Microsoft

Modern Authentication for the Security Admin

Bailey Bercik @baileybercik
Mark Morowczynski @markmorow
Program Managers – Microsoft

Agenda

Why Modern Auth?

SAML

OAuth2

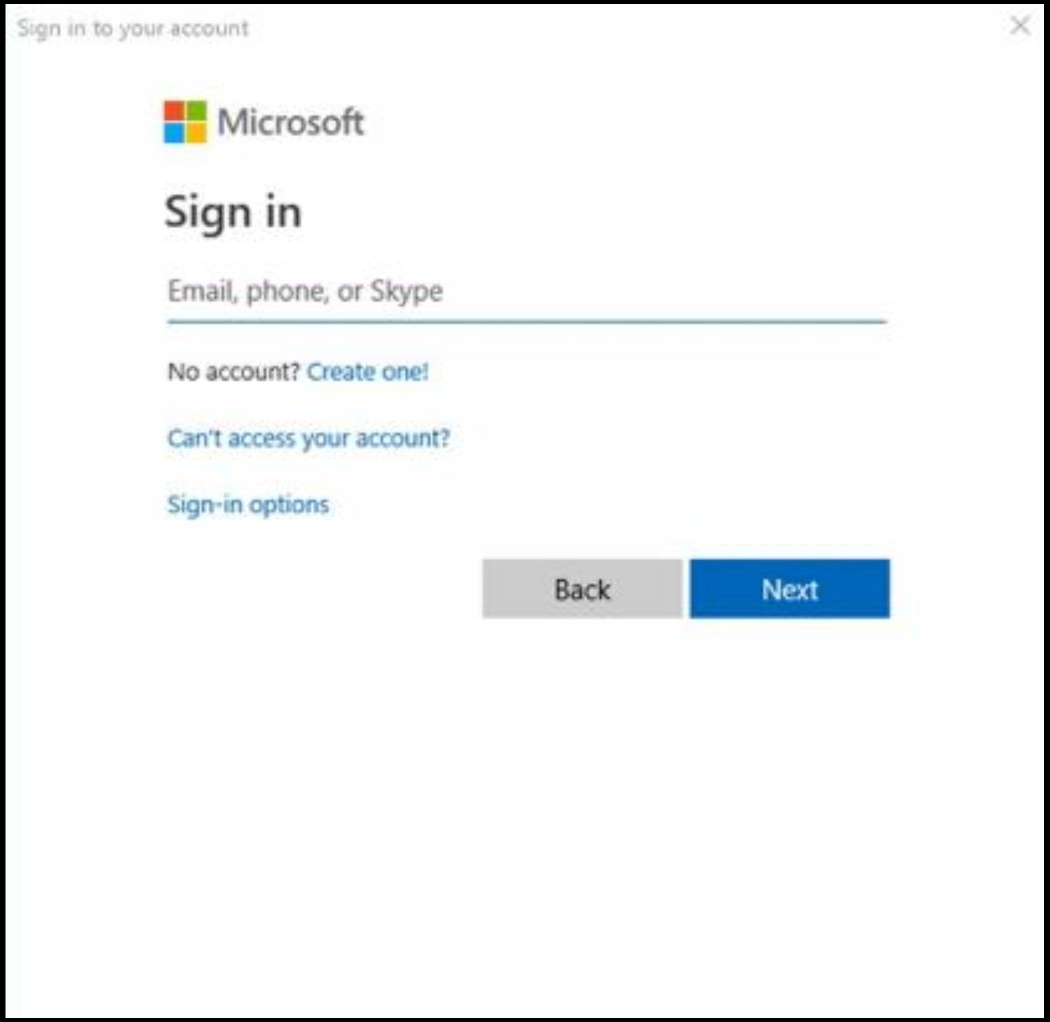
OIDC

App Consent Phishing Attack

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

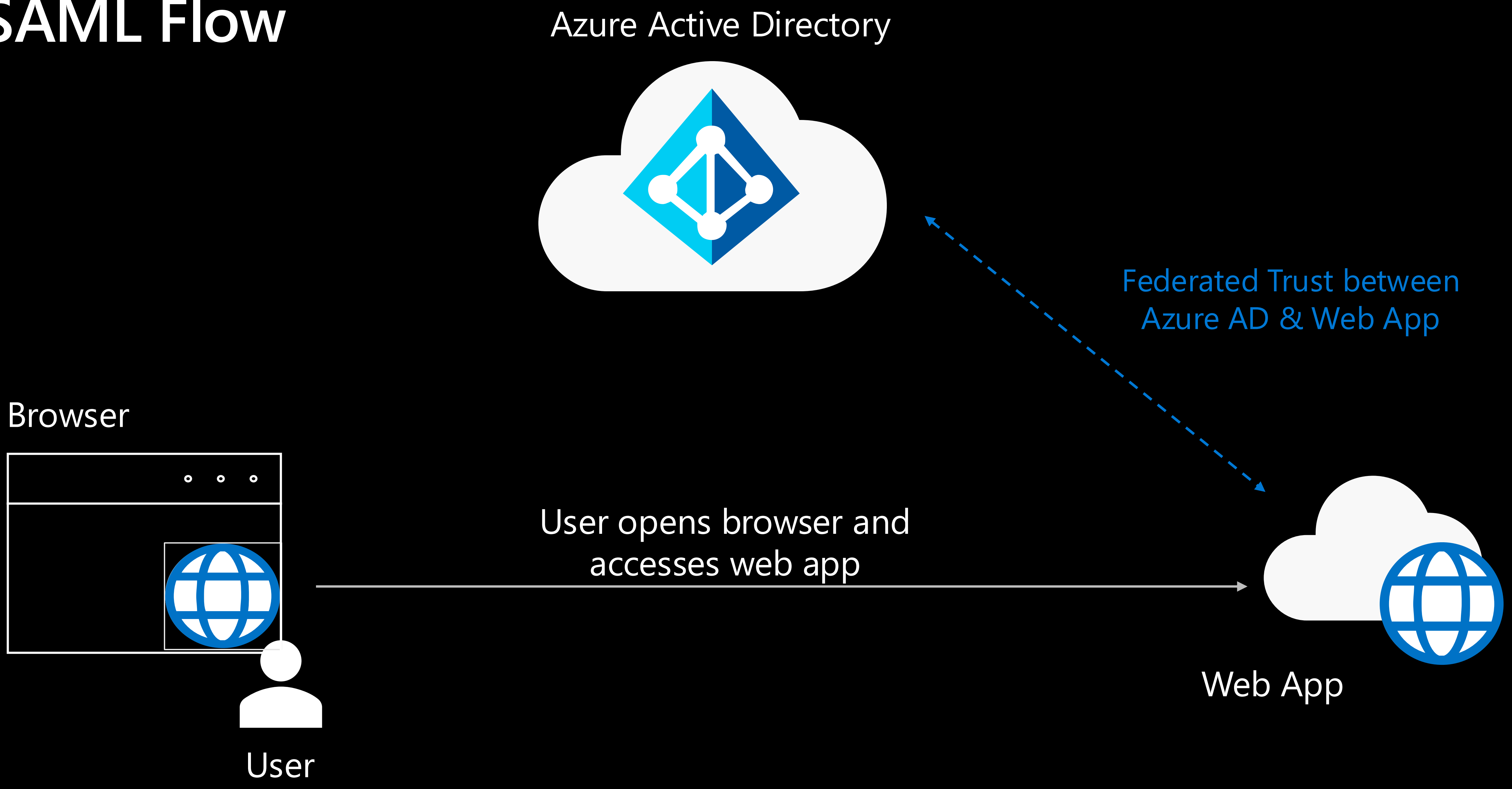
OIDC

App Consent Phishing Attack

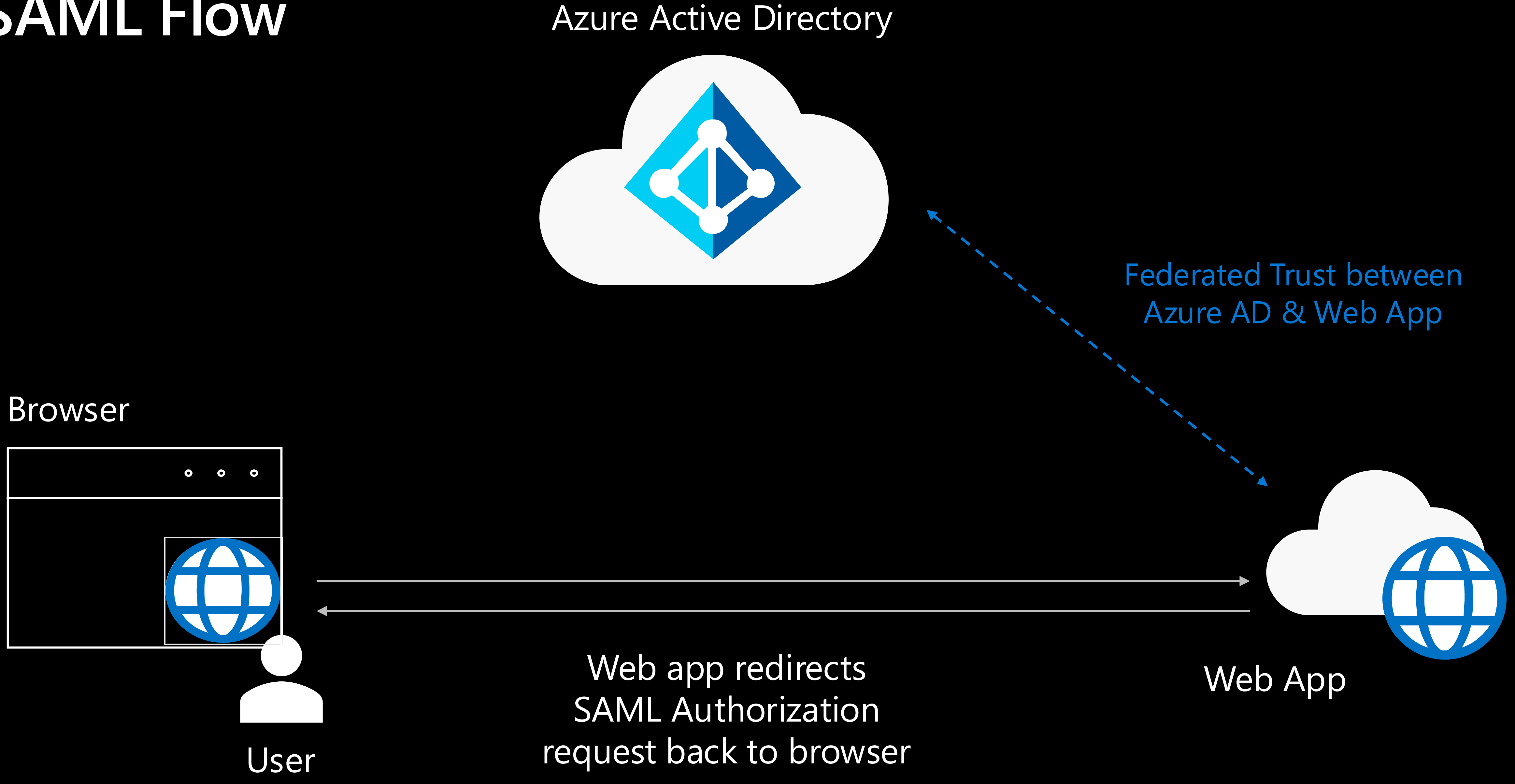
SAML Fundamentals

- 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

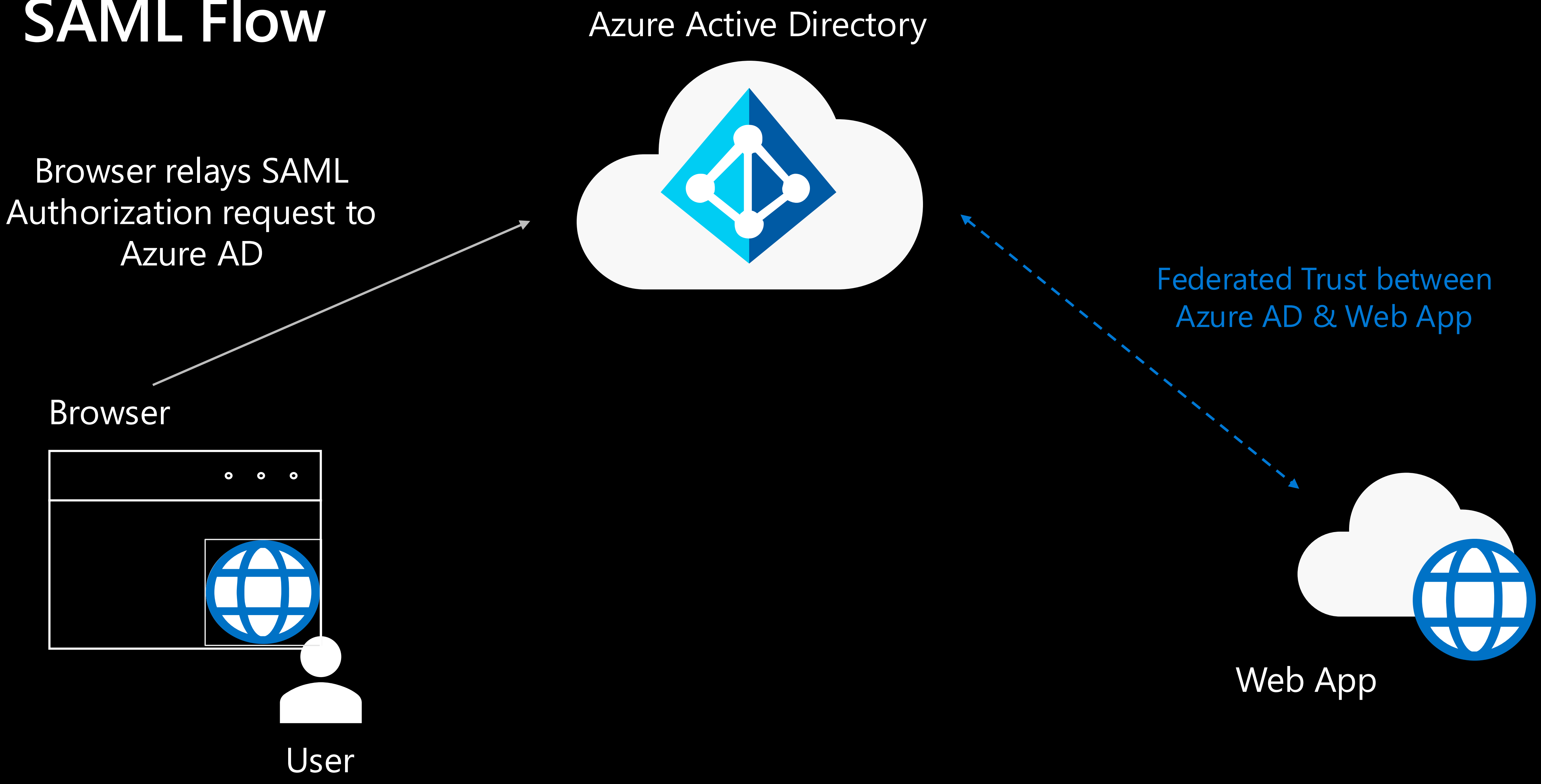
SAML Flow


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

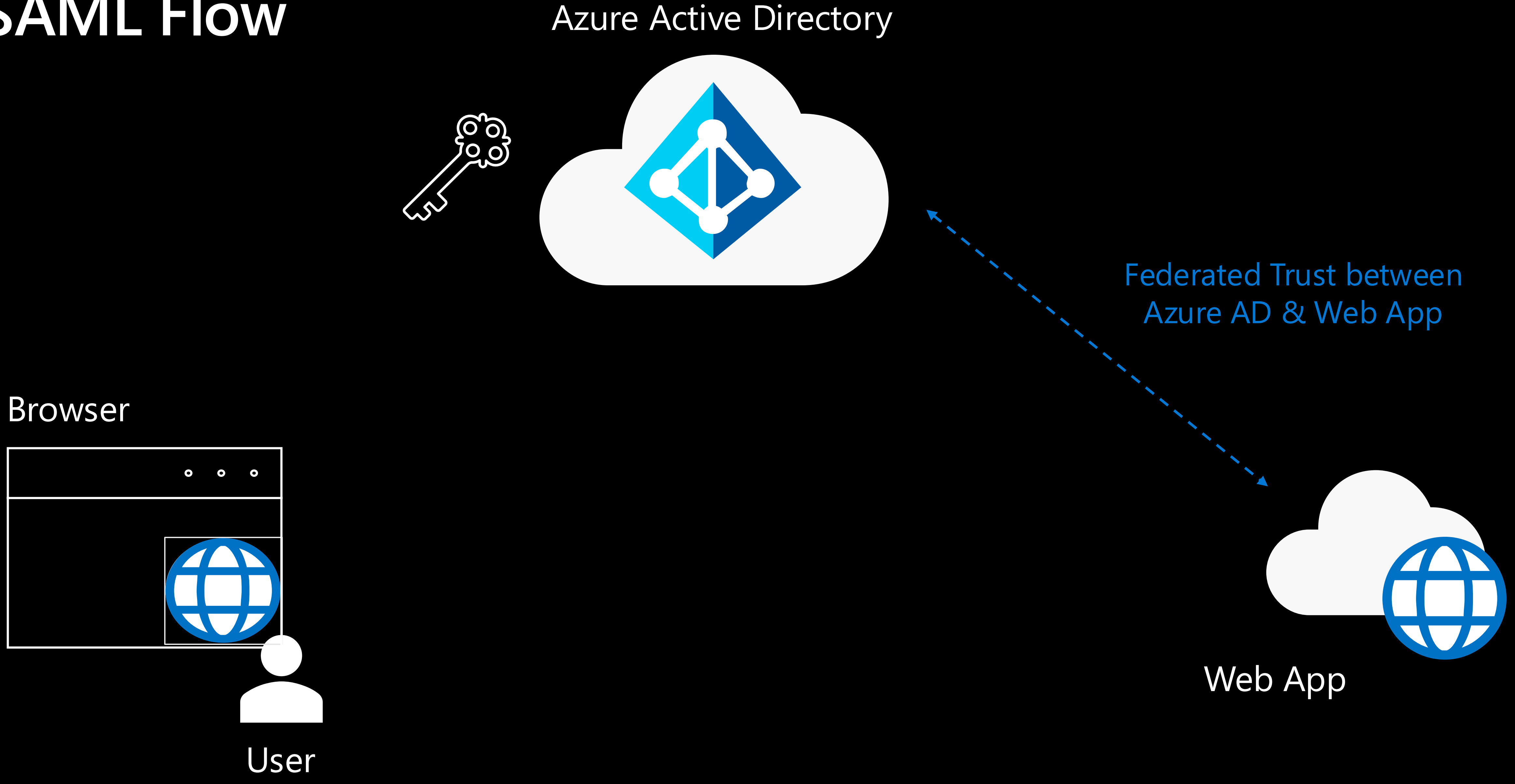
SAML Flow


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

