Enforcing consent conformance in your authorization logic with a fine-grained permissions model

Let’s talk about authorization and consent!

45 min drive
at your own pace
State of authorization
Problem with Consent management
Integrating consent based Authorization in user experience
Key elements for your AuthZ strategy
Promoting Consent as a first class Authorization Policy

{
"alg": "HS256",
"typ": "JWT"
}.
{
“sub”: “**Jeff Lombardo**”,
“role”: “Sr Identity Specialist”,
“issuer”: “AWS”
}.
3ccdQeTNN7AfPj74JJq-RhJd
LwQ_fhR1yXVqzDNJo-Y

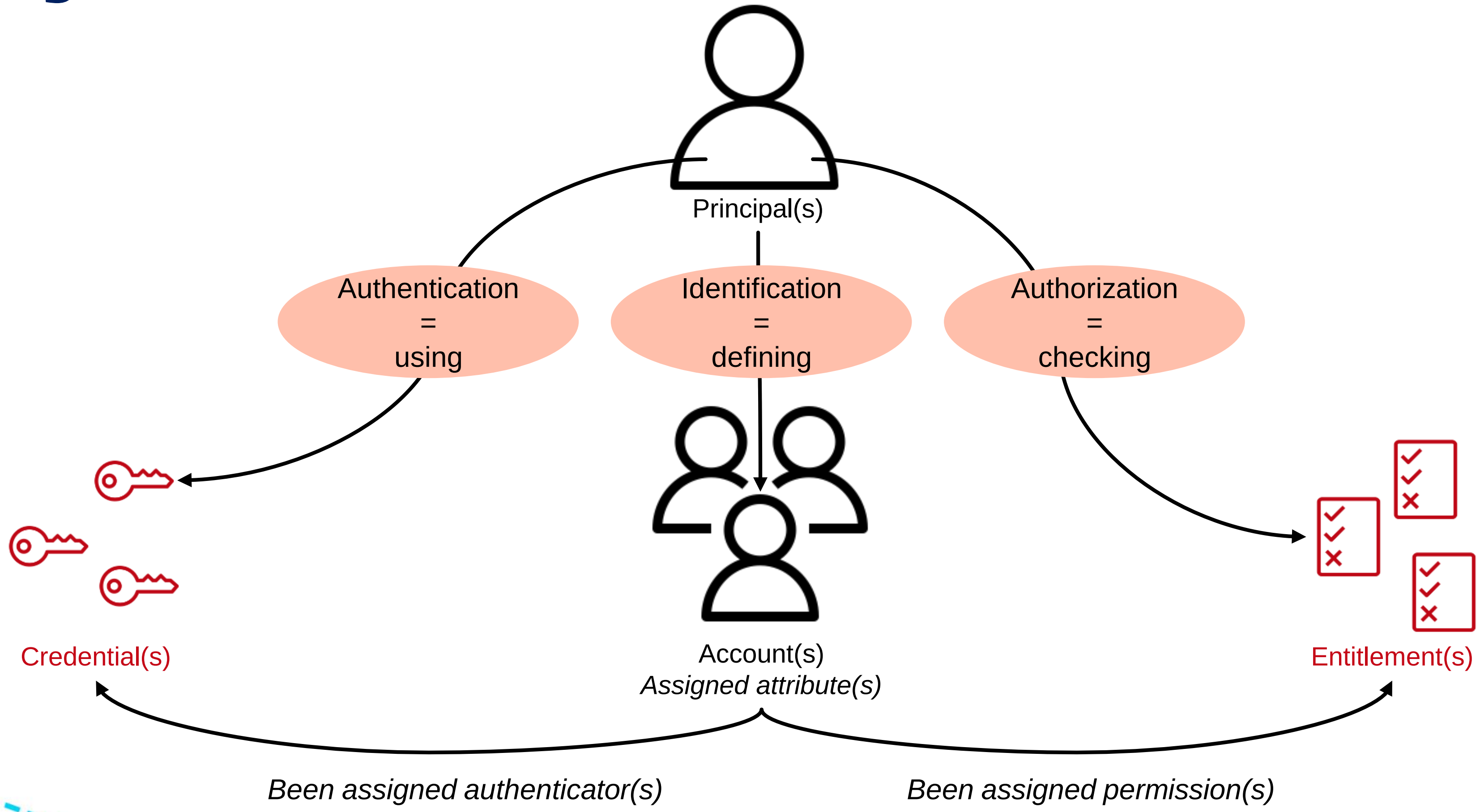
{
"alg": "HS256",
"typ": "JWT"
}.
{
“sub”: “**Jeremy Ware**”,
“role”: “Identity Specialist”,
“issuer”: “AWS”
}.
zTM34eETYMovwhQuB2LwC
7a7TfdYskYaFJzsS1dg3v8

State of authorization

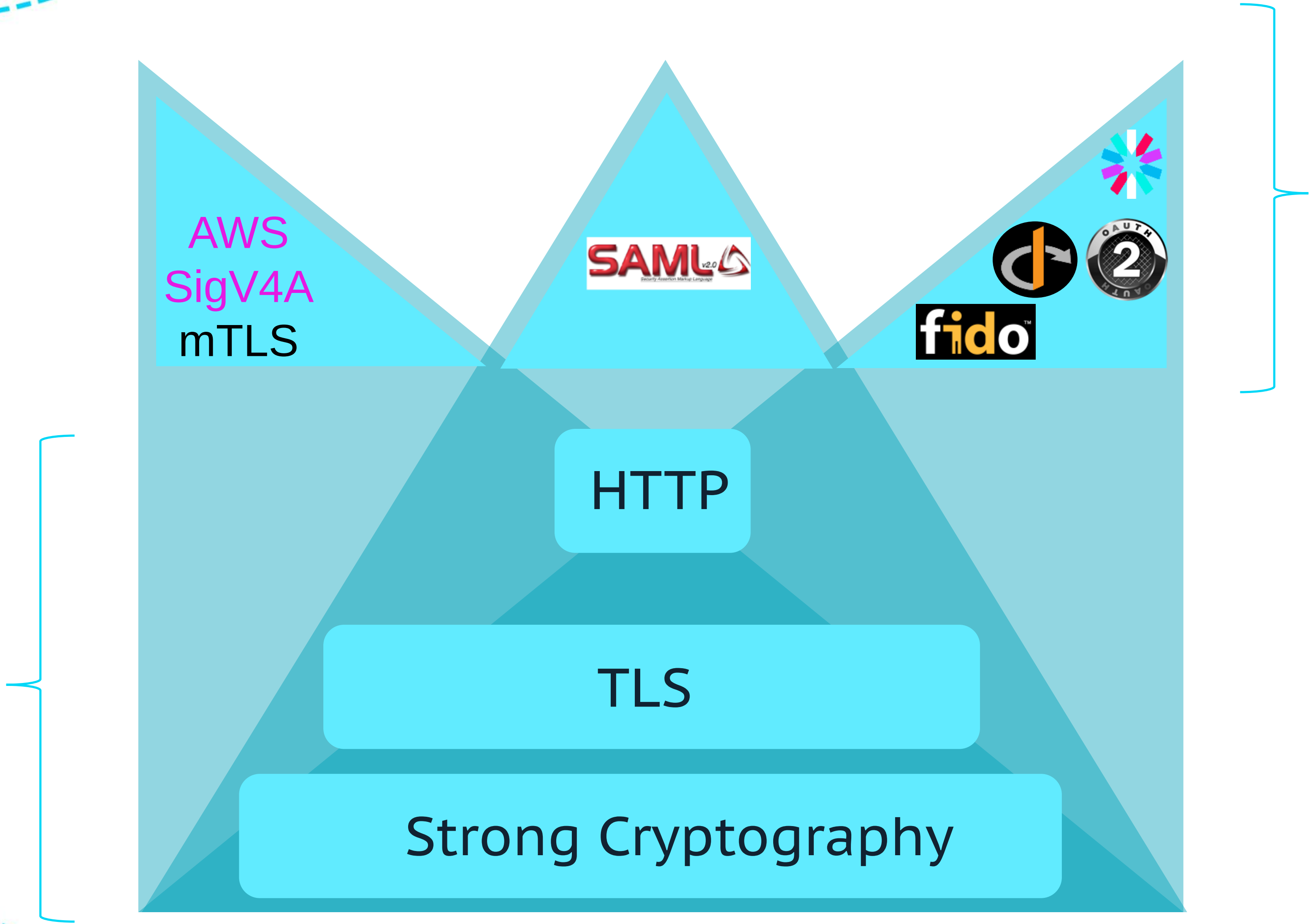
identiverse

#identiverse

Identity reminder...

