Implementing Authorization
Torin Sandall

- Engineer @ Styra
- Co-founder @ Open Policy Agent
"Undifferentiated Heavy Lifting"
- Jeff Bezos (Amazon CEO, 2006)
Authorization is heavy lifting.
...but every app needs authorization.
Rethink how you implement authorization.
Ship secure projects faster.
Authentication $\neq$ Authorization

(auth/n)

Verify identity

(auth/z)

Verify permission
Authentication standards

SAML

```xml
<saml:Assertion>
  <saml:Subject>
    <saml:NameID abcdef>
    </saml:NameID>
    <saml:SubjectConfirmation Method="urn:...:bearer">
      <saml:SubjectConfirmation Data NotOnOrAfter=../>
    </saml:SubjectConfirmation>
  </saml:Subject>
</saml:Assertion>
```

OpenID Connect

```json
{
  "iss": "https://example.com",
  "sub": "bob",
  "aud": "retail",
  "nbf": 123456789,
  "exp": 123456789,
  "amr": ["password", "otp"]
}
```

SPIFFE

```text
spiffe://acmecorp/a/b/c
```

Enterprise
Consumer
Infrastructure
Authentication verifies identity & produces attributes.

```json
{
  iss: acmecorp
  sub: bob
  aud: retail
  nbf: 123456789
  exp: 123456789
  amr: [
    password
    otp
  ]
}
```
Attribute semantics are beyond the scope of the specification.

### 2.2. Path

The path component of a SPIFFE ID allows for the unique identification of a given workload. The meaning behind the path is left open ended and the responsibility of the administrator to define. Paths MAY be hierarchical - similar to filesystem paths. The specific meaning of paths is reserved as an exercise to the implementer and are outside the SVID specification. However some examples and conventions are expressed below.

2. ID Token [...]

The definition of particular values to be used in the amr Claim is beyond the scope of this specification. Parties using this claim will need to agree upon the meanings of the values used, which may be context-specific. [...] ID Tokens MAY contain other Claims.
App must **decide** how identity attributes map to functionality, privileges, etc.
What about OAuth?

RFC 6749

The OAuth 2.0 Authorization Framework

Abstract

The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf.
OAuth 2.0 enables **delegation**.

"Power of Attorney" for web and mobile applications.

Source: https://backstage.forgerock.com/docs/am/5/oauth2-guide/
Application of access tokens is beyond the scope of the specification.

RFC 6749 Section 7

The client accesses protected resources by presenting the access token to the resource server. The resource server MUST validate the access token and ensure that it has not expired and that its scope covers the requested resource. The methods used by the resource server to validate the access token (as well as any error responses) are beyond the scope of this specification but generally involve an interaction or coordination between the resource server and the authorization server.
How does the app **decide** what to do with incoming requests, identity attributes, and access tokens?
Authorization: Problem Statement

Can identity I do operation 0 on resource R?
Example Authorization Scenario

"Employees should be able to read their own salary and the salary of employees they manage."

HTTP API
GET /salary/bob
Authorization: alice
@route("GET", "/salaries/{employee_id}")
def get_salary(req):
    if not authorized(req):
        return error(403)
    return db.read_salary(req.emp_id)

def authorized(req):
    if req.user == req.emp_id:
        return True
    if req.user in managers_of(req.emp_id):
        return True
    return False
This code raises questions!

- How do you enforce policies from security or legal departments?
- How do you delegate control to your end-users?
- How do you roll-out policy changes?
- How do you access HR database or other sources of context?
- How do you render the UI based on the user's permissions?
- How do you audit and test your policies for correctness?
- How do you audit enforcement of the policies?
- What about 100+ other services written in Java, Go, and Ruby?
Authorization: Problem Statement

Can identity I do operation O on resource R?

Goal: Solve for any combination of I, O, and R.