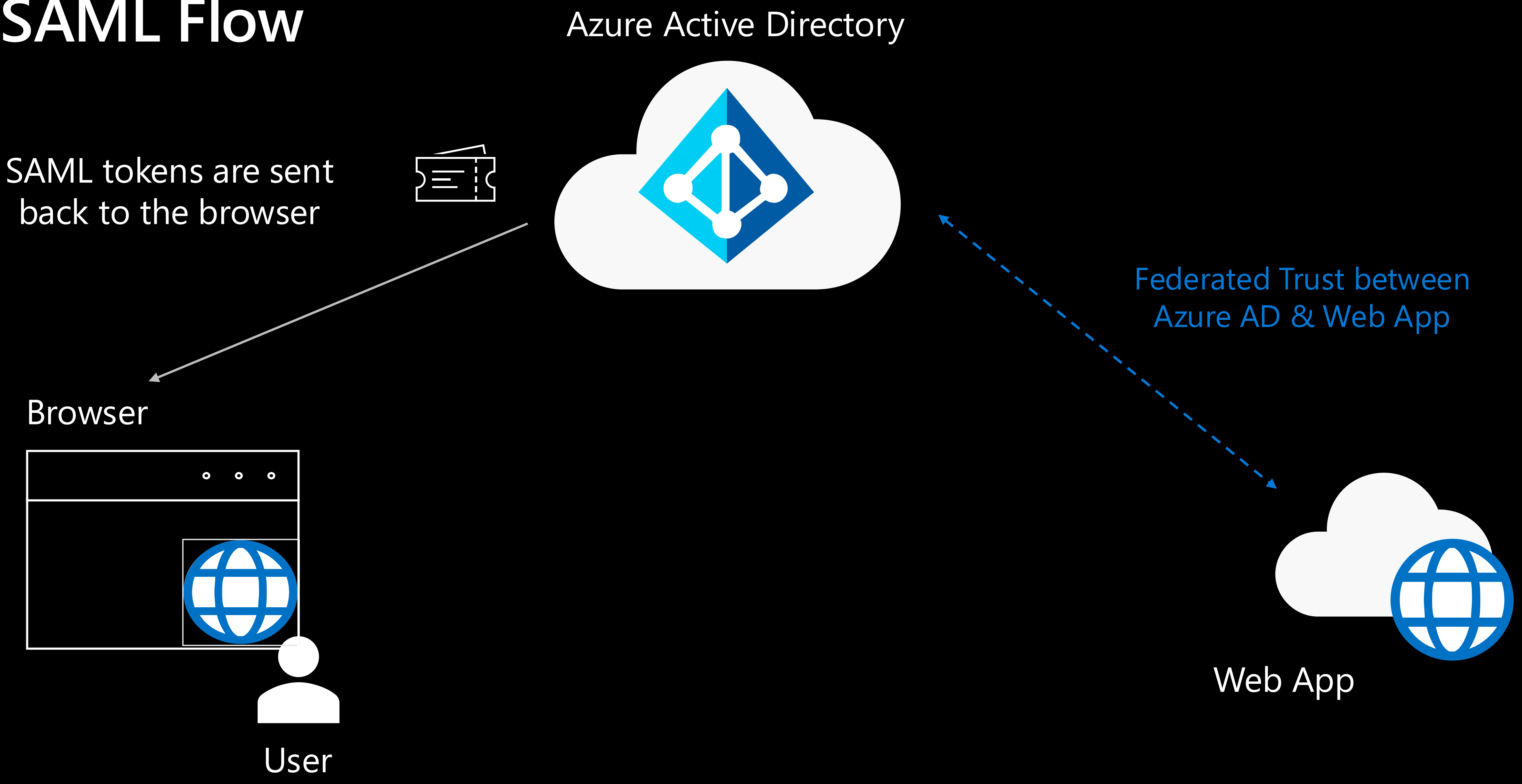
SAML Flow


Azure Active Directory
Browser relays SAML Authorization request to Azure AD
Federated Trust between Azure AD & Web App
Browser
User
Web App

SAML Flow

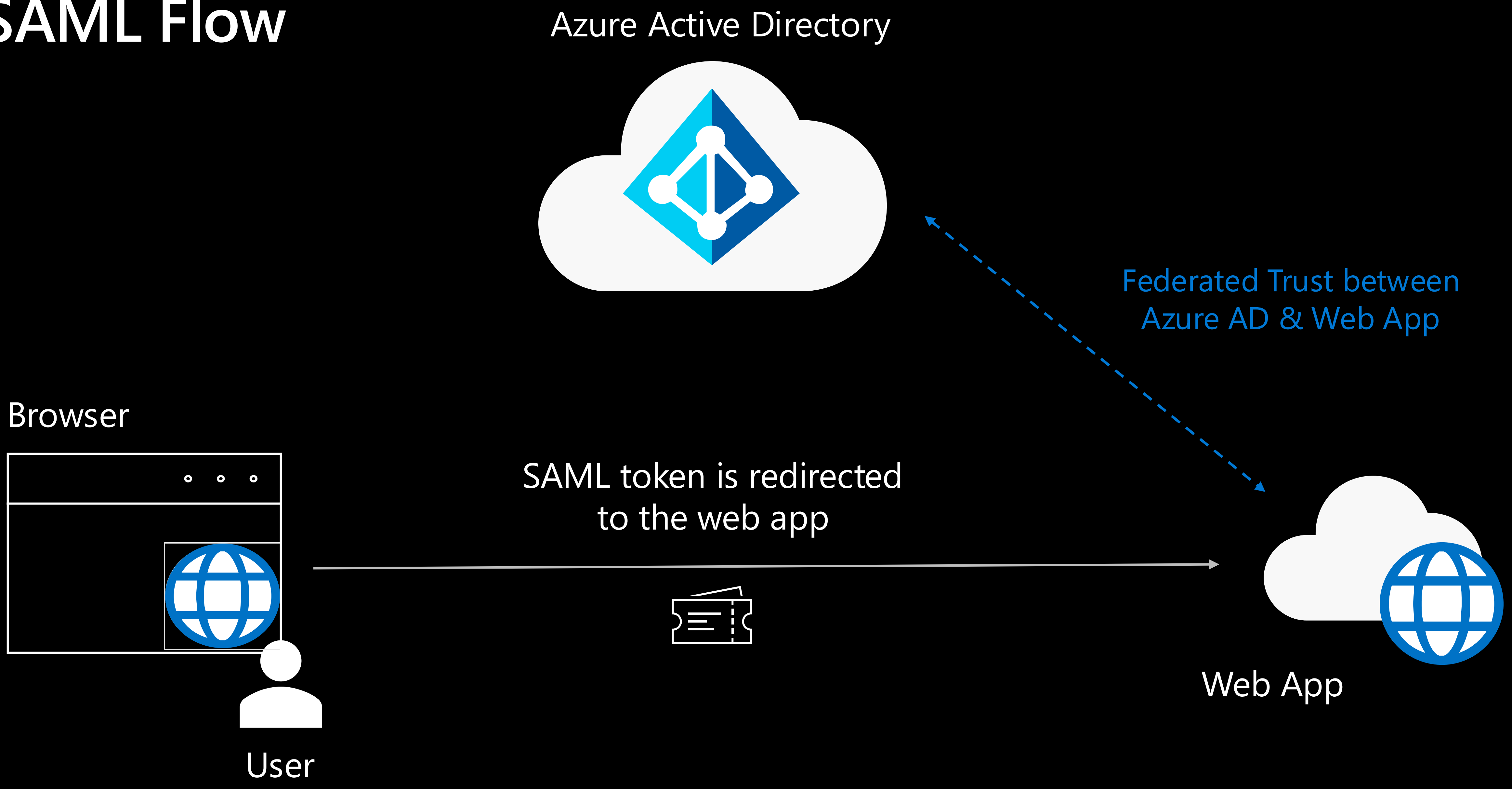

Azure Active Directory
Federated Trust between Azure AD & Web App
Browser
User
Web App

SAML Flow

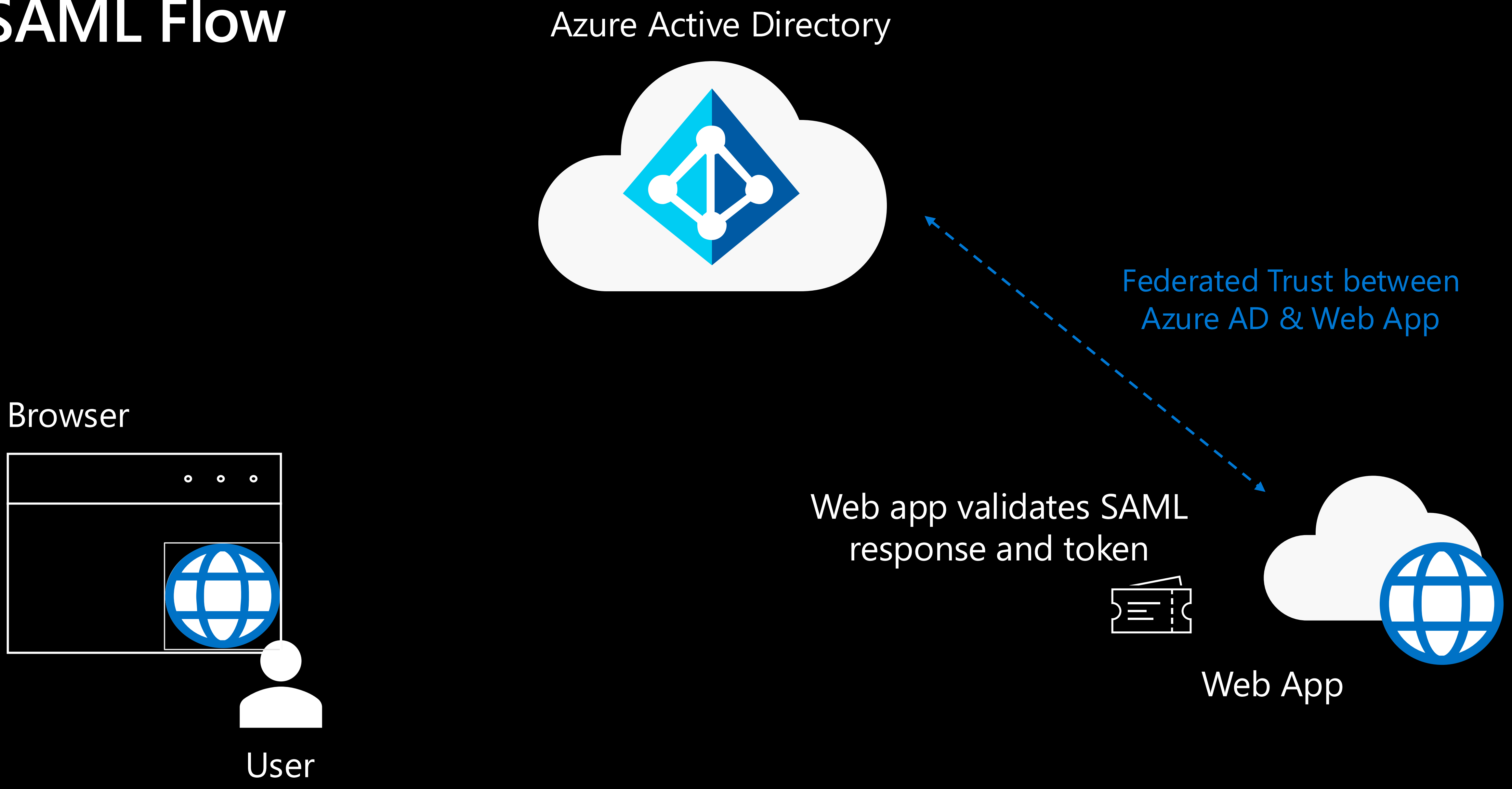


Azure Active Directory
SAML tokens are sent back to the browser
Federated Trust between Azure AD & Web App
Browser
User
Web App

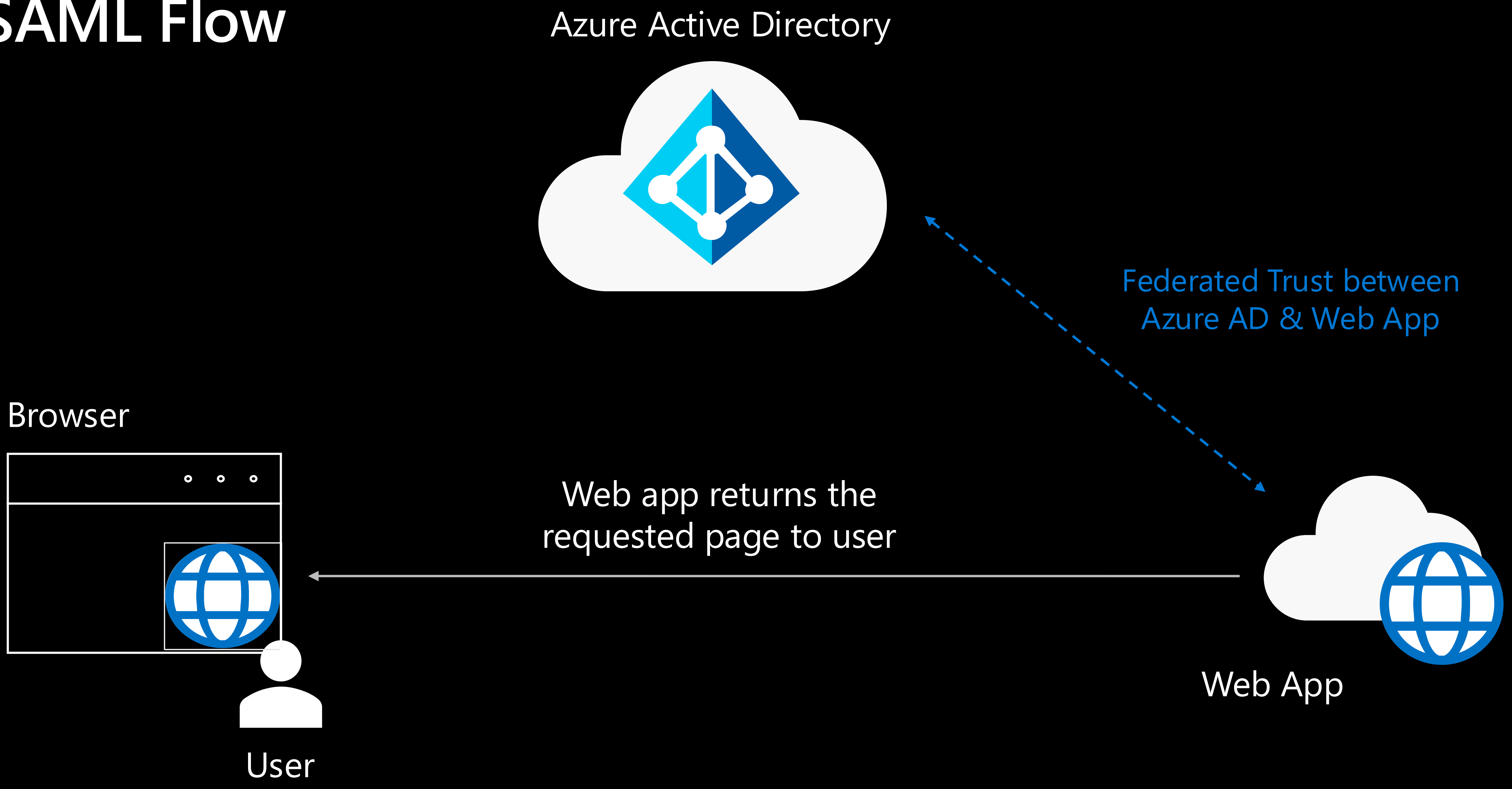
SAML Flow


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
Web app validates SAML response and token
Web App
User

