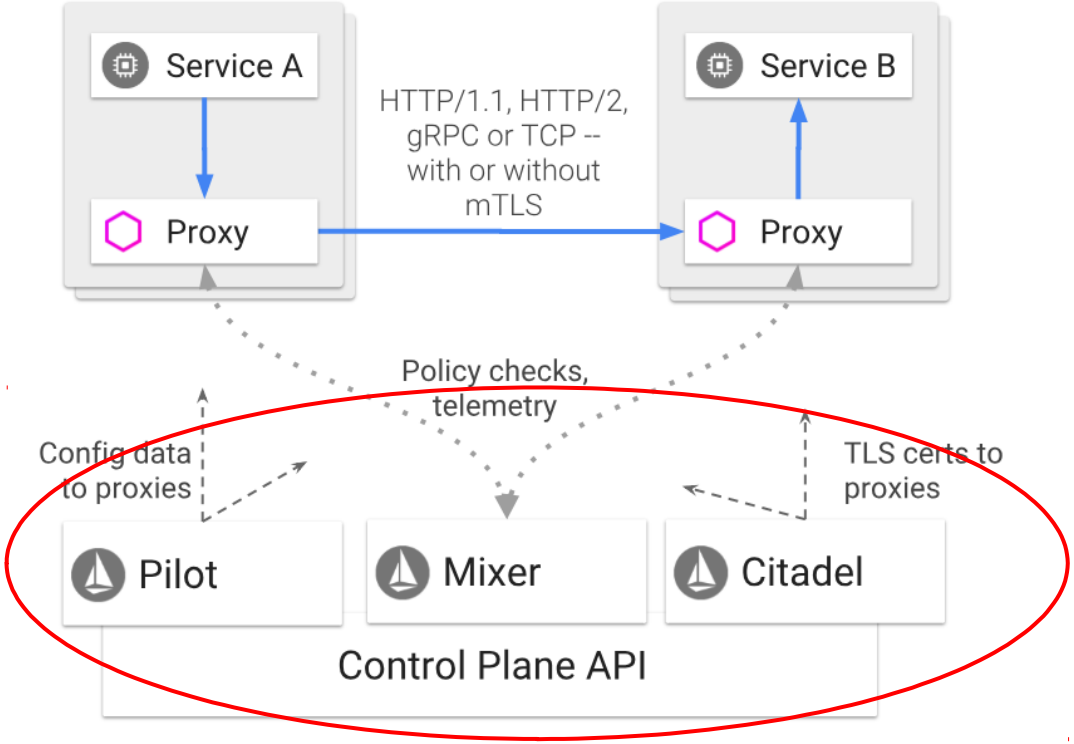
https://www.eclipse.org/community/eclipse_newsletter/2018/september/MicroProfile_istio.php

Typical Service Mesh Architecture



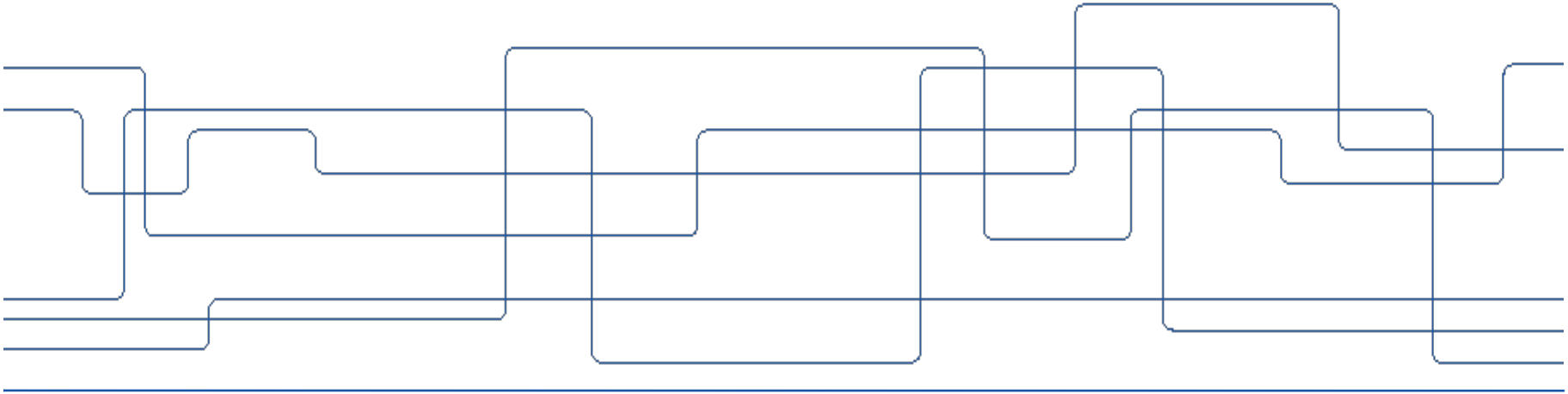
https://www.eclipse.org/community/eclipse_newsletter/2018/september/MicroProfile_istio.php

Typical Service Mesh Architecture

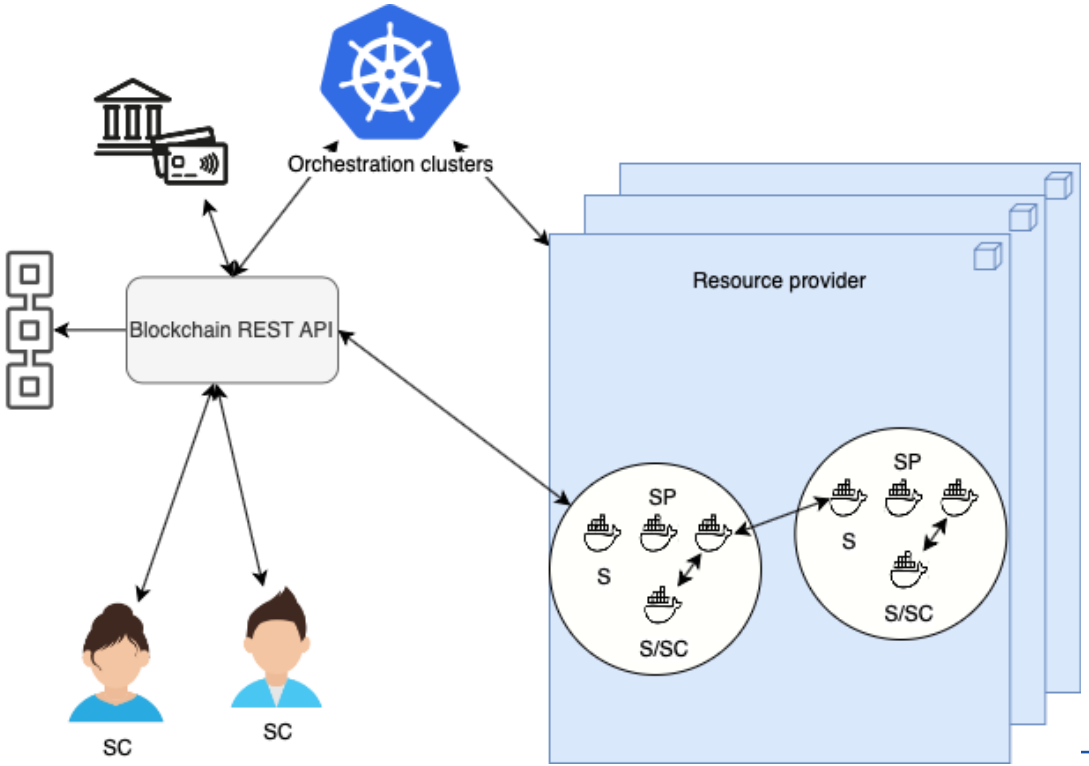


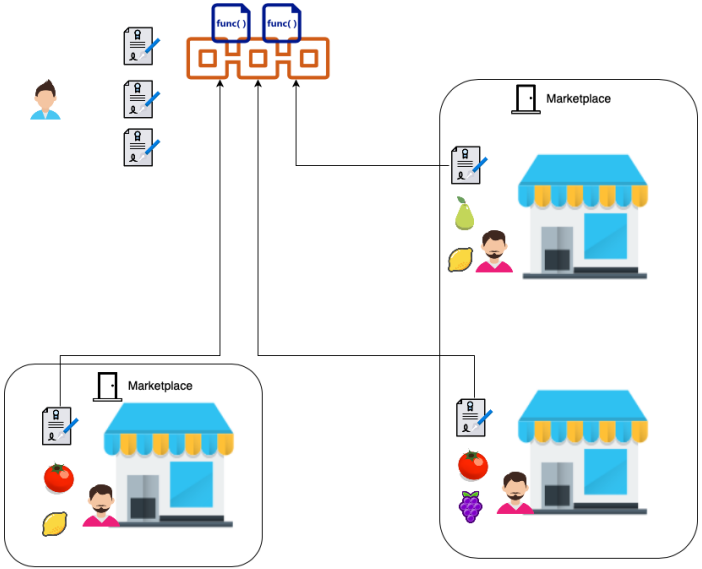
https://www.eclipse.org/community/eclipse_newsletter/2018/september/MicroProfile_istio.php

How is that good for us?



