Microsoft

Modern Authentication for the Security Admin

Grace Picking
Mark Morowczynski @markmorow
Program Managers – Microsoft

Agenda

Why Modern Auth?

SAML

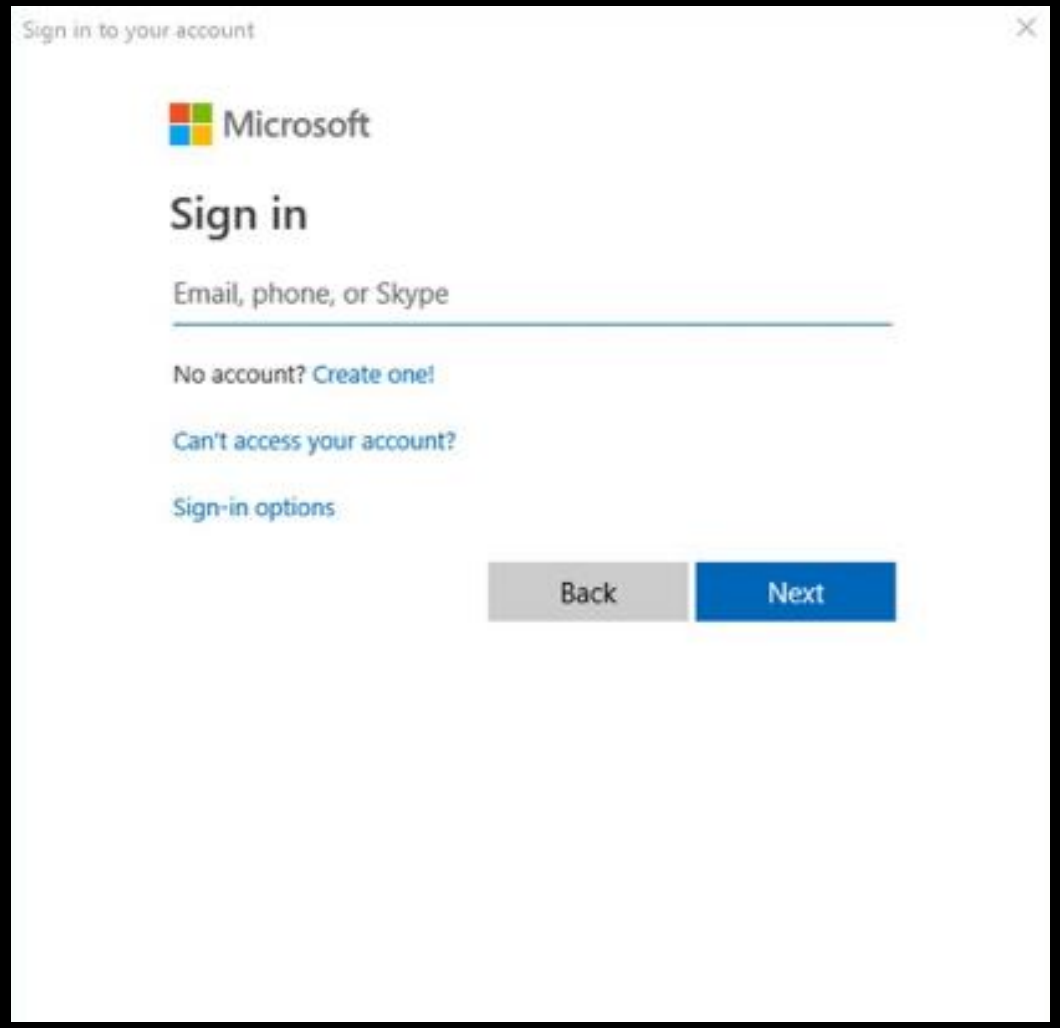
OAuth2

OIDC

Go Do's!

Why You Want To Move To Modern Authentication

- More tools to protect resources
  - Ability to handle an MFA challenge/response
  - Can include additional information about the device (Hybrid Domain Join)
  - Applies to mobile devices as well (MAM Policies)
  - More information an attacker has to guess correctly to spoof (this is good news for us!)
    - User Agent, Application Target
- Not exposing the user credentials to the "client" application