SAML Flow


Azure Active Directory
Federated Trust between Azure AD & Web App
Browser
Web app returns the requested page to user
User
Web App

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

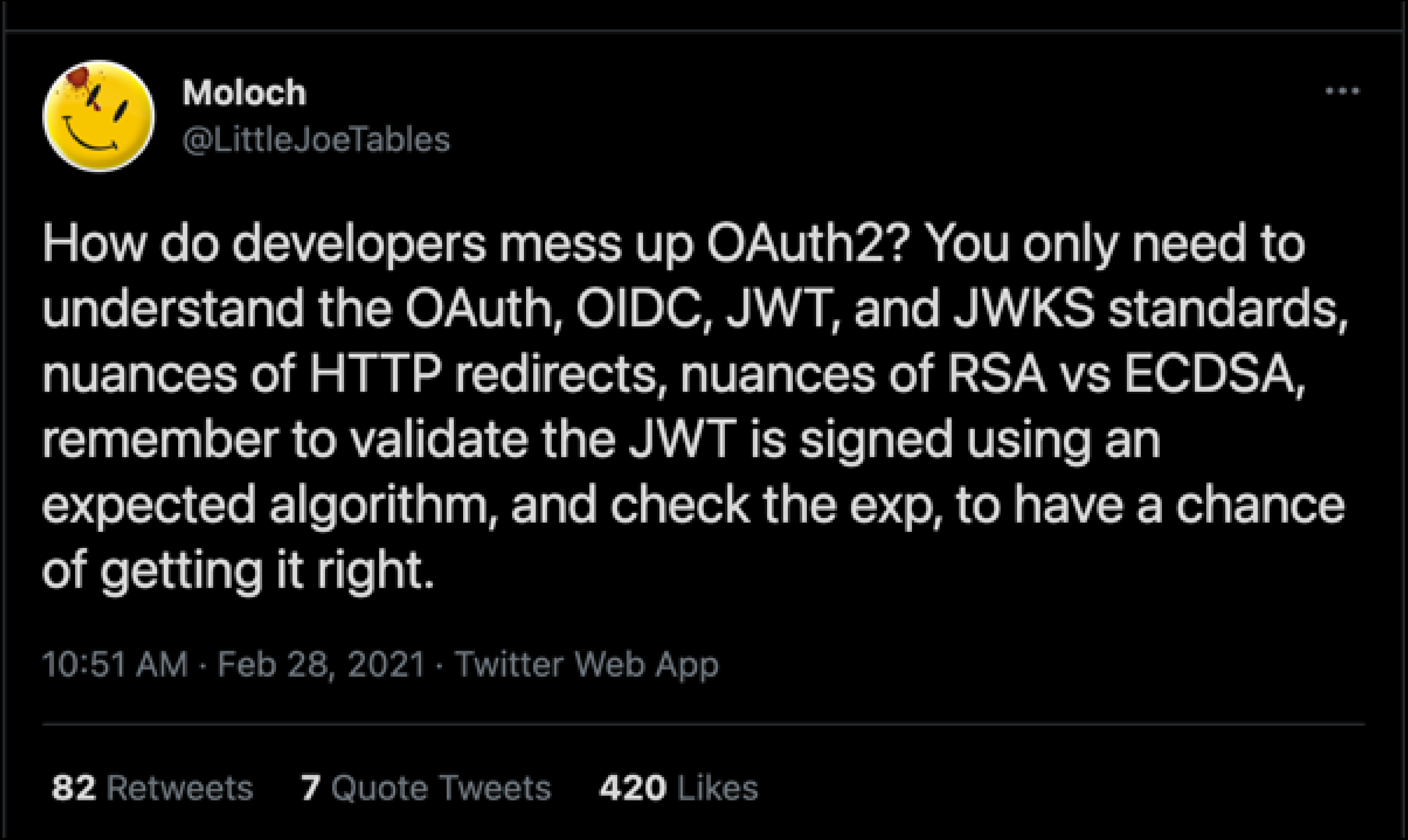
- Golden SAML Attack
  - Bad actor compromises a certificate (the SAML assertion) and can forge SAML requests
  - Can then SSO to any service as any user
- Assertion Consumer Service URL (Reply URL) – required for some apps
  - Specifies where the app expects to receive the SAML token
  - If an attacker 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
App Consent Phishing Attack
Go Do's!



Moloch
@LittleJoeTables

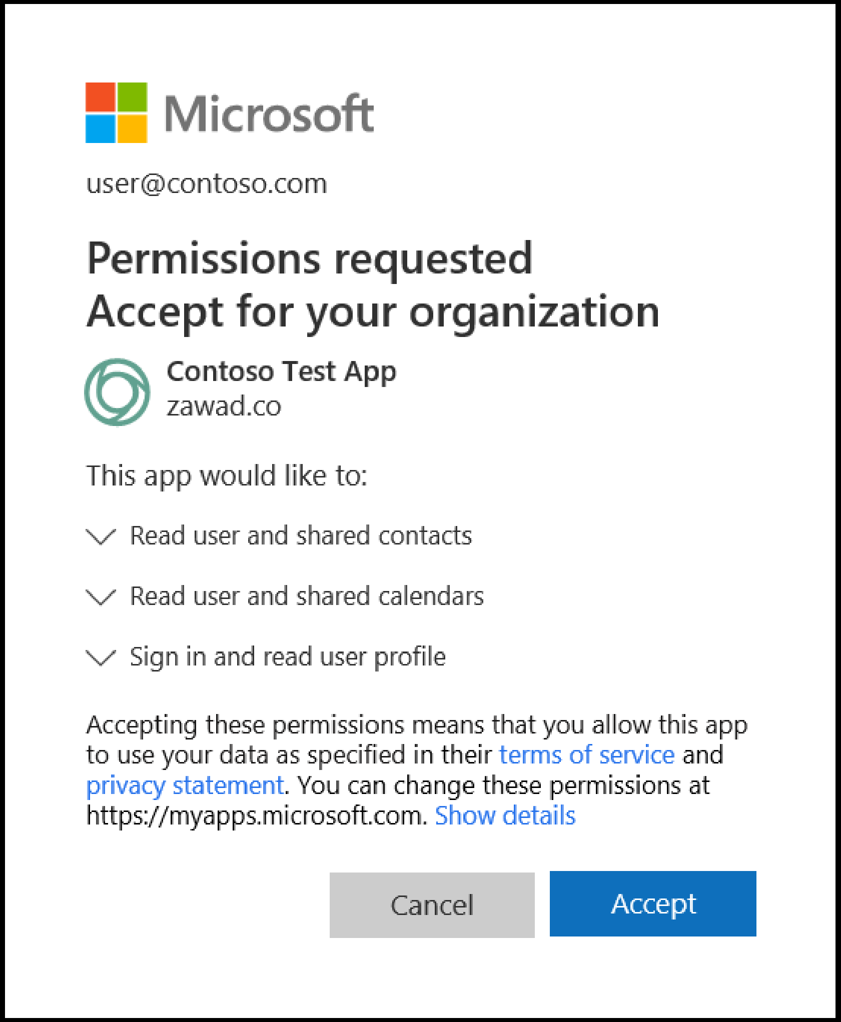
How do developers mess up OAuth2? You only need to understand the OAuth, OIDC, JWT, and JWKS standards, nuances of HTTP redirects, nuances of RSA vs ECDSA, remember to validate the JWT is signed using an expected algorithm, and check the exp, to have a chance of getting it right.

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.



user@contoso.com

Permissions requested
Accept for your organization

Contoso Test App
zawad.co

This app would like to:

- Read user and shared contacts
- Read user and shared calendars
- Sign in and read user profile

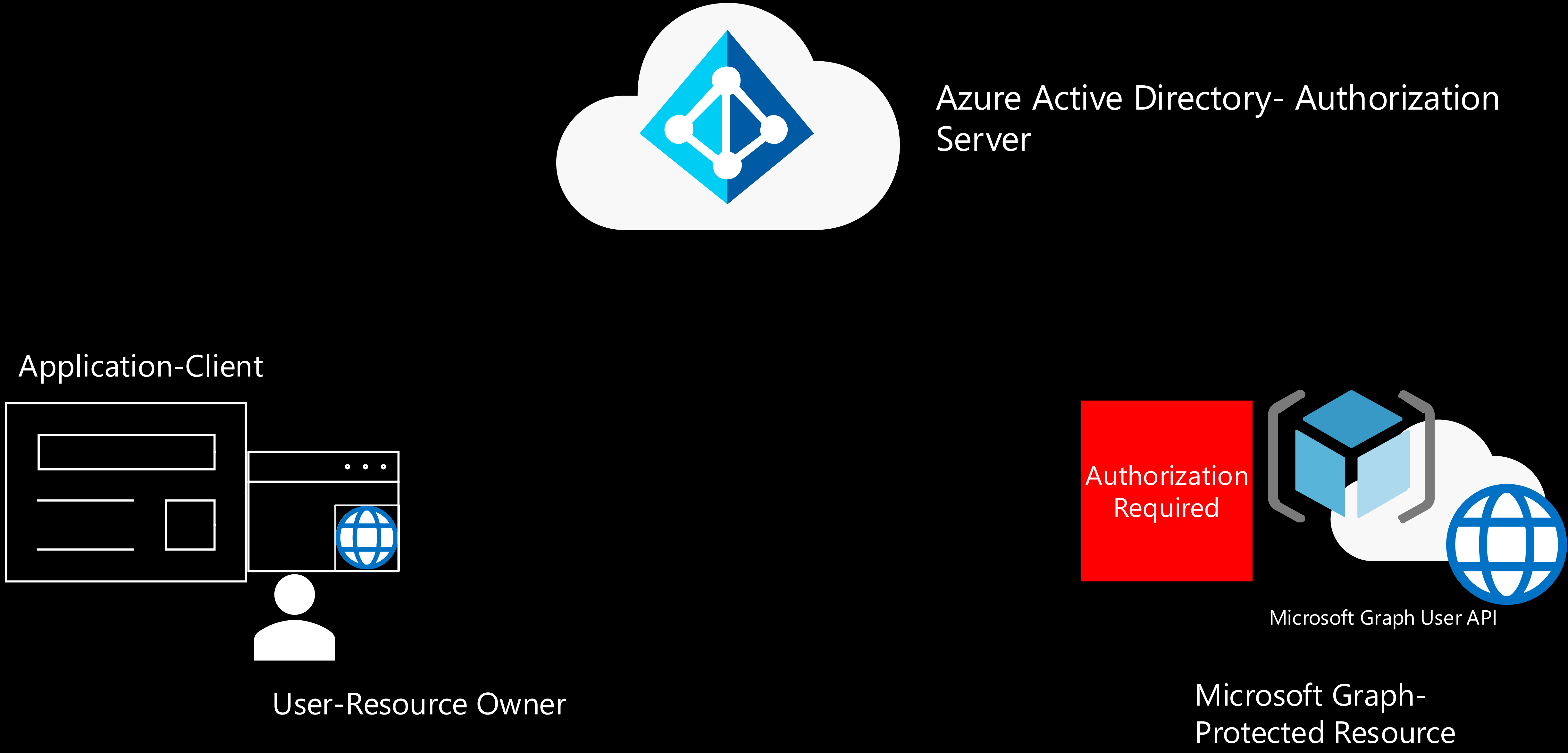
Accepting these permissions means that you allow this app to use your data as specified in their terms of service and privacy statement. You can change these permissions at https://myapps.microsoft.com. Show details

OAuth Components

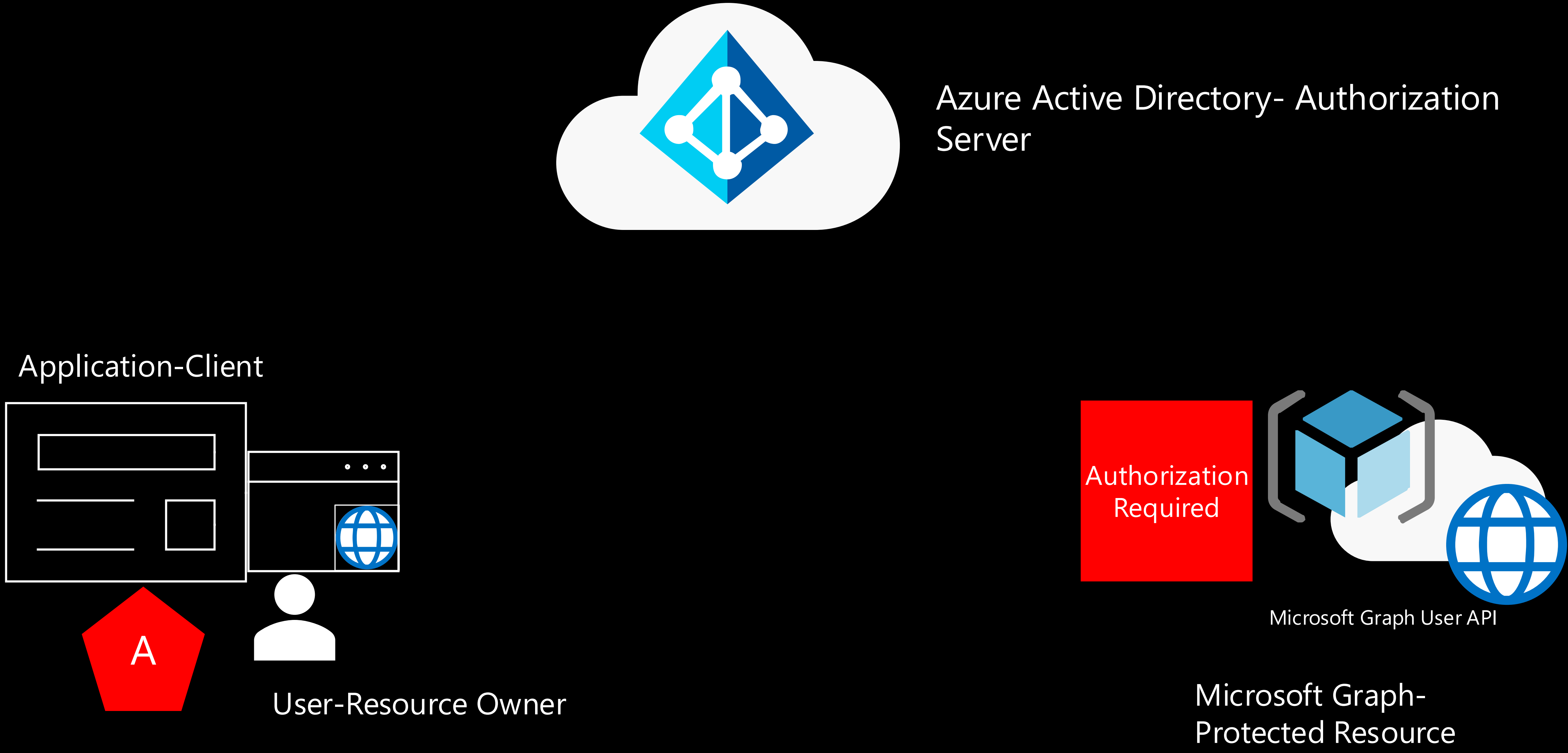
- **Resource Owner**-Usually a person on a “browser”. Has access to an API and can **delegate** access to that API.
- **Protected Resource**-Usually a WebAPI. The thing the resource owner has access to.
- **Client**-Piece of software that is accessing the protected resource on behalf of the resource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