System Model





SP - Service provider

SC - Service consumer

S - Service

RP - Resource provider

Adversary model

Assumptions

Adversary's environment:
- Can be outside and inside the marketplace
- Highly skilled attackers

Adversary's resources:
- Access to configuration files
- Access to compromised servers
- Access to accounts

Goals

Disrupting SLA:
- use more resources than agreed without consequences

Bypassing authorization:
- access resources not intended for this level of authorization

Security properties violation:
- drop messages that are not intended for this user
- tampering messages

Disrupting of availability and resilience
- DOS
- delete config files

Capabilities

Delete configuration files, deployment files, or storage -> disrupt availability of system

Escalating the privilege -> get administrative access to the resources

Get SSH credentials via brute-force or phish -> access to SSH servers

Find leaked credentials available in open access -> access the servers

Use lateral movement tactic to stay inside the system -> learn the confidential information

https://www.microsoft.com/security/blog/2020/04/02/attack-matrix-kubernetes/

Adversary model

Assumptions

Adversary's environment:
- Can be outside and inside the marketplace
- Highly skilled attackers

Adversary's resources:
- Access to configuration files
- Access to compromised servers
- Access to accounts

Goals

Disrupting SLA:
- use more resources than agreed without consequences

Bypassing authorization:
- access resources not intended for this level of authorization

Security properties violation:
- drop messages that are not intended for this user
- tampering messages

Disrupting of availability and resilience
- DOS
- delete config files

Capabilities

Delete configuration files, deployment files, or storage -> disrupt availability of system

Escalating the privilege -> get administrative access to the resources

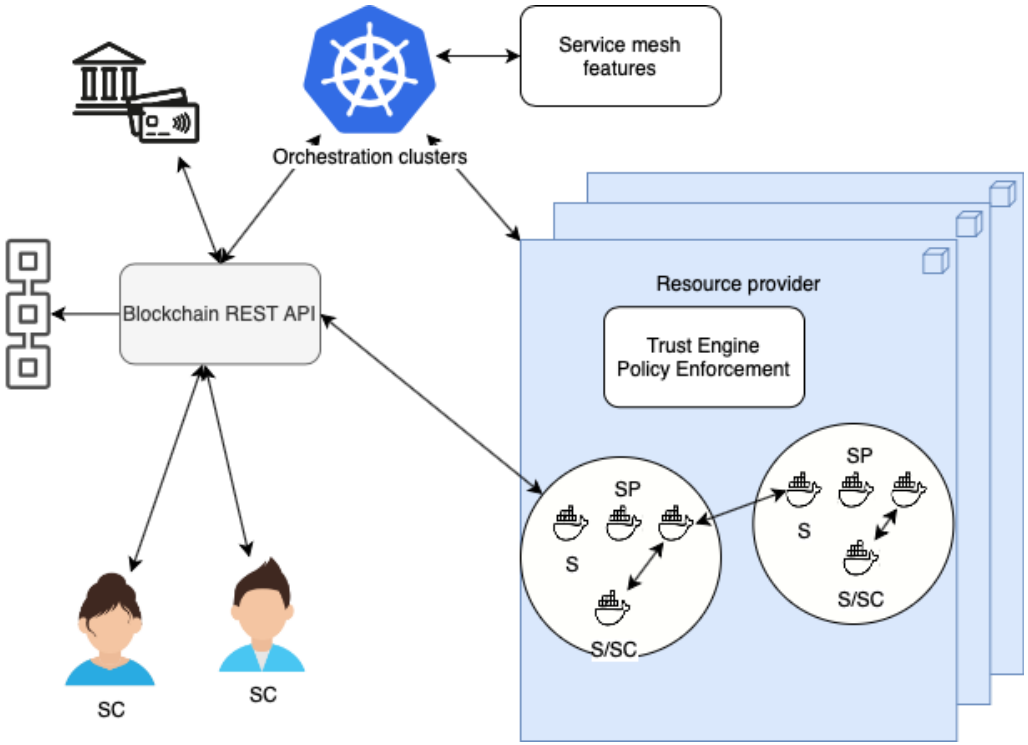
Get SSH credentials via brute-force or phish -> access to SSH servers

Find leaked credentials available in open access -> access the servers

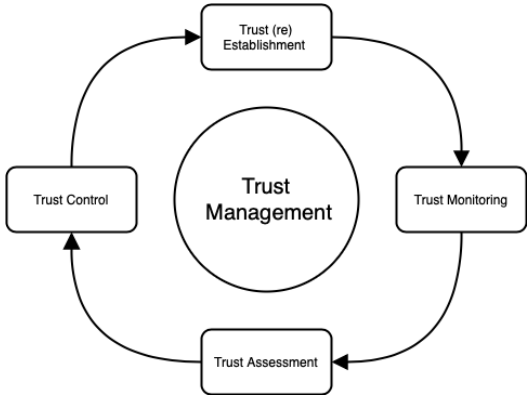
Use lateral movement tactic to stay inside the system -> learn the confidential information

https://www.microsoft.com/security/blog/2020/04/02/attack-matrix-kubernetes/

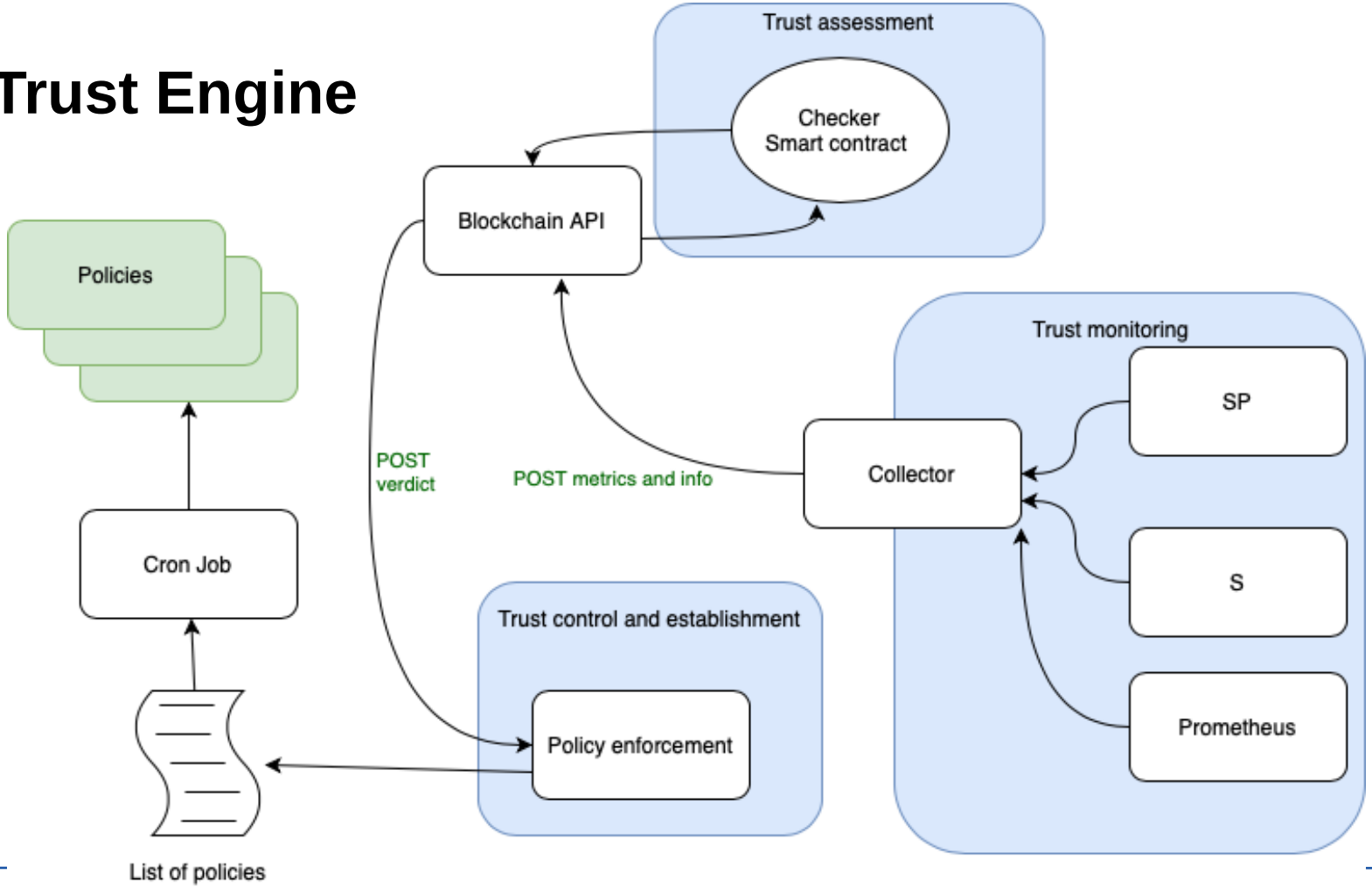
System Model with Service mesh



- Collect assessment metrics
- Create verdict/policy per relation
- Documenting results and metrics in blockchain
- Enforcing the policy in the running deployments