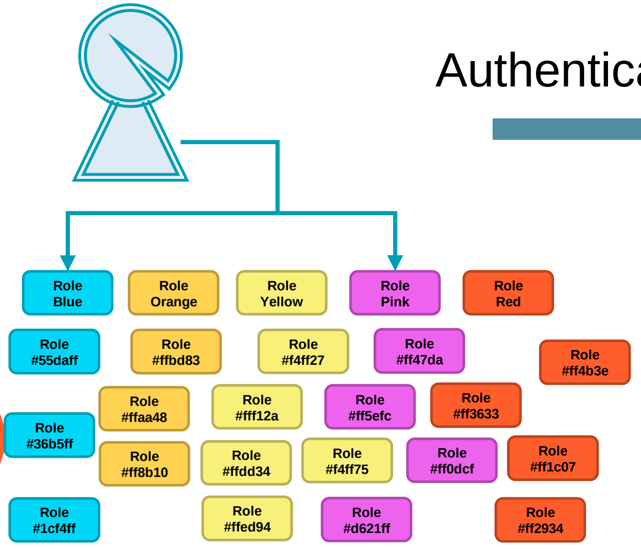


Standards of Digital Identity

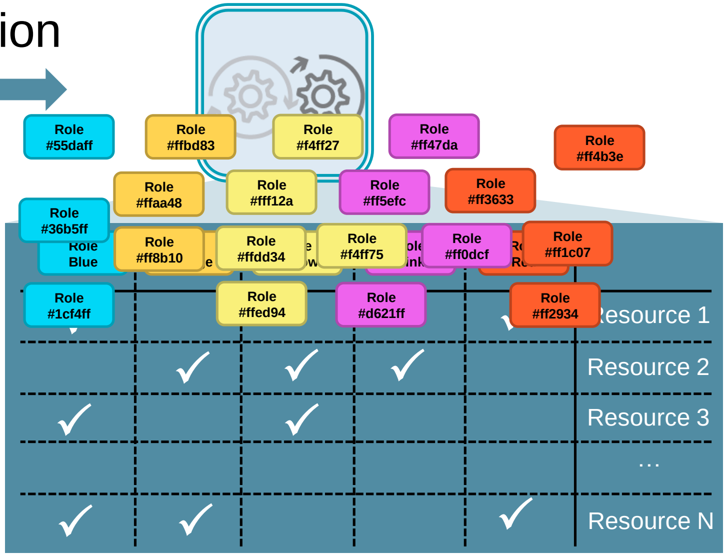


RBAC – Role Based Access Control

Roles are static so we need more…



Authenticates and accesses to the application

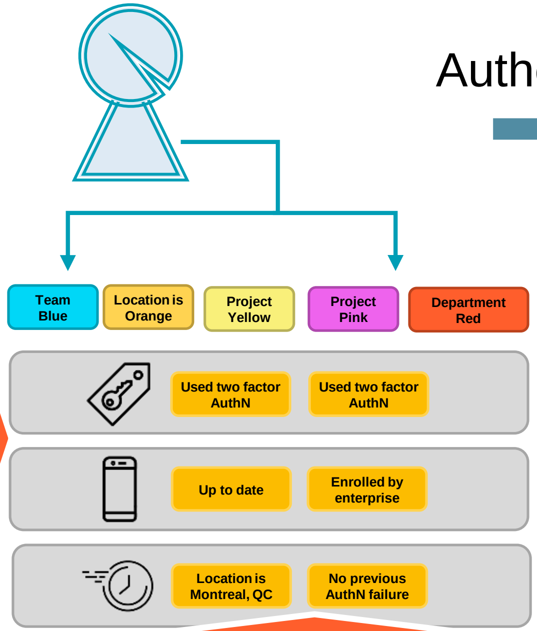


How can our applications handle this kind of explosion?

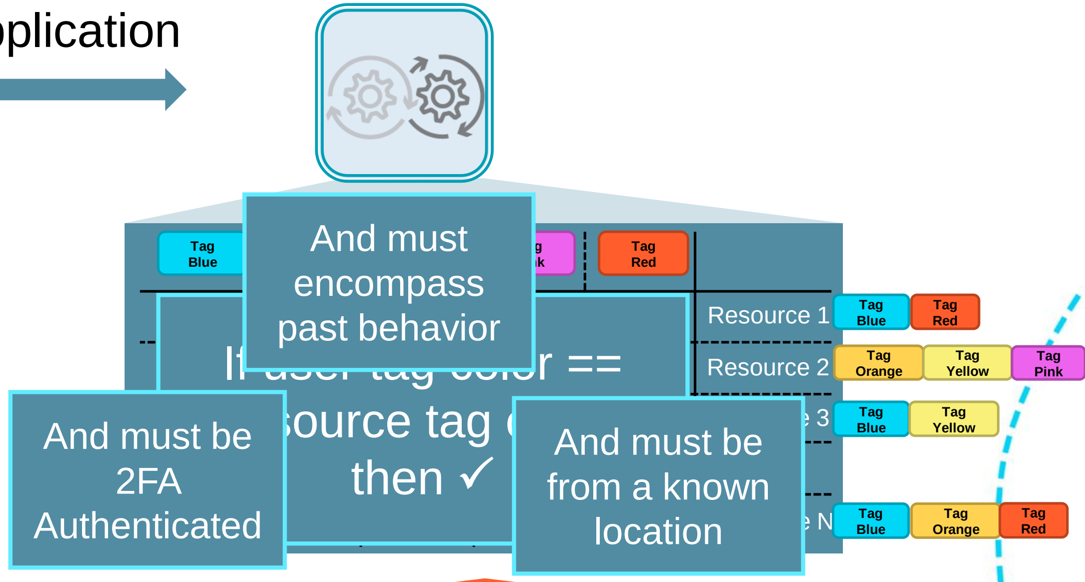
All applications need to be role aware via updates

ABAC - Attribute Based Access Control

Rules grow based on context growth…



Authenticates and accesses to the application



And must encompass past behavior

And must be 2FA Authenticated

And must be from a known location

How can our applications handle this kind of explosion?

All applications need to be rule aware via updates

Good

RBAC

Pre packaged
groups of entitlements

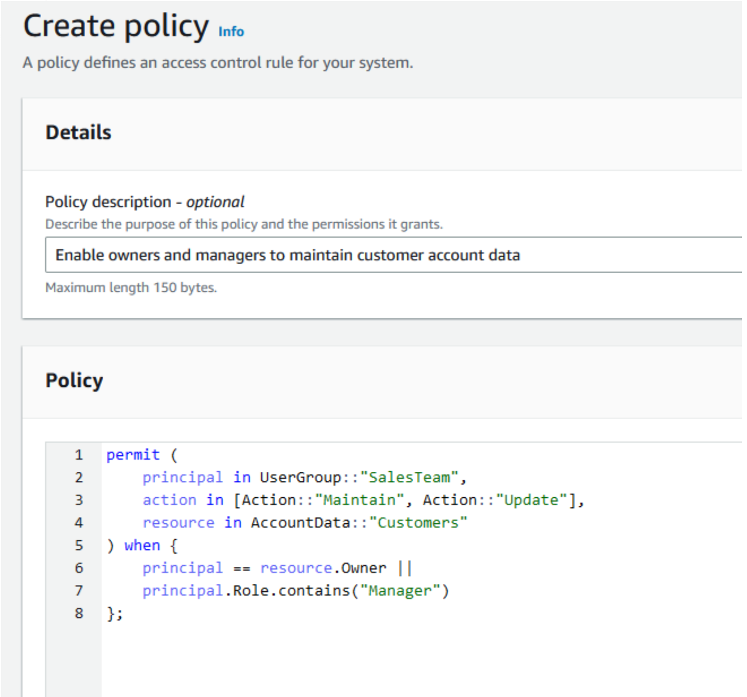
ABAC

Dynamic Access Control
based on
contextual information

We also need:

- One language of expression
  to rule them all
- One source of truth
  to homogenize them all

PBAC - Policy-based access control



Create policy Info

A policy defines an access control rule for your system.