OAuth Components Example



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

- eyJ0eXAiOiJKV1QiLCJub25jZSI6Il9wUHVWdVR0TklDbFZXcEVLd2RYX3B5bnhzczdrOTBVM2E3TElCbERqaFUiLCJhbGciOiJSUzI1NiIsIng1dCI6I
m5PbzNaRHJPRFhFSzFqS1doWHNsSFJfS1hFZyIsImtpZCI6Im5PbzNaRHJPRFhFSzFqS1doWHNsSFJfS1hFZyJ9.eyJhdWQiOiIwMDAwMDAwM
y0wMDAwLTAwMDAtYzAwMC0wMDAwMDAwMDAwMDAiLCJpc3MiOiJodHRwczovL3N0cy53aW5kb3dzLm5ldC9jNzJhMjk1ZC1kN2E1LTQx
ZWEtYTM1MS1iMTVkZDlmNjcyMTUvIiwiaWF0IjoxNjI2MjE4NjIxLCJuYmYiOjE2MjYyMTg2MjEsImV4cCI6MTYyNjIyMjUyMSwiYWNjdCI6MCw
iYWNyIjoiMSIsImFjcnMiOlsidXJuOnVzZXI6cmVnaXN0ZXJzZWN1cml0eWluZm8iLCJ1cm46bWljcm9zb2Z0OnJlcTEiLCJ1cm46bWljcm9zb2Z0
OnJlcTIiLCJ1cm46bWljcm9zb2Z0OnJlcTMiLCJjMSIsImMyIiwiYzMiLCJjNCIsImM1IiwiYzYiLCJjNyIsImM4IiwiYzkiLCJjMTAiLCJjMTEiLCJjMTIiLCJj
MTMiLCJjMTQiLCJjMTUiLCJjMTYiLCJjMTciLCJjMTgiLCJjMTkiLCJjMjAiLCJjMjEiLCJjMjIiLCJjMjMiLCJjMjQiLCJjMjUiXSwiYWlvIjoiQVVRQXUvOFR
BQUFBMm9YSHd5TkZJKzZnZDRlWmZxUm1mNDZaQmxZbXBKbStjVUZjV1c4dm5VV1JZRnRpa2tsYklncWM0eGVOQUhVOXhFdURUMFk0Q
1VWSE1Bc0FiQzdxaXc9PSIsImFtciI6WyJyc2EiLCJtZmEiXSwiYXBwX2Rpc3BsYXluYW1lIjoiTE9CIFdvcmtzaG9wIERlbW8iLCJhcHBpZCI6IjdkOTJ
mYTFmLTk3YzQtNDk4OS1iNjIyLThmYzcxNjI5ZTY1MiIsImFwcGlkYWNyIjoiMCIsImZhbWlseV9uYW1lIjoiTWFyc2giLCJnaXZlbl9uYW1lIjoiS3lsZ
SIsImlkdHlwIjoidXNlciIsImlwYWRkciI6IjE1NC4yMC4xOTYuMTQ5IiwibmFtZSI6Ikt5bGUgTWFyc2giLCJvaWQiOiI4OWQ3NWRiYi1hNjg5LTQ2Z
mMtOWE4Zi01YjA1MzYwZTFhNTgiLCJwbGF0ZiI6IjMiLCJwdWlkIjoiMTAwMzIwMDA0OEZCMzg0MiIsInJoIjoiMC5BUzRBWFNrcXg2WFg2a0d
qVWJGZDJmWnlGUl82a24zRWw0bEp0aUtQeHhZcDVsSXVBQnculiwic2NwIjoib3BlbmlkIFBlb3BsZS5SZWFkIHByb2ZpbGUgVXNlci5SZWFkI
GVtYWlsIiwic3ViIjoiS25aNjRkV1U1N3ZrTUhXT1NmVHBiOGRSeWFrTUhLSDRCZjhBd3hSQzlGRSIsInRlbmFudF9yZWdpb25fc2NvcGUiOiJOQ
SIsInRpZCI6ImM3MmEyOTVkLWQ3YTUtNDFlYS1hMzUxLWIxNWRkOWY2NzIxNSIsInVuaXF1ZV9uYW1lIjoia3lsZW1hckBtaWNyb3NvZnRpZ
GVudGl0eS5kZXYiLCJ1cG4iOiJreWxlbWFyQG1pY3Jvc29mdGlkZW50aXR5LmRldiIsInV0aSI6IlMxbXFKMXlQRWstV2NZSmRKZXpKQUEiLCJ2Z
XIiOiIxLjAiLCJ3aWRzIjpbImNmMWMzOGU1LTM2MjEtNDAwNC1hN2NiLTg3OTYyNGRjZWQ3YyIsImI3OWZiZjRkLTNlZjktNDY4OS04MTQzLT
c2YjE5NGU4NTUwOSJdLCJ4bXNfc3QiOnsic3ViIjoiU1F0bmdzd1pCSjk1NkZWZTVKYkVybVNBZGZwMWhHRF9NbnNEMEh2RTN2YyJ9LCJ4b
XNfdGNkdCI6MTU0NzQwNDUzMH0.VQrBPQfMYPt10YPpHlpjVa1UdYfIDaDvnw5NKU5VicIAmFFLKCi945EHdKKHRQkSf8BjS3ul-
7AKz8VhBxzP7LZHLCseHiHj_9MYMJWaOigNkx2m7vFE-
h_5I1DT_PGnbWSZdT9c7s9pYzHm1p3YUyp04uX5vOadK0o_fUYNtOy2hfR42GgtJDUKb0el3mQRFfdOruRG5W4jptaKVyTvMqSiEHHjlNYJpsY
ZMNvidxDQ16ErIzgpfepvjOSZlnSQkZeA4L-zEBSFBcn-hZcQF4KYmUwdBKPT-KiAt0ajZtb2VurIq9WOtbIrv8Hv-CFDiuTyxsxoo5mwlC7JLIRg3Q

Access Token Decoded Example

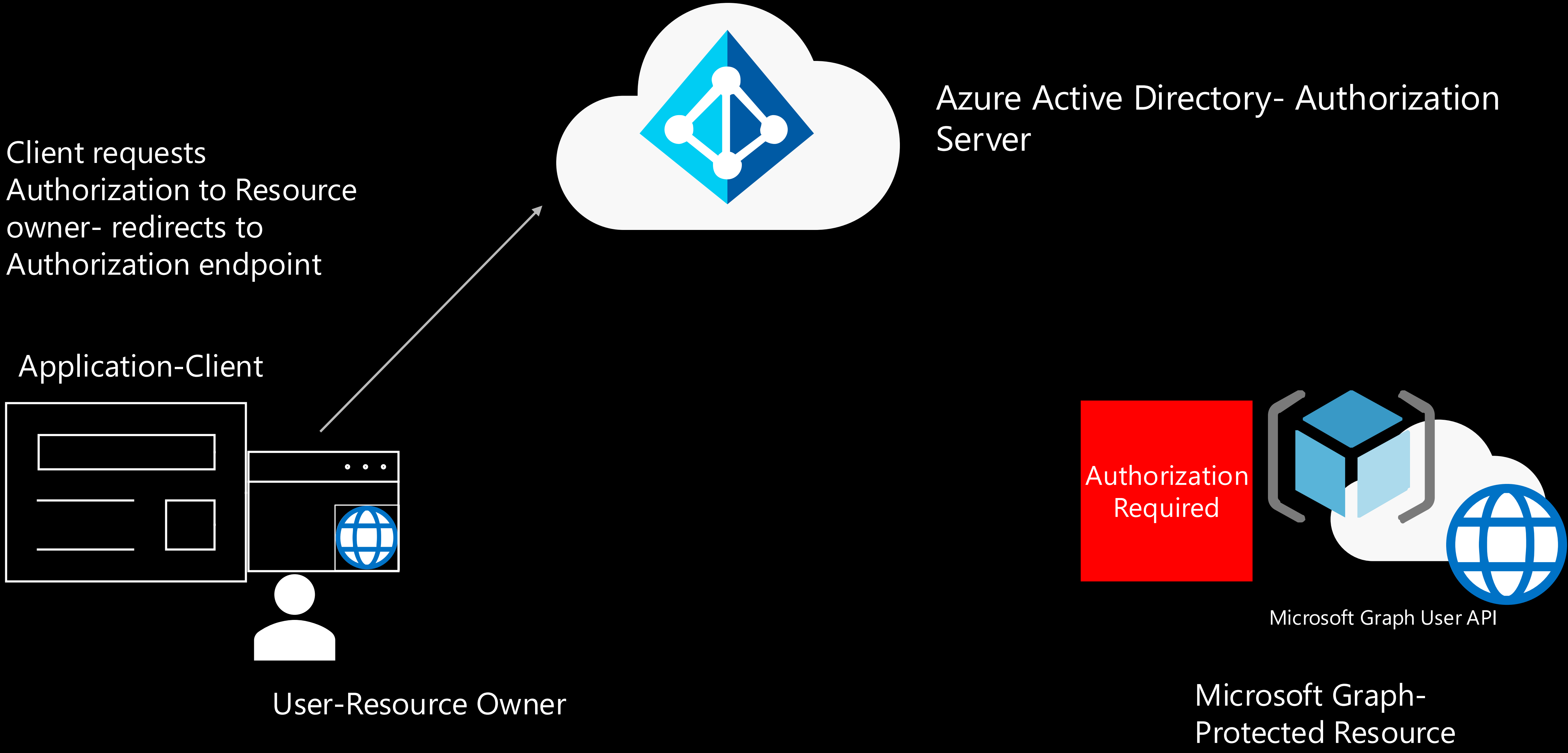
- { "**typ**": "JWT", "nonce": "_pPuVuTtNIClVWpEKwdX_pynxss7k90U3a7LIBlDjhU", "alg": "RS256", "x5t": "nOo3ZDrODXEK1jKWhXslHR_KXEg", "kid": "nOo3ZDrODXEK1jKWhXslHR_KXEg" }.{ "**aud**": "00000003-0000-0000-c000-000000000000", "**iss**": "https://sts.windows.net/c72a295d-d7a5-41ea-a351-b15dd9f67215/", "iat": 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

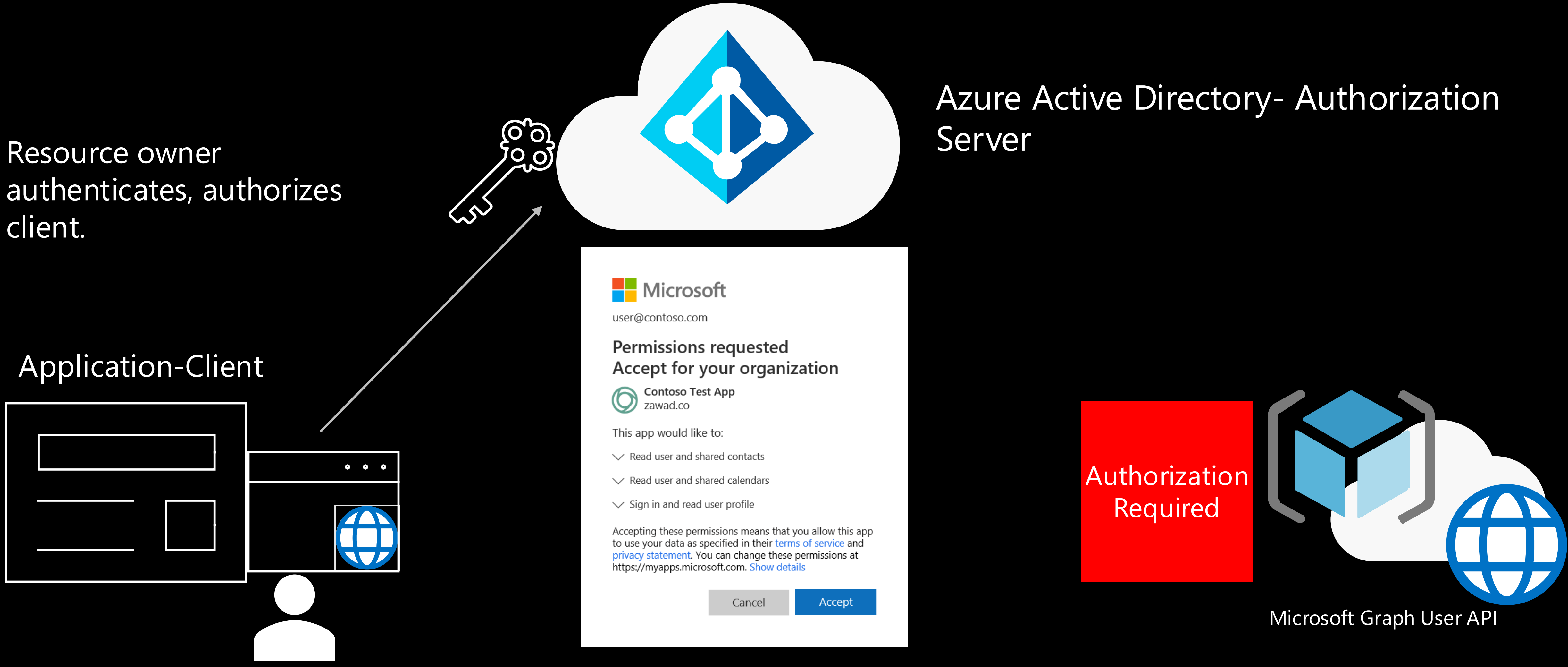
Client requests Authorization to Resource owner- redirects to Authorization endpoint

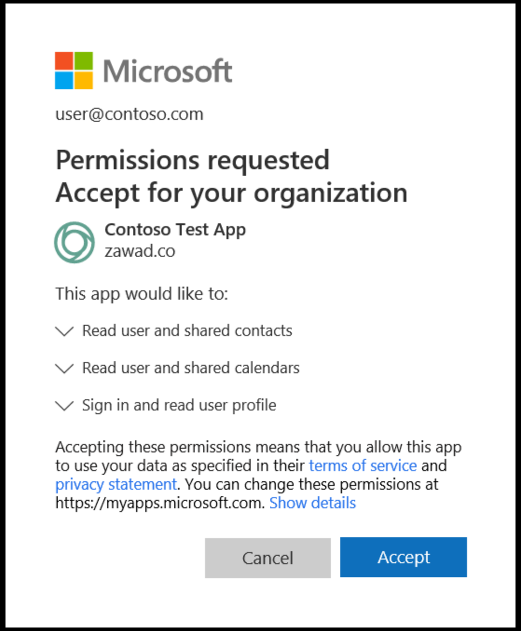


OAuth Authorization Code Flow Simplified Cont.

Resource owner authenticates, authorizes client.

Azure Active Directory- Authorization Server





Microsoft

user@contoso.com

Permissions requested
Accept for your organization

Contoso Test App
zawad.co

This app would like to:

- Read user and shared contacts
- Read user and shared calendars
- Sign in and read user profile

Accepting these permissions means that you allow this app to use your data as specified in their terms of service and privacy statement. You can change these permissions at https://myapps.microsoft.com. Show details