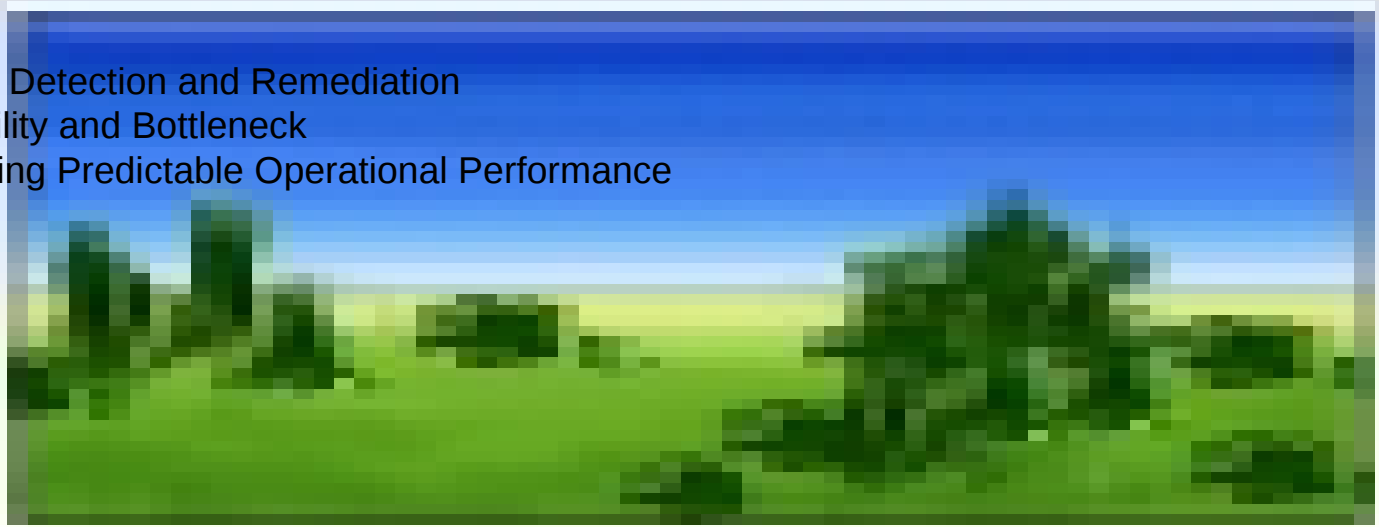
Trust Engine



Results and Analysis

Security and Performance Analysis

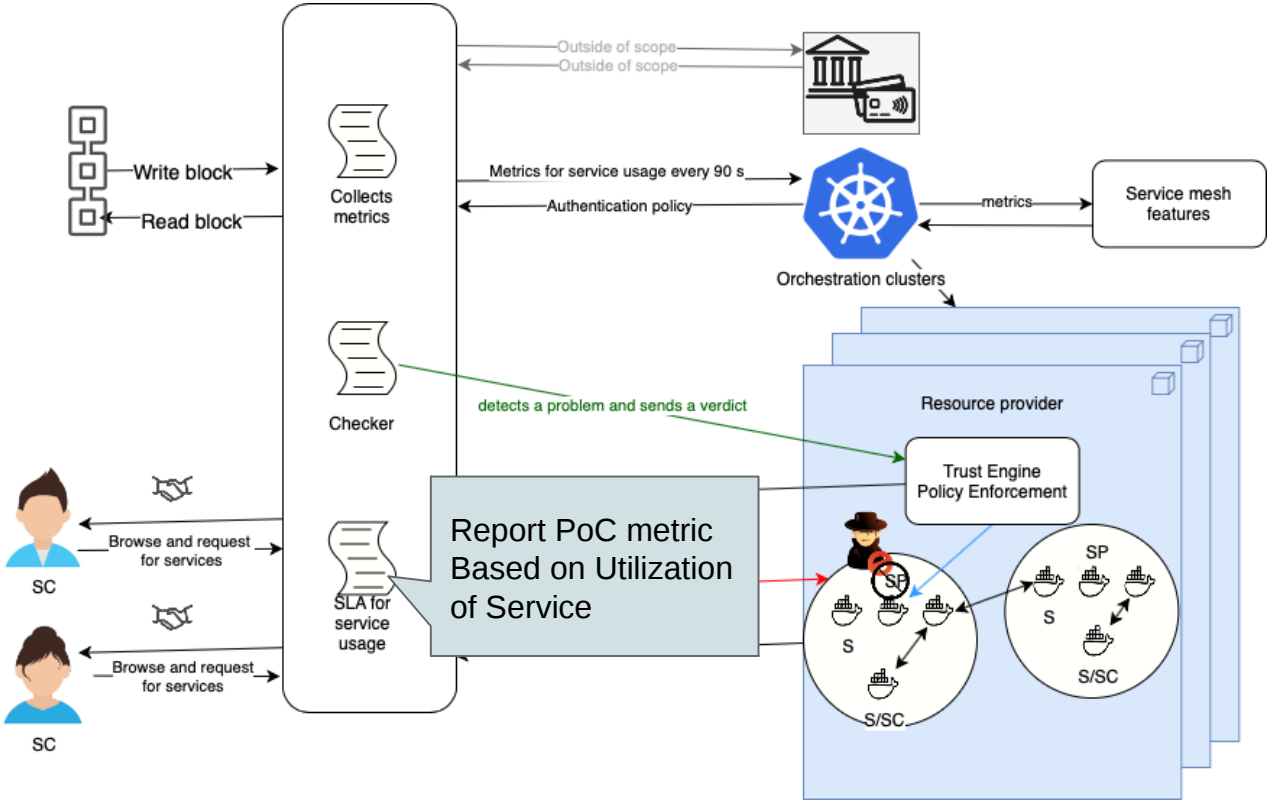
- Overall Detection and Remediation
- Scalability and Bottleneck
- Delivering Predictable Operational Performance



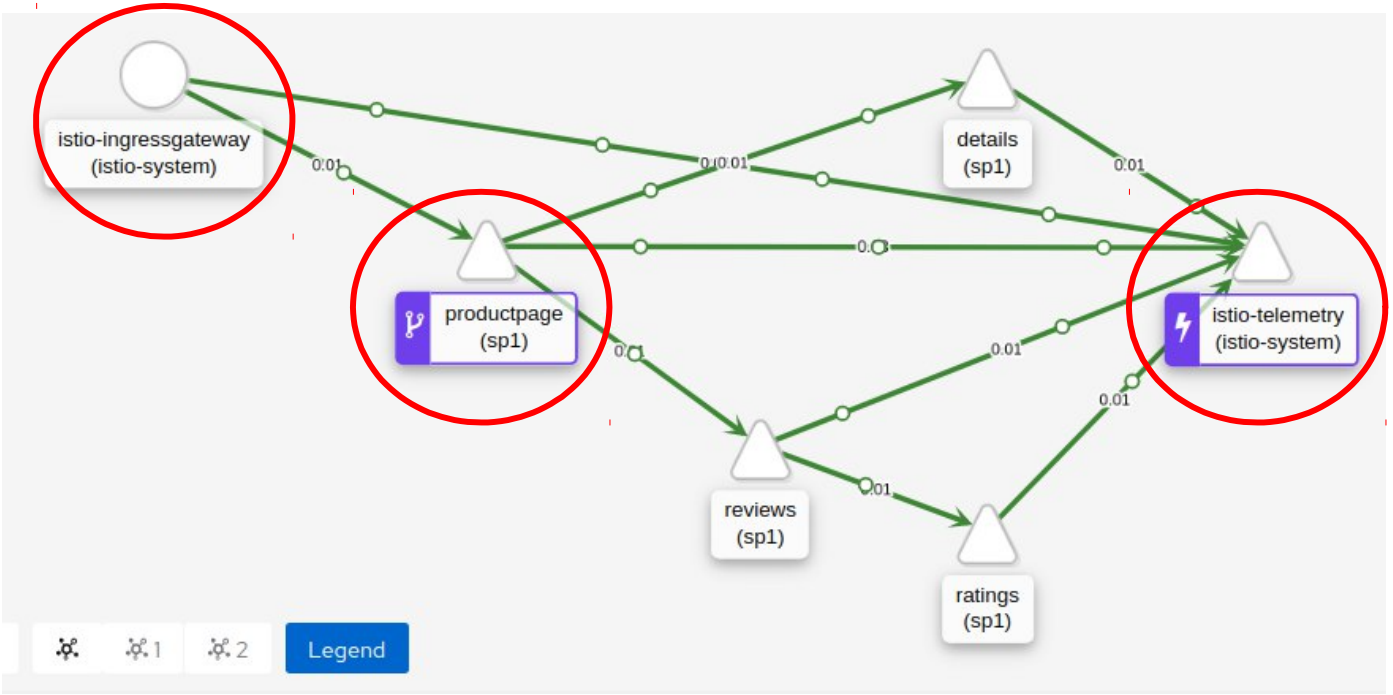
Component	Technology	Description
Service Mesh	Istio v1.4.3	Single mesh deployment with a single control plane and namespace tenancy model
Orchestration	Kubernetes on Minikube	Single cluster, with one master node
Virtual Machines	Openstack Nova	Debian based with 16GB of allocated ram and 4 processing cores
Blockchain	Hyperledger fabric	Single peer and single orderer

