Design Document

Platform Access Control Model (ACM)

Started: October 21, 2022 | Author: Harlow, K
This RFC blueprints a Platform Access Control Model (ACM).

Summary

Goals

- Centralized and Standardized Access Control Model
- Authentication and Authorization at Central Gateway
- Consolidated Approach to Access Assignment / Fully Embrace Policy-Based Model

Method

(I) Having a service→resource attribute map used to describe and evaluate access.

(II) Consolidating how access is assigned to a single strategy: by policy.

(III) Enforcing authentication and authorizations at a centralized gateway.

Summary Architecture

The proposed access control model (ACM) is framed by the principles: (a) access is described via resource; and (b) access is granted via policy.

That is:

(I) We (service owners) describe a universe of resources made available by our public APIs

(II) Consumers (resource owners) define policies for who can access what within the bounds of that universe

(III) Those policies prevent, determine, audit, and grant access in the Platform

The ACM infrastructure models three decoupled parts:

Part I: Resource Attribution and Discoverability Model (RAD\M)

RAD\M infrastructure centralizes the collection of resources exposed in our platform (called the resource universe); and a mapping of service endpoints that act on those resources (called service statements).

Resource Universe

A publicly discoverable list of resources. A resource is an attribute-like name representing a service data type, or collection of types, referenced in authorization policies (PBA) to assign access to APIs which have that resource attributed.

Service Statements

An internal mapping of service endpoints→resource. A service statement attributes a resource to a particular service path (or wildcard) endpoint. The central enforcement point (CEP) evaluates policy access to a requested resource by comparing what the policy allows vs. what is required by the best matched service statement.

How RAD\M Works

The RAD model relies on ingestion of a configuration file at the individual service-level of public APIs. These service-level files are compiled to form both the resource universe and service statements, then made discoverable via a centralized broker-like agent.

Primary Utility

- Authorization to be centrally enforced
- Access assignments to be determinable and auditable
- A determinable rule set for evaluating access
- Anti-corruption layer

RAD\M Principles

(I.1) **Resource Attributions (s manipulates r) are explicitly defined for every service's public API.**

(I.2) **Resource Attributions are decoupled from any authorization-related designations.**

(I.3) **Resource Attributions are decoupled from service functional code.**

(I.4) **Resource Attributions are defined within a specification language format.**

(I.5) **Resource Attributions are validated pre-service deployment.**

(I.6) **Resource Attributions are discoverable from a centralized broker.**

(I.7) **Resource Attributions are auditable.**

Part II: Policy-Based Access Model (PBA\M)

The PBA\M is the interface for access assignment. Its infrastructure defines the policy model, access assignment strategies, and method of assuming a policy's authorization.

Policy Structure

The policy is an account-scoped object that defines access statements and principle requirements:

- Policy Statements: A collection of simple resource-based (via RAD) access grants
- Policy Principles: The set of users who have been assigned permission to assume some policy
- Policy Requirements: An optional set of just-in-time conditions a principle must meet to assume the policy (e.g., ip:address)

Comparison to AWS IAM

The PBA model is similar to a subset of AWS IAM in that principles are given permission to assume an AuthorizationPolicy and the Principle assuming that policy is allowed to execute all accesses defined by it and only it. It is evaluated at the centralized enforcement point (CEP). The model decouples authorization from authentication and consolidates different access strategies for e.g., account vs user scopes, in the Platform.

PBA\M Principles

(II.1) **Resource Policies (p is granted s:r) are persisted with a uniform policy schema in a single store.**

(II.2) **Resource Policies are simple, fine-grained, and bound to the resource universe.**

(II.3) **Resource Policies are defined within a specification language format.**

(II.4) **Resource Policies are decoupled from credential items (e.g., users, tokens).**

(II.5) **Resource Policies are dynamic and have immediate effect.**

(II.6) **Resource Policies are revokable.**

(II.7) **Resource Policies are auditable.**

Part III: Centralized Enforcement Point Model (CEP\M)

The CEP\M is the ACM infrastructure concerned with evaluating access at a central point (i.e., a gateway). It models the authentication of a request's credential token, and the authorization of its policy—removing that responsibility from individual service APIs.

Credential Token

An opaque token that stores the userId and account membership. Its secret is the first part of a request's authorization-header used to identify the user, e.g., "token 1ae2f755xuw-q2-1px/Auditor"

CEP Evaluation Process

The CEP model is the matrix point; granting or denying a request by:

- (a) Validating the request's credential token
- (b) Validating the credential has principle rights to assume the policy
- (c) Validating the credential meets the policy's principle requirements
- (d) Comparing the policy's granted resources to the resource required by the request's best-matched service statement

In this model, the policy's data filters, if present, are passed down to the service.