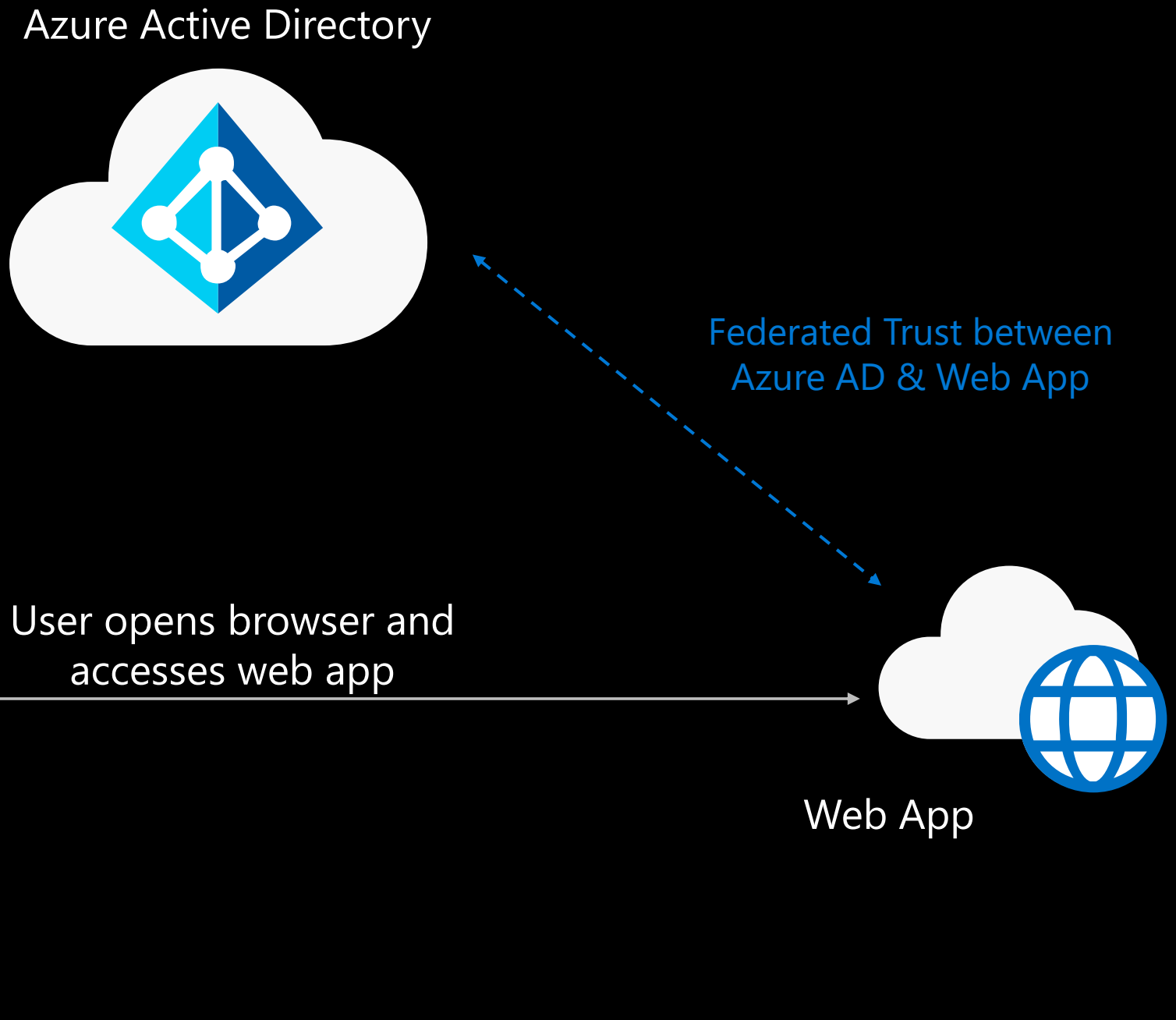
Agenda

Why Modern Auth?
SAML
OAuth2
OIDC

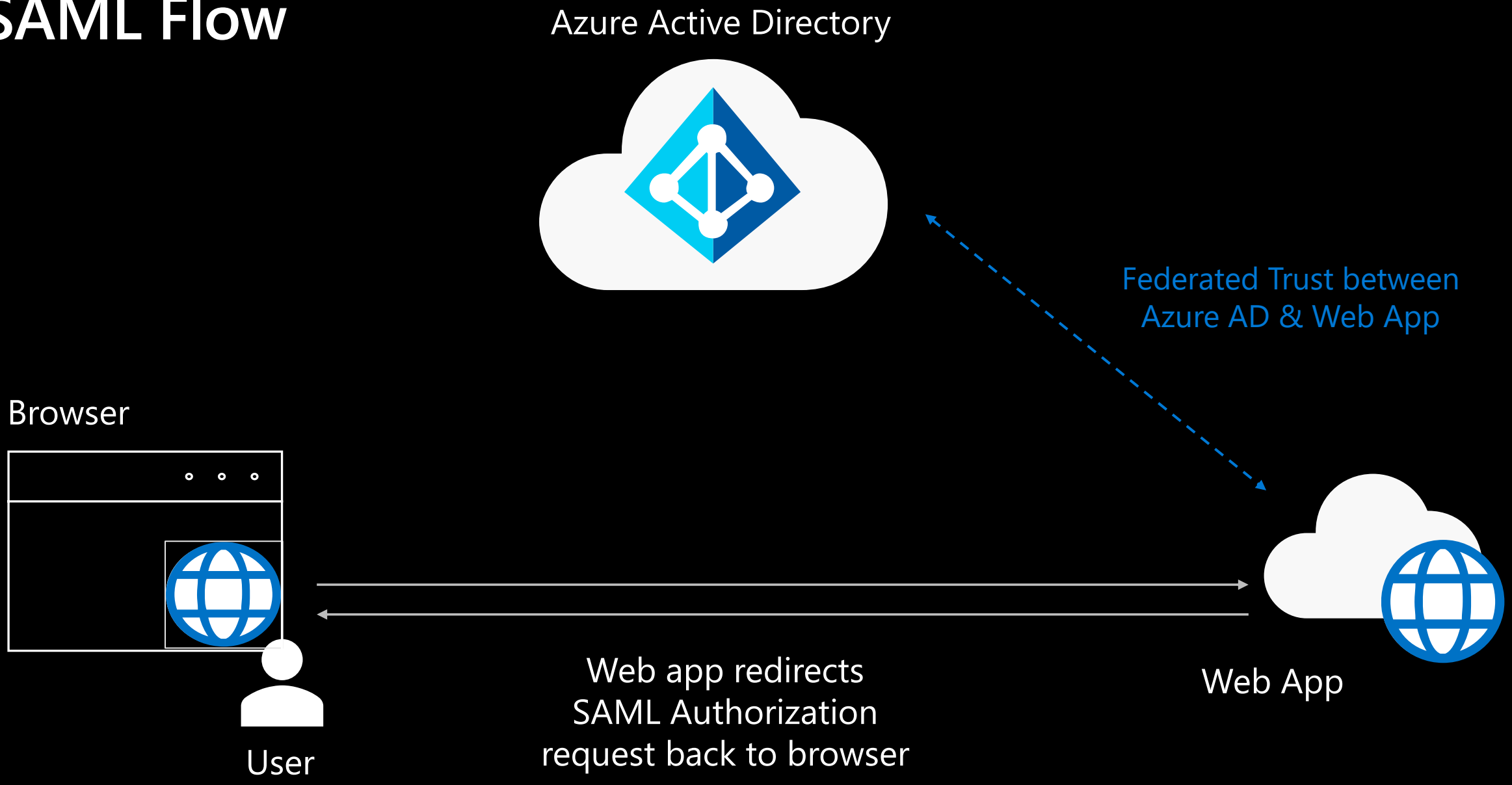
SAML Fundamentals

- Security Assertion Markup Language
- XML-based AuthN standard for SSO to web-based apps
  - Claims in the token can be used for AuthZ
  - Supported by a lot of web apps already
- Reasons SAML is used:
  1. Traditionally easier to implement and been around longer, so more products use it
  2. Still using old on prem IDPs which does not support OIDC/OAuth
  3. You do not want an overhead for customers to consent to an application

SAML Flow


Azure Active Directory
Federated Trust between
Azure AD & Web App
Browser
User opens browser and
accesses web app
Web App
User

SAML Flow


Azure Active Directory
Federated Trust between
Azure AD & Web App
Browser
Web app redirects
SAML Authorization
request back to browser
Web App
User

