Finding Entra ID CA bypasses the structured way

TROOPERS

Intro – Fabian Bader

glueck▫kanja

- Lives in Germany
- Cyber Security Architect / Researcher @ glueckkanja AG
- Microsoft MVP
- First time speaker at TROOPERS
- Organizer of “Purple Elbe Security User Group”
- Author of
  - TokenTacticsV2
  - SentinelARConverter

Socials
Blog/talks: cloudbrothers.info
Twitter/X: @fabian_bader
BlueSky: @fabian.bader.cloud

Intro – Dirk-jan Mollema

/OUTSIDER SECURITY

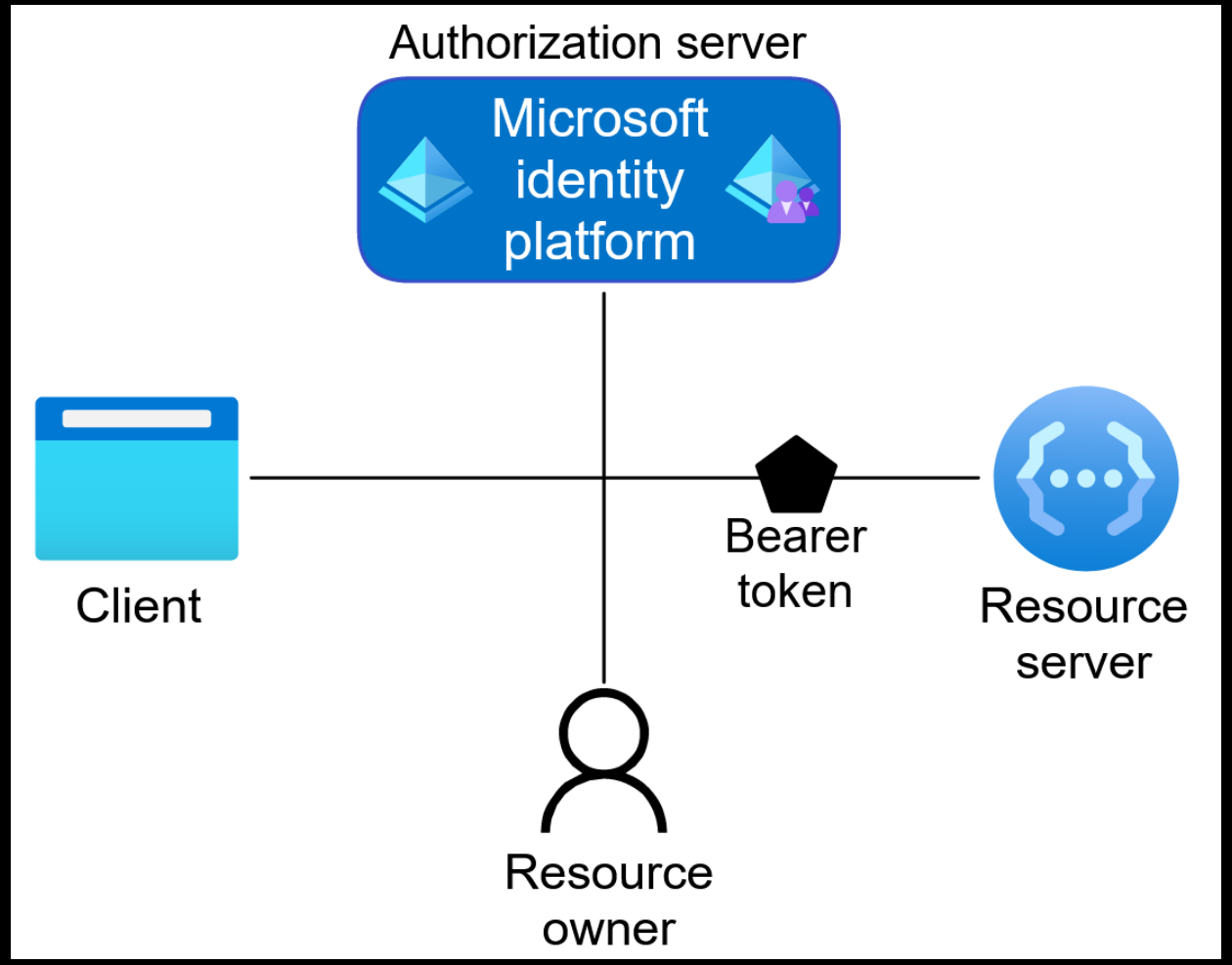
- Lives in The Netherlands
- Hacker / Researcher / Founder / Trainer @ Outsider Security
- Microsoft MVP and MVR
- TROOPERS veteran since 2019 (and some other conferences)
- Author of several Active Directory and Entra ID tools
  - mitm6
  - ldapdomaindump
  - adidnsdump
  - BloodHound.py
  - ntlmrelayx / krbrelayx
  - ROADtools