CEP\M Principles

(III.1) **Resource Authorizations (is p allowed s:r) are decoupled from service API layer.**

(III.2) **Resource Authorizations are evaluated independent of any context information about what service or resource attribute it's being evaluated for.**

(III.3) **Resource Authorizations are decoupled from the principle/credential context.**

(III.4) **Resource Authorizations are applied at a central gateway point.**

(III.5) **Resource Authorizations are evaluated in a pure functional and standardized way.**

(III.6) **Resource Authorizations are adherent to the Failsafe Default principle.**

Example: Access Control Model Flow

This example illustrates the three models' (RAD\M, PBA\M, CEP\M) roles, responsibilities, and interaction points.

Step 1: Resource Universe (RAD\M)

Assume our mock Platform has compiled the following centralized resources and service statements from all services:

Sample Resource Universe

```
compliance:controlsMapping compliance:evidence compliance
```

Sample Service Statements

"compliance:compliance/evidence/*":"compliance:evidenc

Step 2: Define Authorization Policy (PBA\M)

Assume customer account xDev defines an Authorization Policy called "AWS-Auditor":

- User 000-000-000 is assigned the right to assume this policy
- Policy is limited to ip:address 10.00.01
- Policy grants auditor-related access to resources with AWS data filters

Policy Definition

```
{

accountId: 'xDev',

authorizationPolicyId: '934-x23-54
```

Credential Token

```
{

accountId: 'xDev',

principleId: 'user:00000-00000-00000'
```

Policy Assignment

```
{

userId:'00000-00000-00000',

accountId:'xDev',

authori
```

Step 3: User Authentication

The user (000-000-000) logs into the xDev account and receives a valid session credentialToken stored as a browser cookie. On future requests, this credentialToken is attached.

Authentication Flow

1. User visits apps.xxxx.io (AUTHENTICATED:FALSE)
2. User ini

Step 4: Request Evaluation (CEP\M)

The user makes a request:

```
GET/compliance/evidence?type=awsHTTP/1.1
Authorization
```

Gateway Evaluation Process

The Centralized Enforcement Model (CEM) processes the request:

1. Validate AUTHORIZATION token credential

```
http://iam/resolve/1ae2f755xuw-q2-1px
=>{user:'0000-0000
```

2. Fetch AUTHORIZATION token policy

```
http://iam/0000-0000-0000/assume/AWS-Auditor => Returnsful
```

3. Inspect REQUEST and GRANT/DENY

- (a) Check if PrincipleRequirements are met
- (b) Check if requested resource attribution statement is granted

Resource Matching Process

```
GEThttps://xxxxxx.io/compliance/evidence/aws_Xsfha-afg
```

4. Fulfill Request

```
user:'000-000-000' account:'xDev' filters:["*"] =>Forwa
```

Implementation Details: RAD\M

RAD\M defines (a) the standard for service-by-service cataloging of public resources and endpoint→resource relationships; (b) the means to compile those catalogs and make them centrally discoverable.

What is a Resource?

A resource represents some data type or action (or collection thereof), that is acted on by a service's public API.

- It's a public-facing attribute (e.g., s3_bucket, ecs_task)
- It can be hierarchical, e.g., LibraryItem, LibraryItem:Evidence

Note: Policies also use hierarchical statements, e.g., read:LibraryItem: or read:LibraryItem:Evidence:**

Configuration File Standard

The proposed standard is a configuration file at the root of every service with a public API containing two pieces of information:

Service Resources

A listing of resources exposed by the service's public API.

Format: `"service:resource"`

Service Statements

A path-by-path (or query-by-query / mutation-by-mutation) mapping of resources exposed by the service's public API.

Format: `"service:path":"resource"`

Example: Compliance Service

```
//compliance-service"serviceresources""compliance:con
```

Example: Query Service

```
//query-service"service resources"
"query:entity"
"quer
```

Centralization Mechanism

The centralization part of this model refers to the process in which a copy of each service's catalog is persisted in a central location and used for discovery by several different processes, e.g., enforcing authorization at the gateway and validating policies.

Proposed Implementation

- Each service uploads its configuration file to S3 in a pre-deployment hook
- The service configuration files in S3 are either:
    - Fetched and composed on each cache miss request by the centralization broker, OR
    - Stored compiled in S3 and polled for updates on some interval or trigger