Details

Policy description - *optional*

Describe the purpose of this policy and the permissions it grants.

Enable owners and managers to maintain customer account data

Maximum length 150 bytes.

Policy

```
permit (
    principal in UserGroup::"SalesTeam",
    action in [Action::"Maintain", Action::"Update"],
    resource in AccountData::"Customers"
) when {
    principal == resource.Owner ||
    principal.Role.contains("Manager")
};
```

Scalable

Easier to understand and maintain

Dynamically manageable from runtime

Does not require application code changes

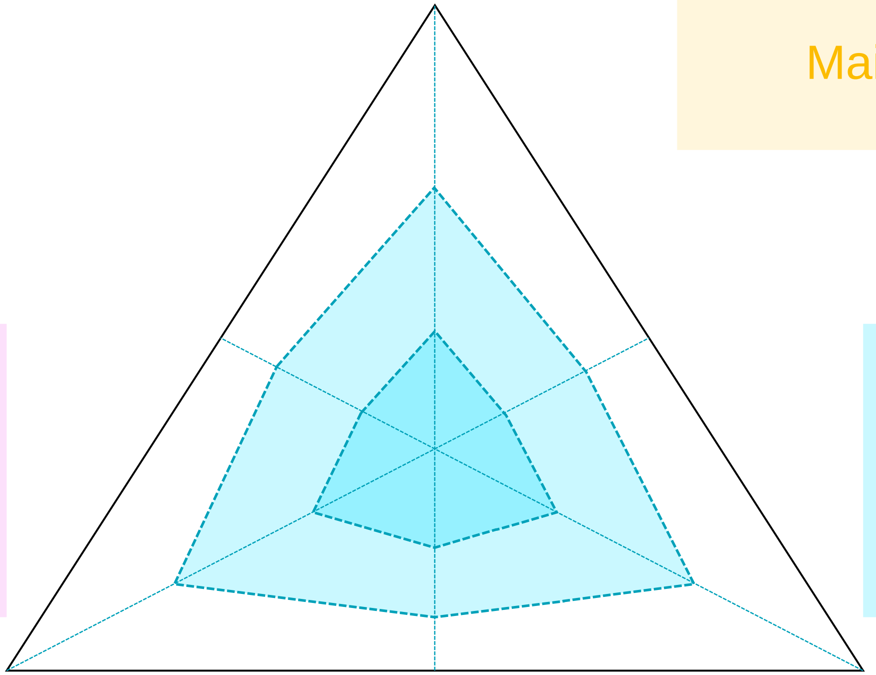
Fine-grained

Access defined down to the level of individual resources and users

PBAC - Core of a Zero Trust strategy

DOD Zero Trust Strategy and Roadmap (2022)	Never Trust, **Always Verify Explicitly**. Treat every user, device, and application as untrusted and unauthenticated. Authenticate and explicitly authorize to the least privilege **using dynamic security policies**
M-22-09 (2022)	Using **centrally managed systems** to provide enterprise identity and access management services […] allowing agencies to more **uniformly enforce security policies that limit access**.
NIST SP800-207 (2020)	**3.1.1 ZTA Using Enhanced Identity Governance** Individual resources or […] components protecting the resource **MUST** have a way to forward requests to a policy engine […] and approve the request before granting access.

PBAC – OK but in which mode?