Socials
Blog/talks: dirkjanm.io
Twitter/X: @_dirkjan
BlueSky: @dirkjanm.io

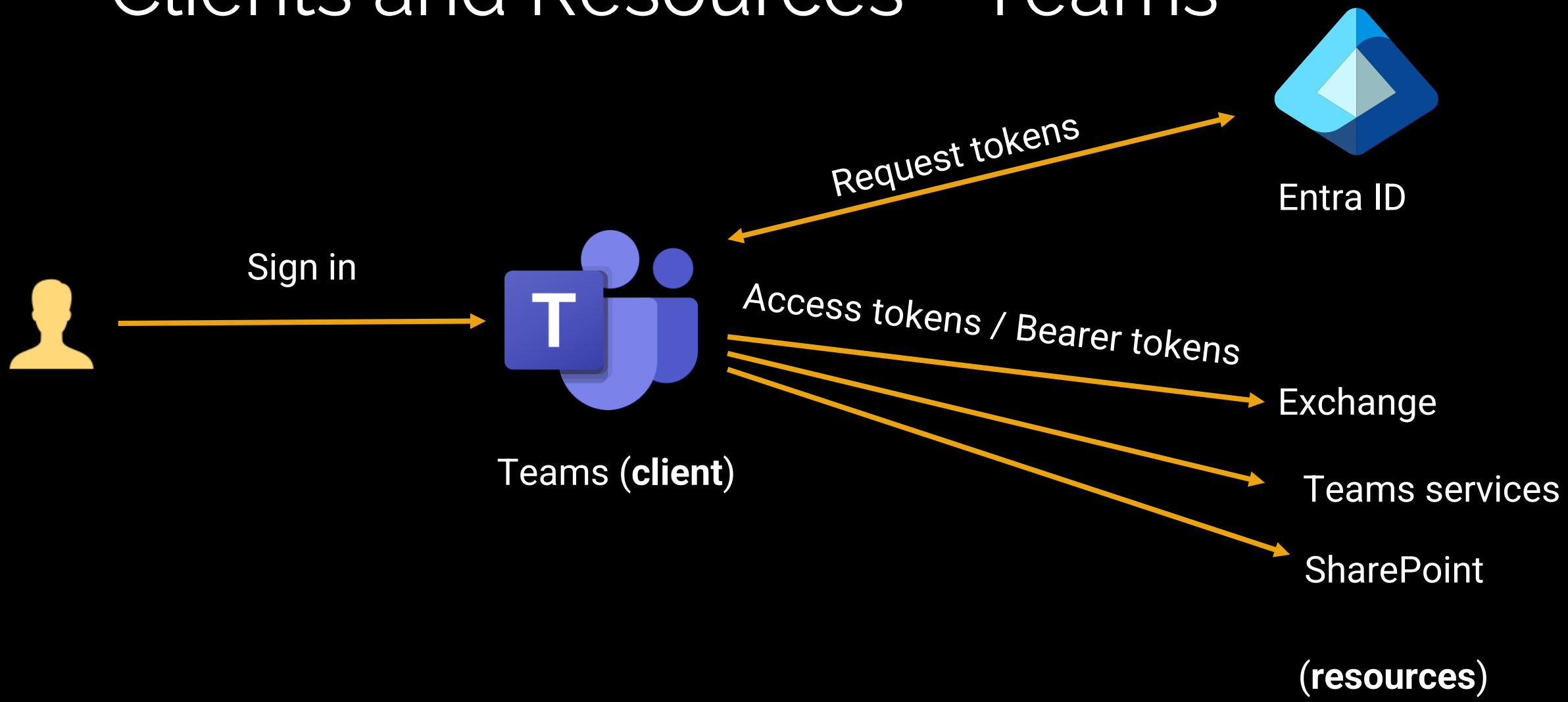
Talk Agenda

- Intro: Apps, clients, resources, scopes and FOCI
- Research starting point and goals
- Conditional access policies and unstructured bypasses/limits
- The structured approach:
  - Finding all apps and permissions
  - Testing all the apps for undocumented exclusions and bypasses
- Findings and conclusions

