Distributed Authorization System: A Netflix case study

Manish Mehta - Chief Security Architect @ Volterra
Torin Sandall - Co-founder of Open Policy Agent project
- Software Engineer @ Styra
Manish Mehta
Senior Security Engineer @ Netflix
Chief Security Architect @ Volterra
manish@ves.io

Projects:
• Bootstrapping Identities
• Secrets Management
• PKI
• Authentication
• Authorization

Torin Sandall
Co-founder of the OPA project
Software Engineer @ Styra
@sometorin
@OpenPolicyAgent

Projects:
• Open Policy Agent
• Kubernetes
• Istio (security SIG)
• Likes: Go, Quality, Good abstractions
1. Verify the Identity of the Requester (Authentication or AuthN)
2. Verify that the Requestor is authorized to perform the requested operation (Authorization or AuthZ)

These 2 steps do not need to be tied together!!
Background - Netflix Architecture

Netflix Backend - Internal Resources

Cloud Provider Resources

Partner Resources

CDN
Background - Netflix Architecture

Netflix Backend - Internal Resources

Cloud Provider Resources

Partner Resources

CDN
AuthZ Problem

A (simple) way to **define** and **enforce** rules that read

Identity $I$
can/cannot perform
Operation $O$
on
Resource $R$

For **ALL** combinations of $I$, $O$, and $R$ in the ecosystem.
Design Considerations

**Company Culture**
- Freedom and Responsibility

**Resource Types**
- REST endpoints, gRPC methods, SSH, Crypto Keys, Kafka Topics, …

**Identity Types**
- VM/Container Services, Batch Jobs, Employees, Contractors, …

**Underlying Protocols**
- HTTP(S), gRPC, Custom/Binary, …

**Implementation Languages**
- Java, Node JS, Python, Ruby, …

**Latency**
- Call depth and Service rate

**Flexibility of Rules**
- Hard-coded structure vs. language-based

**Capture Intent**
- Did you actually do what you think you did?
- Don’t just trust, verify !!
High-level Architecture
High-level Architecture
High-level Architecture

- Policy Portal
- Policy DB
- Aggregator
- Application Ownership DB
- Employee Management System
- Build Manifest
- Service A
  - AuthZ Agent
  - SSH
  - App Code
- Service B
  - AuthZ Agent
  - App Code
High-level Architecture
High-level Architecture

- Policy Portal
- Policy DB
- Aggregator
- Application Ownership DB
- Employee Management System
- Build Manifest
- AuthZ Agent
- Service A
- App Code
- SSH
- Distributor
- Service B
- AuthZ Agent
- App Code

Velocity San Jose '18
High-level Architecture

Policy Portal

Policy DB

Aggregator

Application Ownership DB

Employee Management System

Build Manifest

Service A

AuthZ Agent

SSH

App Code

Distributor

Service B

AuthZ Agent

App Code

Service A

Policy Portal

Policy DB

Aggregator

Application Ownership DB

Employee Management System

Build Manifest

Service B

AuthZ Agent

App Code

Service A

Policy Portal

Policy DB

Aggregator

Application Ownership DB

Employee Management System

Build Manifest

Service B

AuthZ Agent

App Code
AuthZ Agent Internals

AuthZ Agent

Open Policy Agent Engine

Request -> API -> Stager -> Updater

Decision -> Updater

Periodic updates on policies and associated data
Example Setup

Authorization Policy

1. Employees can read their own salary and the salary of anyone who reports to them.

2. Report Generator Job should be able to Read all users' salaries

3. Performance Review Application should be able to update all users' salaries
Open Policy Agent
What about RBAC?
RBAC solves XX% of the problem.
RBAC is not enough.

"Allow all HTTP requests from 10.1.2.0/24."

"Restrict employees from accessing the service outside of work hours."

"QA must sign-off on images deployed to the production namespace."

"Restrict ELB changes to senior SREs that are on-call."

"Analysts can read client data but PII must be redacted."

"Prevent developers from running containers with privileged security contexts in the production namespace."

"Give developers SSH access to machines listed in JIRA tickets assigned to them."

"Workloads for euro-bank must be deployed on PCI-certified clusters in the EU."
OPA is a general-purpose policy engine.
Decisions are decoupled from enforcement.
Evaluate policies locally.

- Daemon (HTTP API)
- Library (Go)
- Service Mesh (Istio)
Fate Sharing
✔ Low latency
✔ High availability

Host Failures

Network Partitions

Node
Service
OPA

Node
Service
OPA

Node
Service

Node
OPA

Node
OPA
Policy and data are stored in-memory.

No external dependencies during enforcement.
Declarative Language (Rego)

- Is Identity I allowed to perform Operation O on Resource R?
- What labels must be applied to Deployment X?
- Which users can SSH into production servers?
"Employees can read their own salaries and the salaries of their subordinates."
"Employees can read their own salaries [...]"
'Employees can read their own salaries [...]'