Table 4.3.1: Testbed specification overview

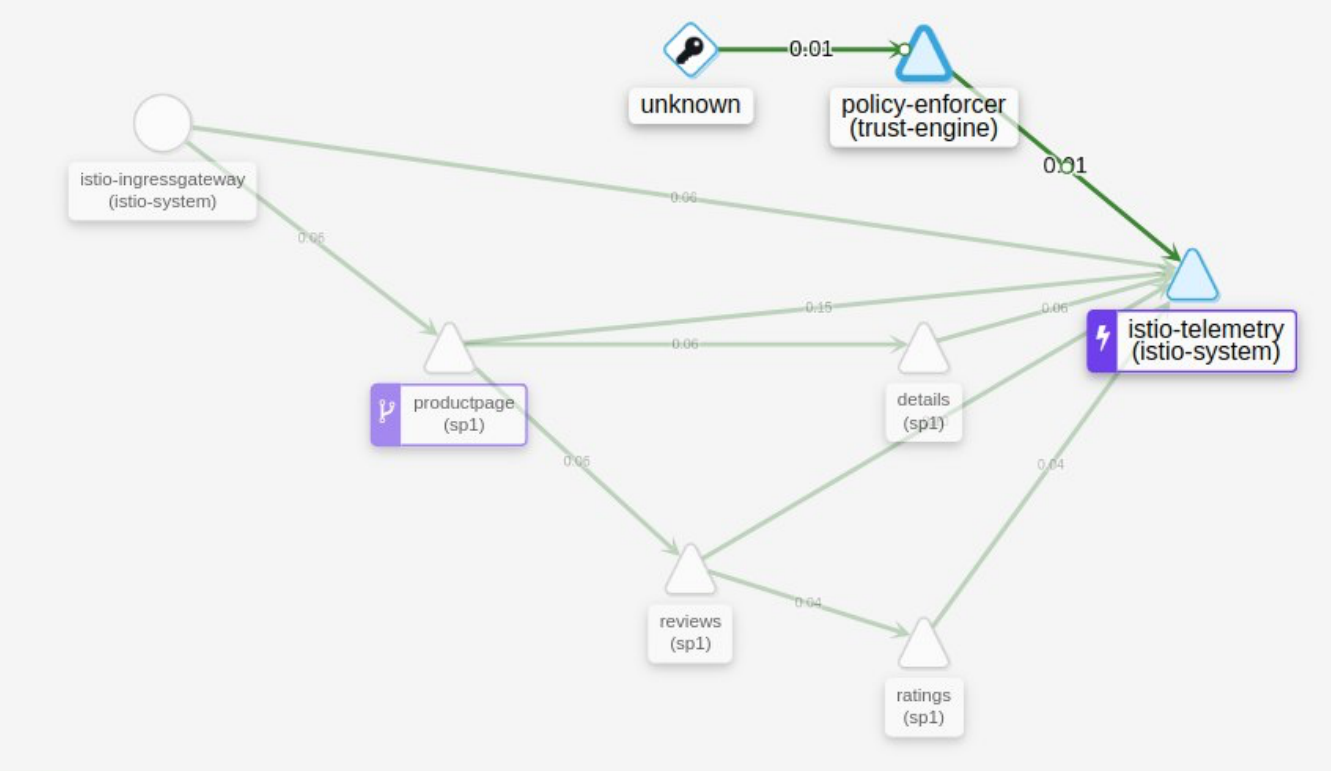
Overall Detection and Remediation - Attack Scenario



Overall Detection and Remediation

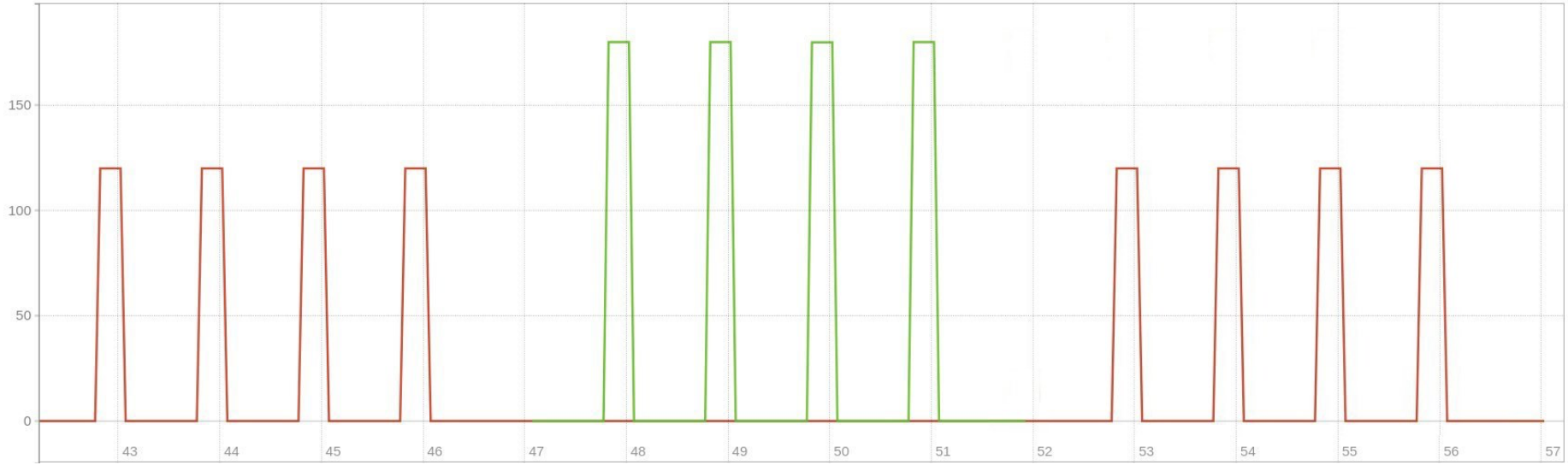


Overall Detection and Remediation

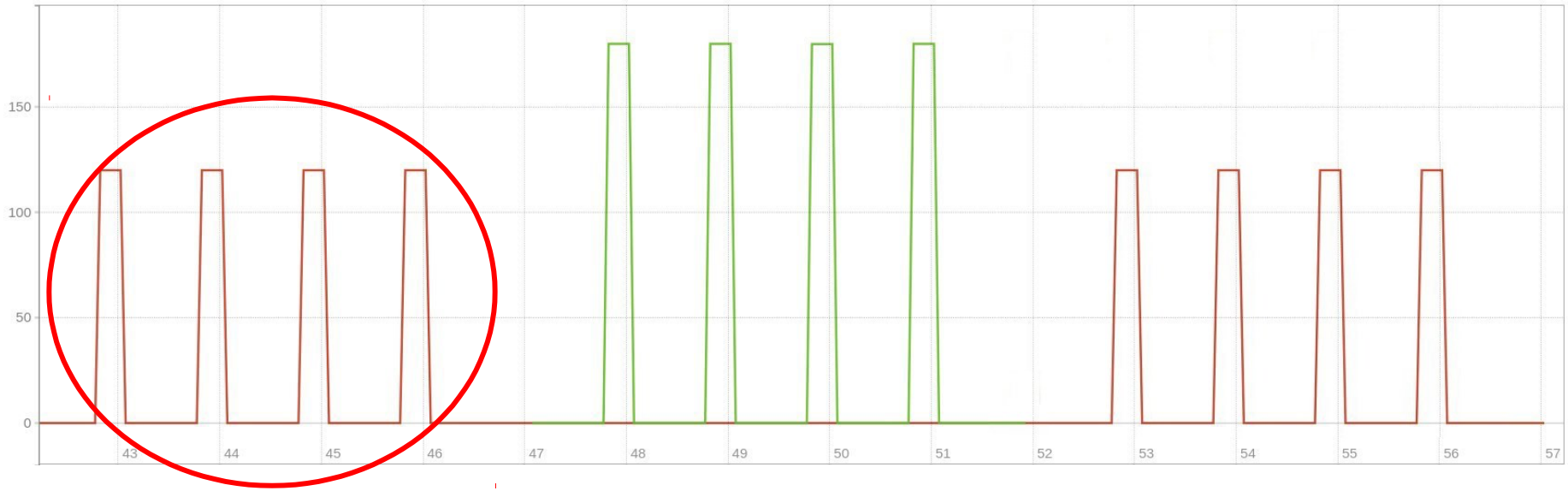


Overall Detection and Remediation

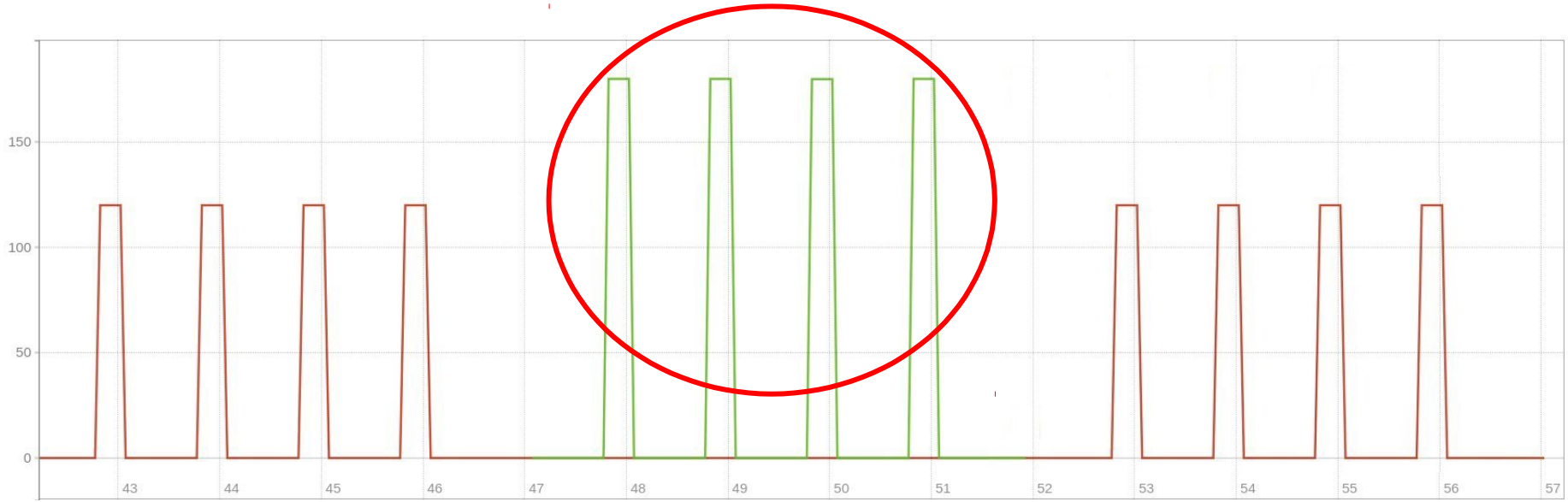
Reported PoC value over a 15 min time period pulled from Prometheus.