Clients and Resources – OAuth2



Clients and Resources - Teams



Token scopes

- Each combination of client and resource will result in a token with a specific **scope**
- The scope indicates what can be done with a token on behalf of a user: e.g. Mail.Read, User.Read, Files.ReadWrite
- For most apps, theses scopes need to be consented to by admins or individual users (Microsoft docs and portals call this delegated permissions)

First Party Apps

- Microsoft apps do not require admin consent – they are pre-consented by Microsoft.
- These “first-party“ apps exist in every tenant.
- There are hundreds of them, however we cannot query the scopes since that is determined on the Microsoft side.

- Many of these are **public clients**, which means we can “borrow” their client ID to get a token without approval.

Family of Client IDs

- Undocumented Microsoft “feature” that creates a family of applications / clients (FOCI)
- Each of these apps can use each others **refresh tokens**.
- We can get the **pre-consented scopes** of any of these clients after authenticating to a single client.
- Clear violation of the OAuth2 standard.
- At the start of our research **38** known FOCI clients

Reference: https://github.com/secureworks/family-of-client-ids-research

Research goals

- With all first-party apps we could find from the internet + customer environments:
  1. Sign in to all the clients
  2. Get tokens for all the resources
  3. Find all the scopes
  4. Find all the possible CA bypasses for these scopes
  5. ~~Profit~~ Give a cool talk at TROOPERS

How to find first party Microsoft apps

- Use sign-in logs
- Request key information about the app from Azure AD Graph API
- If member of SharePoint MSFT tenant = First Party App
- Other method
  - Compare with list of registered app ids in the tenant that are not from Microsoft
- Result: A long list of >500 app ids that belong to MSFT
  - The more environments you have, there more you will find

