The ArcGIS Platform

Security Model for Apps

Jeremy Bartley, Keyur Shah, Sud Menon
Security

Built in Access Control based on Sharing Rules

https: for secure information transport

SAML 2 Web SSO: Use your Enterprise Login to Sign in to your ArcGIS Account

Authenticate using PKI, Integrated Windows, or Forms

ArcGIS

OAuth 2 for Developers. Supports User Logins as well as App Logins.
Sharing and Collaboration

• ArcGIS includes Organizations, Users, and Groups
• Enables Secure and Private sharing of information
The ArcGIS Platform - Cloud Information Model

- Organization
  - Group
  - Application
  - Item
  - Web Map
  - User
    - Layer
      - Service
      - Dataset
  - Dataset
Types of Apps

- Apps that allow users of the Platforms to log in and work with content that is accessible to the logged in user.

- Apps who’s users are unknown to the platform and who connect to the platform on behalf of the app itself.
Types of Logins

- **User Logins**
  - Users of the App are users of the Platform
  - Users login to the platform through the App
  - Users must possess user credentials

- **App Logins**
  - Users of the App are unknown to the platform
  - App logs in to the platform on behalf of itself
  - App must possess app credentials
Existing Authentication Models
Existing Authentication Models

• **ArcGIS Token Based Authentication**
  - Introduced in ArcGIS Server 9.3
  - Supported both On Premises and Online
    - Portal for ArcGIS, ArcGIS Server
    - ArcGIS Online

• **Web Server / JEE Container managed Authentication**
  - PKI, IWA, HTTP Basic and Digest
ArcGIS Token Based Authentication

- Introduced in ArcGIS Server 9.3
- Supported both On Premises and Online
  - Portal for ArcGIS, ArcGIS Server
  - ArcGIS Online
- Based on a generateToken endpoint
  - similar to OAuth2 resource owner password credentials grant
User Logins

- App is responsible for presenting the user with a login dialog
  - User enters platform credentials into app’s login dialog

- App signs the user in using platform’s token service endpoint
  - App obtains access token that allows it to act on behalf of user

- Client side IdentityManagers in Web and Runtime API’s take care of the entire flow
Demo

User Login
Javascript Identity Manager
Token Based
App Logins

- Apps are modeled as users - ‘app user’

- App logs in using app user credentials, eg
  - app_username + app_password

- App obtains access token for itself

- Assumes app has a server side component that keeps the credentials secure
Web Server / Container Managed Authentication

• Server side :
  - Authentication is carried out by the fronting web server hosting the ArcGIS Server or Portal web adaptor

• Client side :
  - Authentication is carried out by the OS HTTP communication stack on which the App and the ESRI client api is built

• Authentication is mostly transparent to both the App and to the ESRI client and server stacks
  - Authenticated identity is flowed into the server
OAuth2 Authentication
OAuth2 Authentication

- Introduced in the March 2013 ArcGIS Online Update

- Based on the concept of registered applications who’s identity is known to the platform

- May be introduced in Portal for ArcGIS at a future release if there is interest
  - Not part of Portal 10.2 June 2013 Release
OAuth2 Authentication

• A token based authentication scheme

• User Logins are centralized on the server

• Usable by Applications built on the ArcGIS web and runtime API’s using the existing versions of those API’s with some work by the Application

• Complete support by the web and runtime API’s in their June release
Types of Logins - OAuth2

• User Logins
  - App guides the user to a login page hosted on the server
  - What the login page does is transparent to the App

• App Logins
  - App must present credentials (appid + appsecret)
User Logins - OAuth2

- Different flows for
  - browser based apps
  - device based apps
  - server based apps

- Browser based Apps - 1 step flow - implicit grant
- Device based Apps - 2 step flow - authorization code grant
**OAuth Workflows**

- **User Logins**
  - Implicit Grant (1-Step Flow)
  - Authorization Grant (2-Step Flow)