Centralized

Main objective is **Governance**

Main gain is **Accuracy**

Decentralized

Main objective is **Enforcement**

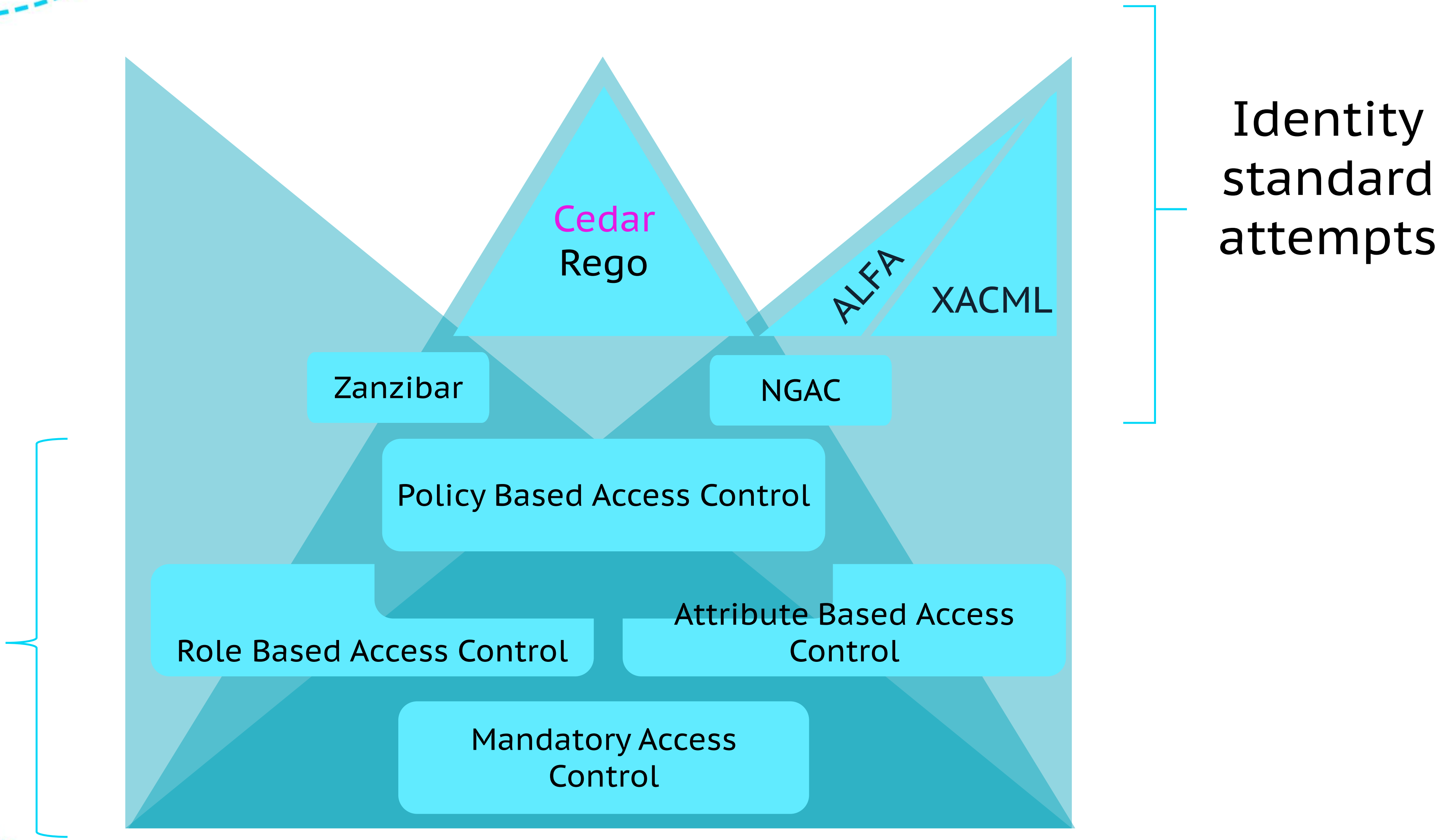
Main gain is **Latency**

Distributed

Main objective is **Definition**

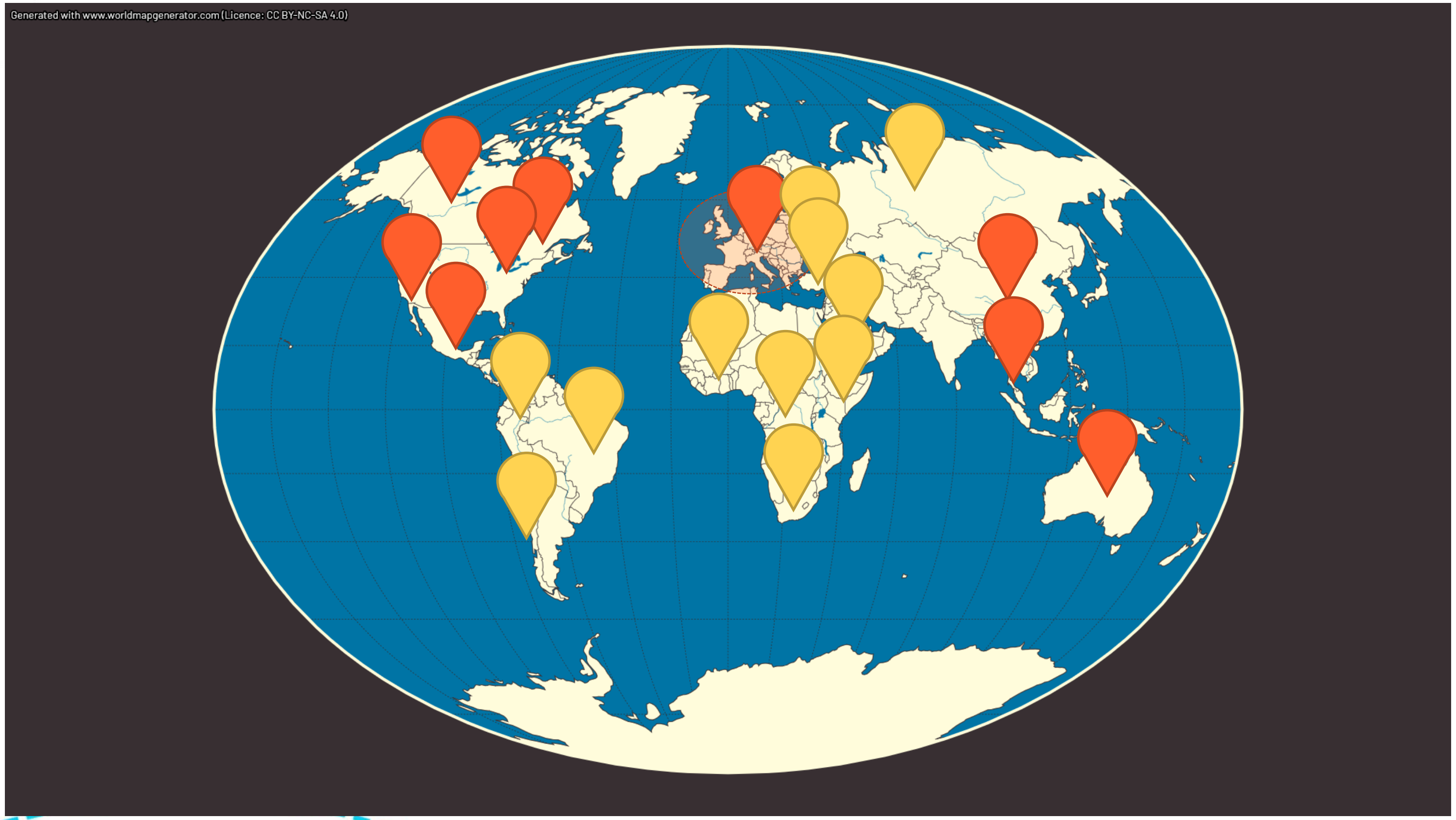
Main gain is **Dynamism**

PBAC – OK but which one?


Cedar
Rego
ALFA
XACML
Zanzibar
NGAC
Policy Based Access Control
Role Based Access Control
Attribute Based Access Control
Mandatory Access Control
Identity standard attempts
Predicate Standards

Problem with consent management

Data Privacy regulations are the new normal



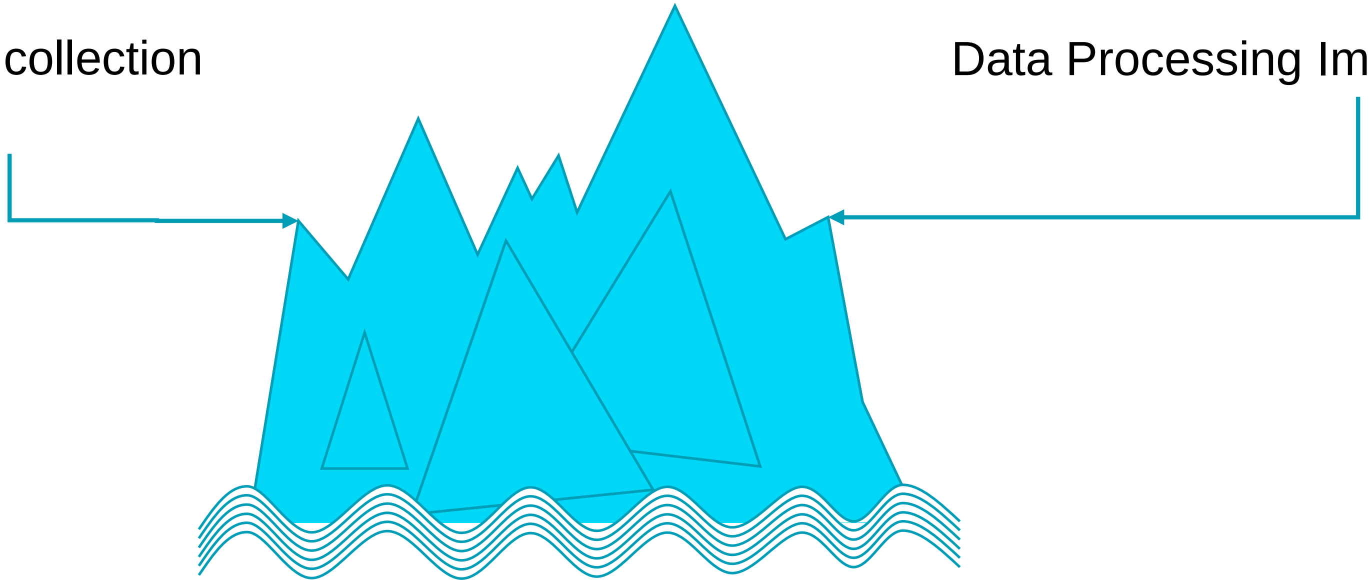
Generated with www.worldmapgenerator.com (Licence: CC BY-NC-SA 4.0)

- GDPR
- CCPA
- ePrivacy
- LGPD
- QC-L25 / C-27
- and many more…

Data Privacy principle in one line

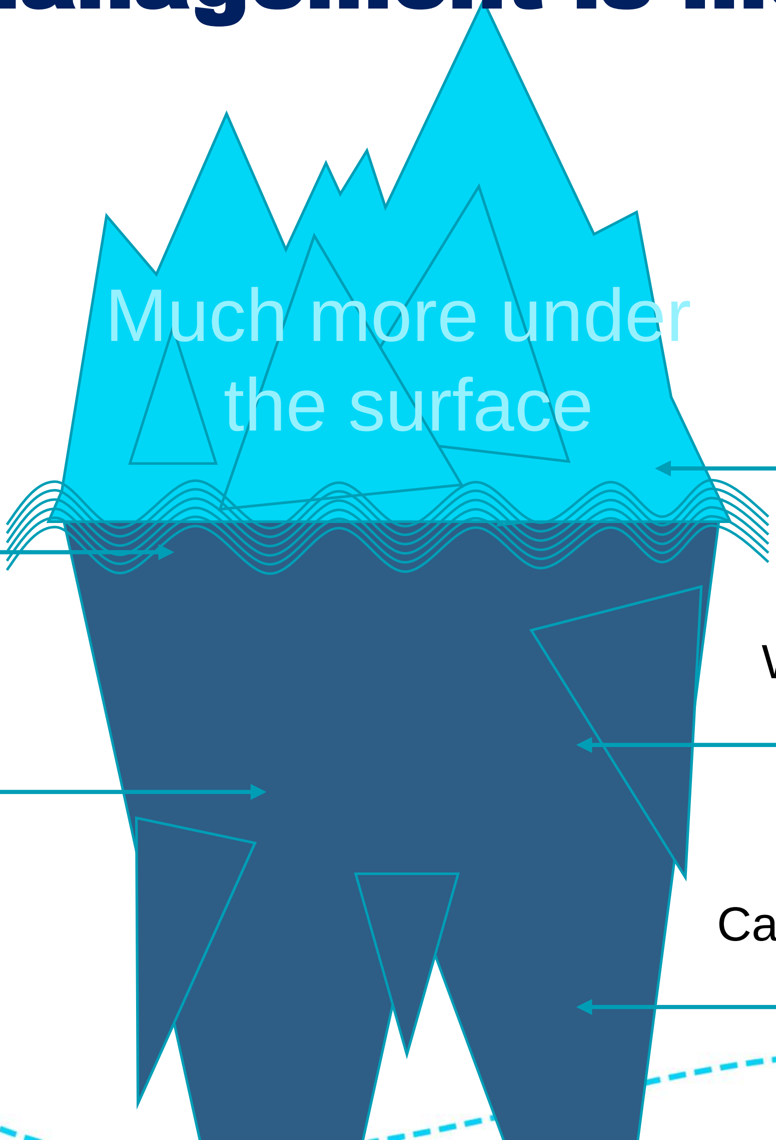
"Ensure that only the principals that shall have justified access to Personal Information effectively have access to it."