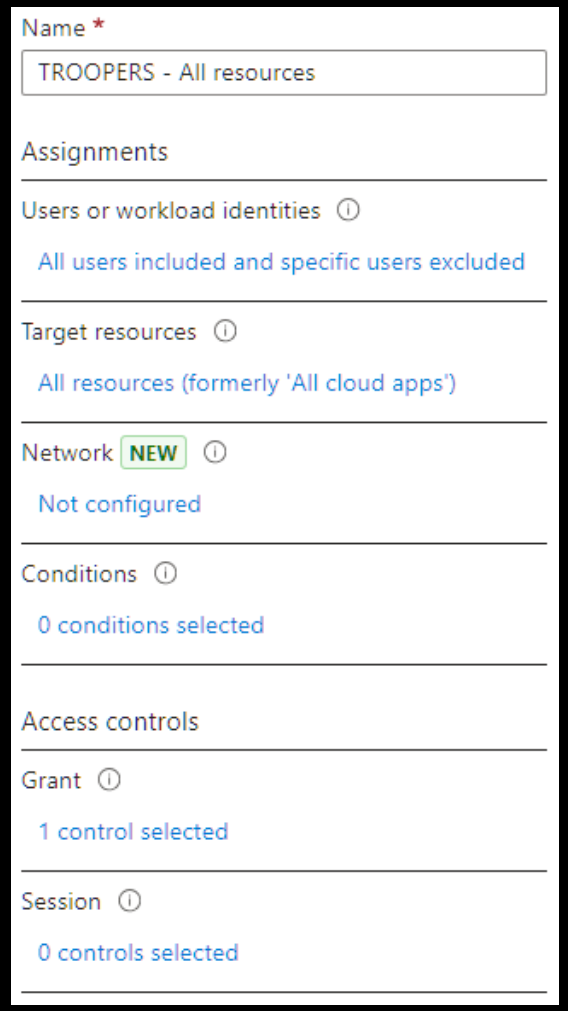
Conditional Access Policies

And why they not work as you sometimes expect

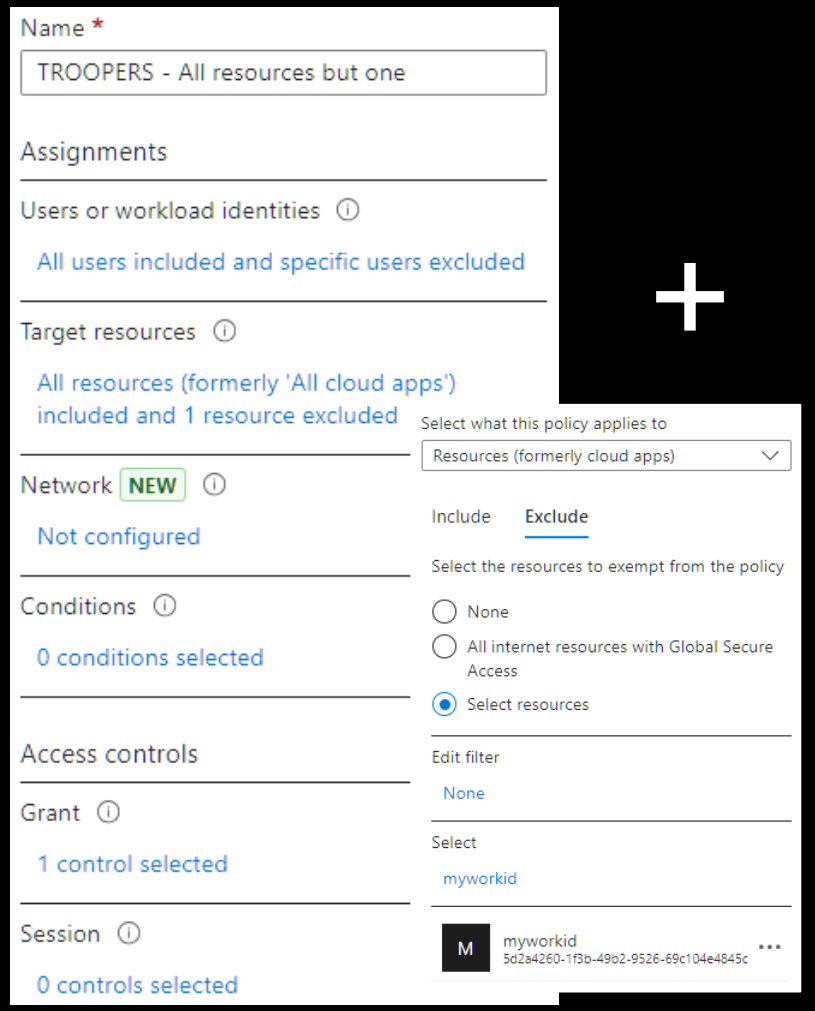
Conditional Access Policies

- Security feature in Microsoft Entra that enforces access decisions based on specific conditions like user location, device compliance, or risk level.
- Scoped to
  - Resources OR
  - User actions OR
  - Authentication context
- Enforce access controls like
  - Multifactor required
  - Compliant device required