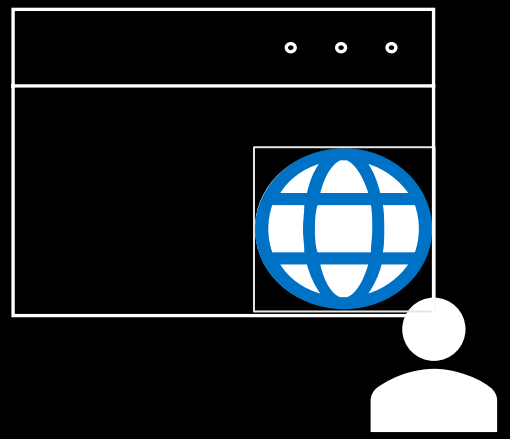
SAML Flow

Azure Active Directory

Browser relays SAML Authorization request to Azure AD

Federated Trust between Azure AD & Web App

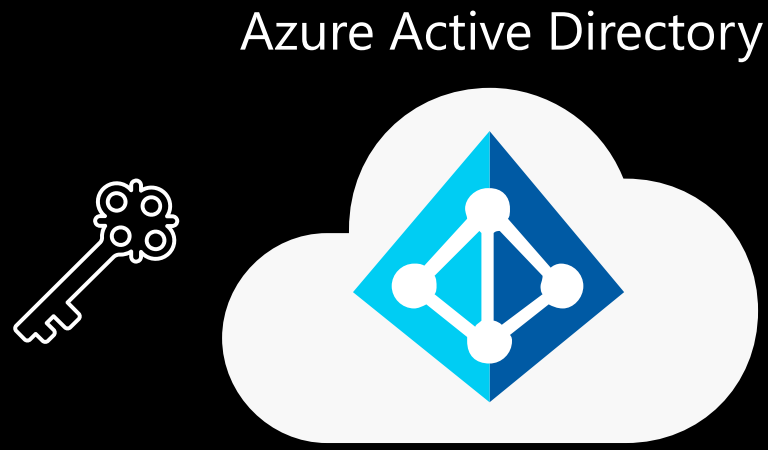
Browser



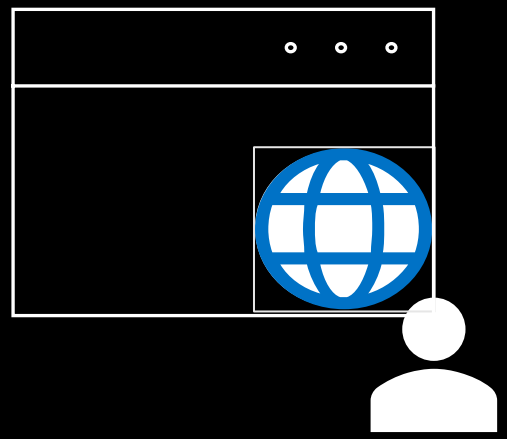
User

Web App

SAML Flow



Federated Trust between
Azure AD & Web App



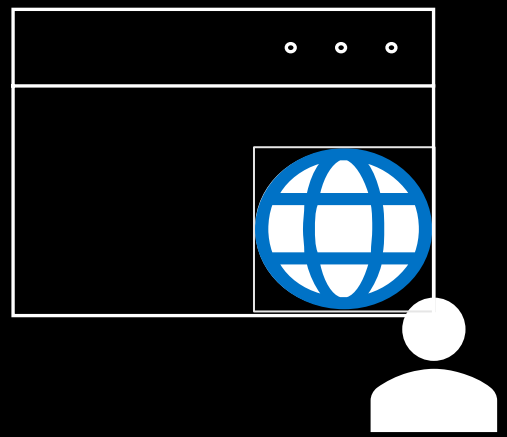
SAML Flow

Azure Active Directory

SAML tokens are sent back to the browser

Federated Trust between Azure AD & Web App

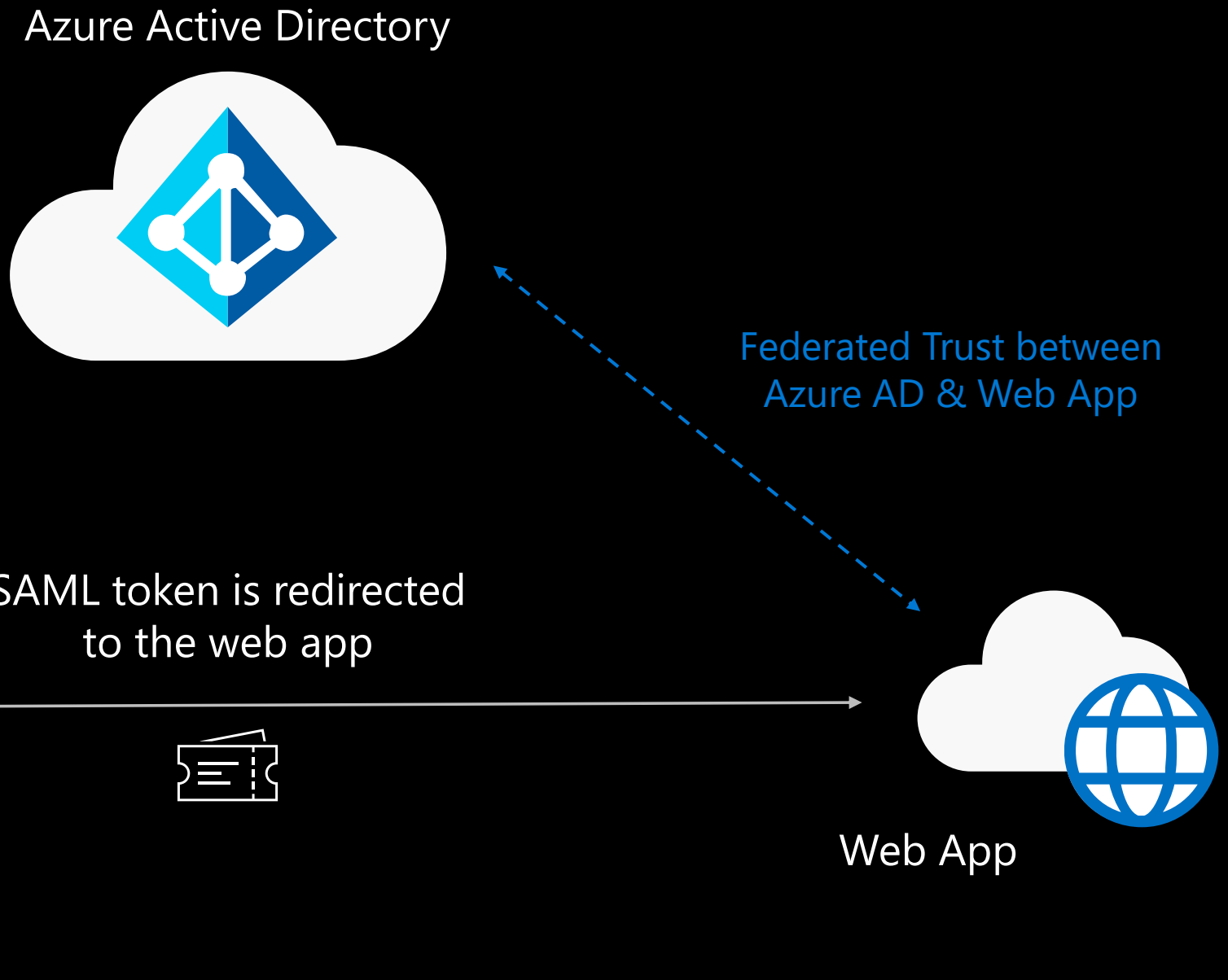
Browser



User

Web App

SAML Flow

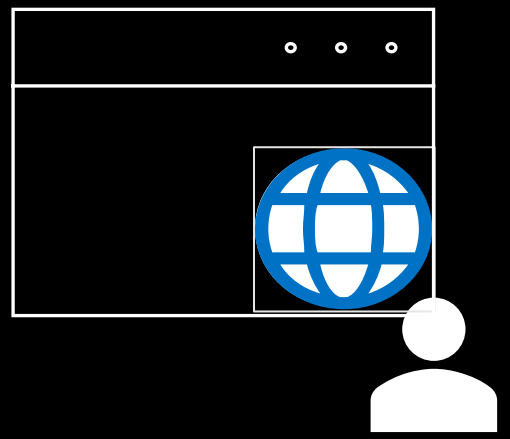

Azure Active Directory
Federated Trust between
Azure AD & Web App
Browser
SAML token is redirected
to the web app
Web App
User

SAML Flow

Azure Active Directory

Federated Trust between
Azure AD & Web App

Browser



User

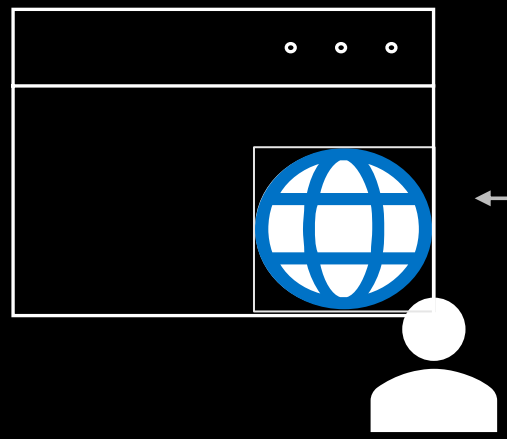
Web app validates SAML
response and token