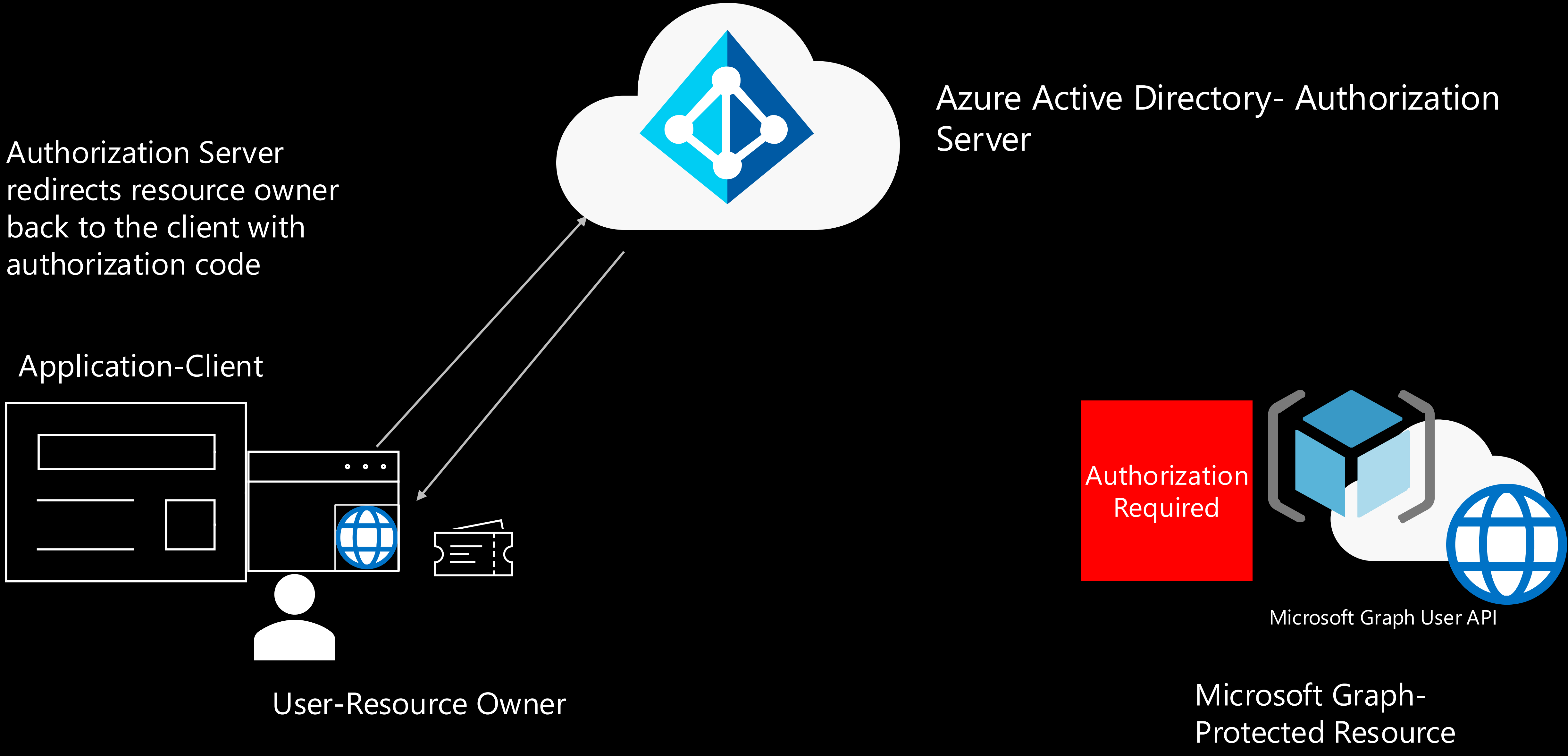
Application-Client

User-Resource Owner

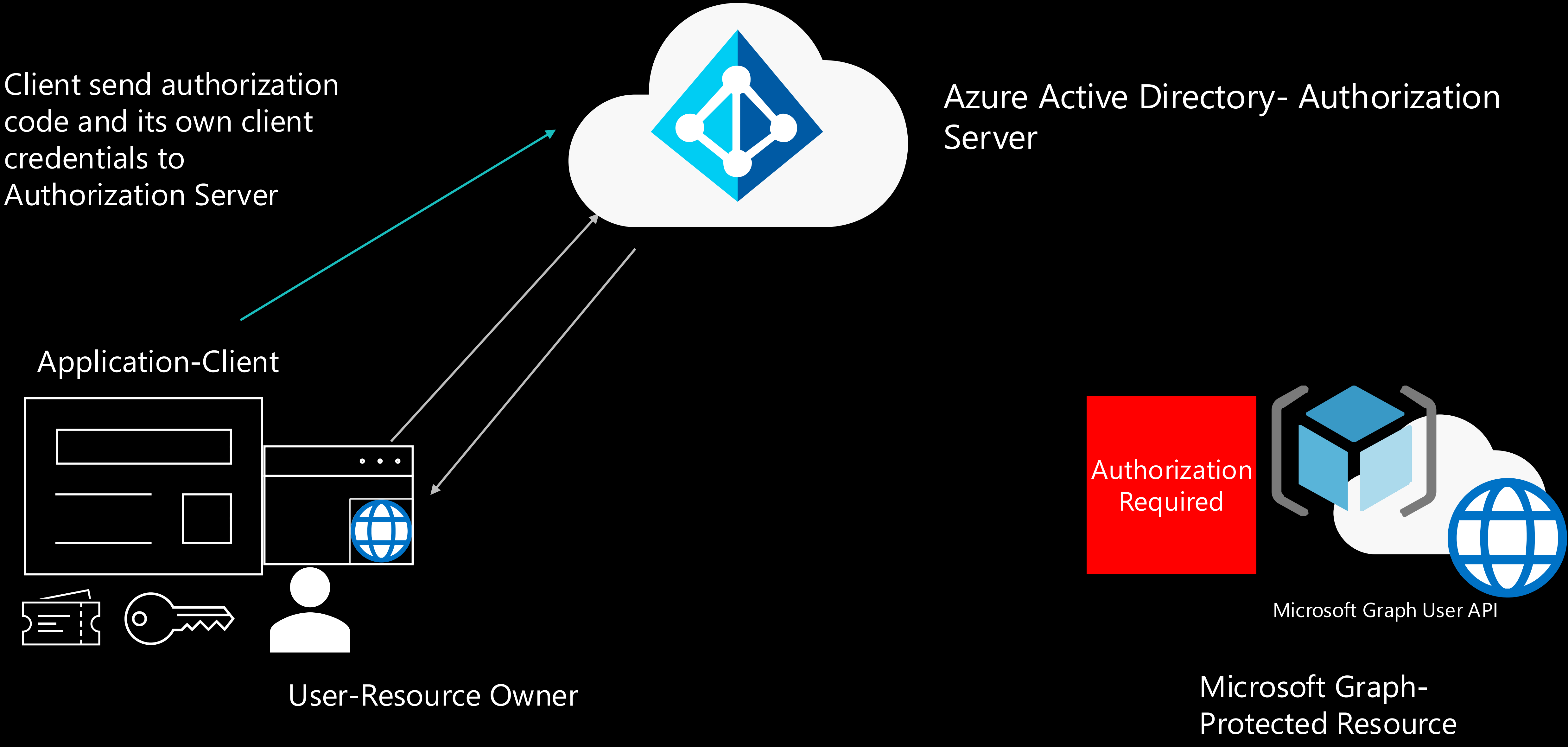
Microsoft Graph- Protected Resource

OAuth Authorization Code Flow Simplified Cont.

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

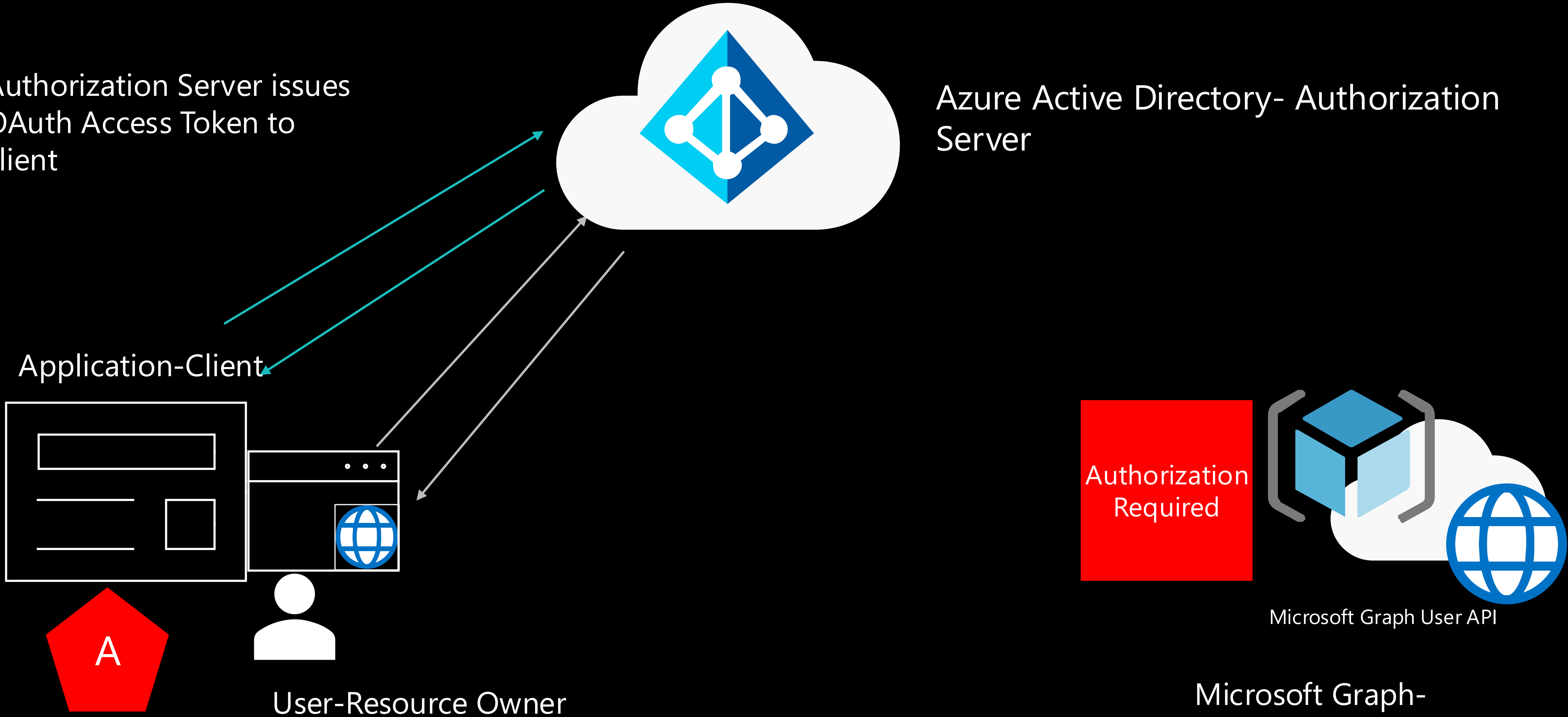


OAuth Authorization Code Flow Simplified Cont.

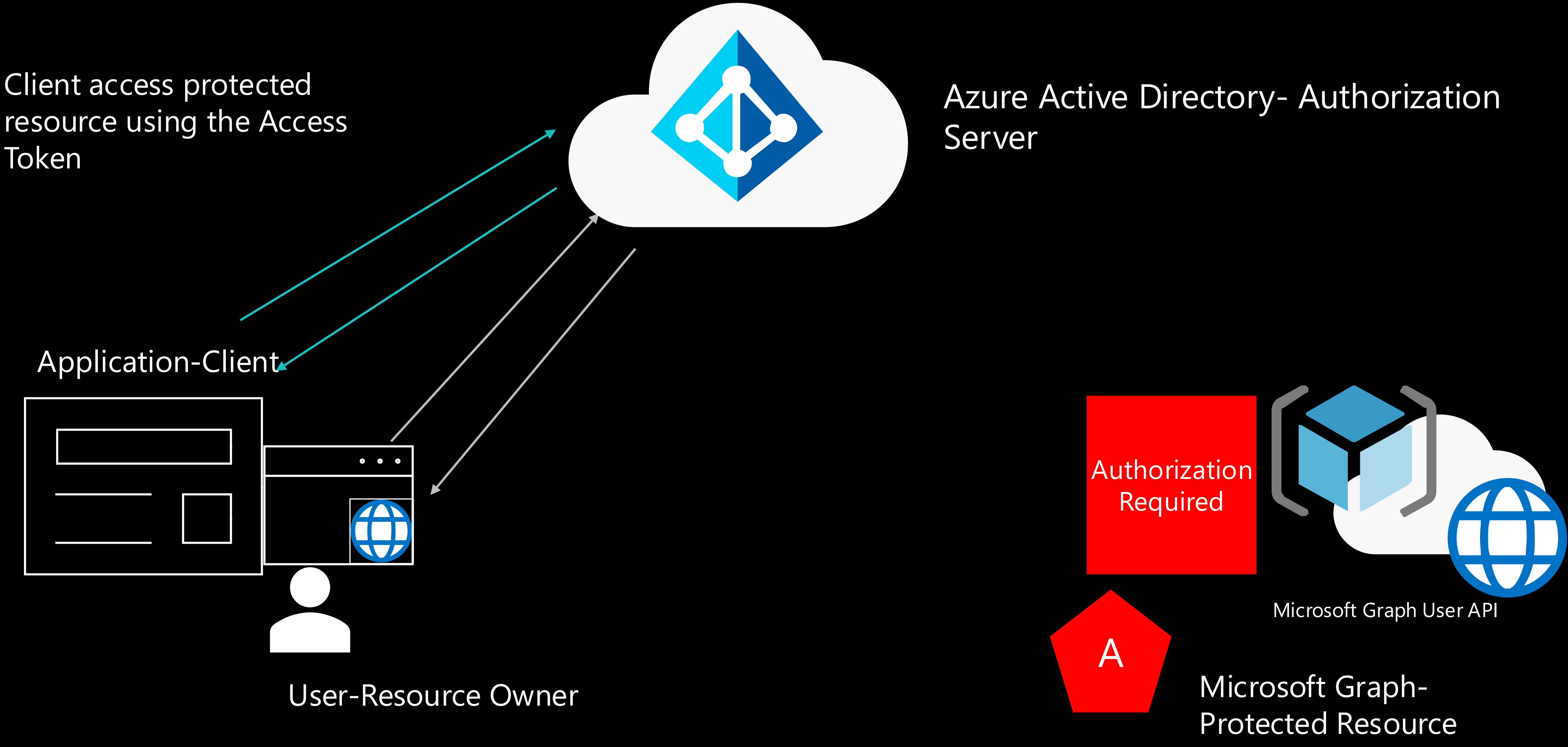

Client send authorization code and its own client credentials to Authorization Server
Azure Active Directory- Authorization Server
Application-Client
User-Resource Owner
Authorization Required
Microsoft Graph User API
Microsoft Graph- Protected Resource

OAuth Authorization Code Flow Simplified Cont.

Authorization Server issues OAuth Access Token to client



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
A
Microsoft Graph User API
Microsoft Graph- Protected Resource

OAuth Flow Recap

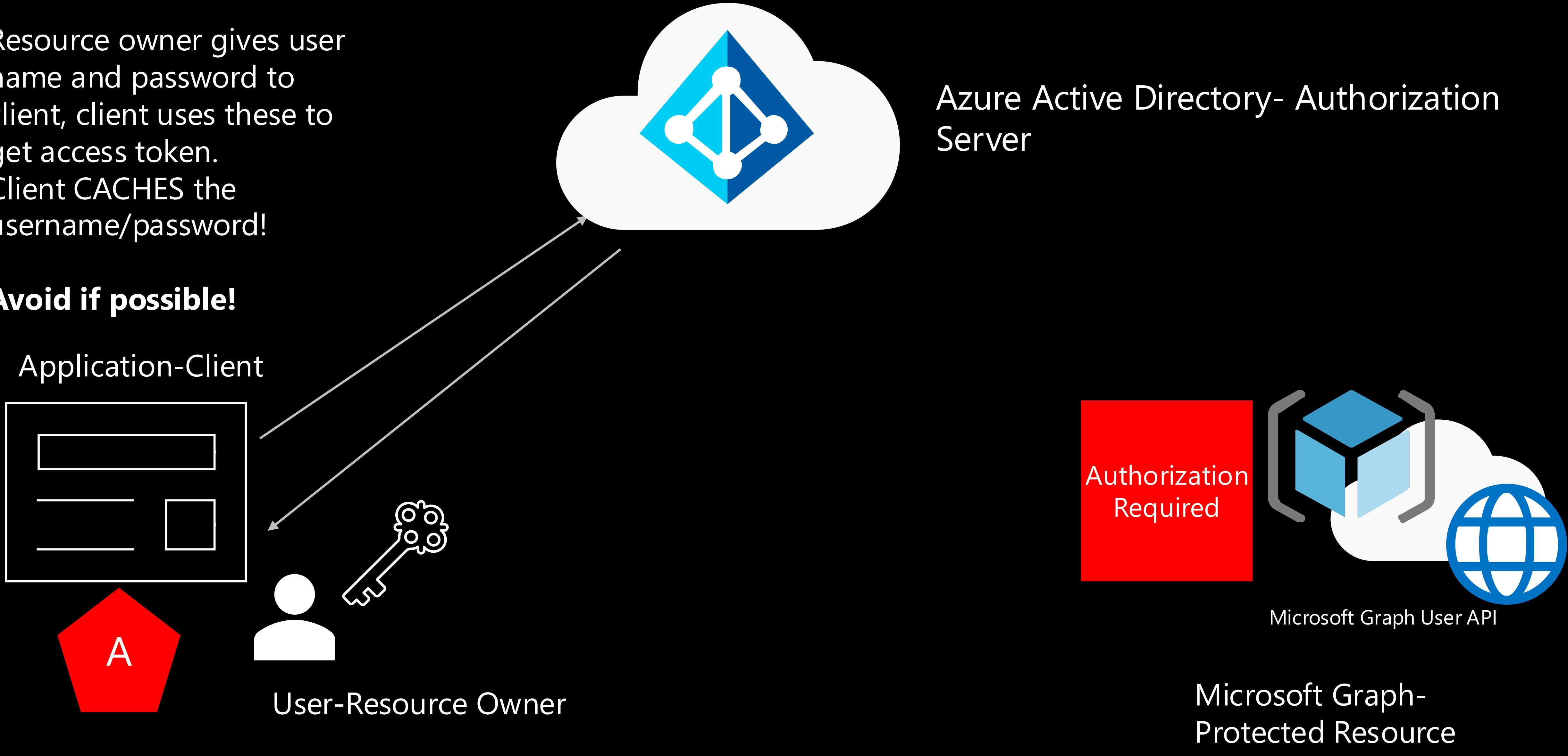
- Resource owner never provided credentials to the client app.
  - Only to the Authorization Server (AAD)
- Resource owner delegated the permissions needed for client
  - Provided in the scope of the access token.
- Client accessed protected resource with access token
  - Protected resource trusts the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



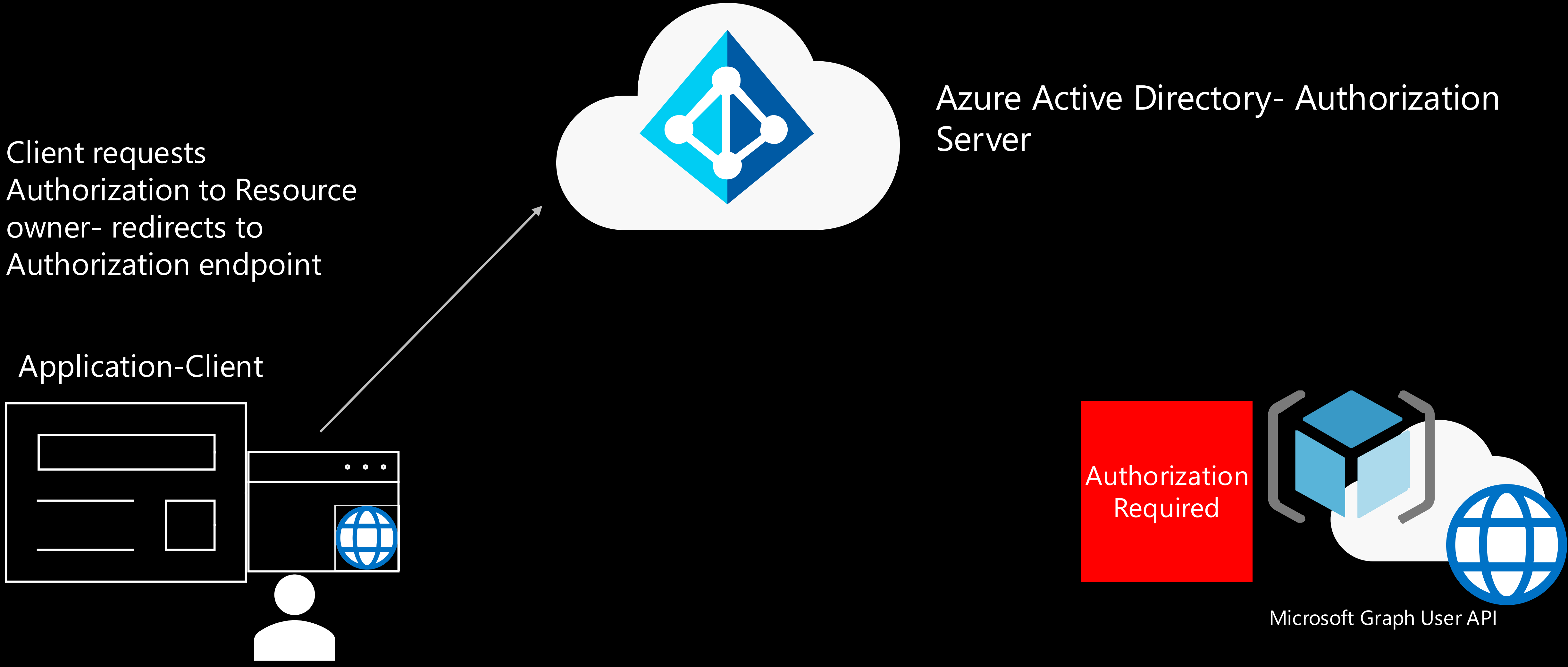
Agenda

Why Modern Auth?
SAML
OAuth2
OIDC
App Consent Phishing Attack
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