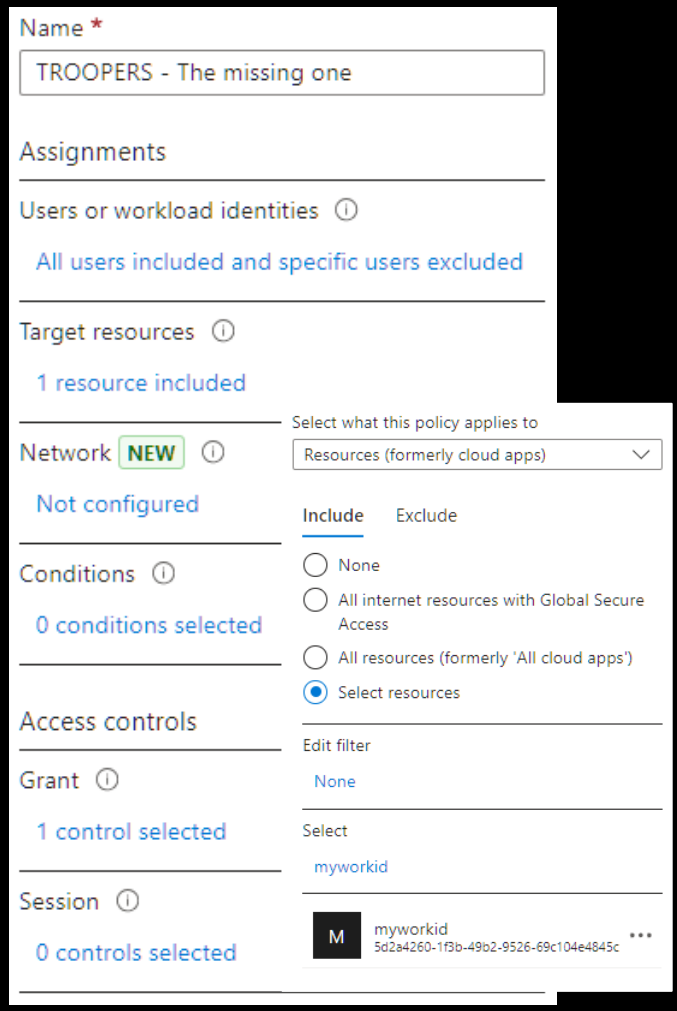
Conditional Access Policies



vs.



+



Conditional Access Policies

Conditional Access behavior when an all resources policy has an app exclusion

If any app is excluded from the policy, in order to not inadvertently block user access, certain low privilege scopes are excluded from policy enforcement. These scopes allow calls to the underlying Graph APIs, like `Windows Azure Active Directory` (00000002-0000-0000-c000-000000000000) and `Microsoft Graph` (00000003-0000-0000-c000-000000000000), to access user profile and group membership information commonly used by applications as part of authentication. For example: when Outlook requests a token for Exchange, it also asks for the `User.Read` scope to be able to display the basic account information of the current user.

Most apps have a similar dependency, which is why these low privilege scopes are automatically excluded whenever there's an app exclusion in an **All resources** policy. These low privilege scope exclusions don't allow data access beyond basic user profile and group information. The excluded scopes are listed as follows, consent is still required for apps to use these permissions.

- Native clients and Single page applications (SPAs) have access to the following low privilege scopes:
  - Azure AD Graph: `email`, `offline_access`, `openid`, `profile`, `User.Read`
  - Microsoft Graph: `email`, `offline_access`, `openid`, `profile`, `User.Read`, `People.Read`
- Confidential clients have access to the following low privilege scopes, if they're excluded from an **All resources** policy:
  - Azure AD Graph: `email`, `offline_access`, `openid`, `profile`, `User.Read`, `User.Read.All`,`User.ReadBasic.All`
  - Microsoft Graph: `email`, `offline_access`, `openid`, `profile`, `User.Read`, `User.Read.All`, `User.ReadBasic.All`, `People.Read`, `People.Read.All`, `GroupMember.Read.All`, `Member.Read.Hidden`

For more information on the scopes mentioned, see Microsoft Graph permissions reference and Scopes and permissions in the Microsoft identity platform.

Reference: https://learn.microsoft.com/en-us/entra/identity/conditional-access/concept-conditional-access-cloud-apps

The *User.ReadBasic.All* permission constrains app access to reading a limited set of properties for other users' work or school accounts. This basic profile includes only the following prop

- displayName

People.Read.All

Expand table

	Delegated
5	b89f9189-71a5-4e70-b041-9887f0bc7e4a

User.Read.All

Expand table

Category	Application
Identifier	df021288-bdef-4463-88db-
Displa	
Descr	
Admi	

GroupMember.Read.All

Expand table

153-4739-b217-4326f2e966d0
memberships
app to list groups, read basic group properties embership of all groups the signed-in user to.