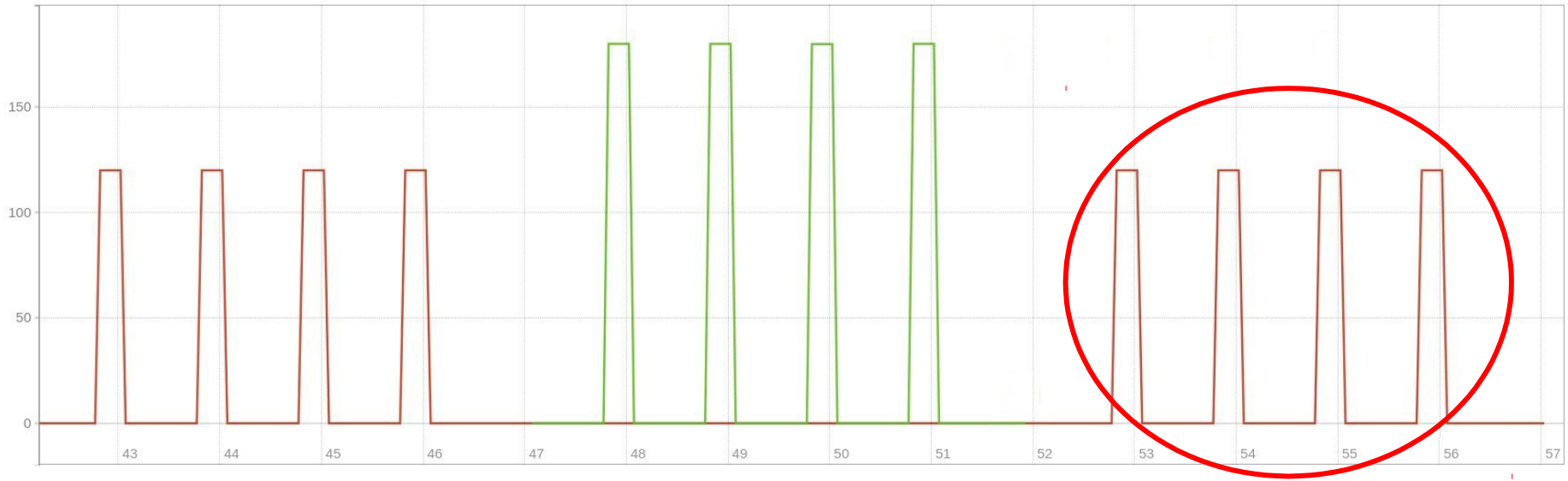
Overall Detection and Remediation



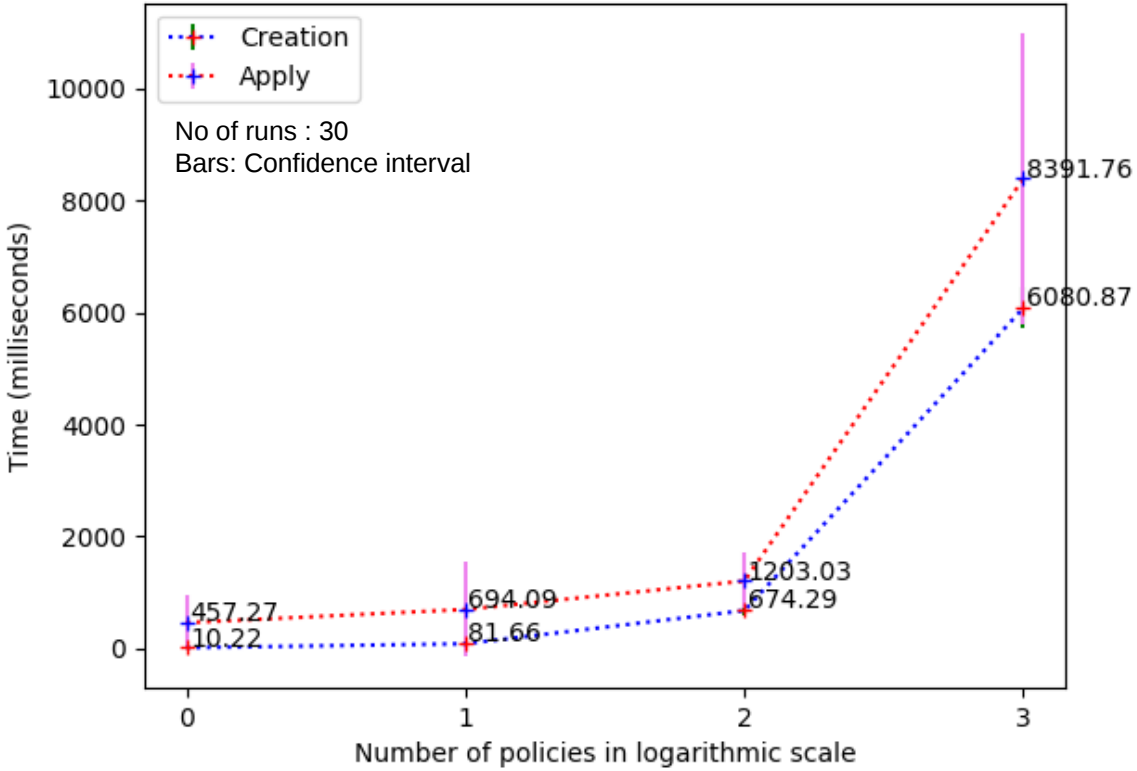
Overall Detection and Remediation



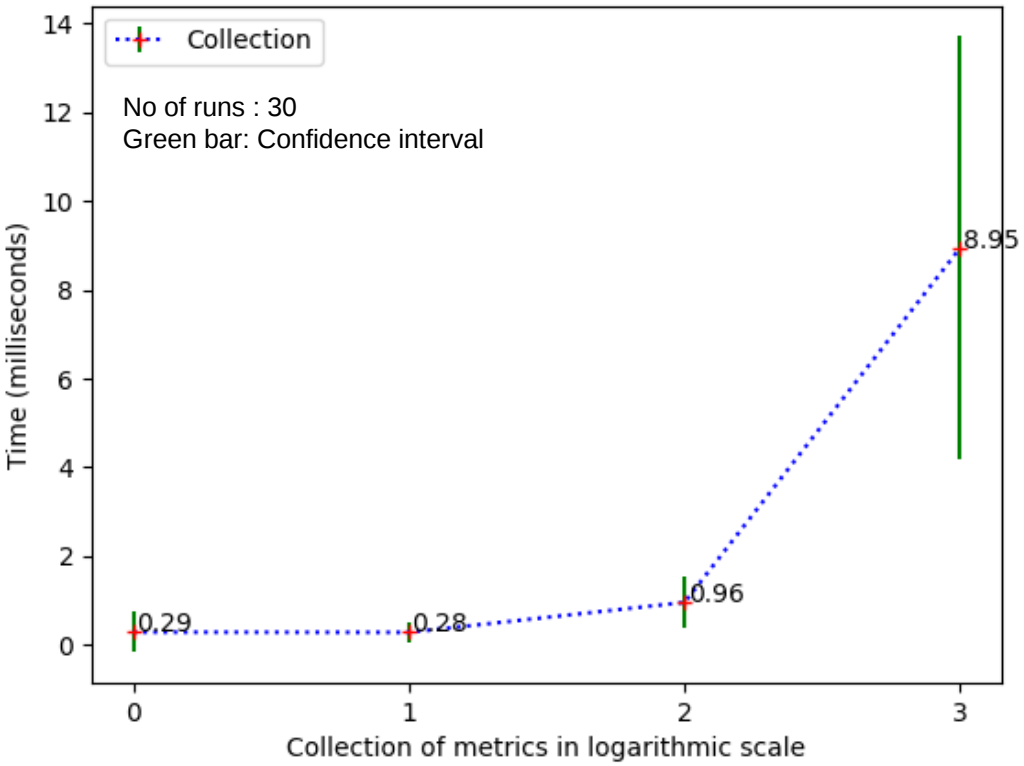
Overall Detection and Remediation



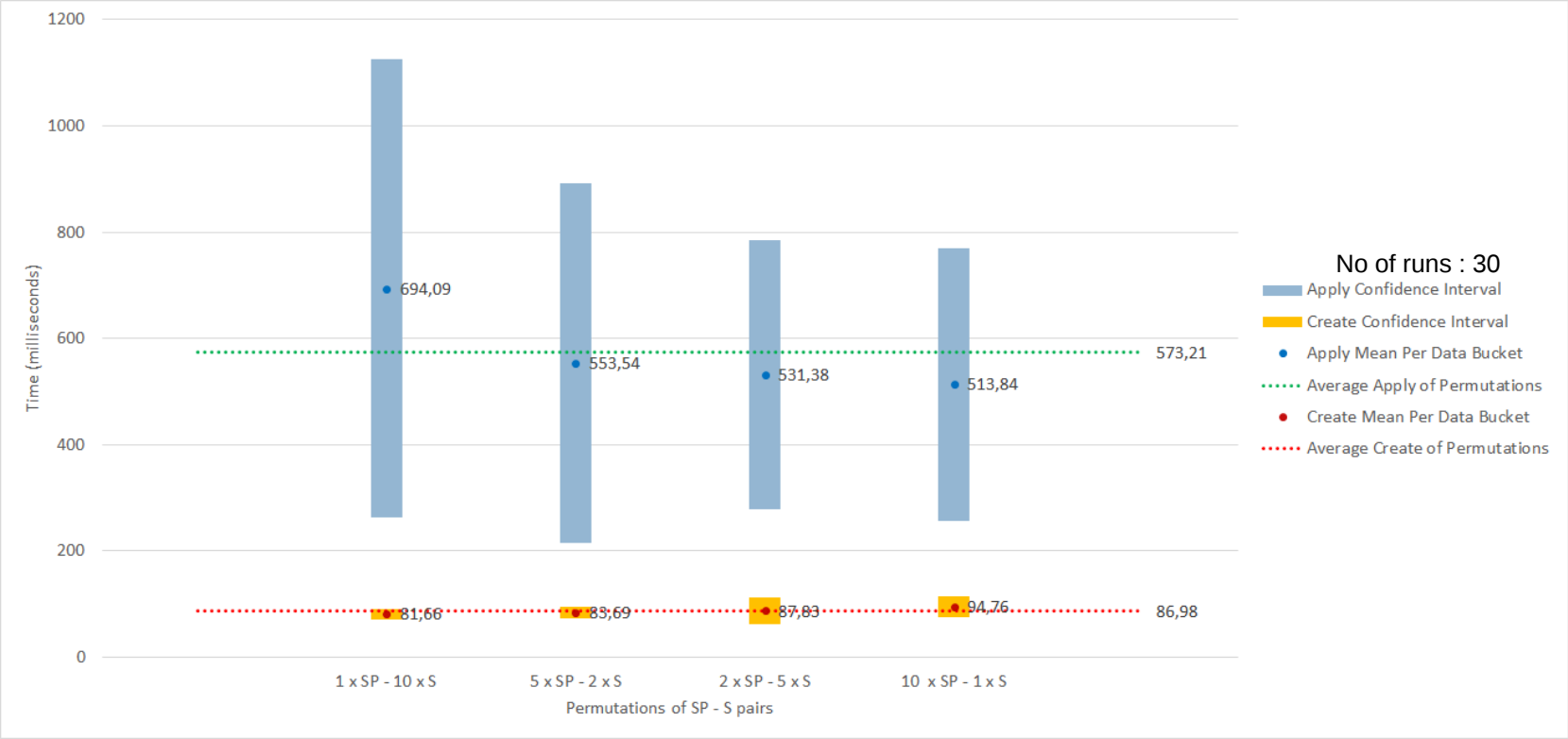
Scalability and Bottleneck - Policy Enforcement