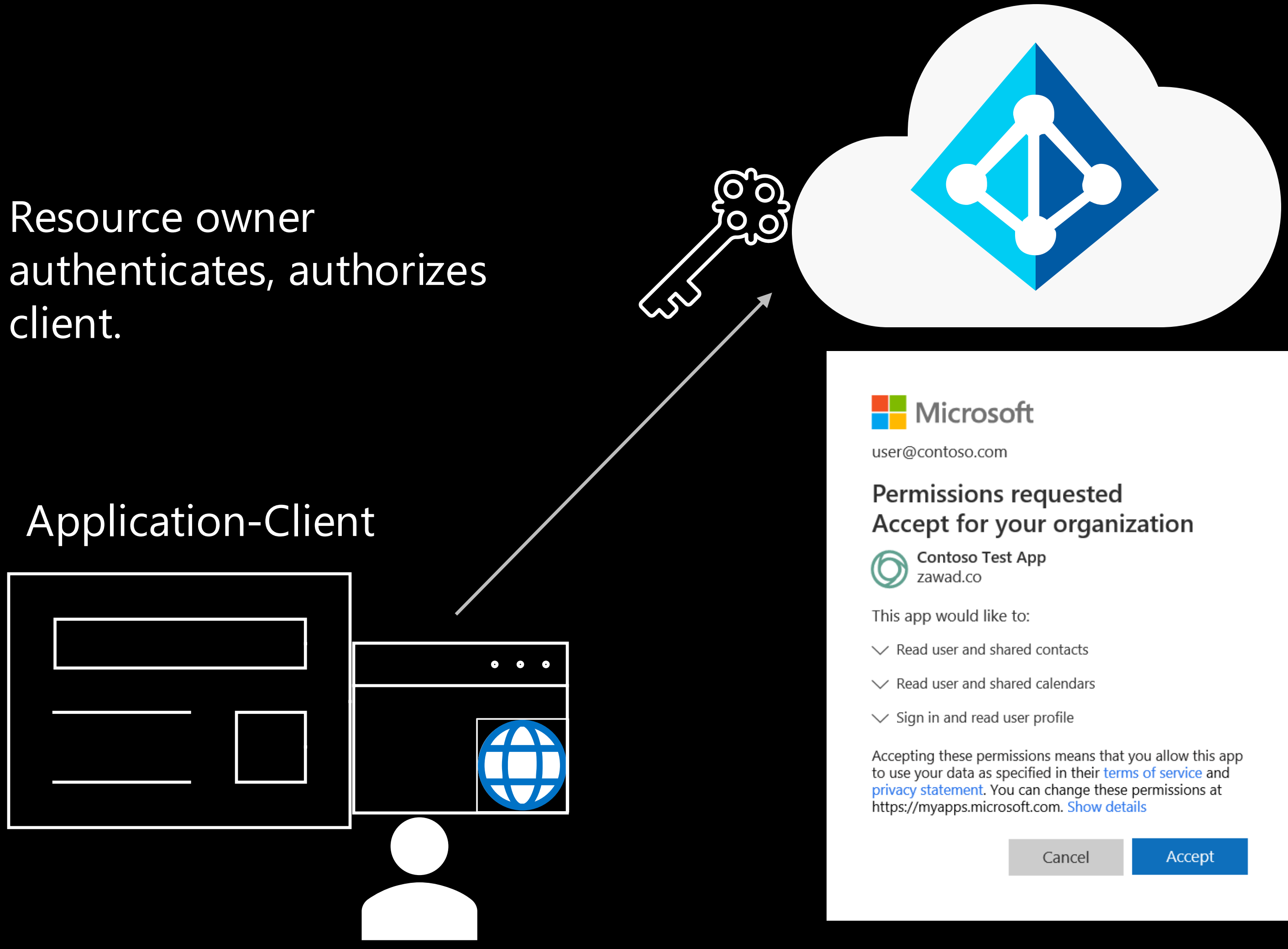
Azure Active Directory- Authorization Server
Client requests Authorization to Resource owner- redirects to Authorization endpoint
Application-Client
Authorization Required
Microsoft Graph User API

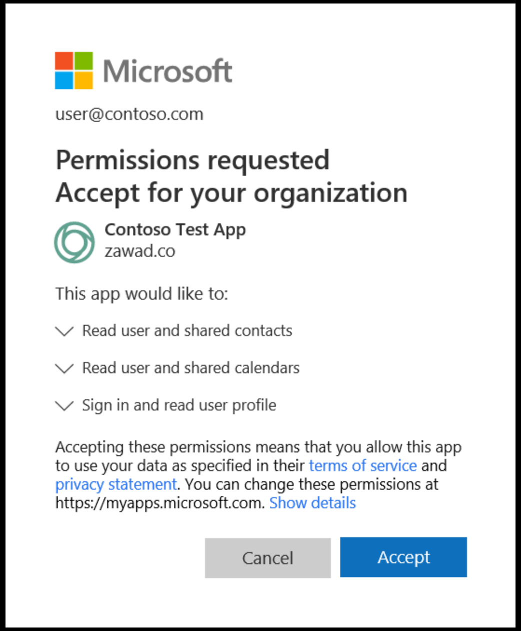
User-Resource Owner

Microsoft Graph- Protected Resource

OpenID Connect Flow Simplified Simplified Cont.

Resource owner authenticates, authorizes client.





user@contoso.com

Permissions requested
Accept for your organization

Contoso Test App
zawad.co

This app would like to:

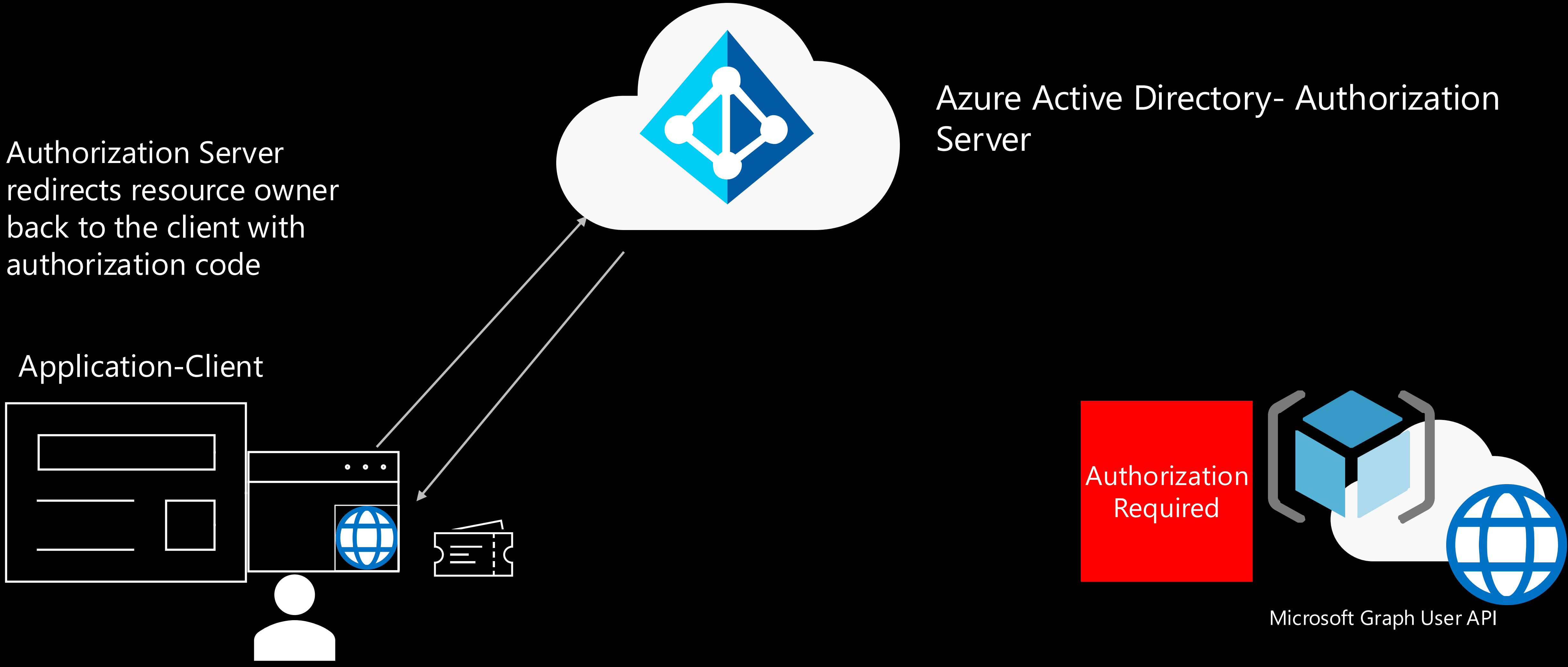
- Read user and shared contacts
- Read user and shared calendars
- Sign in and read user profile

Accepting these permissions means that you allow this app to use your data as specified in their terms of service and privacy statement. You can change these permissions at https://myapps.microsoft.com. Show details

Authorization Required
Microsoft Graph User API

Microsoft Graph- Protected Resource

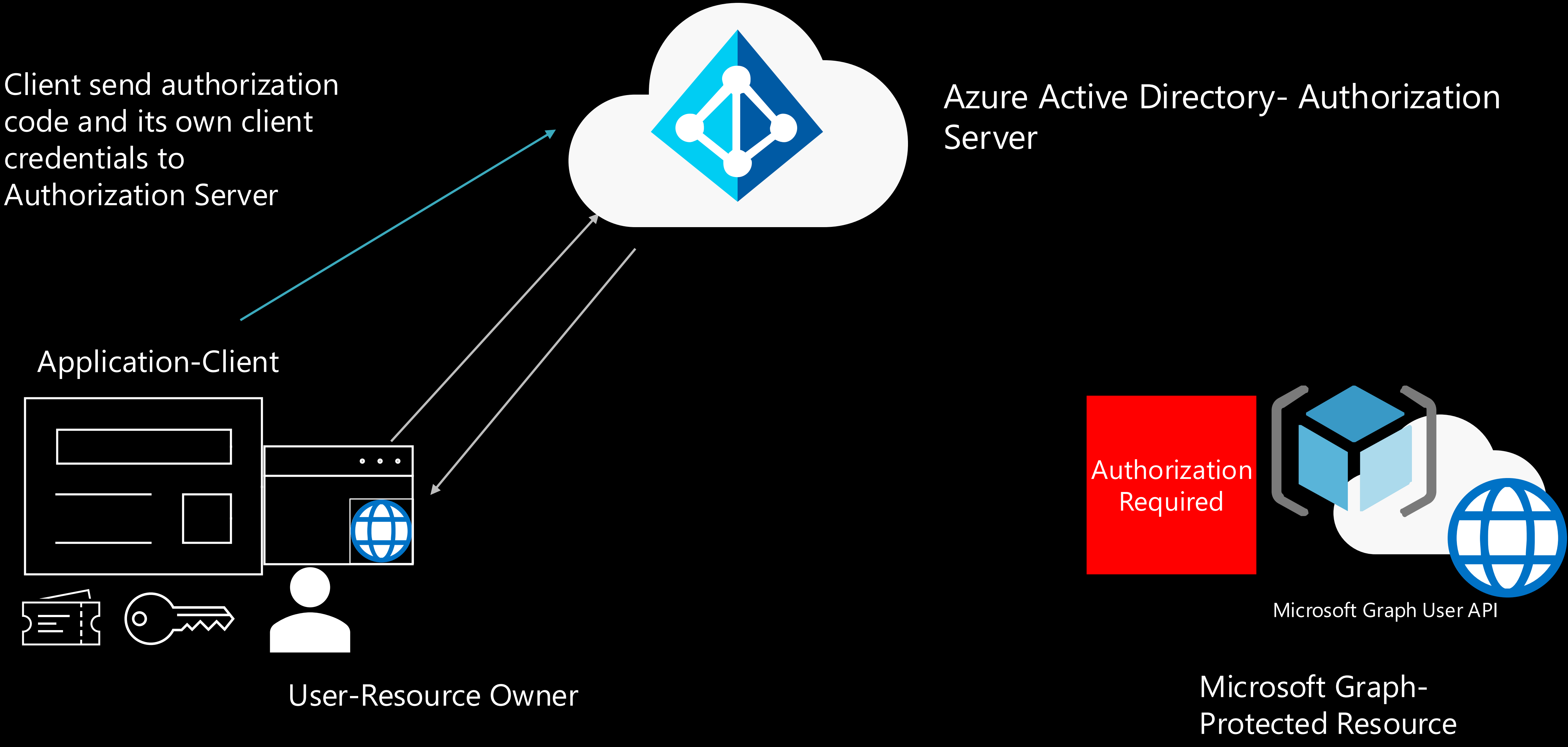
OpenID Connect Flow Simplified Cont.


Azure Active Directory- Authorization Server
Authorization Server redirects resource owner back to the client with authorization code
Application-Client
Authorization Required
Microsoft Graph User API

User-Resource Owner

Microsoft Graph- Protected Resource

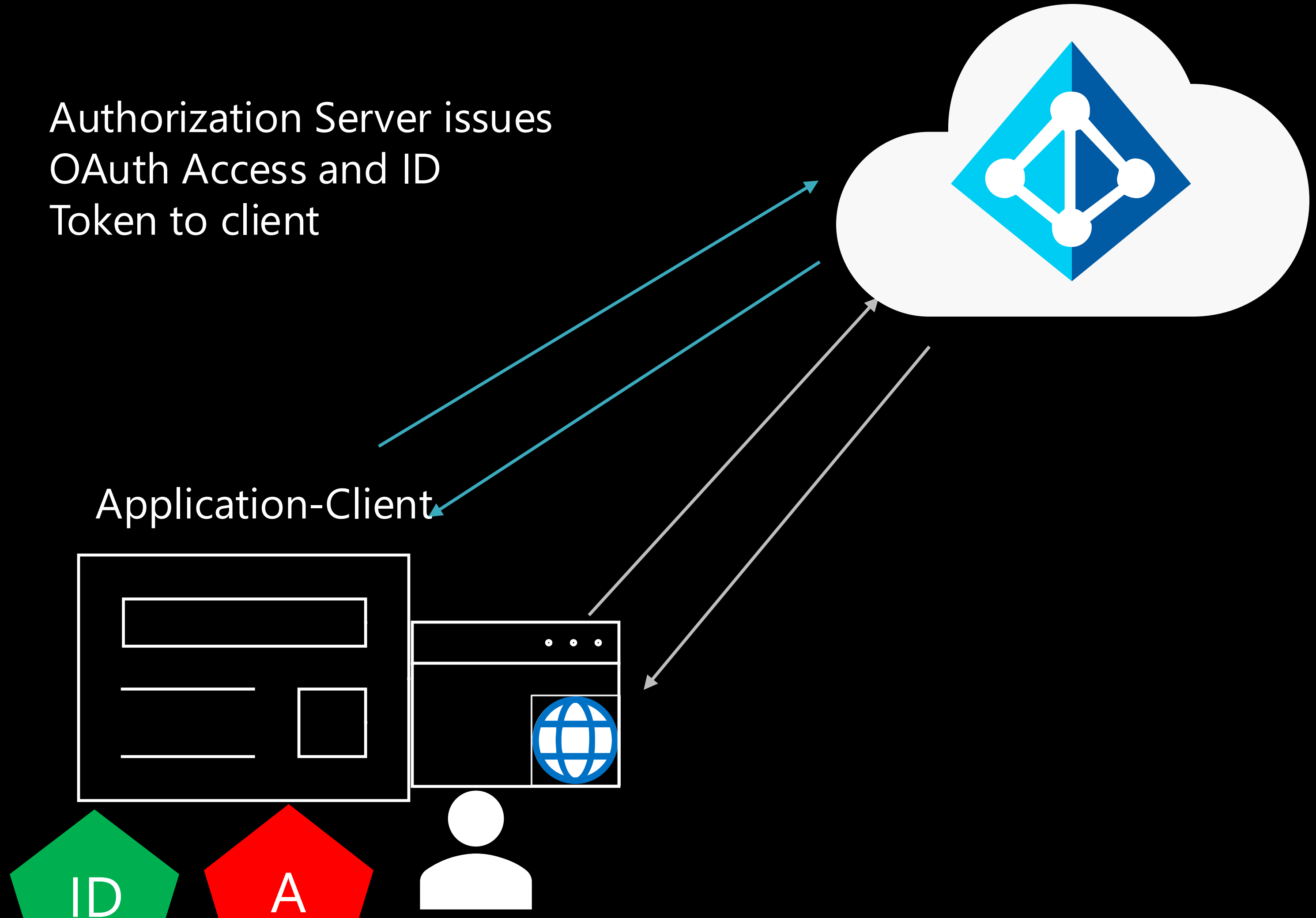
OpenID Connect Flow Simplified Cont.