Member.Read.Hidden

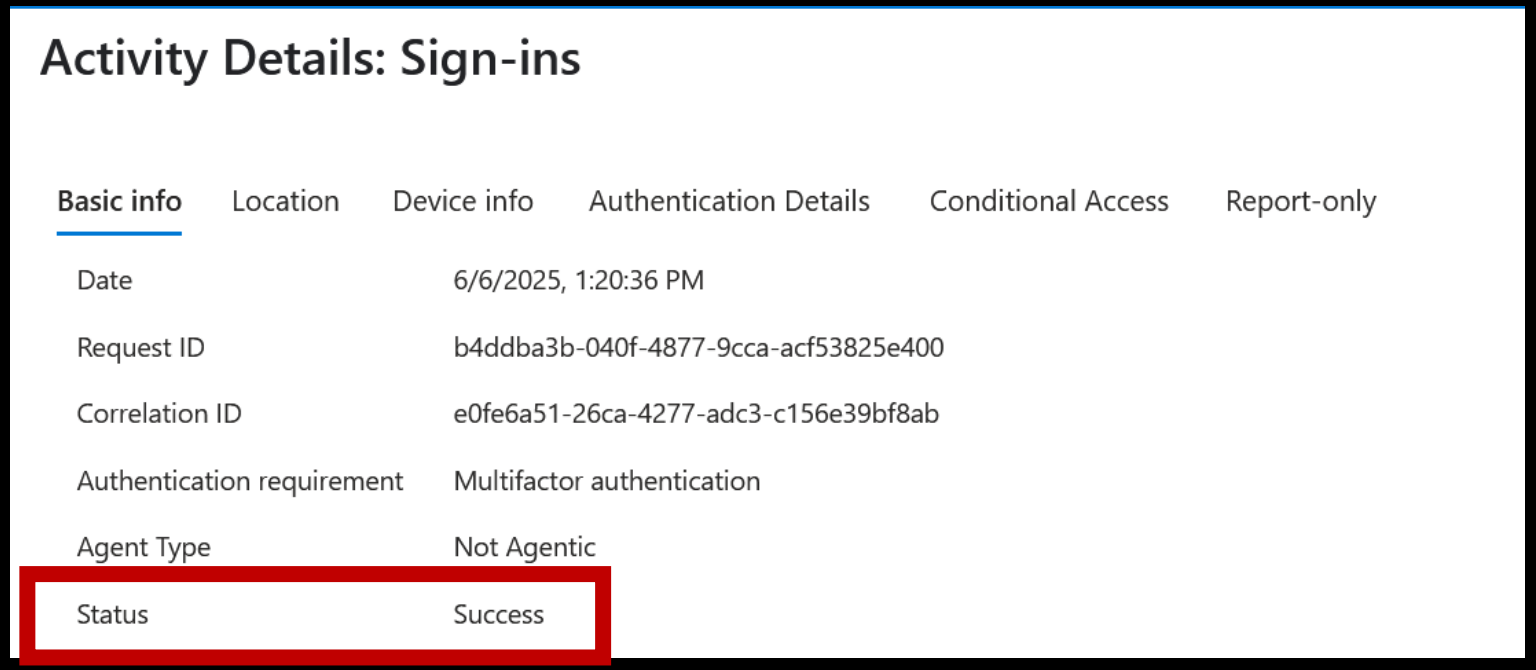
Expand table

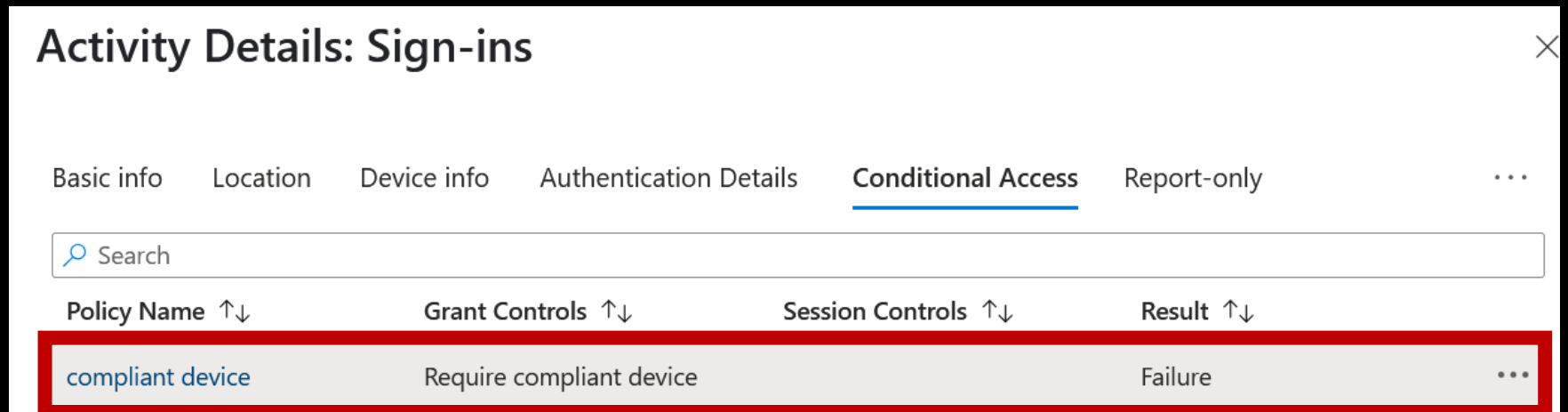
Category	Application	Delegated
Identifier	658aa5d8-239f-45c4-aa12-864f4fc7e490	f6a3db3e-f7e8-4ed2-a414-557c8c9830be
DisplayText	Read all hidden memberships	Read hidden memberships
Description	Allows the app to read the memberships of hidden groups and administrative units without a signed-in user.	Allows the app to read the memberships of hidden groups and administrative units on behalf of the signed-in user, for those hidden groups and administrative units that the signed-in user has access to.
AdminConsentRequired	Yes	Yes