Input

```json
{"method": "GET", "path": ["salaries", "bob"], "user": "bob"}
```
"Employees can read their own salaries [...]"

allow = true { 
  input.method = "GET"
  input.path = ["salaries", employee_id]
  input.user = employee_id
}

Input
{
  "method": "GET",
  "path": ["salaries", "bob"],
  "user": "bob"}
*Employees can read their own salaries [...]*

```javascript
allow = true {
    input.method = "GET"
    input.path = ["salaries", "bob"]
    input.user = "bob"
}
```

Input

```json
{"method": "GET",
"path": ["salaries", "bob"],
"user": "bob"}
```
"Employees can read their own salaries [...]"

allow = true { 
  input.method = "GET" # OK
  input.path = ["salaries", "bob"] # OK
  input.user = "bob" # OK
}

Input
{"method": "GET", "path": ["salaries", "bob"], "user": "bob"}
allow = true {
    input.method = "GET"
    input.path = ["salaries", employee_id]
    input.user = employee_id
}
"Employees can read their own salaries [...]"

allow = true {
    input.method = "GET" # OK
    input.path = ["salaries", "bob"] # OK
    "alice" = "bob" # FAIL
}
"Employees can read [...] the salaries of their subordinates."

allow = true {
    input.method = "GET"  # OK
    input.path = ["salaries", "bob"]  # OK
    "alice" = "bob"  # FAIL
}
"Employees can read [...] the salaries of their subordinates."

allow = true {
    input.method = "GET"
    input.path = ["salaries", employee_id]
    input.user = employee_id
}

Input

{ "method": "GET",  
  "path": ["salaries", "bob"],
  "user": "alice" }

Data (in-memory)

{ "manager_of": {  
  "bob": "alice",
  "alice": "janet"} }
"Employees can read [...] the salaries of their subordinates."

allow = true {
  input.method = "GET"
  input.path = ["salaries", employee_id]
  input.user = employee_id
}

allow = true {
  input.method = "GET"
  input.path = ["salaries", employee_id]
  input.user = data.manager_of[employee_id]
}

Input

```json
{"method": "GET",
  "path": ["salaries", "bob"],
  "user": "alice"}
```

Data (in-memory)

```json
{"manager_of": {
"bob": "alice",
"alice": "janet"}}
```
"Employees can read [...] the salaries of their subordinates."

allow = true {
    input.method = "GET"
    input.path = ["salaries", employee_id]
    input.user = employee_id
}

allow = true {
    input.method = "GET"
    input.path = ["salaries", "bob"]
    input.user = data.manager_of["bob"]
}

Input

{"method": "GET",
 "path": ["salaries", "bob"],
 "user": "alice"}

Data (in-memory)

{"manager_of": {
 "bob": "alice",
 "alice": "janet"}}
"Employees can read [...] the salaries of their subordinates."

```
allow = true {
  input.method = "GET"
  input.path = ["salaries", employee_id]
  input.user = employee_id
}

allow = true {
  input.method = "GET"
  input.path = ["salaries", "bob"]
  input.user = "alice"
}

Input
{"method": "GET",
 "path": ["salaries", "bob"],
 "user": "alice"}

Data (in-memory)
{"manager_of": {
 "bob": "alice",
 "alice": "janet"}}
```
"Employees can read [...] the salaries of their subordinates."

```
allow = true {
    input.method = "GET"
    input.path = ["salaries", employee_id]
    input.user = employee_id
}

allow = true {
    input.method = "GET" # OK
    input.path = ["salaries", "bob"] # OK
    input.user = "alice" # OK
}
```

Input

```
{"method": "GET",
    "path": ["salaries", "bob"],
    "user": "alice"}
```

Data (in-memory)

```
{"manager_of": {
    "bob": "alice",
    "alice": "janet"}}
```

@sometorin
OPA enables flexible

- RBAC
- ABAC
- Admission Control
- Data Protection
- Risk Management
- ...

OPA supports any

- Resource Type
- Identity Type
- Implementation Language
- Underlying Protocol

```allow
    input.method = "GET"
    input.path = ["salary", user]
    input.user = user'

method: GET
path: /salary/bob
service.source:
    namespace: production
    service: landing_page
service.target:
    namespace: production
    service: details
user: alice
```
- Submillisecond Latency
- Composition
- External Context
- Partial Evaluation
- Rule Indexing
- Tracing
- Interactive Shell (REPL)
- IDE Integrations (VS Code)
- Test Framework
- Coverage
- Dependency Analysis
open-policy-agent/opa

★ Star 1,285

CLOUD NATIVE SANDBOX

@sometorin @OpenPolicyAgent
Capturing Intent
Capturing Intent
# Summary

<table>
<thead>
<tr>
<th><strong>Resource types</strong></th>
<th>REST, gRPC method, SSH Login, Keys, Kafka Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Identity types</strong></td>
<td>VM/Container Services, Batch Jobs, FTEs, Contractors</td>
</tr>
<tr>
<td><strong>Underlying Protocols</strong></td>
<td>HTTP, gRPC, SSH, Kafka Protocol</td>
</tr>
<tr>
<td><strong>Implementation Languages</strong></td>
<td>Java, Node JS, Ruby, Python</td>
</tr>
<tr>
<td><strong>Latency</strong></td>
<td>&lt; 0.2 ms for basic policies</td>
</tr>
<tr>
<td><strong>Flexibility of Rules</strong></td>
<td>OPA Policy Engine</td>
</tr>
<tr>
<td><strong>Company Culture</strong></td>
<td>Policy Portal - Exercising Freedom, Responsibly</td>
</tr>
<tr>
<td><strong>Capture Intent</strong></td>
<td>Policy Portal UI hides Policy Syntax</td>
</tr>
</tbody>
</table>
Take Away

• AuthZ is a fundamental security problem

• Comprehensive solution gives better Control and Visibility

• Get there faster with Open Source Tools (like OPA)

• Get involved in communities (like PADME)
Questions?

(Volterra is hiring!)

Manish Mehta
manish@ves.io

Torin Sandall
@sometorin