And consent management guidance is mostly...



Visible surface level

But consent management is more than that



Who has access to what?

Do we keep this data or not?

Who did consent to what?

Who did access what?

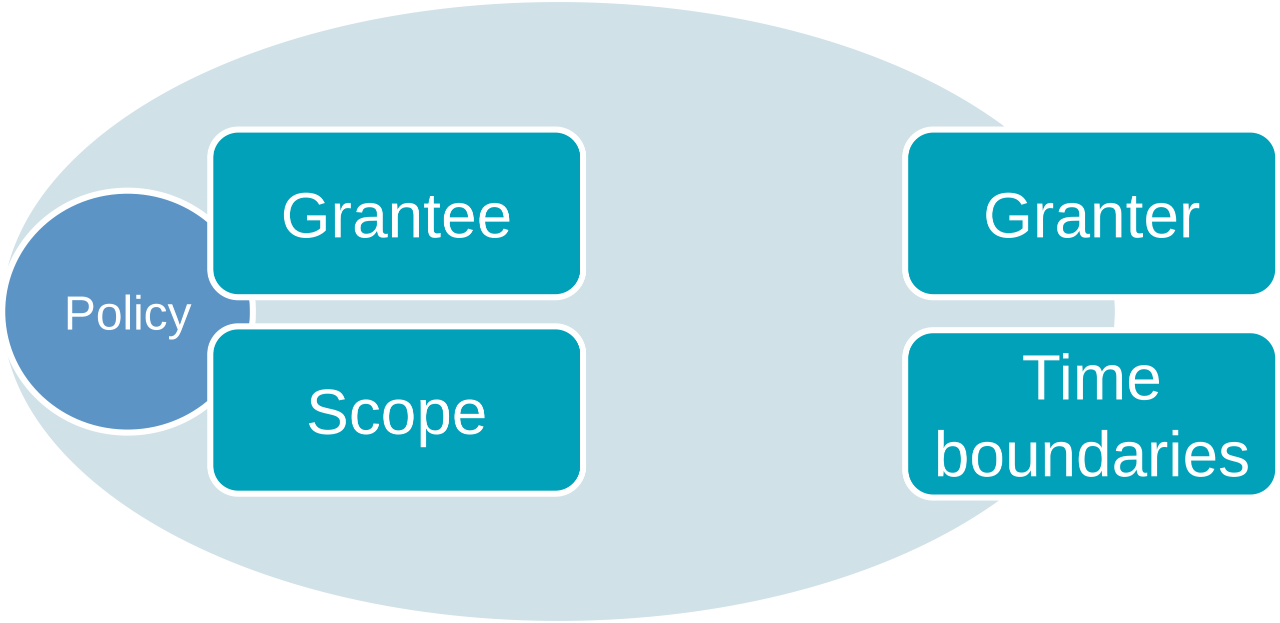
Can you prove who consented?

How much does Consent differ from Policy?

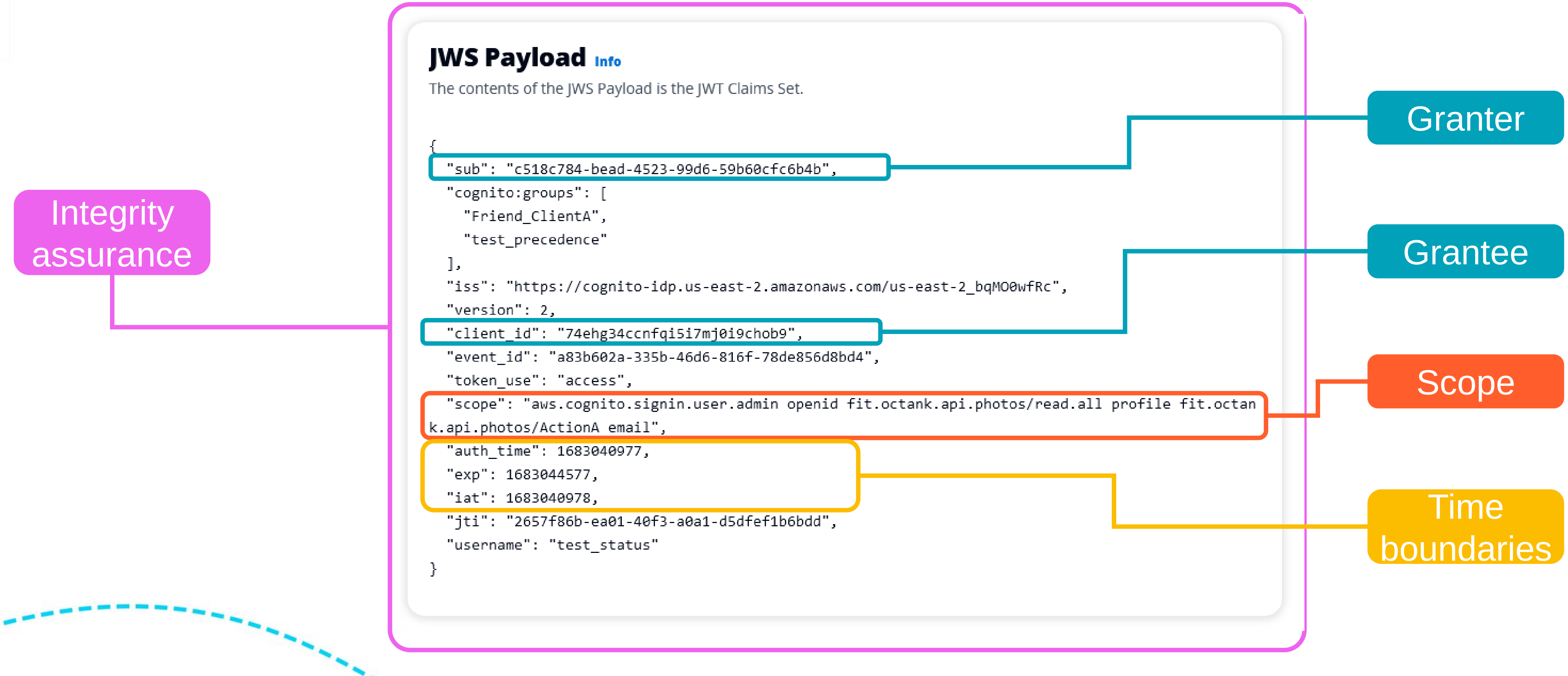
- Consent and policy evaluation to allow are **both required to access to data**
- Consent is **scoped**, and so is an authorization policy
- Consent is **time-bound**, whereas for policies… it is more complicated
- Consent has a **granter** and a **grantee**, policy has mostly a grantee