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.

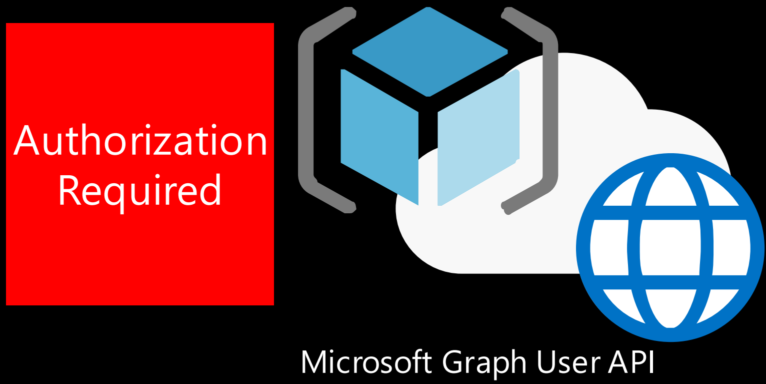
Authorization Server issues OAuth Access and ID Token to client

Azure Active Directory- Authorization Server



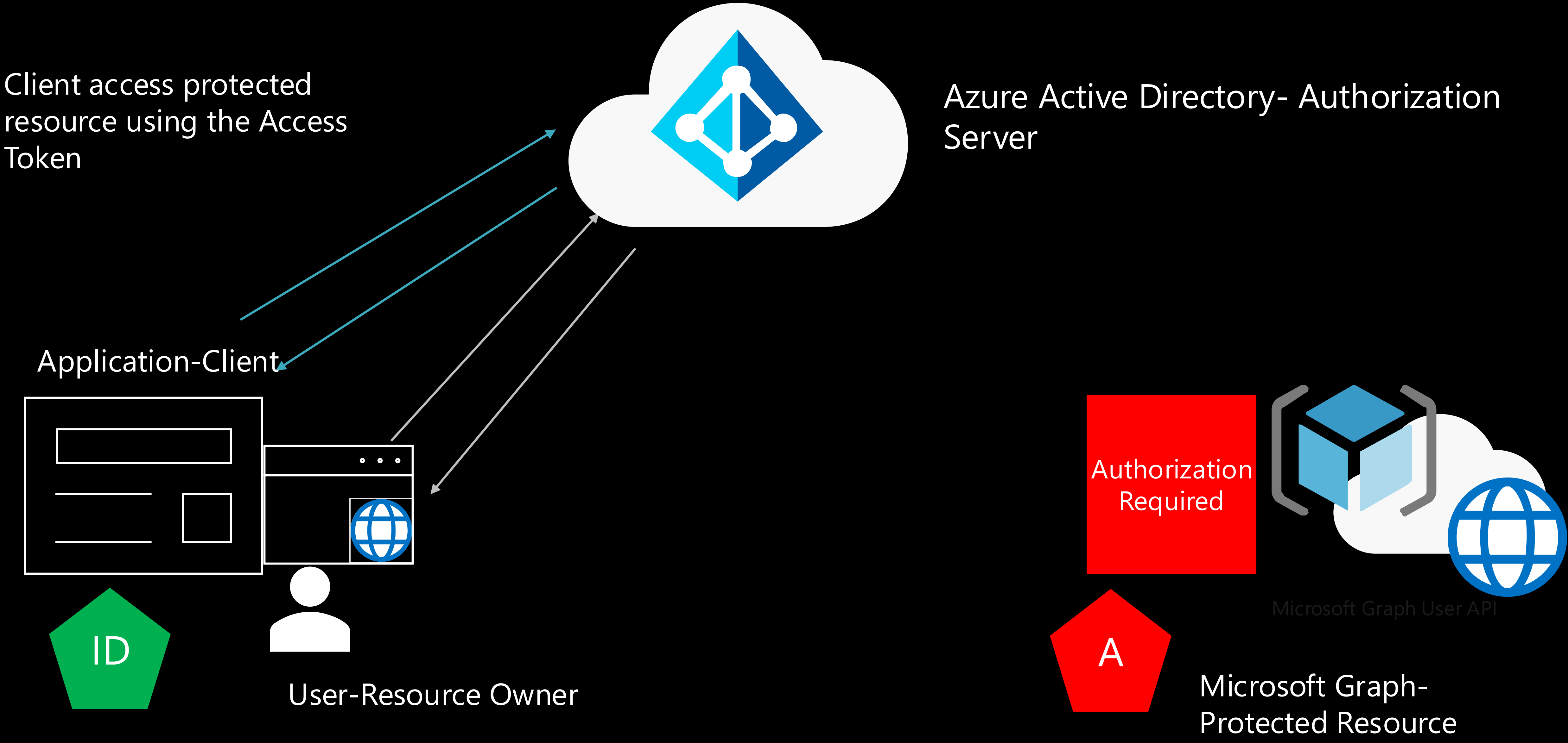
Application-Client

User-Resource Owner



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

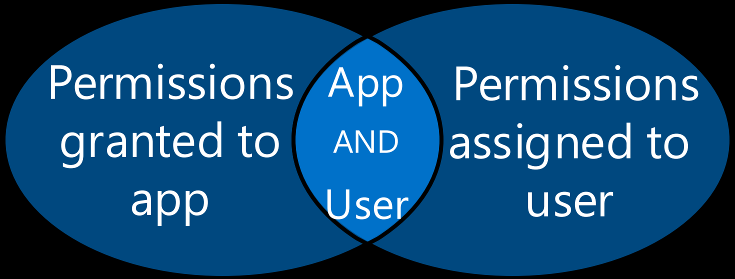
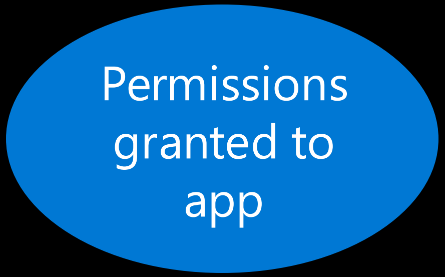
Agenda

Why Modern Auth?
SAML
OAuth2
OIDC
App Consent Phishing Attack
Go Do's!

Microsoft

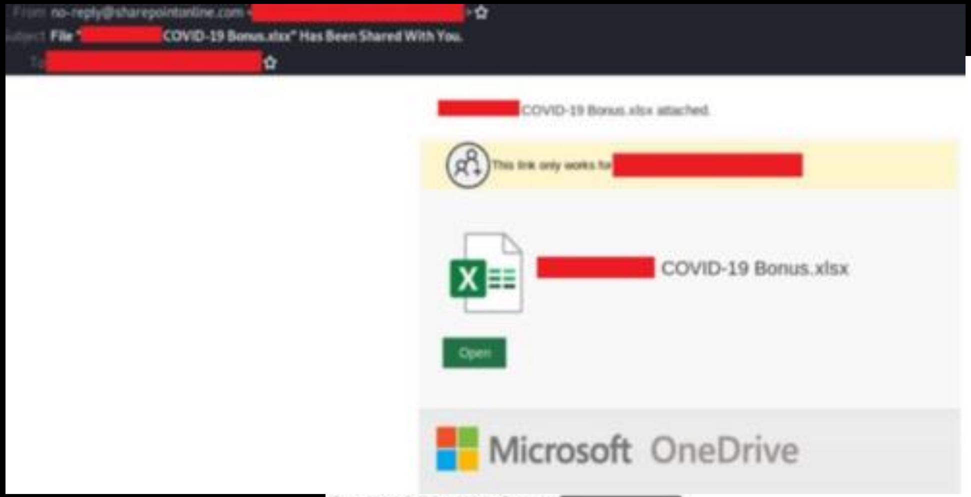
Application Consent Phishing Attacks

Permission Types

	Delegated Permissions	Application Permissions
App	Mobile / Web / SPA	Service / Daemon
Scenario	Get access on behalf of user	Get access as a service
Consent	Users for self / IT admin for all users	Only by IT admin
Effective Permissions	 Permissions granted to app App AND User Permissions assigned to user	

Evolving Threat Landscape | Consent Phishing

- Business-themed email
- Covid-19-themed email
- Malicious webapp



From no-reply@sharepointonline.com <[redacted]>
Subject File "[redacted] COVID-19 Bonus.xlsx" Has Been Shared With You.
To [redacted]

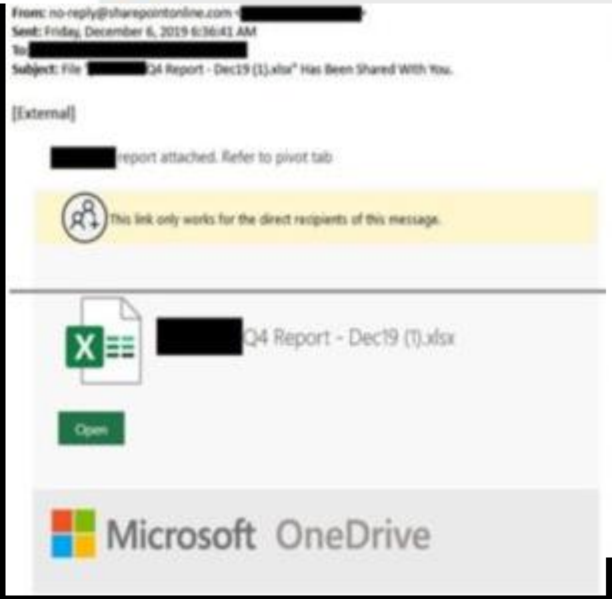
[redacted] COVID-19 Bonus.xlsx attached.

This link only works for [redacted]

[redacted] COVID-19 Bonus.xlsx

Open

Microsoft OneDrive



From: no-reply@sharepointonline.com <[redacted]>
Sent: Friday, December 6, 2019 6:36:41 AM
To: [redacted]
Subject: File "[redacted] Q4 Report - Dec19 (1).xlsx" Has Been Shared With You.

[External]

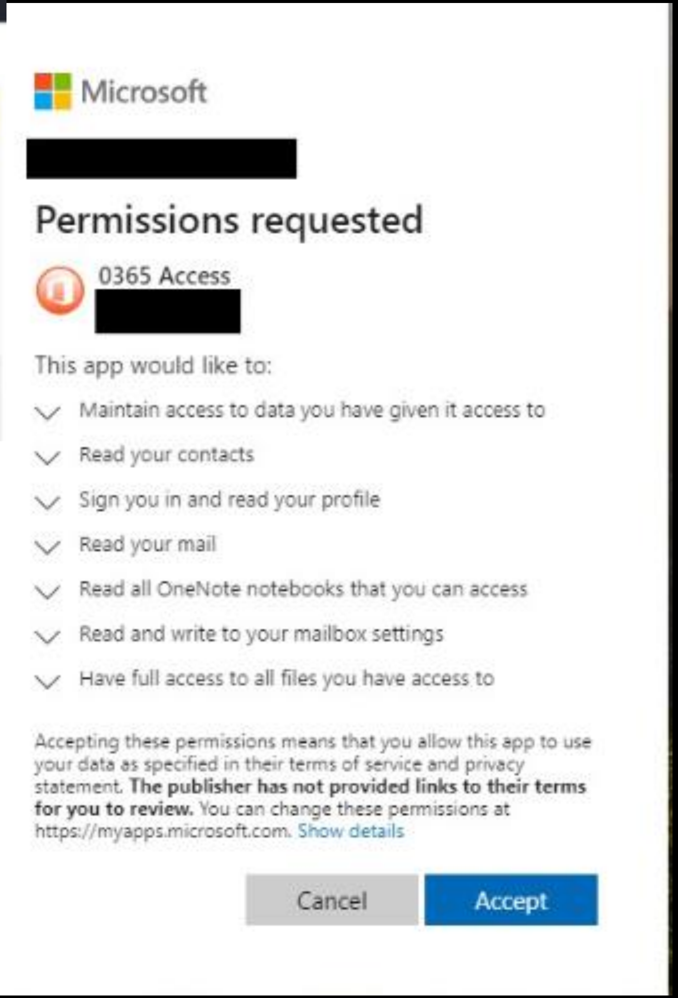
[redacted] report attached. Refer to pivot tab

This link only works for the direct recipients of this message.

[redacted] Q4 Report - Dec19 (1).xlsx

Open

Microsoft OneDrive



Microsoft

[redacted]

Permissions requested

0365 Access
[redacted]

This app would like to:

- Maintain access to data you have given it access to
- Read your contacts
- Sign you in and read your profile
- Read your mail
- Read all OneNote notebooks that you can access
- Read and write to your mailbox settings
- Have full access to all files you have access to

Accepting these permissions means that you allow this app to use your data as specified in their terms of service and privacy statement. **The publisher has not provided links to their terms for you to review.** You can change these permissions at https://myapps.microsoft.com. Show details

Cancel | Accept

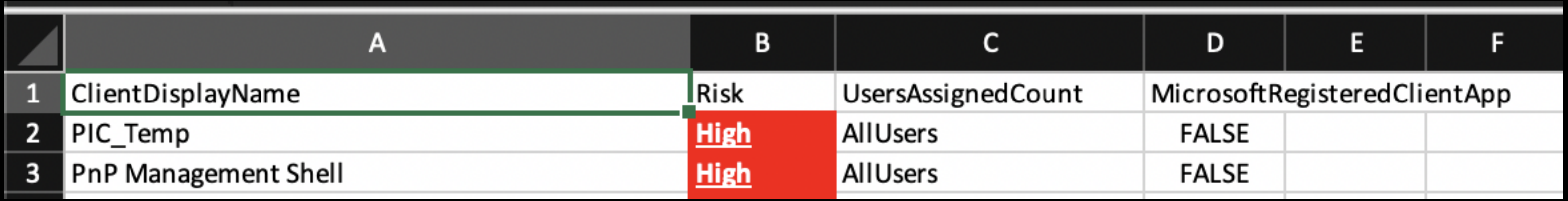
If the user consents, the attacker can gain access to their mail, forwarding rules, files, contacts, notes, profile and other sensitive data and resources.

How to find illicit consent?

- Office 365 Portal
  - Search the audit logs apps and look for signs, also called Indicators of Compromise (IOC) of attack
  - Review the Security&Compliance Center audit logs
    - If **IsAdminContent** is set to **True** it indicates that someone with Global Administrator access may have granted broad access to data.
- Azure AD Portal
  - Enterprise Apps – Permissions
  - Audit logs
- PowerShell
  - Inventory applications and their granted permissions
  - This is the fastest and most thorough method, with the least amount of overhead.
  - https://aka.ms/getazureadpermissions
- Microsoft Cloud App Security (w/ applicable license)

A few other things to look for...

- Start with HighRiskApps tab & UserAssignedCount AllUsers
  - Every non-Microsoft application with this should be reviewed carefully



	A	B	C	D	E	F
1	ClientDisplayName	Risk	UsersAssignedCount	MicrosoftRegisteredClientApp		
2	PIC_Temp	High	AllUsers	FALSE		
3	PnP Management Shell	High	AllUsers	FALSE		

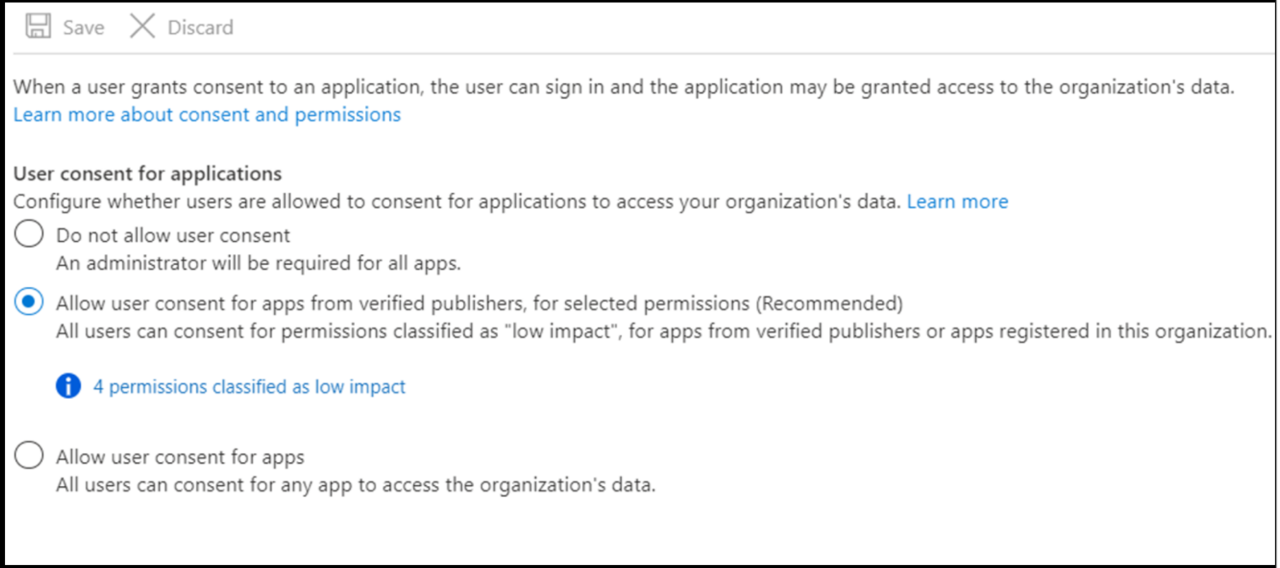
- Review HighRiskUsers Tab
  - Start with those that have high privilege or access to sensitive info (C suite, finance, etc)

- Review Permissions for each delegated application
  - Look for "Read" and "Write" permission or "*.All" permission, and review these carefully because they may not be appropriate.