Enforce in any language, framework, or environment.
Authorization: Common Approaches

ACLs

- deny by default
- admin controlled
- user, action, resource
Authorization: Common Approaches

ACLs
- deny by default
- admin controlled
- user, action, resource

RBAC
- deny by default
- group users
- grant groups permissions
- inheritance
- separation of duty (SOD)
## Authorization: Common Approaches

<table>
<thead>
<tr>
<th>ACLs</th>
<th>RBAC</th>
<th>IAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>- deny by default</td>
<td>- deny by default</td>
<td>- allow and deny</td>
</tr>
<tr>
<td>- admin controlled</td>
<td>- group users</td>
<td>- users, groups, resources</td>
</tr>
<tr>
<td>- user, action, resource</td>
<td>- grant groups permissions</td>
<td>- negation &amp; built-ins</td>
</tr>
<tr>
<td></td>
<td>- inheritance</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- separation of duty (SOD)</td>
<td></td>
</tr>
</tbody>
</table>
Authorization: Common Approaches

<table>
<thead>
<tr>
<th>ACLs</th>
<th>RBAC</th>
<th>IAM</th>
<th>ABAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>- deny by default</td>
<td>- deny by default</td>
<td>- allow and deny</td>
<td>- boolean logic</td>
</tr>
<tr>
<td>- admin controlled</td>
<td>- group users</td>
<td>- users, groups, resources</td>
<td>- context</td>
</tr>
<tr>
<td>- user, action, resource</td>
<td>- grant groups permissions</td>
<td>- negation &amp; built-ins</td>
<td>- relationships</td>
</tr>
</tbody>
</table>
Authorization: Trade-offs

ACLs  RBAC  IAM  ABAC

Ease of use  Flexibility
ACLs, RBAC, and IAM are 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.
Open Policy Agent (OPA) is a general-purpose policy engine.
Open Policy Agent (OPA)

Decouple policy decisions from enforcement and codify decisions using a declarative language.
Open Policy Agent (OPA)

- Supports multiple authorization models
  - ✔ ACLs
  - ✔ RBAC
  - ✔ IAM
  - ✔ ABAC
Demo
Authorization: Where does OPA stand?

ACLs

Ease of use

RBAC

Flexibility

IAM

ABAC
Authorization: Where does OPA stand?

- ACLs
- RBAC
- IAM
- ABAC

Ease of use → Flexibility
GET /pets
Authorization: bob

SELECT * FROM pets

<table>
<thead>
<tr>
<th>name</th>
<th>owner</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fluffy</td>
<td>Bob</td>
<td>7</td>
</tr>
<tr>
<td>Muffin</td>
<td>Alice</td>
<td>3</td>
</tr>
<tr>
<td>King</td>
<td>Janet</td>
<td>12</td>
</tr>
</tbody>
</table>
GET /pets
Authorization: bob

SELECT * FROM pets

<table>
<thead>
<tr>
<th>name</th>
<th>owner</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fluffy</td>
<td>Bob</td>
<td>7</td>
</tr>
<tr>
<td>Muffin</td>
<td>Alice</td>
<td>3</td>
</tr>
<tr>
<td>King</td>
<td>Janet</td>
<td>12</td>
</tr>
</tbody>
</table>
Example Policy

"Users should only be allowed to see details of pets they own."
Example Policy

"Users should only be allowed to see details of pets they own."
"Users should only be allowed to see details of pets they own."
GET /pets
Authorization: bob

SELECT * FROM pets

Petdetails

policy query

allow or deny

OPA

Policy (rego)

DB
GET /pets
Authorization: bob

SELECT * FROM pets

Requires OPA to have access to the data.
GET /pets
Authorization: bob

SELECT * FROM pets
WHERE pets.owner = "bob"

See blog.openpolicyagent.org for details.
https://goo.gl/uCPTuF
Authorization is heavy lifting.
Rethink how you implement authorization.

openpolicyagent.org
Thank you!

slack.openpolicyagent.org

open-policy-agent/opa