Managing authentication and authorization for your Substrate application

Document Level Classification

200

Introduction

This page describes how service owners can configure their Substrate application to authenticate and authorize requests from Okta users (internal to Epic) or from other services (machine-to-machine).

Substrate applications can use https://confluence-epicgames.atlassian.net/wiki/spaces/CE/pages/93488860 for authenticating (authN) and authorizing (authZ) requests with Okta and UAM. The authentication workflow (parse headers, check for and validate JWT, perform redirects as needed) is handled by https://confluence-epicgames.atlassian.net/wiki/spaces/CE/pages/93488602 that is deployed as a container alongside your application in the same *Pod*.

Adding UAP to your Substrate application

UAP is intended to be used as a *request proxy*. UAP should sit in front of each instance of the application and should be configured to point to that instance of the service as its *upstream*. Incoming traffic should always go through UAP instead of to the application directly. This means that *Ingress* should be reconfigured to point to UAP, and for security reasons, the upstream application should not be exposed on the network. When UAP receives a request with a valid JWT, it will proxy the request to the configured upstream. If the upstream application receives a request, it can safely assume that the request has already been authenticated and authorized by Okta (for requests from users) or UAM (for requests from machines). The token will be included in the Authorization header of the incoming request and its claims can be inspected using a typical JWT library to retrieve various information. To learn more about the JWT payload, see UAS JWT Schema.

If UAP receives a request that either does not include a JWT in the authorization header, or contains an invalid JWT, it will assume that the request needs to be authenticated and that user authentication is desired. The request will be redirected to the centralized auth server component to go through the user authentication flow. Clients making requests to a UAS-enabled service should have redirects enabled, as the user authentication flow follows a number of necessary redirects before ultimately reaching the upstream application.

A service that has UAS enabled will support both user authentication (via Okta), as well as machine-to-machine (m2m) authentication (via a special protocol called UASSv1).

⌄ Example: Snippet to add UAP container to your Pod with epic-app

```
epic-app:
  containers:
    # UAP
    uap:
      image:
        name: hub.ol.epicgames.net/infra-platform/uap
        # See https://github.ol.epicgames.net/cloud-eng/uas/releases fo
        tag: v1.9.0
      ports:
        - 8000
      environment:
        #
        # Reference:
        # * https://github.ol.epicgames.net/cloud-eng/uas#uap-flags-and
        #
        UAS_OAUTH_AUDIENCE: uas
        UAS_SERVICE_NAME: <YOUR_SERVICE_NAME>
        UAS_UPSTREAM: http://127.0.0.1:<YOUR_APPLICATION_PORT>

  service:
    enabled: true
    type: ClusterIP
    ports:
      - 8000 # This should point to the UAP port, and not the applicati
```

Configuring Path-based Authorization (PBA) with UAP

UAP supports Path-based Authorization that allows you to:

1. Require specific UAM policies for specific routes.
2. Restrict the HTTP methods for routes.
3. Ignore specific routes (for example, `/health`).

To use path-based authorization, you will need to provide a *route configuration file* to the UAP container. An example of how to do this for Substrate applications is available here.