Steps to protect your organization

#1 Set Policies

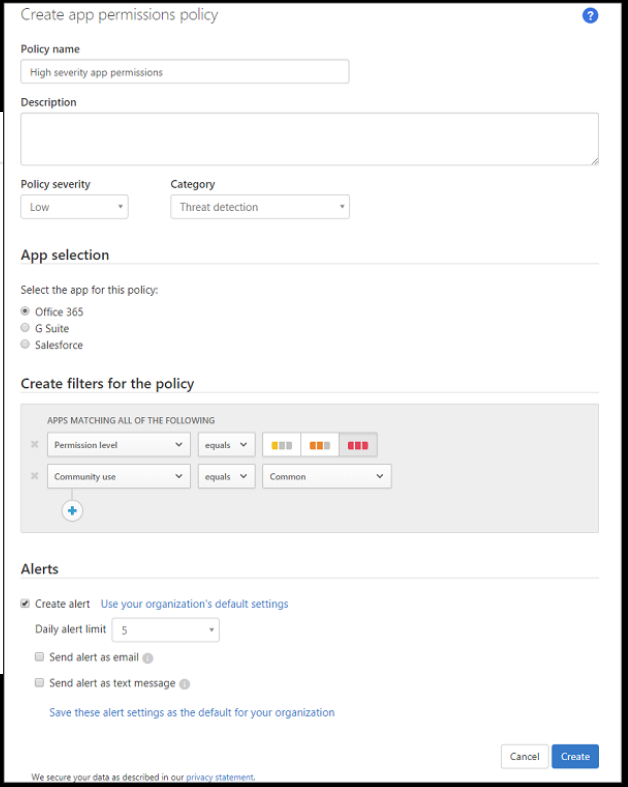
- Use app consent policies to limit user consent to apps- e.g. only from verified publishers requesting low risk permissions
- Use Microsoft Cloud App Security to automatically revoke an app or a specific user from an app when risk is detected



When a user grants consent to an application, the user can sign in and the application may be granted access to the organization's data.
Learn more about consent and permissions

User consent for applications
Configure whether users are allowed to consent for applications to access your organization's data. Learn more

- ○ Do not allow user consent
  An administrator will be required for all apps.
- [✓ Allow user consent for apps from verified publishers, for selected permissions (Recommended)]
  All users can consent for permissions classified as "low impact", for apps from verified publishers or apps registered in this organization.
  4 permissions classified as low impact
- ○ Allow user consent for apps
  All users can consent for any app to access the organization's data.



Create app permissions policy

Policy name: High severity app permissions

Description:

Policy severity: Low
Category: Threat detection

App selection

Select the app for this policy:
- [✓ Office 365]
- G Suite
- Salesforce

Create filters for the policy

APPS MATCHING ALL OF THE FOLLOWING
- Permission level equals ▨ ▨ 🟥
- Community use equals Common

Alerts

- [✓ Create alert] Use your organization's default settings
  Daily alert limit: 5
- ☐ Send alert as email
- ☐ Send alert as text message

Save these alert settings as the default for your organization

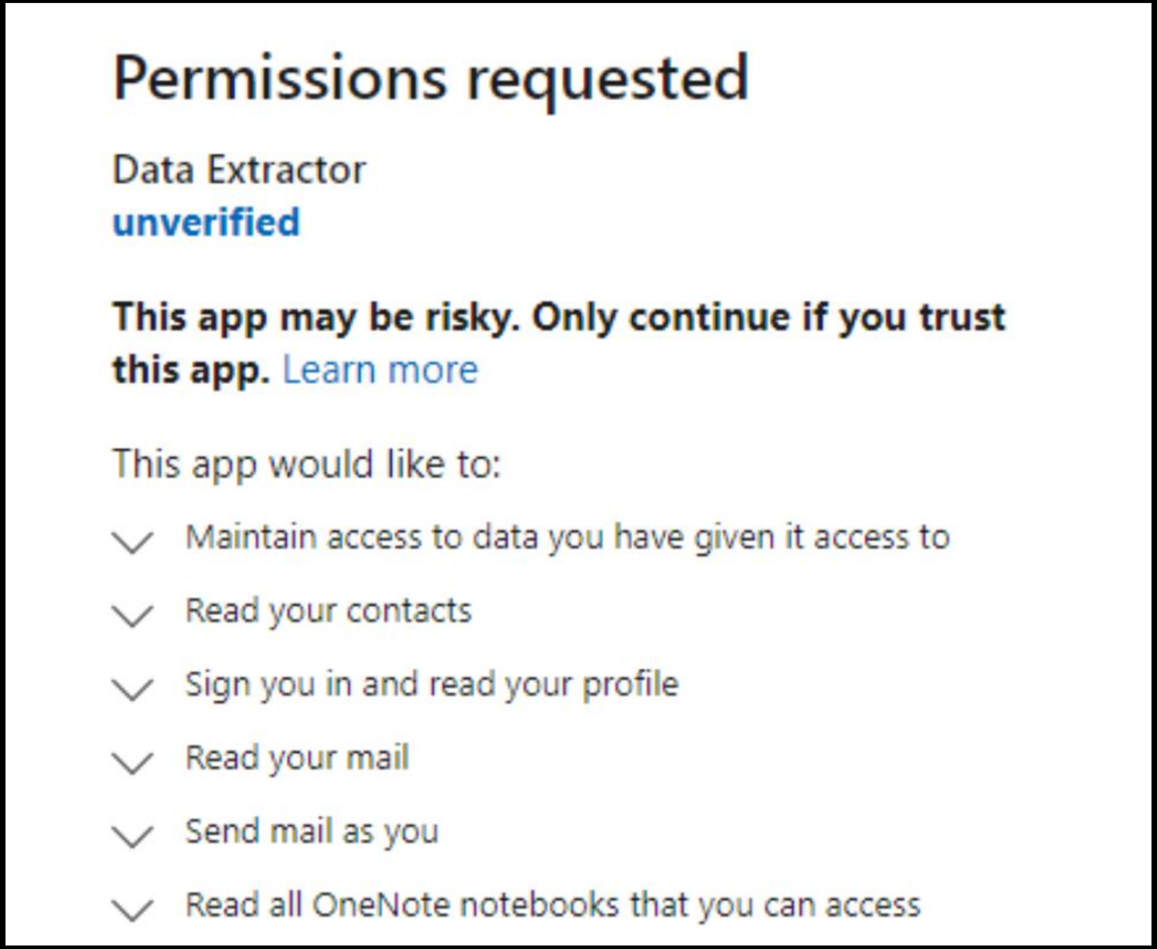
We secure your data as described in our privacy statement.

Steps to protect your organization

#2 Risk-based user step-up consent (enabled by default)

Risk-based step-up consent:

- When a **risky consent** request is detected, request will be "stepped up" to **require admin approval**
- **Warning will be shown** to users and admins, but only admin can grant permissions
- Audit event will be logged



Permissions requested

Data Extractor
unverified

This app may be risky. Only continue if you trust this app. Learn more

This app would like to:

- Maintain access to data you have given it access to
- Read your contacts
- Sign you in and read your profile
- Read your mail
- Send mail as you
- Read all OneNote notebooks that you can access

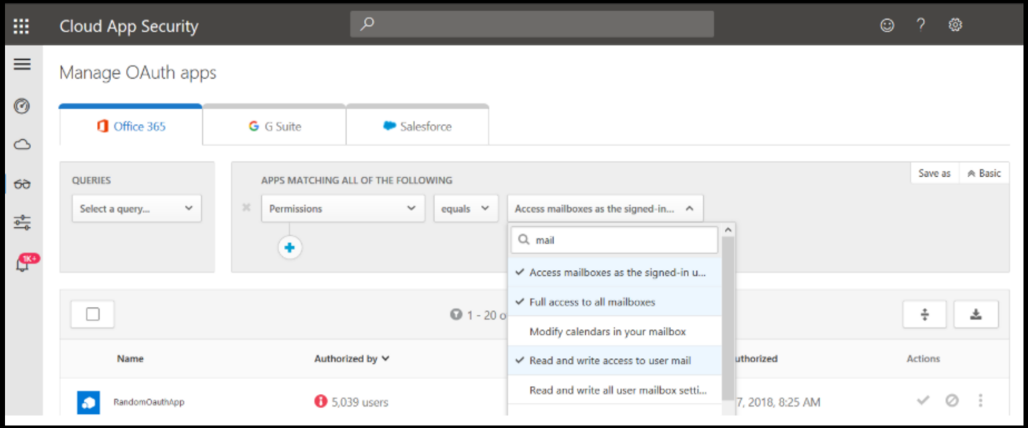
Steps to protect your organization

#3 Detect risky OAuth apps

- Good: Audit apps and consented permissions
  - https://aka.ms/getazureadpermissions

- Better: Use Azure Monitor to set alerts to automatically send you notifications when an OAuth app meets certain criteria
  - App requires high permissions
  - App was authorized by >50 users

- Best: Detect risky apps by hunting using CASB like MCAS
  - Permission level high security
  - Community use not common
  - Apps authorized by external users


Cloud App Security

Manage OAuth apps

Office 365 | G Suite | Salesforce

QUERIES: Select a query...

APPS MATCHING ALL OF THE FOLLOWING: Permissions equals Access mailboxes as the signed-in...

Save as | Basic

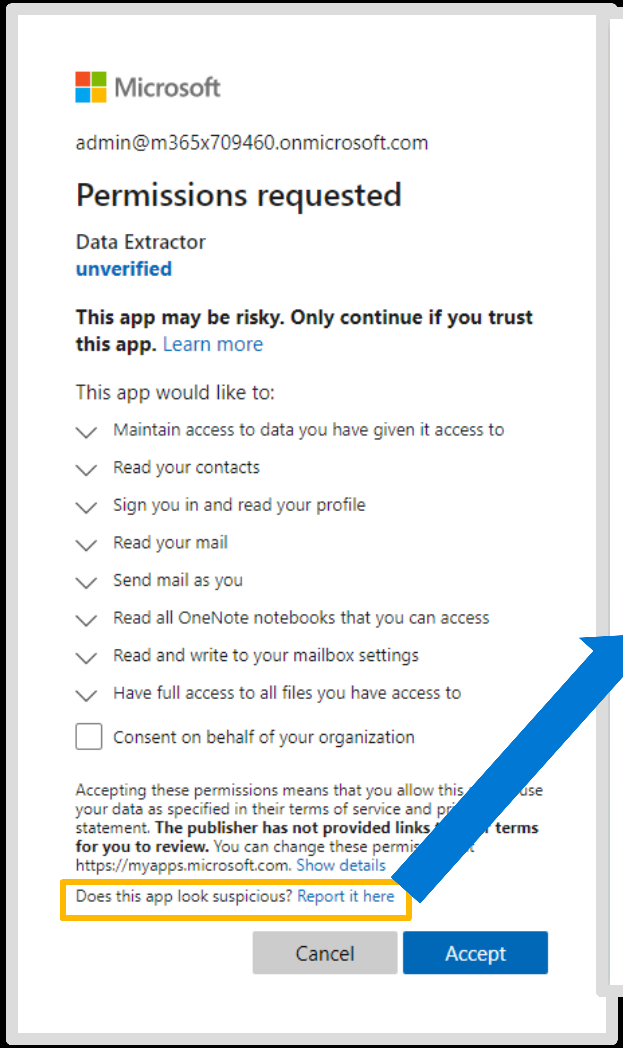
mail
- ✓ Access mailboxes as the signed-in u...
- ✓ Full access to all mailboxes
- Modify calendars in your mailbox
- ✓ Read and write access to user mail
- Read and write all user mailbox setti...

1 - 20 o

Name	Authorized by	uthorized	Actions
RandomOauthApp	5,039 users	7, 2018, 8:25 AM	

Report suspicious apps

- Report suspicious apps to Microsoft for investigation directly from the consent screen or using MCAS



Microsoft

admin@m365x709460.onmicrosoft.com

Permissions requested

Data Extractor
unverified

This app may be risky. Only continue if you trust this app. Learn more

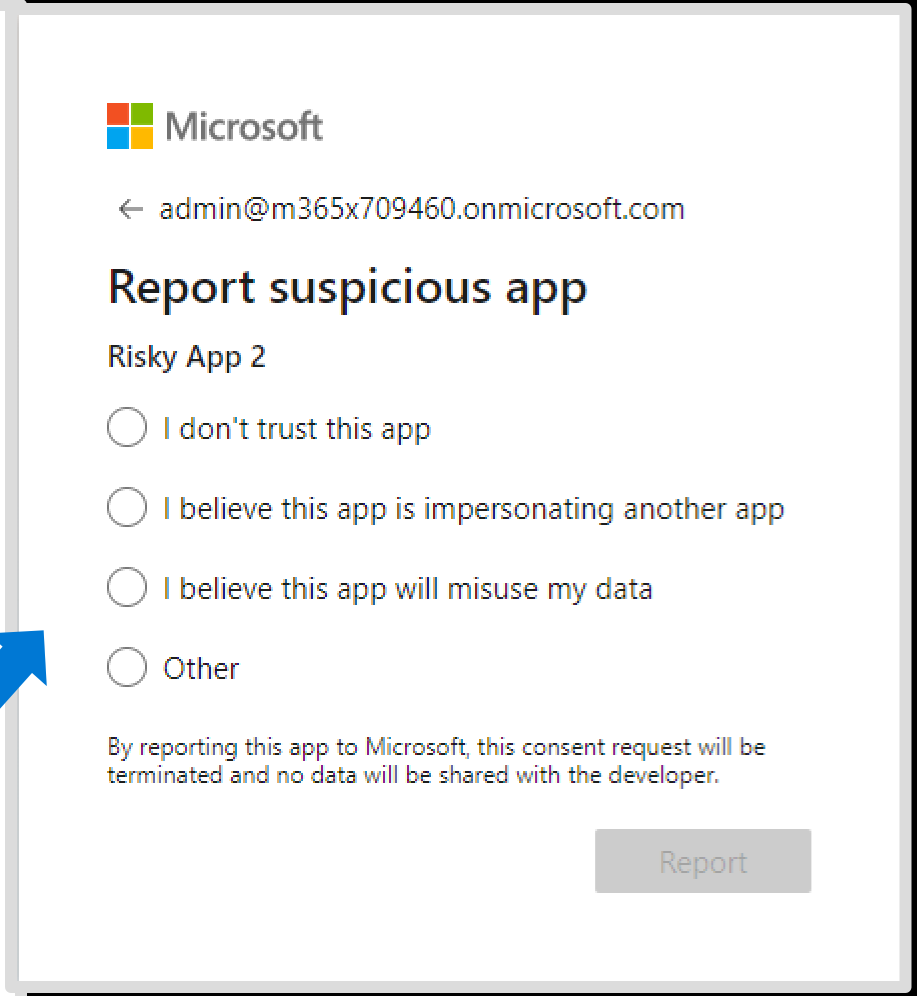
This app would like to:

- Maintain access to data you have given it access to
- Read your contacts
- Sign you in and read your profile
- Read your mail
- Send mail as you
- Read all OneNote notebooks that you can access
- Read and write to your mailbox settings
- Have full access to all files you have access to

☐ Consent on behalf of your organization

Accepting these permissions means that you allow this use your data as specified in their terms of service and pri statement. **The publisher has not provided links r terms for you to review.** You can change these permis t https://myapps.microsoft.com. Show details

Does this app look suspicious? Report it here



Microsoft

← admin@m365x709460.onmicrosoft.com

Report suspicious app

Risky App 2

- ○ I don't trust this app
- ○ I believe this app is impersonating another app
- ○ I believe this app will misuse my data
- ○ Other

By reporting this app to Microsoft, this consent request will be terminated and no data will be shared with the developer.

App Consent Further Resources

- Have you updated your IR playbooks?
  - https://aka.ms/irplaybooks
- Sample "Malicious" App Consent
  - Written in Python
  - Dumps the directory to demonstrate impact
  - https://github.com/MarkMorow/AppConsentPhishingSample
- A full talk on this problem
  - https://aka.ms/BSidesCT2020AppConsent

Agenda

Why Modern Auth?

SAML

OAuth2

OIDC

App Consent Phishing Attack

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
  - 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
- Go deeper on OAuth2/OpenID Connect/SAML/JWT
  - This is the new NTLM/Kerberos/Tickets

Q&A

Bailey Bercik @baileybercik
Mark Morowczynski @markmorow
Program Managers – Microsoft