Web App

SAML Flow

Federated Trust between
Azure AD & Web App

Browser



Web app returns the
requested page to user

Web App

User

Two Ways to Initiate SAML Flow

- Service Provider Initiated
  - If the user starts at a web app
- Identity Provider Initiated
  - If the user starts at the identity provider
  - For example, in Azure AD, this is the My Apps page

What To Look For As Defenders

- Golden SAML Attack
  - Bad actor compromises a certificate (the SAML assertion) and can forge SAML requests
  - Can then SSO to any service as any user
- Assertion Consumer Service URL (Reply URL) – required for some apps
  - Specifies where the app expects to receive the SAML token
  - If an attacker could compromise this URL, your users could be routed to a malicious app
  - Validate that the request is signed and verify data or limit what reply URLs can be used
- SAML logout for the app
  - If a rule is too complex, the user may not get logged out
  - If the user was on a shared machine, the next user will have the previous user's access
- Token Monitoring
  - Swapping SAML Token for Session Token
    - Ensure inactivity timeout and maximum token lifetime
  - Token sniffing

How to reduce risk

- Protect IDP like you protect your domain controllers
- Protect your certificates
  - Use an HSM
  - Monitoring for certificate expiration, configuration changes, addition of certificates
- Check out our Sec Ops guide:
  - https://aka.ms/AzureADSecOps

Agenda

Why Modern Auth?
SAML
OAuth2
OIDC
Go Do's!

Moloch
@LittleJoeTables

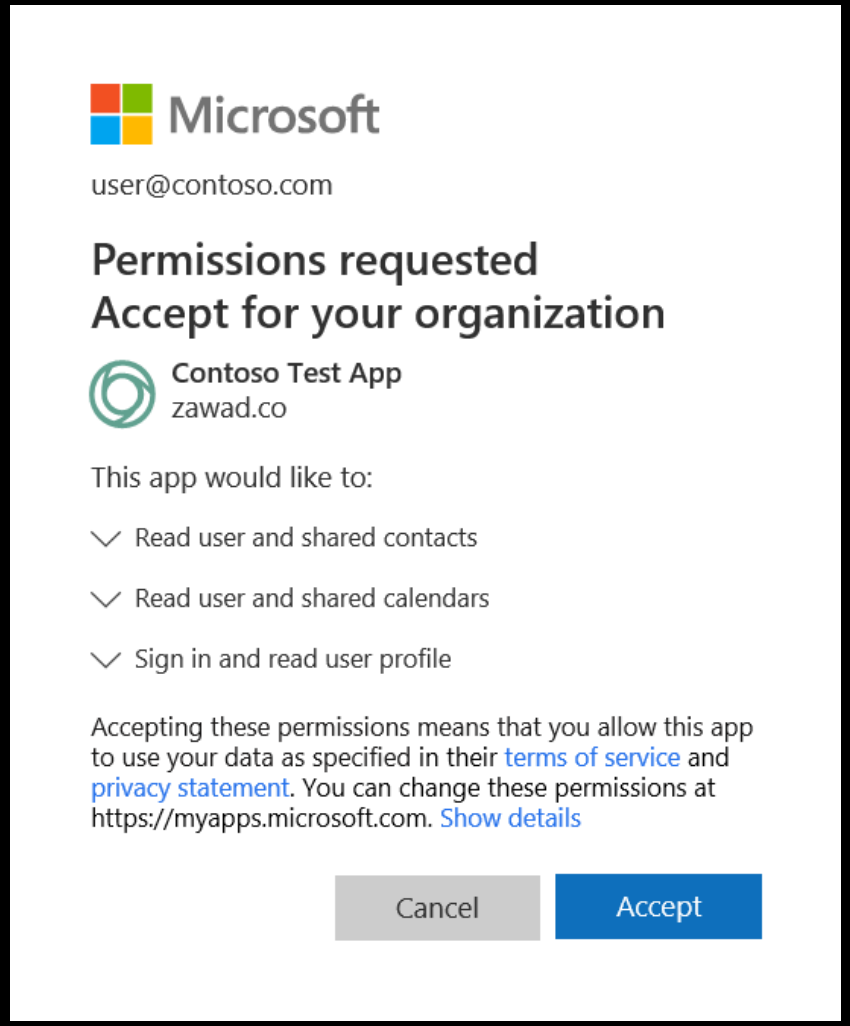
How do developers mess up OAuth2? You only need to understand the OAuth, OIDC, JWT, and JWKS standards, nuances of HTTP redirects, nuances of RSA vs ECDSA, remember to validate the JWT is signed using an expected algorithm, and check the exp, to have a chance of getting it right.

10:51 AM · Feb 28, 2021 · Twitter Web App

82 Retweets **7** Quote Tweets **420** Likes

OAuth Fundamentals

- AuthoriZation framework
  - Really a delegation protocol.
- "Getting the right of access from one component of a system to another."
- Leverages HTTP, tokens, and scopes.



OAuth Components

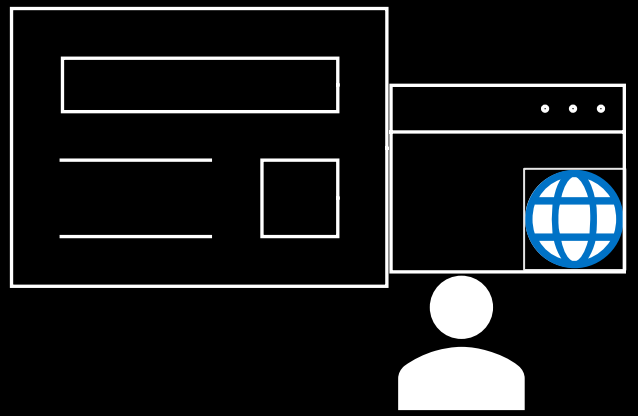
- **Resource Owner**-Usually a person on a “browser”. Has access to an API and can **delegate** access to that API.
- **Protected Resource**-Usually a WebAPI. The thing the resource owner has access to.
- **Client**-Piece of software that is accessing the protected resource on behalf of the resource owner. CONSUMING the WebAPI
- **Authorization Server**-Trusted by the protected resource to issue access tokens to the Client.

- Resource owner credentials never exposed to the Client!

OAuth Components Example

Azure Active Directory- Authorization Server

Application-Client



User-Resource Owner

Authorization Required

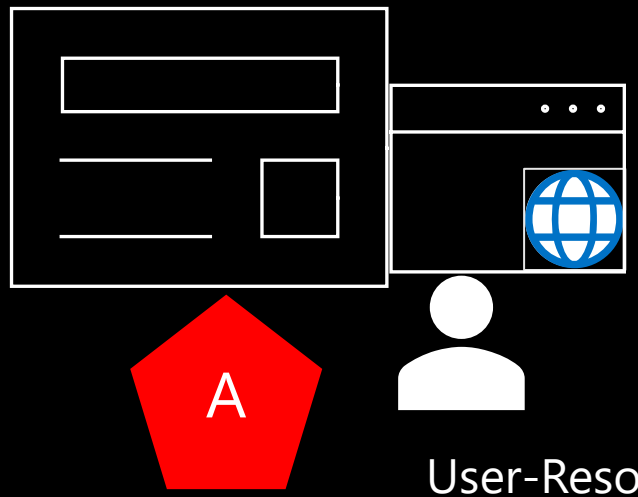
Microsoft Graph User API

Microsoft Graph- Protected Resource

OAuth Components Example

Azure Active Directory- Authorization Server

Application-Client



User-Resource Owner

Authorization Required

Microsoft Graph User API

Microsoft Graph- Protected Resource

Access Tokens

- Frequently called bearer tokens or tokens.
  - Whoever bearers/carries this has the right to use it.
- OAuth doesn’t define a token format or message signature. Not meant to be used outside of HTTPS.
- JWT (JSON Web Token) is a commonly used format, includes ability to sign and encode. RFC 7519 for more info.
- Opaque to the client app. Client has no need to look at the token.
- Authorization server issues access token.
- Protected resource consumes access token.
- Defines what access has been granted to the client. (Ex. User.Read)

