Exploring Microsoft Entra: In-Depth Look at Authentication Flows

— Cédric Braekevelt

. Who am I?

- Cédric Braekevelt
- Azure & DevOps Engineer @ Lebon.IT
- Focused on
  - Entra (Identity)
  - Azure Governance / Landing Zones
  - IaC
  - Scripting

Agenda

01 Basics

02 Authorization Code Flow

03 Client Credentials Flow

04 Device Code Flow

05 Managed Identity Flow

01. Basics

- Reason & goal
- ADAL or MSAL
- Components
- Tokens

Reason

- Powershell
- Connect-AzAccount
  - Microsoft Azure PowerShell
- Connect-MgGraph
  - Microsoft Graph Powershell
- Azure CLI
- Azure Portal
- Admin Portals
- ...

. Goal

- List ARC-Enabled server
- 7 different scenario’s
- Interactive / Non-Interactive

. Abstraction

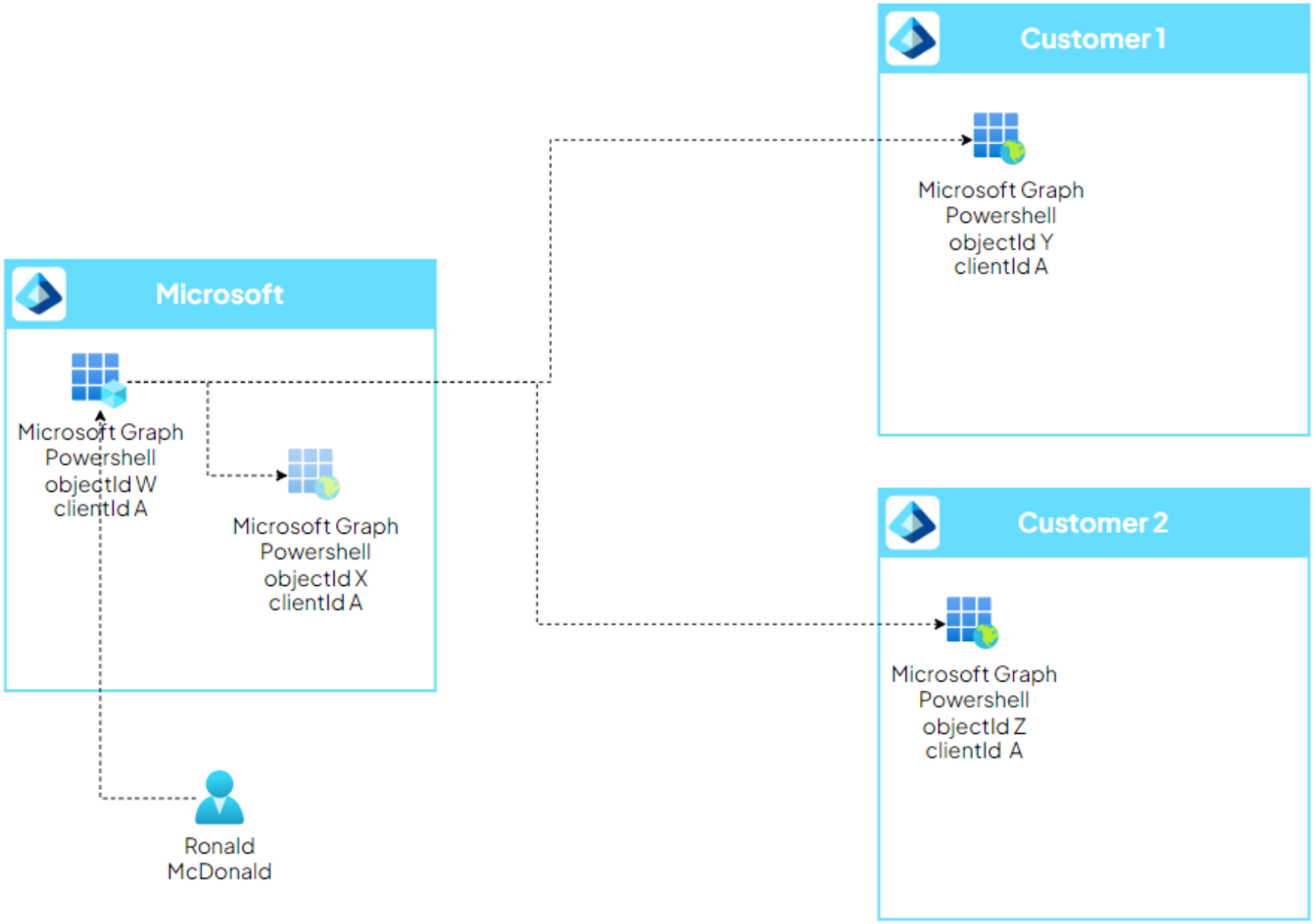
- ADAL
- MSAL
  - Platforms
    - .NET
    - Java (Script)
    - Python
    - …
  - Scenarios
    - Web API
    - SPA
    - Web Apps
    - Mobile Apps
    - Server Apps
  - Done for you!