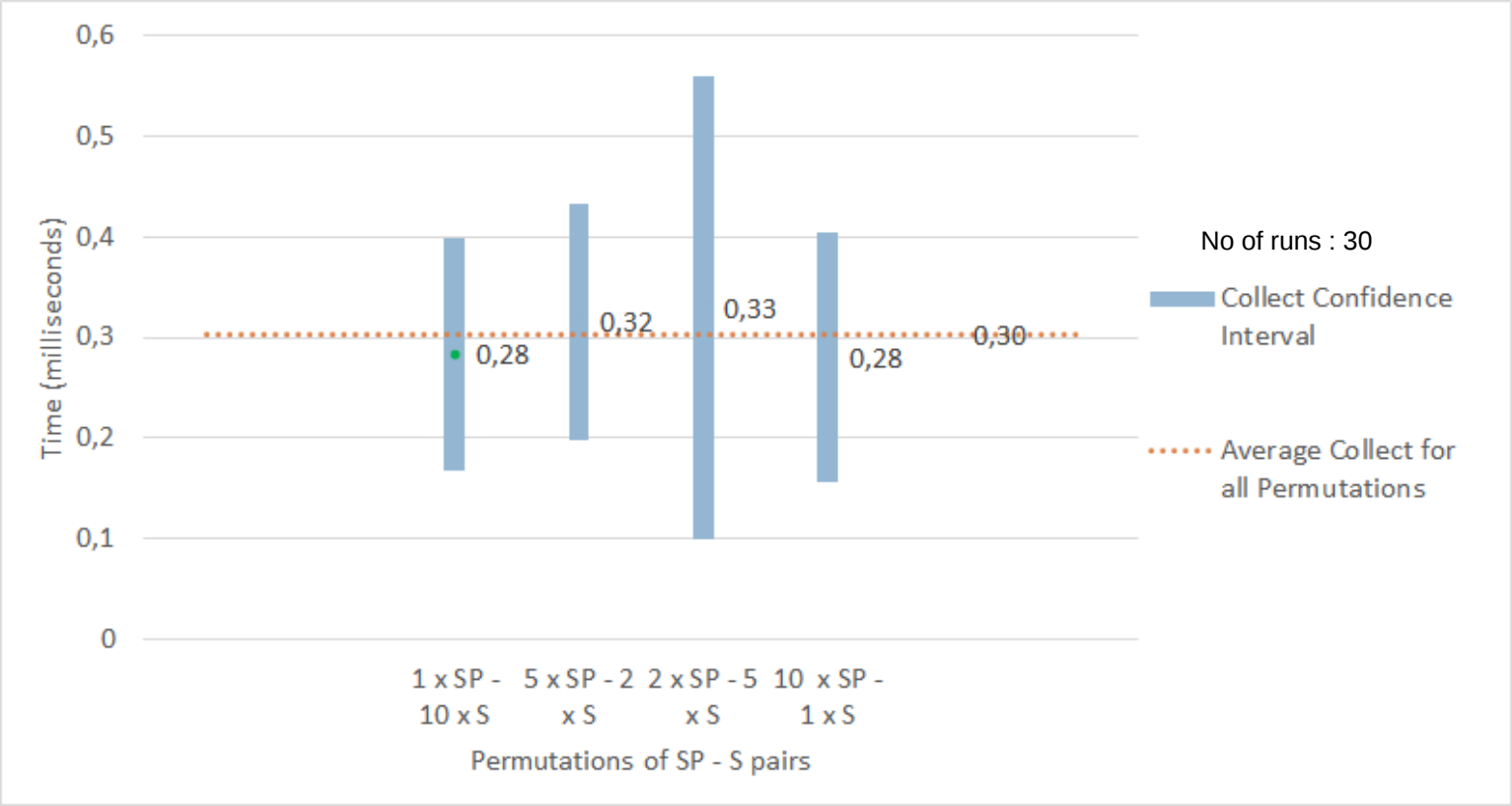
Scalability and Bottleneck - Collector



Predictable Operational Performance - Policy Enforcement



Predictable Operational Performance - Collector



Future work

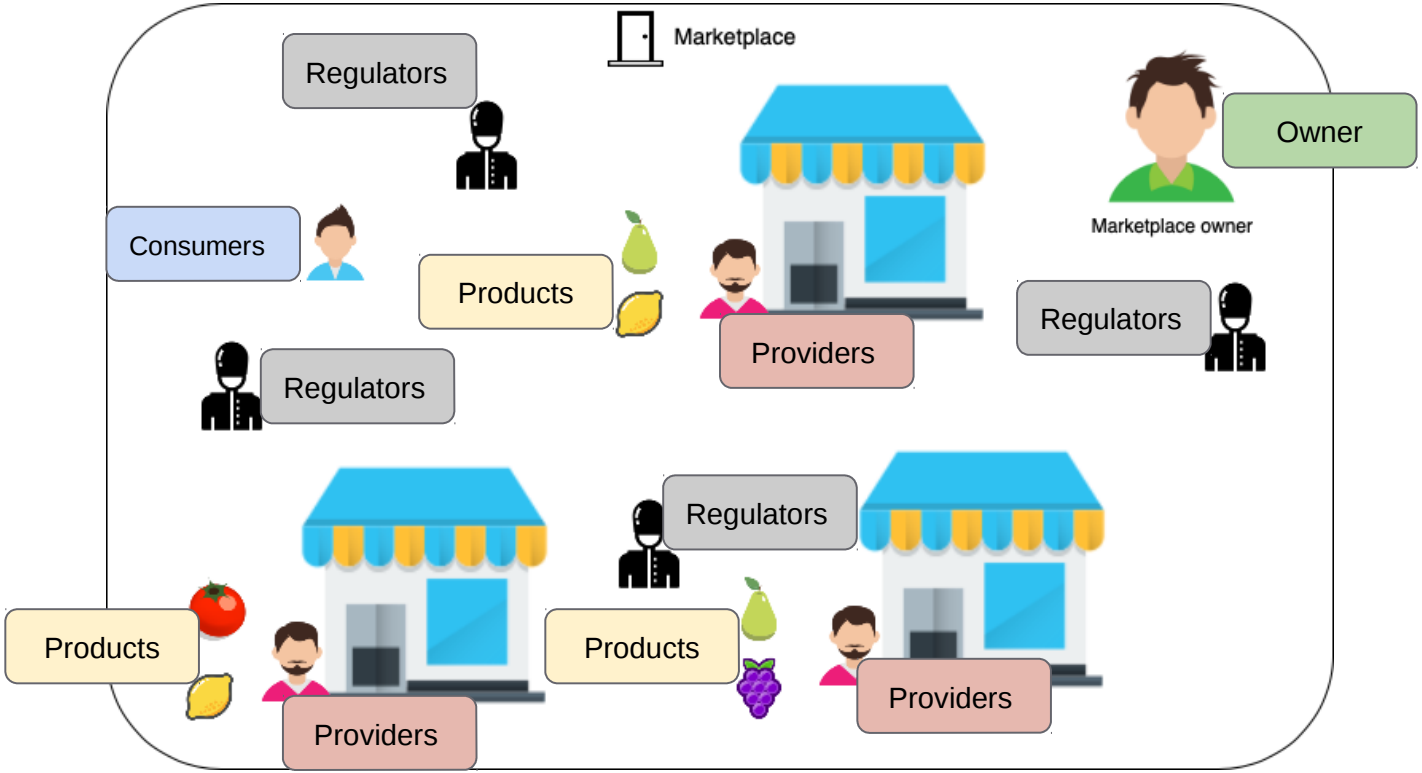
- Multi-mesh and Multi-cluster Istio Deployments Architectures
- Incentives and Deterrents for Blockchain Peers