Access Token Encoded Example

- eyJ0eXAiOiJKV1QiLCJub25jZSI6Il9wUHVWdVR0TklDbFZXcEVLd2RYX3B5bnhzczdrOTBVM2E3TElCbERqaFUiLCJhbGciOiJSUzI1NiIsIng1dCI6Im5PbzNaRHJPRFhFSzFqS1doWHNsSFJfS1hFZyIsImtpZCI6Im5PbzNaRHJPRFhFSzFqS1doWHNsSFJfS1hFZyJ9.eyJhdWQiOiIwMDAwMDAwMy0wMDAwLTAwMDAtYzAwMC0wMDAwMDAwMDAwMDAiLCJpc3MiOiJodHRwczovL3N0cy53aW5kb3dzLm5ldC9jNzJhMjk1ZC1kN2E1LTQxZWEtYTM1MS1iMTVkZDlmNjcyMTUvIiwiaWF0IjoxNjI2MjE4NjIxLCJuYmYiOjE2MjYyMTg2MjEsImV4cCI6MTYyNjIyMjUyMSwiYWNjdCI6MCwiYWNyIjoiMSIsImFjcnMiOlsidXJuOnVzZXI6cmVnaXN0ZXJzZWN1cml0eWluZm8iLCJ1cm46bWljcm9zb2Z0OnJlcTEiLCJ1cm46bWljcm9zb2Z0OnJlcTIiLCJjMSIsImMyIiwiYzMiLCJjNCIsImM1IiwiYzYiLCJjNyIsImM4IiwiYzkiLCJjMTAiLCJjMTEiLCJjMTIiLCJjMTMiLCJjMTQiLCJjMTUiLCJjMTYiLCJjMTciLCJjMTgiLCJjMTkiLCJjMjAiLCJjMjEiLCJjMjIiLCJjMjMiLCJjMjQiLCJjMjUiXSwiYWlvIjoiQVVRQXUvOFRBQUFBMm9YSHd5TkZJKzZnZDRlWmZxUm1mNDZaQmxZbXBKbStjVUZjV1c4dm5VV1JZRnRpa2tsYklncWM0eGVOQUhVOXhFdURUMFk0Q1VWSE1Bc0FiQzdxaXc9PSIsImFtciI6WyJyc2EiLCJtZmEiXSwiYXBwX2Rpc3BsYXluYW1lIjoiTE9CIFdvcmtzaG9wIERlbW8iLCJhcHBpZCI6IjdkOTJmYTFmLTk3YzQtNDk4OS1iNjIyLThmYzcxNjI5ZTY1MiIsImFwcGlkYWNyIjoiMCIsImZhbWlseV9uYW1lIjoiTWFyc2giLCJnaXZlbl9uYW1lIjoiS3lsZSIsImlkdHlwIjoidXNlciIsImlwYWRkciI6IjE1NC4yMC4xOTYuMTQ5IiwibmFtZSI6Ikt5bGUgTWFyc2giLCJvaWQiOiI4OWQ3NWRiYi1hNjg5LTQ2ZmMtOWE4Zi01YjA1MzYwZTFhNTgiLCJwbGF0ZiI6IjMiLCJwdWlkIjoiMTAwMzIwMDA0OEZCMzg0MiIsInJoIjoiMC5BUzRBWFNrcXg2WFg2a0dqVWJGZDJmWnlGUl82a24zRWw0bEp0aUtQeHhZcDVsSXVBQncuIiwic2NwIjoib3BlbmlkIFBlb3BsZS5SZWFkIHByb2ZpbGUgVXNlci5SZWFkIGVtYWlsIiwic3ViIjoiS25aNjRkV1U1N3ZrTUhXT1NmVHBiOGRSeWFrTUhLSDRCZjhBd3hSQzlGRSIsInRlbmFudF9yZWdpb25fc2NvcGUiOiJOQSIsInRpZCI6ImM3MmEyOTVkLWQ3YTUtNDFlYS1hMzUxLWIxNWRkOWY2NzIxNSIsInVuaXF1ZV9uYW1lIjoia3lsZW1hckBtaWNyb3NvZnRpZGVudGl0eS5kZXYiLCJ1cG4iOiJreWxlbWFyQG1pY3Jvc29mdGlkZW50aXR5LmRldiIsInV0aSI6IlMxbXFKMXlQRWstV2NZSmRKZXpKQUEiLCJ2ZXIiOiIxLjAiLCJ3aWRzIjpbImNmMWMzOGU1LTM2MjEtNDAwNC1hN2NiLTg3OTYyNGRjZWQ3YyIsIml3OWZiZjRkLTNlZjktNDY4OS04MTQzLTc2YjE5NGU4NTUwOSJdLCJ4bXNfc3QiOnsic3ViIjoiU1F0bmdzd1pCSjk1NkZWZTVKYkVybVNBZGZwMWhHRF9NbnNEMEh2RTN2YyJ9LCJ4bXNfdGNkdCI6MTU0NzQwNDUzMH0.VQrBPQfMYPt10YPpHlpjVa1UdYfIDaDvnw5NKU5VicIAmFFLKCi945EHdKKHRQkSf8BjS3ul-7AKz8VhBxzP7LZHLCseHiHj_9MYMJWaOigNkx2m7vFE-h_5I1DT_PGnbWSZdT9c7s9pYzHm1p3YUyp04uX5vOadK0o_fUYNtOy2hfR42GgtJDUKb0el3mQRFfdOruRG5W4jptaKVyTvMqSiEHHjlNYJpsYZMNvidxDQ16ErIzgpfepvjOSZlnSQkZeA4L-zEBSFBcn-hZcQF4KYmUwdBKPT-KiAt0ajZtb2Vurlq9WOtbIrv8Hv-CFDiuTyxsxoo5mwlC7JLIRg3Q