Components

- Entra Id
  - Directory service

- App Registration | Application
  - Secrets, Certificates, Federated Credentials
  - Token Configuration
  - Multi-tenancy
  - Login page branding

- Enterprise Application | Service Principal
  - Single tenant
  - Allowed Users / Groups
  - SSO settings
  - Consent


Customer 1
Microsoft Graph
Powershell
objectId Y
clientId A
Microsoft
Microsoft Graph
Powershell
objectId W
clientId A
Microsoft Graph
Powershell
objectId X
clientId A
Customer 2
Microsoft Graph
Powershell
objectId Z
clientId A
Ronald
McDonald

Tokens

- Access token
  - General name
  - Access to resources
- ID Token
  - Information about the user
- Refresh Token
  - Allow user to renew access token
- Bearer token
  - Type of access token
  - Allows the **bearer** of the token access
  - No need for username,...

02. Authorization Code Flow

- Type: Interactive login

- Identity: User

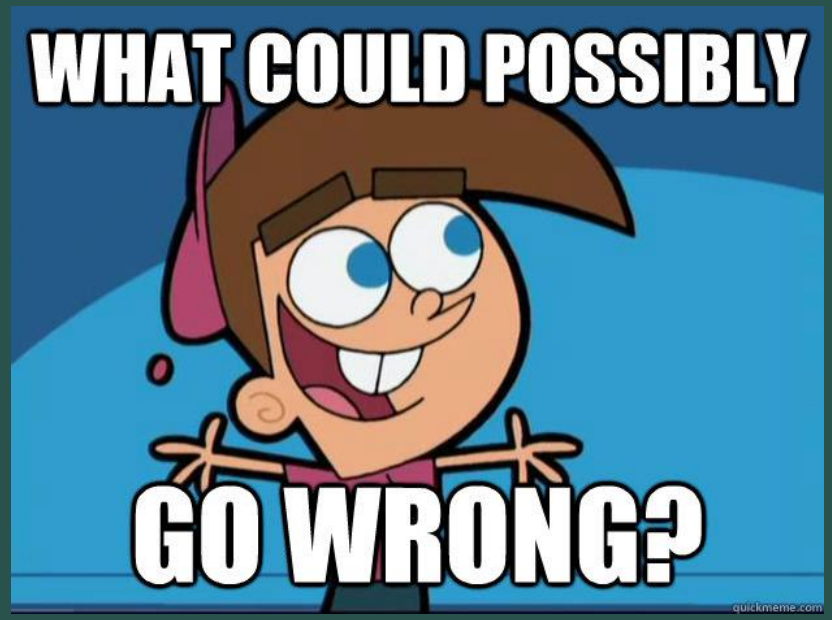
- Use case:
  - Single Page Web Application (React, Angular)
  - Server Web Application (ASP.NET)
  - Desktop & Mobile Applications (.NET [Framework])

- Special:
  - Proof Key for Code Exchange (MiTM)

. Technical

- 2 steps
  - /authorize
    - PKCE: Generate Code Challenge (Code Verifier -> hashed)
    - Listen on localhost port 8400
    - Interactive login via browser
      - Client Id / Tenant Id
      - Redirect URL
      - Scope(s)
      - PKCE: Code Challenge
    - Wait for browser to return authorization code
  - /token
    - Non-interactive API call
      - Client Id / Tenant Id
      - Authorization Code
      - PKCE: Code Verifier
    - Returns bearer access token

. Demo!