Q&A

Ahmed Beder

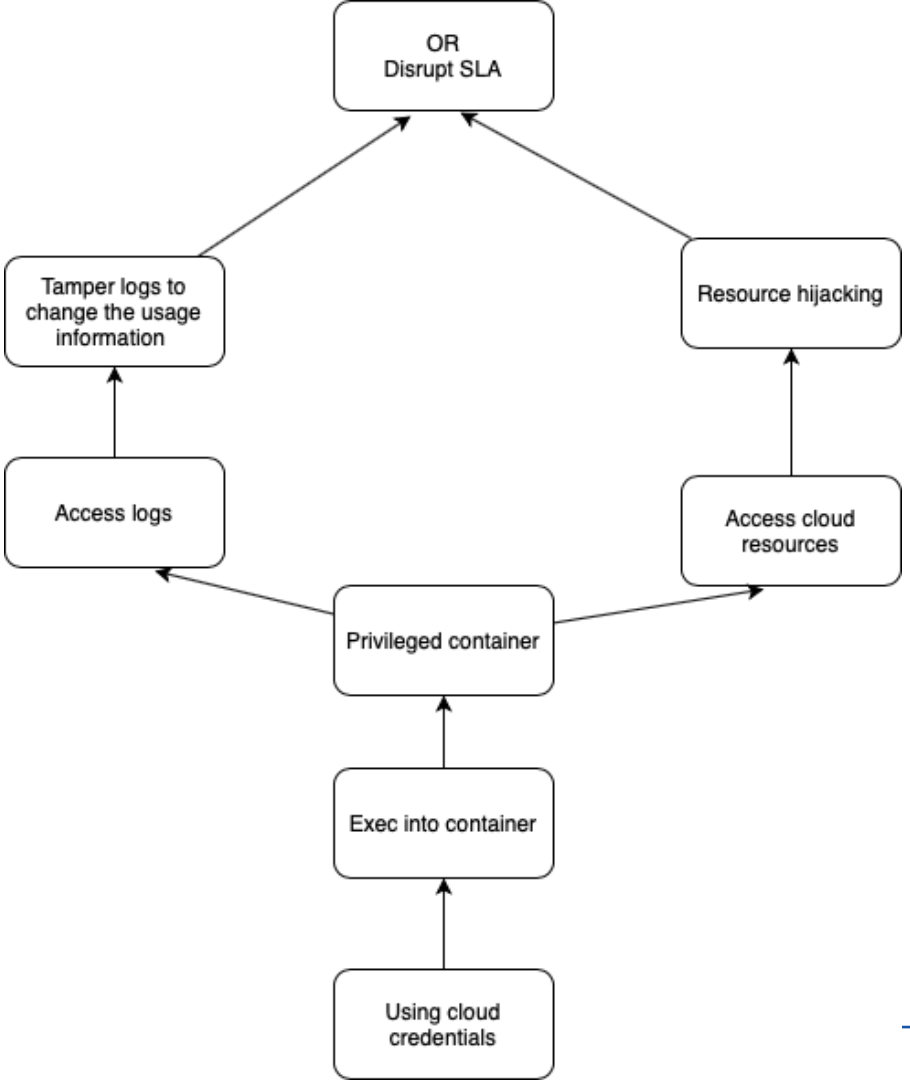
Appendix

Central marketplace

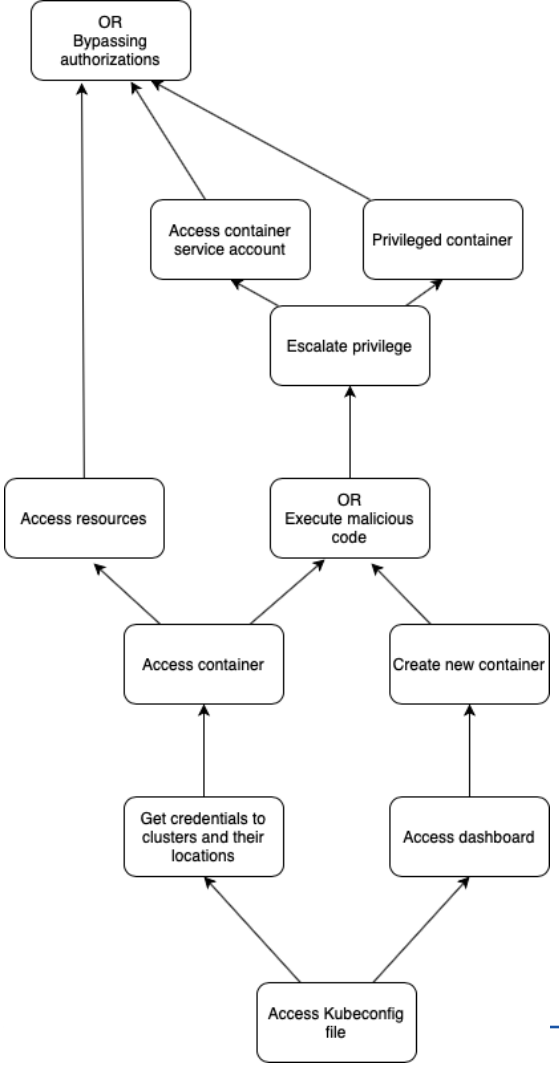


Policy Template

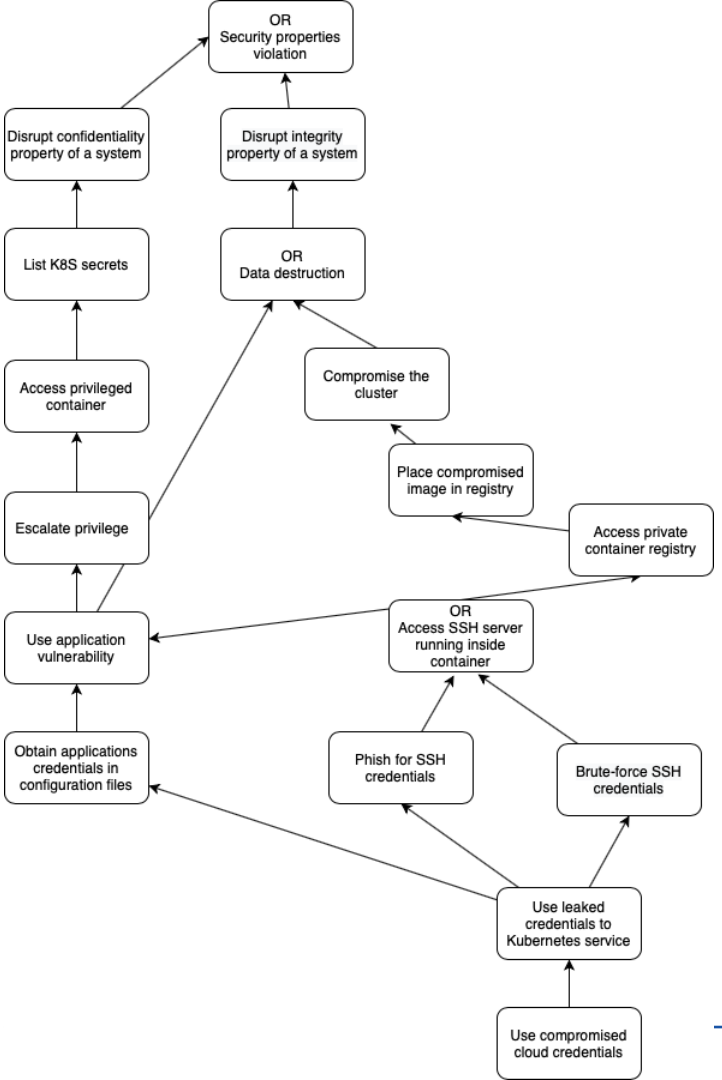
```yaml
apiVersion: config.istio.io/v1alpha2
kind: instance
metadata:
  name: pocmetric
  namespace: '{{SP_NAMESPACE}}'
spec:
  compiledTemplate: metric
  params:
    value: "90" # poc value
    dimensions:
      reporter:
        conditional((context.reporter.kind | "inbound") == "outbound", "client", "server")
      source: source.workload.name | "unknown"
      destination: destination.workload.name | "unknown"
      message: "agreed in sla"
      namespace: "{{SP_NAMESPACE}}"
    monitored_resource_type: '"UNSPECIFIED"'
```


OR
Disrupt SLA
Tamper logs to change the usage information
Resource hijacking
Access logs
Access cloud resources
Privileged container
Exec into container
Using cloud credentials