Access Token Decoded Example

- { **"typ"**: "JWT", "nonce": "_pPuVuTtNIClVWpEKwdX_pynxss7k90U3a7LIBlDjhU", "alg": "RS256", "x5t": "nOo3ZDrODXEK1jKWhXslHR_KXEg", "kid": "nOo3ZDrODXEK1jKWhXslHR_KXEg" }.{ **"aud"**: "00000003-0000-0000-c000-000000000000", **"iss"**: "https://sts.windows.net/c72a295d-d7a5-41ea-a351-b15dd9f67215/", "iat": 1626218621, "nbf": 1626218621, "exp": 1626222521, "acct": 0, "acr": "1", "acrs": ["urn:user:registersecurityinfo", "urn:microsoft:req1", "urn:microsoft:req2", "urn:microsoft:req3", "c1", "c2", "c3", "c4", "c5", "c6", "c7", "c8", "c9", "c10", "c11", "c12", "c13", "c14", "c15", "c16", "c17", "c18", "c19", "c20", "c21", "c22", "c23", "c24", "c25"], "aio": "AUQAu/8TAAAA2oXHwyNFI+6gd4eZfqRmf46ZBlYmpJm+cUFcWW8vnUWRYFtikklbIgqc4xeNAHU9xEuDT0Y4CUVHMAsAbC7qiw==", "amr": ["rsa", "mfa"], **"app_displayname"**: "LOB Workshop Demo", "appid": "7d92fa1f-97c4-4989-b622-8fc71629e652", "appidacr": "0", "family_name": "Marsh", "given_name": "Kyle", "idtyp": "user", "ipaddr": "154.20.196.149", "name": "Kyle Marsh", "oid": "89d75dbb-a689-46fc-9a8f-5b05360e1a58", "platf": "3", "puid": "1003200048FB3842", "rh": "0.AS4AXSkqx6XX6kGjUbFd2fZyFR_6kn3El4lJtiKPxxYp5lIuABw.", "scp": "openid People.Read profile User.Read email", "sub": "KnZ64dWU57vkMHWOSfTpb8dRyakMHKH4Bf8AwxRC9FE", "tenant_region_scope": "NA", "tid": "c72a295d-d7a5-41ea-a351-b15dd9f67215", "unique_name": "kylemar@microsoftidentity.dev", **"upn"**: "kylemar@microsoftidentity.dev", "uti": "S1mqJ1yPEk-WcYJdJezJAA", "ver": "1.0", "wids": ["cf1c38e5-3621-4004-a7cb-879624dced7c", "b79fbf4d-3ef9-4689-8143-76b194e85509"], "xms_st": { "sub": "SQtngswZBJ956FVe5JbErmSAdfp1hGD_MnsD0HvE3vc" }, "xms_tcdt": 1547404530 }.[Signature]

The Right OAuth Flows For The Job

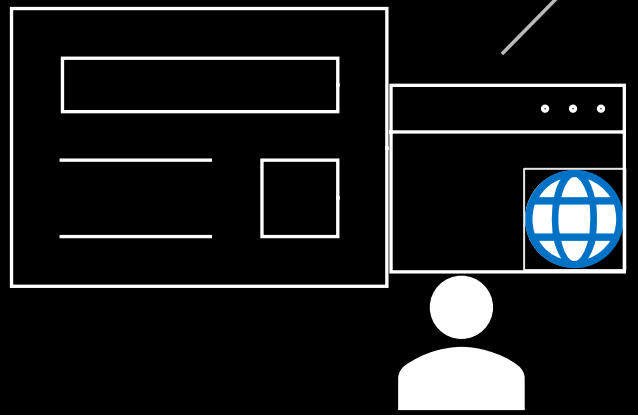
- **Authorization code grant**-Majority of app types will use this
- **Implicit grant**-Used for Single Page Apps (SPA). Move to authorization code flow if possible.
- **On-behalf-of grant-** Client calls a WebAPI and that WebAPI needs to call ANOTHER WebAPI.
- **Device code grant**-Input constrained devices, IoT, printers, etc.
- **Client credentials grant**-Deamons or service accounts. Service to Service calls
- **Resource owner password credentials grant (ROPC)**-User gives password to client app. **DO NOT USE** unless absolutely have to and understand the risks.

OAuth Authorization Code Flow Simplified

Azure Active Directory- Authorization Server

Client requests Authorization to Resource owner- redirects to Authorization endpoint

Application-Client



User-Resource Owner

Authorization Required

Microsoft Graph User API

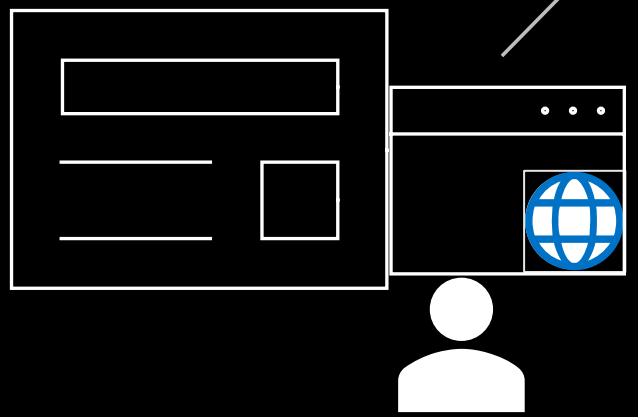
Microsoft Graph- Protected Resource

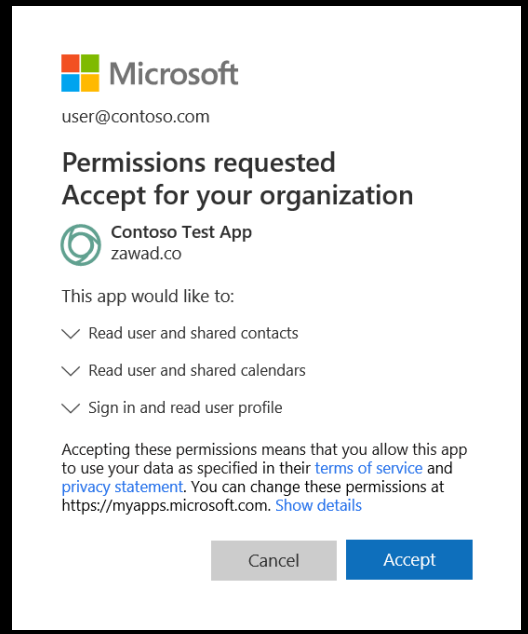
OAuth Authorization Code Flow Simplified Cont.

Resource owner authenticates, authorizes client.

Azure Active Directory- Authorization Server

Application-Client





Authorization Required

Microsoft Graph User API

User-Resource Owner

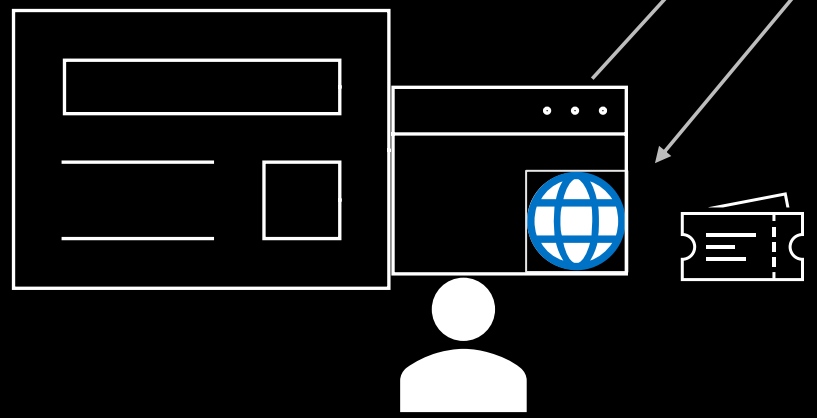
Microsoft Graph- Protected Resource

OAuth Authorization Code Flow Simplified Cont.

Azure Active Directory- Authorization Server

Authorization Server redirects resource owner back to the client with authorization code

Application-Client



User-Resource Owner

Authorization Required

Microsoft Graph User API

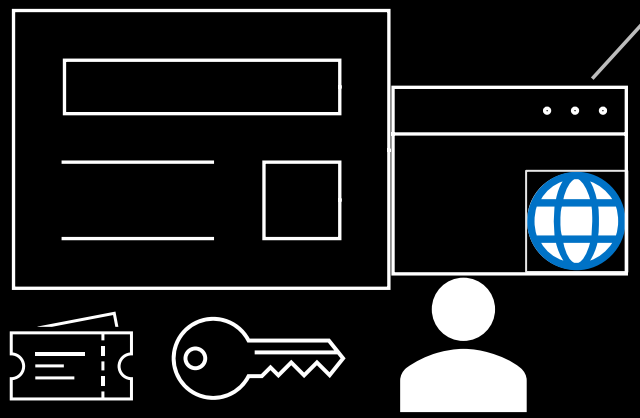
Microsoft Graph- Protected Resource

OAuth Authorization Code Flow Simplified Cont.