- Azure Powershell Enterprise App

- Scope: openid + profile + email + offline_access
  - Basic user identification data
  - offline_access provides refresh token
- PKCE implemented
- Retrieve resources in resource group

03. Client Credential Flow

- Type: Non-Interactive login
- Identity: Service Principal
- Use case:
  - Server to server interaction
- Special:
  - Shared Secret
  - Certificate
  - Federated Credential

Technical

- 1 step
  - /token
    - In a regular scenario only 1 non-interactive API call
      - Client Id / Tenant Id
      - Scope(s)
      - Client Secret / JWT token / Federated token

- Federated credentials
  - Platform specific federated token retrieval
  - “Convert” federated token into Entra access token

Secret vs Certificate

- Secret based authentication
  - Max 2 year via GUI
  - No limit via Graph
  - Very simple | unsecure
  - Key rotation | management!

- Certificate based authentication
  - Upload public key to app registration
  - Connect with JWT based on private key
    - Headers = Base64 (JSON (Headers with encryption algorithm + Base64 (private key hash)))
    - Payload = Base64 (JSON (Payload with clientId, startdate, enddate, audience and a GUID))
    - SignedPackage = SHA512 (Base64 (Headers . Payload)
    - Token = Base64 (Headers . Payload . SignedPackage)

Federated Credential

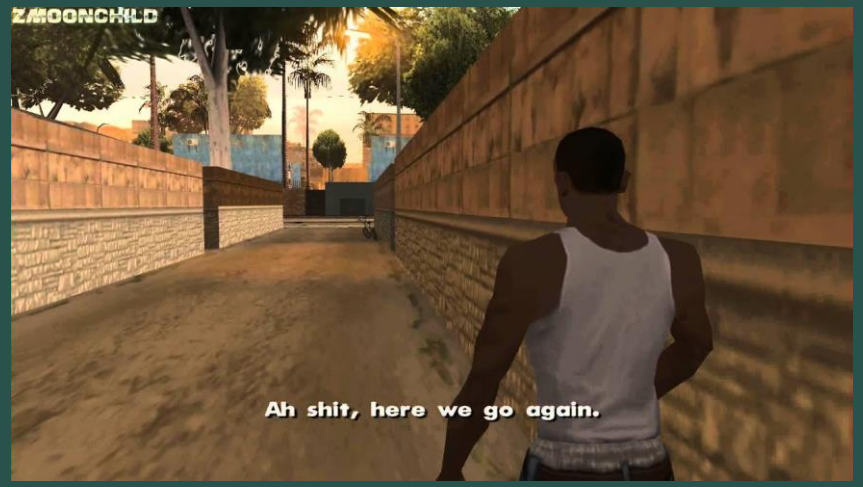
- Endpoint per platform
- Issuer -> specific resource on platform

- Practical
  - Endpoint and federated token mapped as env variables
  - Post request to endpoint return access token

- Github Actions
  - Endpoint: https://token.actions.githubusercontent.com
  - Issuer: repo:{Organization}/{Repository}:{Entity}

- Azure DevOps Pipelines
  - Endpoint: https://vstoken.dev.azure.com/4c9b2d35-c2f5-4bd0-a6ea-f1bd824a40a2
  - Issuer: sc://{Organization}/{Project}/{ServiceConnection}

. Demo!



- Custom Enterprise Applications

- Client Secret Login
- Certificate Login
- Azure DevOps Login (Federated)
- Github Actions Login (Federated)
- Retrieve resources in resource group

04. Device Code Flow

- Type: Interactive login
- Identity: User
- Use case:
  - On devices that lack way to login
    - Printer
    - Smart TV
    - IoT
- Special:
  - Remote login

Technical

- 2 steps
  - /devicecode
    - Generate device code
    - Interactive login via browser
      - https://microsoft.com/devicelogin
      - Insert code in browser
  - /token
    - Non-interactive API call
      - Client Id / Tenant Id
      - Device Code
    - Poll for succes with do-until
    - Returns bearer access token

. Demo!

- Azure Powershell Enterprise App

- Generate a device code
- Login with device code
- Wait for verification
- Retrieve resources in resource group

05. Managed Identity Flow

- Type: Non-Interactive login
- Identity: Service Principal
- Use case:
  - IaaS / PaaS resource to M365 / Azure
- Special:
  - IMDS service (localhost auth service)

Technical

- 2 steps
  - Go to IMDS (localhost) service
  - Retrieve private key of certificate
  - Connect to Entra using private key of certificate
  - Retrieve resources in resource group

. Demo!

- Server **IS** an Enterprise Application

- ARC enrolled server
- Managed Identity
- Retrieve resources in resource group

Thanks!

- Noest
- AZUG

. Questions?