- **App Logins**
  - Client Credentials Grant
OAuth URL End-Points

• /authorize
  - User Logins are always initiated with this call
  - Displays a form for users to enter credentials

• /token
  - Generates access tokens for users (authorization grant) and apps
User Login - Implicit Grant

- Login workflow is completed in 1 Step

- Used by browser-based apps
  - JavaScript, Flex, Silverlight
User Login - Implicit Grant - 1 Step

- `/authorize?response_type=token` ...
  - Returns `access_token` on successful authorization
  - Completes the login flow
User Login - Authorization Grant

• Login workflow is completed in 2 Steps
• Used by mobile, desktop and server-side web apps
User Login - Authorization Grant - Steps

• `/authorize?response_type=code&...`
  - Returns authorization `code` on successful authorization

• `/token?grant_type=authorization_code&code=...`
  - Returns `access_token` and `refresh_token`
  - Completes the login flow

• Refreshing tokens
  - `/token?grant_type=refresh_token&refresh_token=...`
    - Returns a new `access_token`
App Login

- /token?
  grant_type=client_credentials&
  client_id=...&
  client_secret=...

  - Returns an app access_token on successful validation
  - Completes the login flow

- App tokens are restricted in what they can access in this release
  - Can only access items owned by the app owner
- More sharing options will be available in the future
Demo
User Logins
OAuth2 - direct use of Rest
User Logins - OAuth2

- Once the app gets a token for the portal it can set it into the client side identity manager

- Identity Manager takes care of the rest and will send the token on all requests
  - Will deal with associated federated servers

- Token expiry will trigger a callback

- In the two step flow the app can use the refresh token to obtain a new access token
App Logins - OAuth2

- App requests an access token in exchange for app credentials
  - appid and appsecret.
  - aka ‘client credentials’ grant

- App is responsible for keeping the app secret secure
  - Implies a server side component for Browser and Device Apps
App Logins - OAuth2

- App developer is responsible for binding the server side component in a secure fashion to the client side App
  - By App Session
    - if the app authenticates its own users unknown to the platform
  - By Referer
    - does not guard against server side code that can set the referer
  - Rate limiting against server based misuse
  - Captcha
Demo

User Logins

JavaScript Identity Manager
Android / iOS Identity Manager
OAuth2
The ArcGIS Platform - Registered Applications

- Application Developers can build Apps for:
  - Users within their own organizations
  - Users within other ArcGIS organizations
  - The larger community

- Developers register Applications that are assigned AppIDs

- Developers use OAuth 2 to authenticate with the platform

- Platform is aware of both the consuming user as well as the consuming application for all requests
The ArcGIS Platform - Usage Reporting and Billing

- Organizations get usage reports on the platform services they consume
  - By type of service
  - By day, week, month

- The reports also show credits consumed for billable services

- Developers can get:
  - Usage reports on the use of their App by platform users.
  - Usage reports on platform services consumed by their App
Portal w/ Registered, Federated, & Hosted Services

Integrated System with Full publishing capabilities

- Client Applications & Browsers
- Server
- Portal
- Identity Store
- Single Sign On across the system
- Your Enterprise
- Registered Services
- Federated Services
- Hosted Services
- Basemaps & Geocoding
- Public Services

Additional login may be required
Portal - Server Federation - Hosted Servers

- Portal + Federated Servers
  - A single logical system with one identity space and one permission model
  - Mutual trust between portal and servers established by administrators

- Each service published by a user is a service item in the portal.

- Server inherits identity space from the portal
- Server inherits service permissions based on item permissions in the portal
Portal - Server Federation - Hosted Servers

- Clients authenticate against the portal

- Servers identify their portal via an `owningSystemURL` property (`.../rest/info`)

- Clients obtain a server specific token from the portal in exchange for a portal token
  - Use `generateToken on portal passing in serverUrl`

- Portal will issue server specific token only if the server is trusted (server was registered by a portal admin)

- Client Side Identity Managers take care of this flow include getting server specific tokens