Promoting Consent as a first class Authorization Policy

We need to expand a policy to be Consent aware



We can bound object to consent

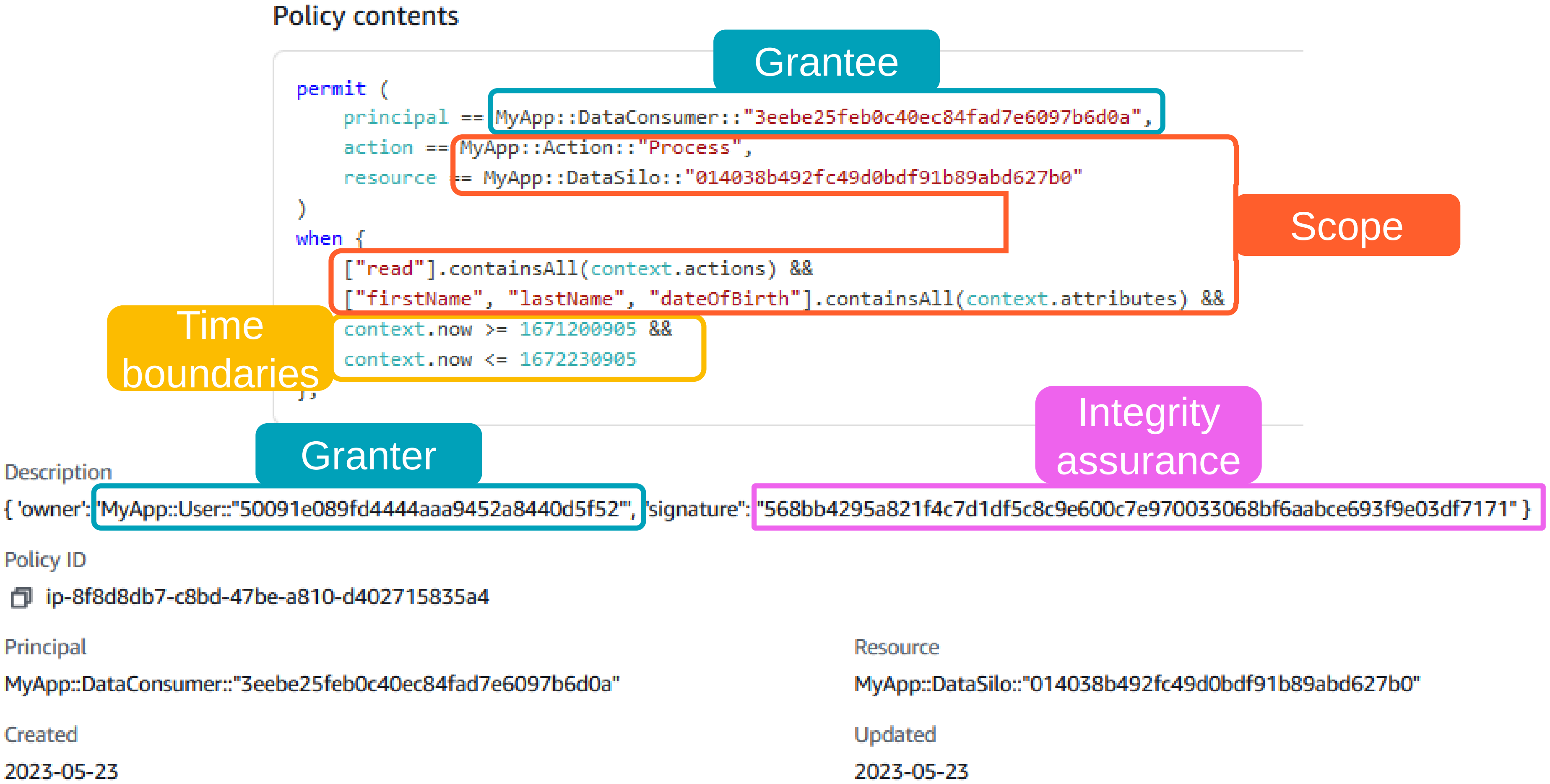


JWS Payload Info

The contents of the JWS Payload is the JWT Claims Set.

```
{
  "sub": "c518c784-bead-4523-99d6-59b60cfc6b4b",
  "cognito:groups": [
    "Friend_ClientA",
    "test_precedence"
  ],
  "iss": "https://cognito-idp.us-east-2.amazonaws.com/us-east-2_bqMO0wfRc",
  "version": 2,
  "client_id": "74ehg34ccnfqi5i7mj0i9chob9",
  "event_id": "a83b602a-335b-46d6-816f-78de856d8bd4",
  "token_use": "access",
  "scope": "aws.cognito.signin.user.admin openid fit.octank.api.photos/read.all profile fit.octan
k.api.photos/ActionA email",
  "auth_time": 1683040977,
  "exp": 1683044577,
  "iat": 1683040978,
  "jti": "2657f86b-ea01-40f3-a0a1-d5dfef1b6bdd",
  "username": "test_status"
}
```

Let’s apply that to Authorization policies



Policy contents

```
permit (
    principal == MyApp::DataConsumer::"3eebe25feb0c40ec84fad7e6097b6d0a",
    action == MyApp::Action::"Process",
    resource == MyApp::DataSilo::"014038b492fc49d0bdf91b89abd627b0"
)
when {
    ["read"].containsAll(context.actions) &&
    ["firstName", "lastName", "dateOfBirth"].containsAll(context.attributes) &&
    context.now >= 1671200905 &&
    context.now <= 1672230905
};
```

Description

{ 'owner': 'MyApp::User::"50091e089fd4444aaa9452a8440d5f52"', 'signature': "568bb4295a821f4c7d1df5c8c9e600c7e970033068bf6aabce693f9e03df7171" }

Policy ID

ip-8f8d8db7-c8bd-47be-a810-d402715835a4

Principal

MyApp::DataConsumer::"3eebe25feb0c40ec84fad7e6097b6d0a"

Resource

MyApp::DataSilo::"014038b492fc49d0bdf91b89abd627b0"

Created

2023-05-23

Updated

2023-05-23

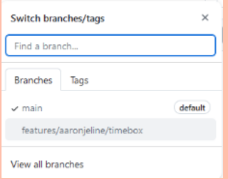
Integrating consent based Authorization in user experience

Let’s share things!

TinyTodo

An example application to learn Cedar, a new language for expressing Authorization rules

Find it at:



Switch branches/tags ×

Find a branch...

Branches | Tags

✓ main default

features/aaronjeline/timebox

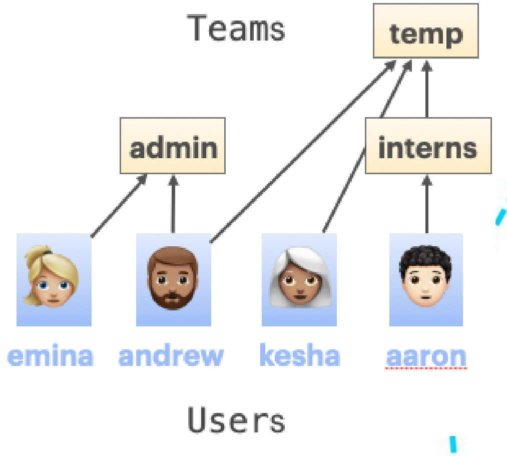
View all branches

1 Basic bootstrapping of Authorization – Default deny

2 Allowing Groups to privileged actions - RBAC

3 Sharing with individuals - ABAC

4 Sharing with individuals (suite) – ABAC and consent

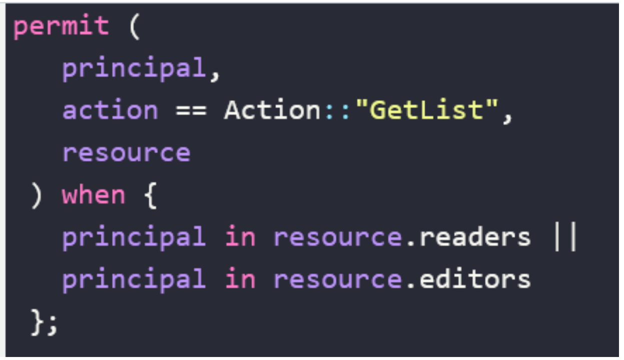


Demo

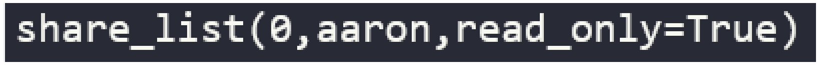
A tale of two sharings

3

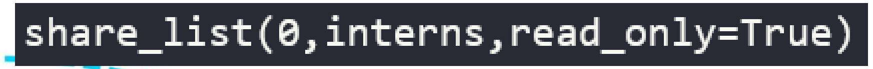
Sharing with individuals - ABAC



```
permit (
   principal,
   action == Action::"GetList",
   resource
 ) when {
   principal in resource.readers ||
   principal in resource.editors
 };
```