Client send authorization code and its own client credentials to Authorization Server

Azure Active Directory- Authorization Server

Application-Client



User-Resource Owner

Authorization Required

Microsoft Graph User API

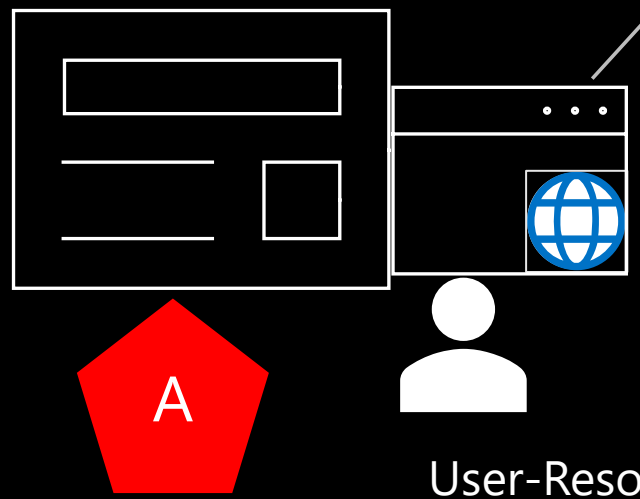
Microsoft Graph- Protected Resource

OAuth Authorization Code Flow Simplified Cont.

Authorization Server issues OAuth Access Token to client

Azure Active Directory- Authorization Server

Application-Client



User-Resource Owner

Authorization Required

Microsoft Graph User API

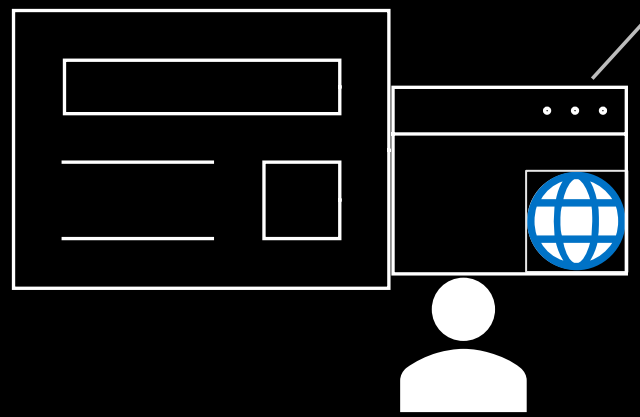
Microsoft Graph- Protected Resource

OAuth Authorization Code Flow Simplified Cont.

Client access protected resource using the Access Token

Azure Active Directory- Authorization Server

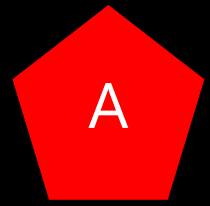
Application-Client



User-Resource Owner

Authorization Required

Microsoft Graph User API



Microsoft Graph- Protected Resource

OAuth Flow Recap

- Resource owner never provided credentials to the client app.
  - Only to the Authorization Server (AAD)
- Resource owner delegated the permissions needed for client
  - Provided in the scope of the access token.
- Client accessed protected resource with access token
  - Protected resource trusts the Authorization server

- What about that Resource Owner Password Credentials Grant (ROPC)?

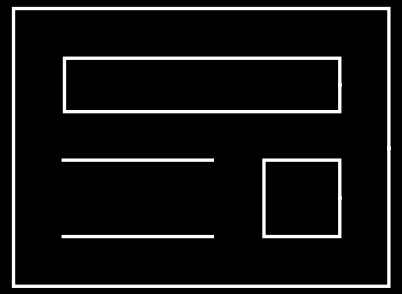
ROPC Code Flow

Resource owner gives user name and password to client, client uses these to get access token.
Client CACHES the username/password!

Avoid if possible!

Azure Active Directory- Authorization Server

Application-Client



User-Resource Owner

Authorization Required

Microsoft Graph User API

Microsoft Graph- Protected Resource

Agenda

Why Modern Auth?
SAML
OAuth2
OIDC
Go Do's!

OpenID Connect

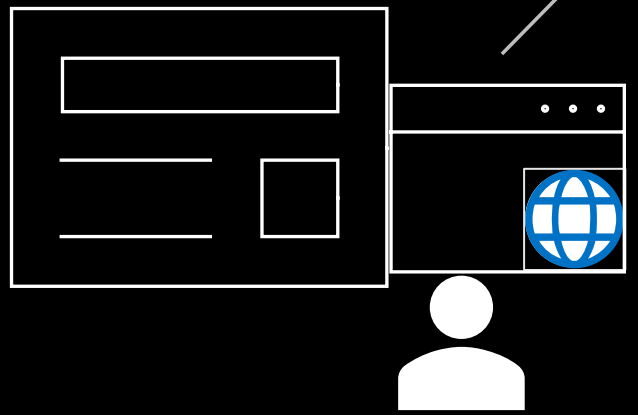
- Open standard built on top of OAuth2 to perform user authe**N**tication.
- Uses JWT with JSON Object Signing and Encrypting (JOSE)
- Get an ID token along side the access token for OAuth2
  - Client is now a Relying Party
- Also has claims like who issued the token, who the subject of the token is, who the audience the token is for, and how long the token is good for.

OpenID Connect Flow Simplified

Azure Active Directory- Authorization Server

Client requests Authorization to Resource owner- redirects to Authorization endpoint

Application-Client



User-Resource Owner

Authorization Required

Microsoft Graph User API

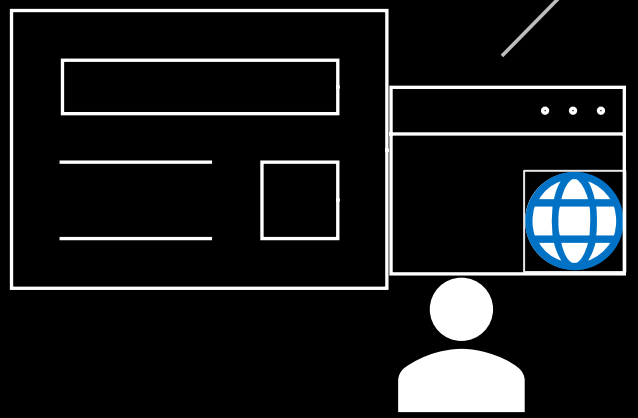
Microsoft Graph- Protected Resource

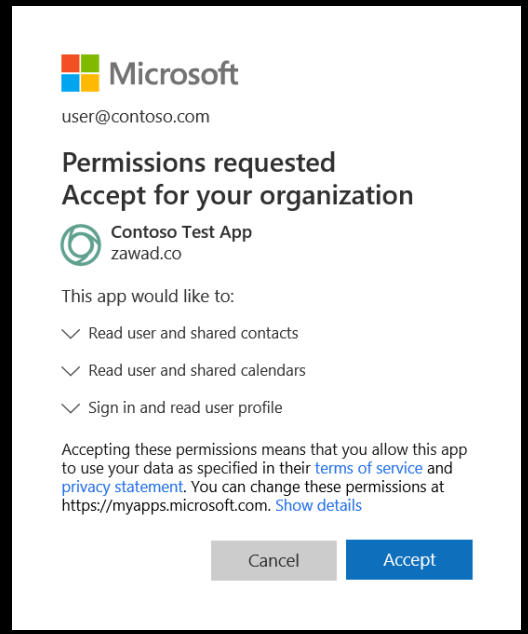
OpenID Connect Flow Simplified Simplified Cont.