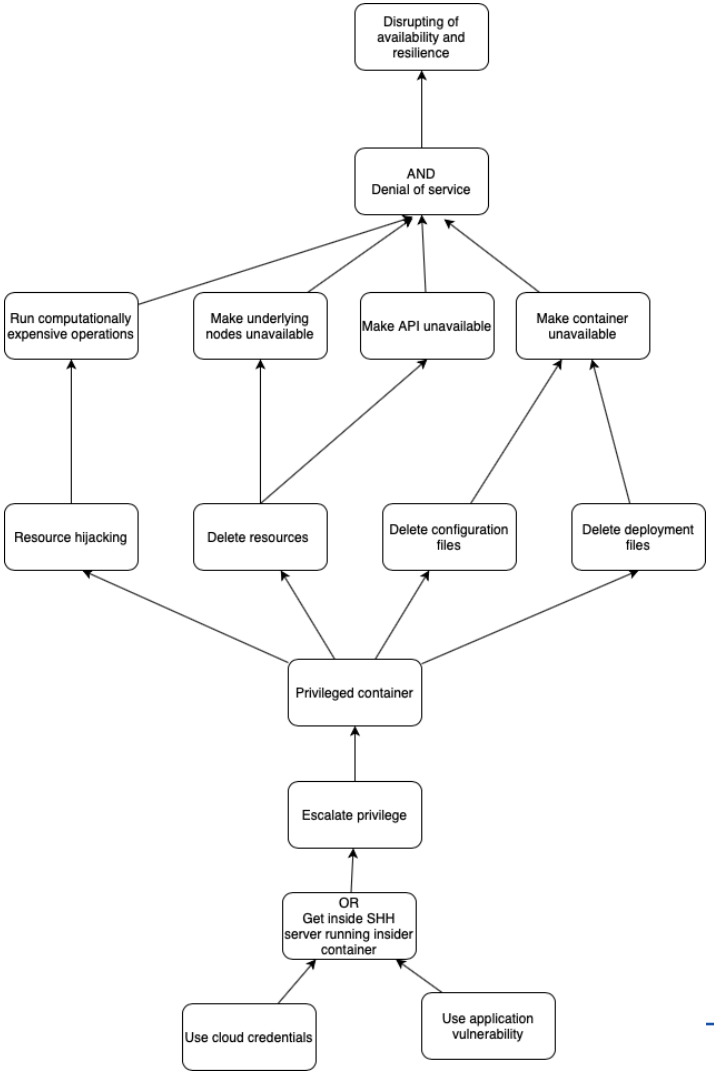
Disrupt SLA



Bypass authorizations

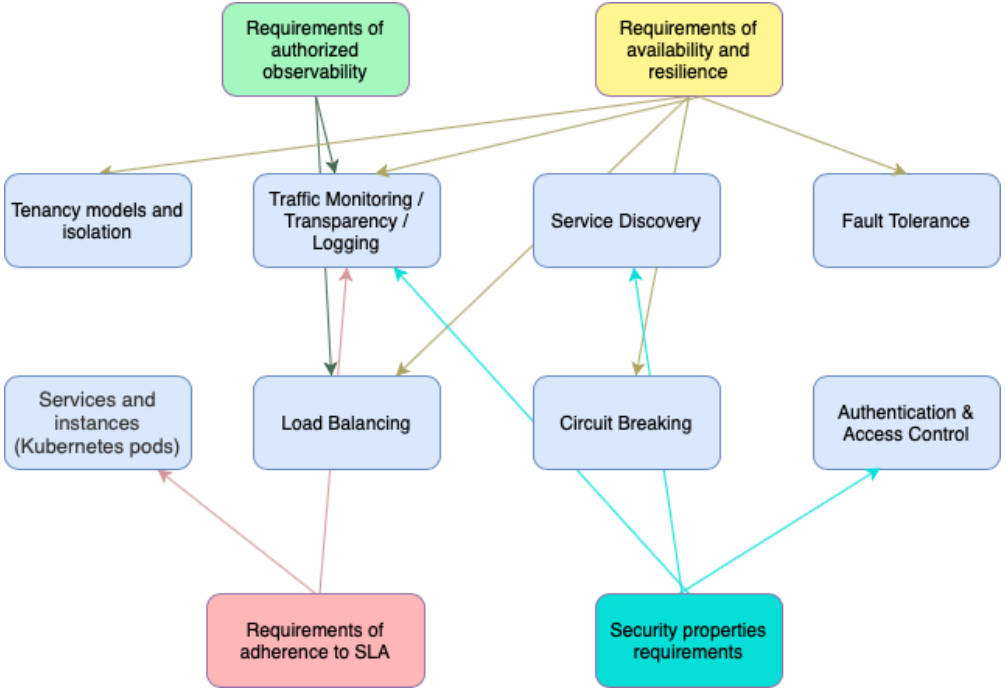


Security properties violation



Denial of availability

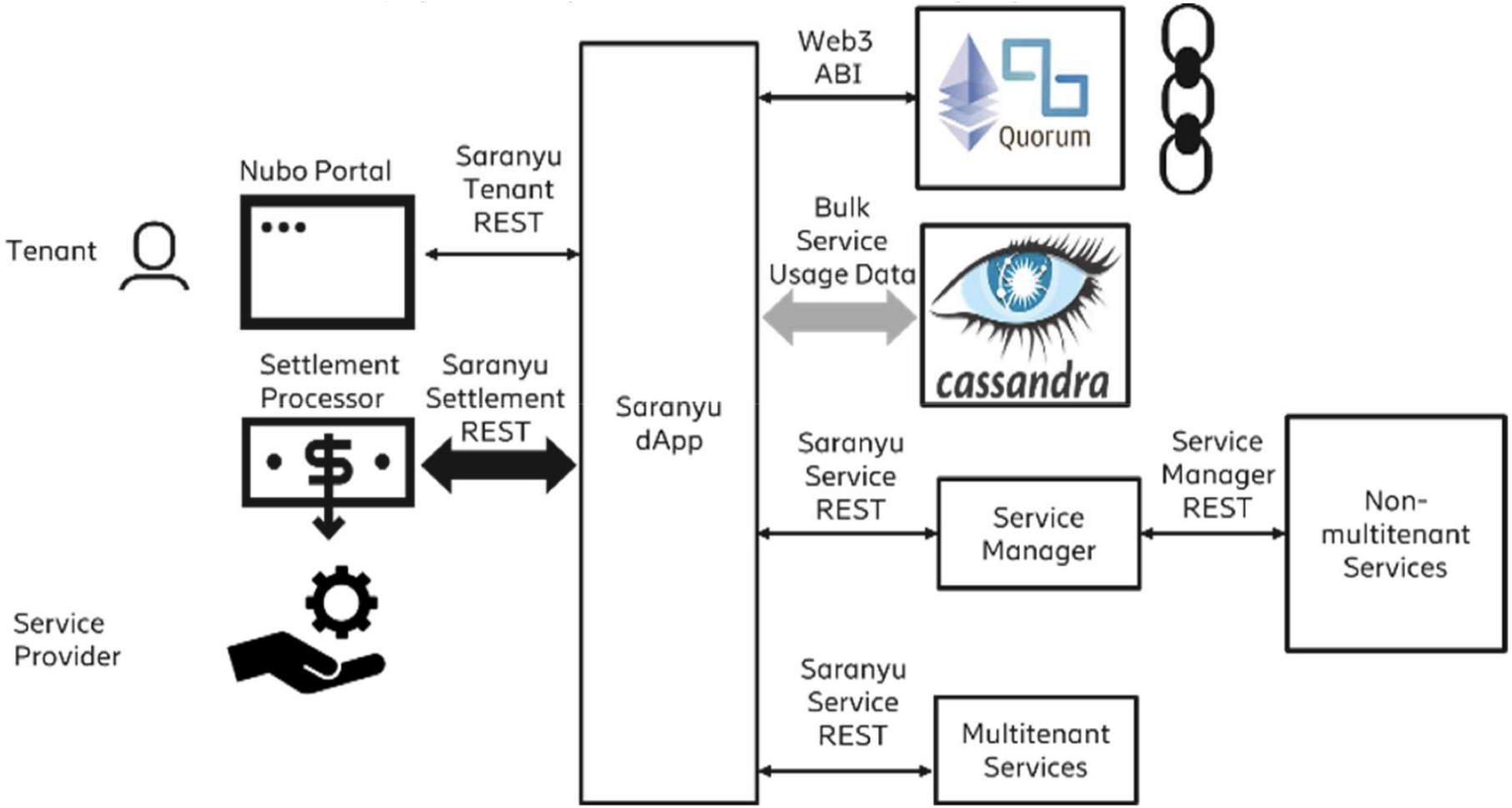
Mapping the requirements into service mesh features



Service Level Account

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: modify-policy
rules:
  - apiGroups: ["authentication.istio.io", "security.istio.io", "config.istio.io"]
    resources:
      - policies
      - authorizationpolicies
      - instances
    verbs:
      - get
      - list
      - delete
      - update
      - create
      - patch
      - watch
      - deletecollection
```

```yaml
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: modify-policy-to-sa
subjects:
  - kind: ServiceAccount
    name: internal-kubectl
    namespace: "trust-engine"
roleRef:
  kind: ClusterRole
  name: modify-policy
  apiGroup: rbac.authorization.k8s.io
---
```


Tenant
Nubo Portal
Saranyu Tenant REST
Settlement Processor
Saranyu Settlement REST
Service Provider
Saranyu dApp
Web3 ABI
Quorum
Bulk Service Usage Data
cassandra
Saranyu Service REST
Service Manager
Service Manager REST
Non-multitenant Services
Saranyu Service REST
Multitenant Services

Body Parameters:

```
{
  "prometheus_query_hash": <string>,
  "prometheus_response_hash": <string>
  "prometheus_metrics": [{
      "service": {
          "name": <string>
          "did": <string>
          "requests_sent": [{
            "destination_name":  <string>
            "destination_id": <string>
            "number_of_requests": int
          }],
          "requests_received":[{
            "sender_name":  <string>
            "sender_id": <string>
            "number_of_requests": int
          }],
        },
      }],
    "timestamp": <long>
    "nonce": <long>
  ...(other_metrics?)
}
```