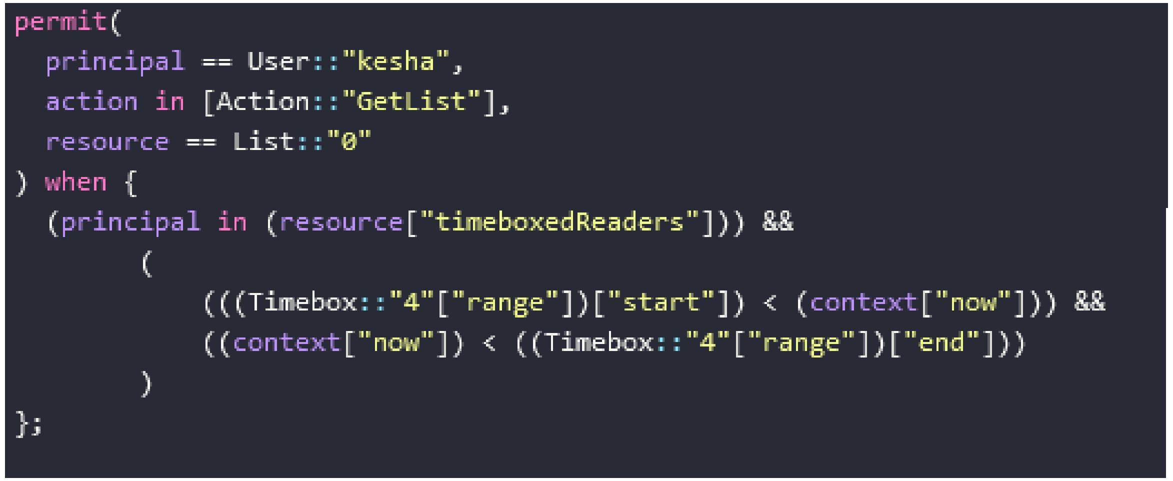
```
share_list(0,aaron,read_only=True)
```



```
share_list(0,interns,read_only=True)
```

4

Sharing with individuals (suite) – ABAC and consent



```
permit(
  principal == User::"kesha",
  action in [Action::"GetList"],
  resource == List::"0"
) when {
  (principal in (resource["timeboxedReaders"])) &&
        (
            (((Timebox::"4"["range"])["start"]) < (context["now"])) &&
            ((context["now"]) < ((Timebox::"4"["range"])["end"]))
        )
};
```

Key elements for your AuthZ strategy

Align with PARC mental model

Principal

Action

Resource

Condition

Easier for humans to review

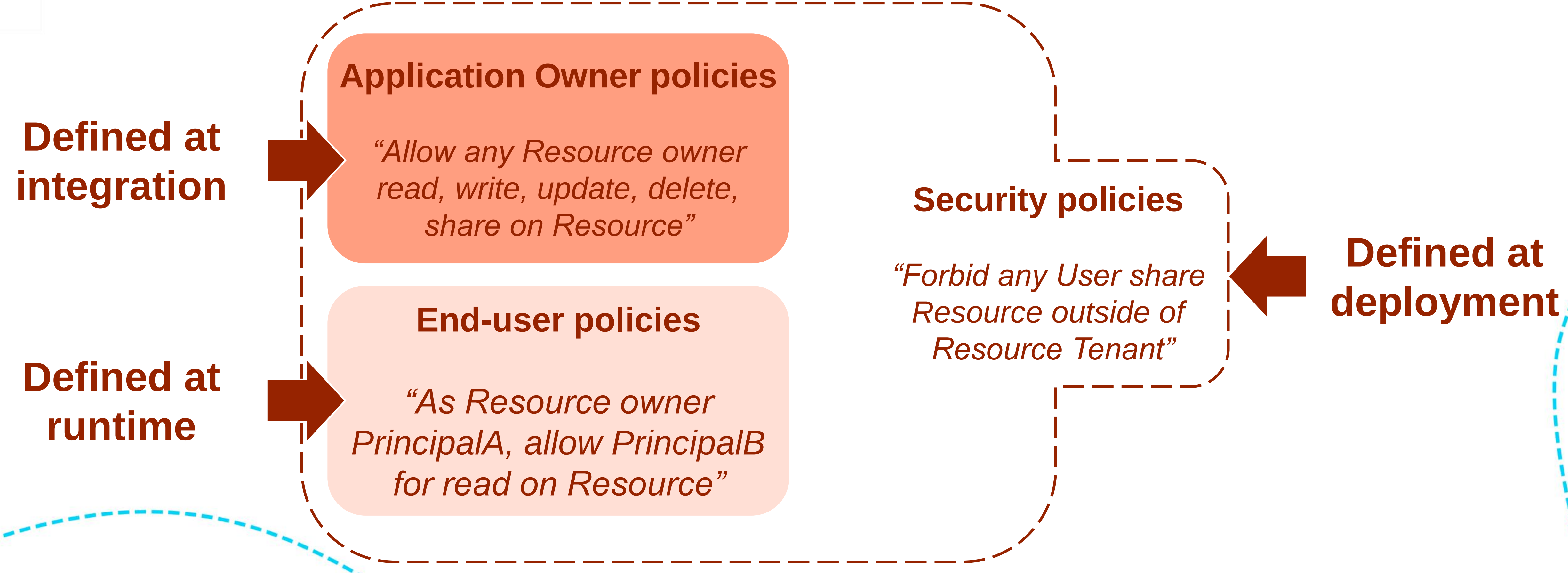
More efficient for systems to review[1] and enforce

More than 500 Millions calls PER SECONDS[2]

[1] https://www.youtube.com/watch?v=6DX7p-OirGU

[2] in 2021, for more: https://youtu.be/8_Xs8Ik0h1w?t=3053

Build policies over 3 layers



Defined at integration

Application Owner policies

"Allow any Resource owner read, write, update, delete, share on Resource"

Defined at runtime

End-user policies

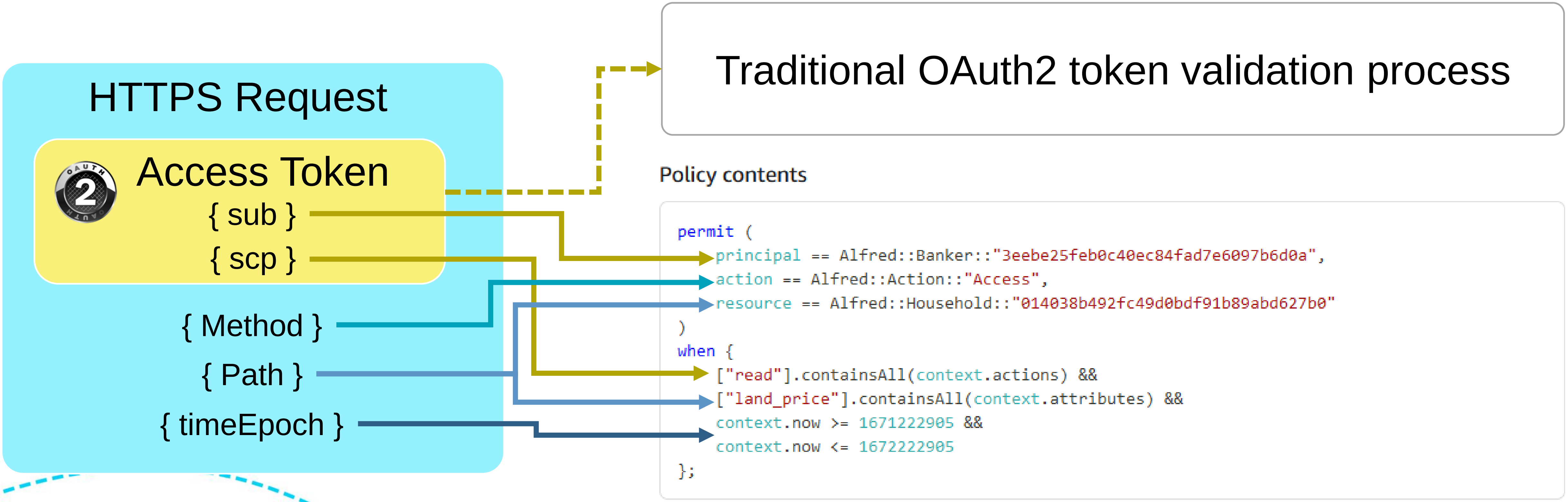
"As Resource owner PrincipalA, allow PrincipalB for read on Resource"