Resource owner authenticates, authorizes client.

Azure Active Directory- Authorization Server

Application-Client





Authorization Required

Microsoft Graph User API

User-Resource Owner

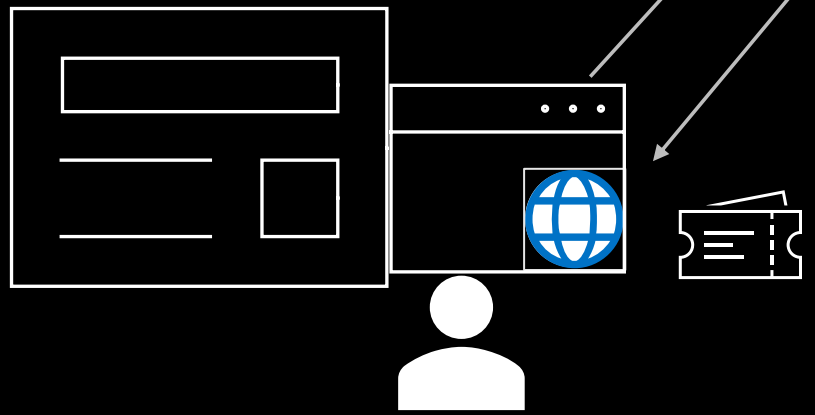
Microsoft Graph- Protected Resource

OpenID Connect Flow Simplified Cont.

Azure Active Directory- Authorization Server

Authorization Server redirects resource owner back to the client with authorization code

Application-Client



User-Resource Owner

Authorization Required

Microsoft Graph User API

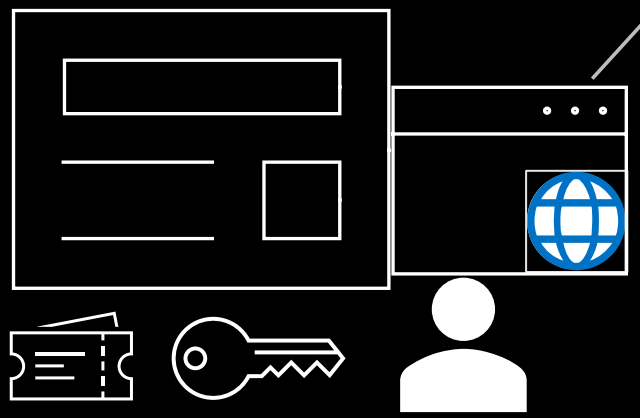
Microsoft Graph- Protected Resource

OpenID Connect Flow Simplified Cont.

Client send authorization code and its own client credentials to Authorization Server

Azure Active Directory- Authorization Server

Application-Client



User-Resource Owner

Authorization Required

Microsoft Graph User API

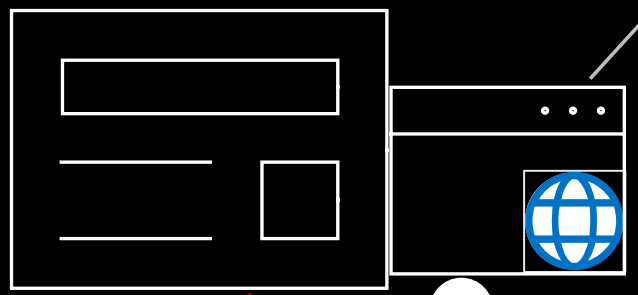
Microsoft Graph- Protected Resource

OpenID Connect Flow Simplified Cont.

Authorization Server issues
OAuth Access and ID
Token to client

Azure Active Directory- Authorization Server

Application-Client



User-Resource Owner

Authorization Required

Microsoft Graph User API

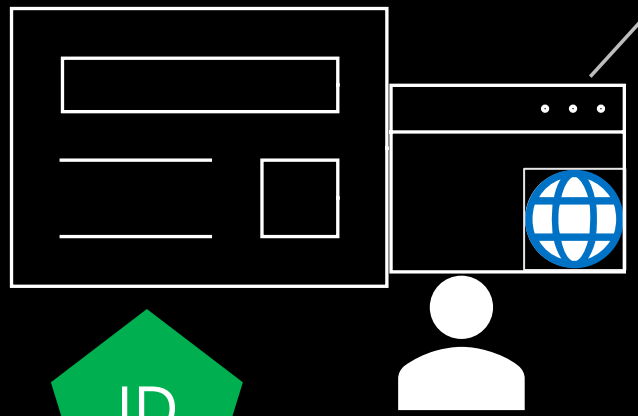
Microsoft Graph-
Protected Resource

OpenID Connect Flow Simplified Cont.

Client access protected resource using the Access Token

Azure Active Directory- Authorization Server

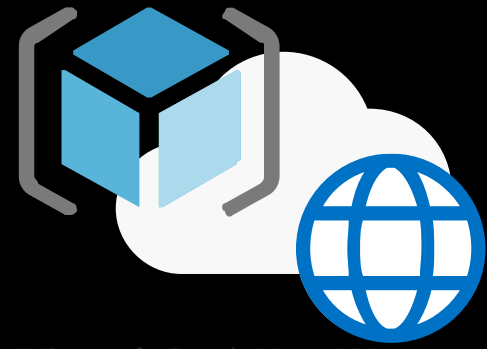
Application-Client



ID

User-Resource Owner

Authorization Required



Microsoft Graph User API

A

Microsoft Graph- Protected Resource

What To Look For As Defenders

- Ask ISV to support OpenID Connect. No more legacy protocols!
  - Gives us additional controls!
- Use an OAuth/OpenID Connect Library, do not let your devs roll their own.
  - If using Azure AD, MSAL library handles lots of this for you. https://aka.ms/aaddev
- Do not use ROPC flow unless you absolutely have to and trust the client app.
- Ensure HTTPS is used and protect access tokens.
  - Whoever bearers/carries this has the right to use it.
- Focus on least privileges on application consent
  - For your LOB apps & ISV
  - Emerging attack https://aka.ms/BSidesCT2020AppConsent

Agenda

Why Modern Auth?
SAML
OAuth2
OIDC
Go Do's!

Go Do's!

- Start moving your apps to modern auth
  - OpenIDConnect or OAuth2 if possible, if not SAML
- Protect your IDP like your DCs, Protect and monitor your certificates
  - Ensure inactivity timeout and maximum token lifetime
  - https://aka.ms/AzureADSecOps
- Use the correct Oauth2/OpenID Connect flow for the job
  - Use a library, don't roll your own. MSAL for Azure Active Directory
  - Try to move away from ROPC if you've already have apps using this
- Ensure least privilege is being followed by internal and ISV apps
  - Look at current consented apps for suspicious apps
  - Update IR playbooks for this type of attack https://aka.ms/IRPlaybooks
- Go deeper on OAuth2/OpenID Connect/SAML/JWT
  - This is the new NTLM/Kerberos/Tickets

Q&A

https://aka.ms/SansBlueTeamRecruit2021

Grace Picking
Mark Morowczynski @markmorow
Program Managers – Microsoft