Security policies

"Forbid any User share Resource outside of Resource Tenant"

Defined at deployment

Bake scope and time-boundaries into policies



HTTPS Request

Access Token

{ sub }

{ scp }

{ Method }

{ Path }

{ timeEpoch }

Traditional OAuth2 token validation process

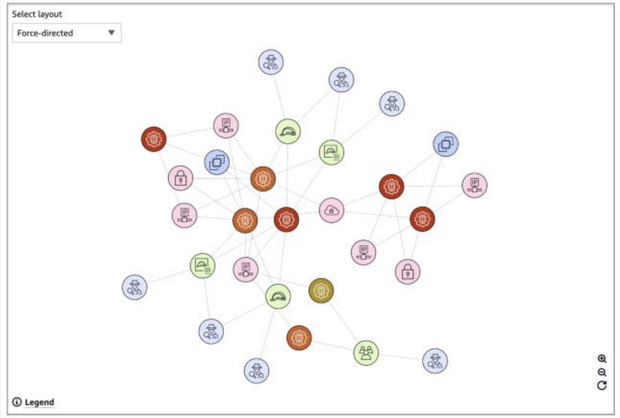
Policy contents

```
permit (
    principal == Alfred::Banker::"3eebe25feb0c40ec84fad7e6097b6d0a",
    action == Alfred::Action::"Access",
    resource == Alfred::Household::"014038b492fc49d0bdf91b89abd627b0"
)
when {
    ["read"].containsAll(context.actions) &&
    ["land_price"].containsAll(context.attributes) &&
    context.now >= 1671222905 &&
    context.now <= 1672222905
};
```

Unblock capabilities

Review entitlements through graph

Direct Acyclic representations can show more that standard queries



Generate proof of consent

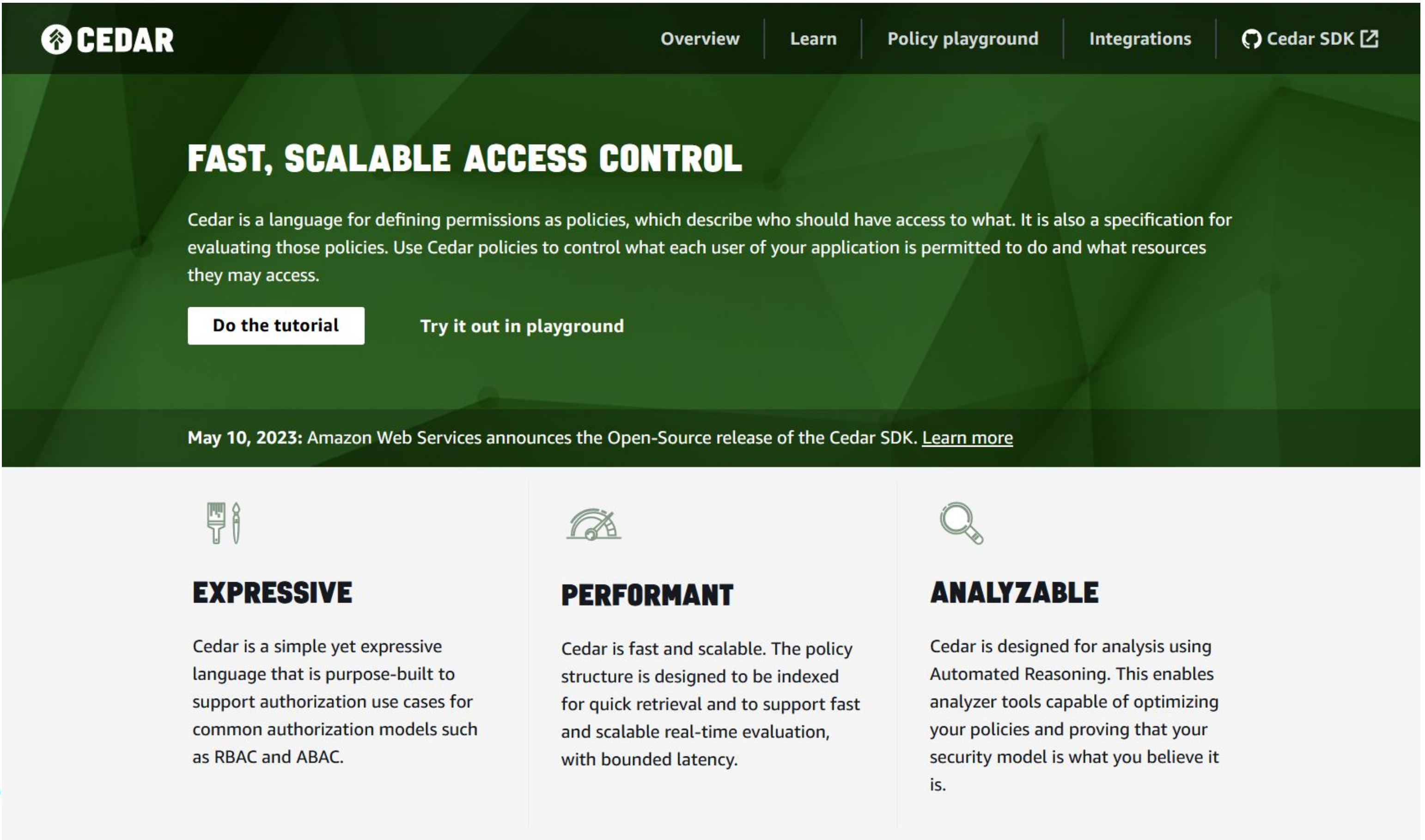
For auditors

For data subjects

Consent Receipt Specification
Version: 1.1.0
Editors: Mark Lizar, David Turner

https://kantarainitiative.org/file-downloads/consent-receipt-specification-v1-1-0/

Try Cedar, it is OpenSource



FAST, SCALABLE ACCESS CONTROL

Cedar is a language for defining permissions as policies, which describe who should have access to what. It is also a specification for evaluating those policies. Use Cedar policies to control what each user of your application is permitted to do and what resources they may access.

May 10, 2023: Amazon Web Services announces the Open-Source release of the Cedar SDK. Learn more

EXPRESSIVE

Cedar is a simple yet expressive language that is purpose-built to support authorization use cases for common authorization models such as RBAC and ABAC.

PERFORMANT

Cedar is fast and scalable. The policy structure is designed to be indexed for quick retrieval and to support fast and scalable real-time evaluation, with bounded latency.

ANALYZABLE

Cedar is designed for analysis using Automated Reasoning. This enables analyzer tools capable of optimizing your policies and proving that your security model is what you believe it is.

SDK

Documentation

Examples

How we built Cedar with automated reasoning and differential testing

https://github.com/cedar-policy

Your turn to play

Using Open Source Cedar to write and enforce custom AuthZ policies

A blog post to implement you first application using Cedar for authorization

Blog posts to learn more Amazon Verified Permissions our own managed Cedar oriented Policy engine

AWS Community Builders

Join AWS Community Builders program to build relationships with AWS product teams, AWS Heroes, and the AWS community

THANK YOU!