Company Portal CA bypass

- Company Portal app is used on mobile devices to enroll the device into Intune
- Logically we cannot yet have a compliant device at the moment we are enrolling the device.
- The Company Portal client is a hardcoded exclusion for device compliance policies.
- However, the Company Portal had extensive scopes on the Azure AD Graph (**user_impersonation**), allowing for full tenant enumeration and modification by admins without CA enforcing device compliance.

Company Portal CA bypass





Company Portal CA bypass disclosure

- Encountered by me in 2023 during Intune research and shared privately with vetted red teams.
- Publicly disclosed by Yuya Chudo (Secureworks) at Black Hat Europe in 2024.
- Despite being classified as “not an issue” by MSRC still fixed quite fast after public disclosure.
- Still excluded from CA but scopes have been heavily reduced.

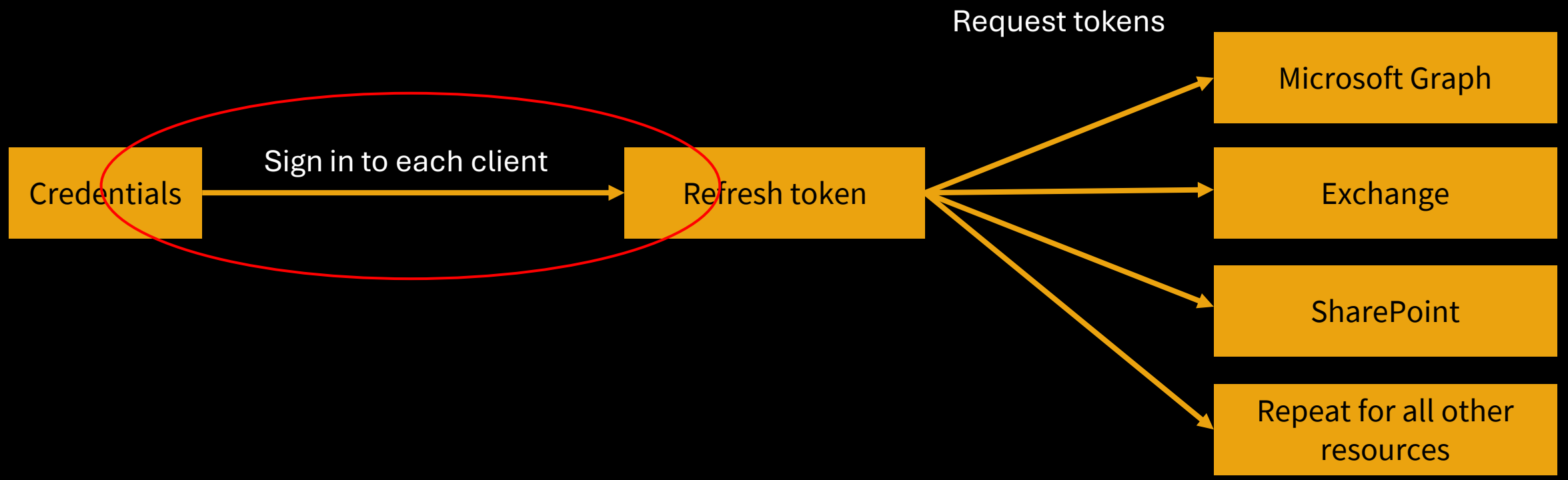
Device registration CA bypass

- Device registration needs a separate policy
- Often forgotten for MFA enforcement: register devices without MFA
- Also used for some MFA method registrations (Windows Hello)
- Especially relevant if enforcing authentication strengths, registration service often forgotten and can be used as bypass.

Signing in to all the apps

Finding all first party clients and their pre-approved scopes

Basic approach

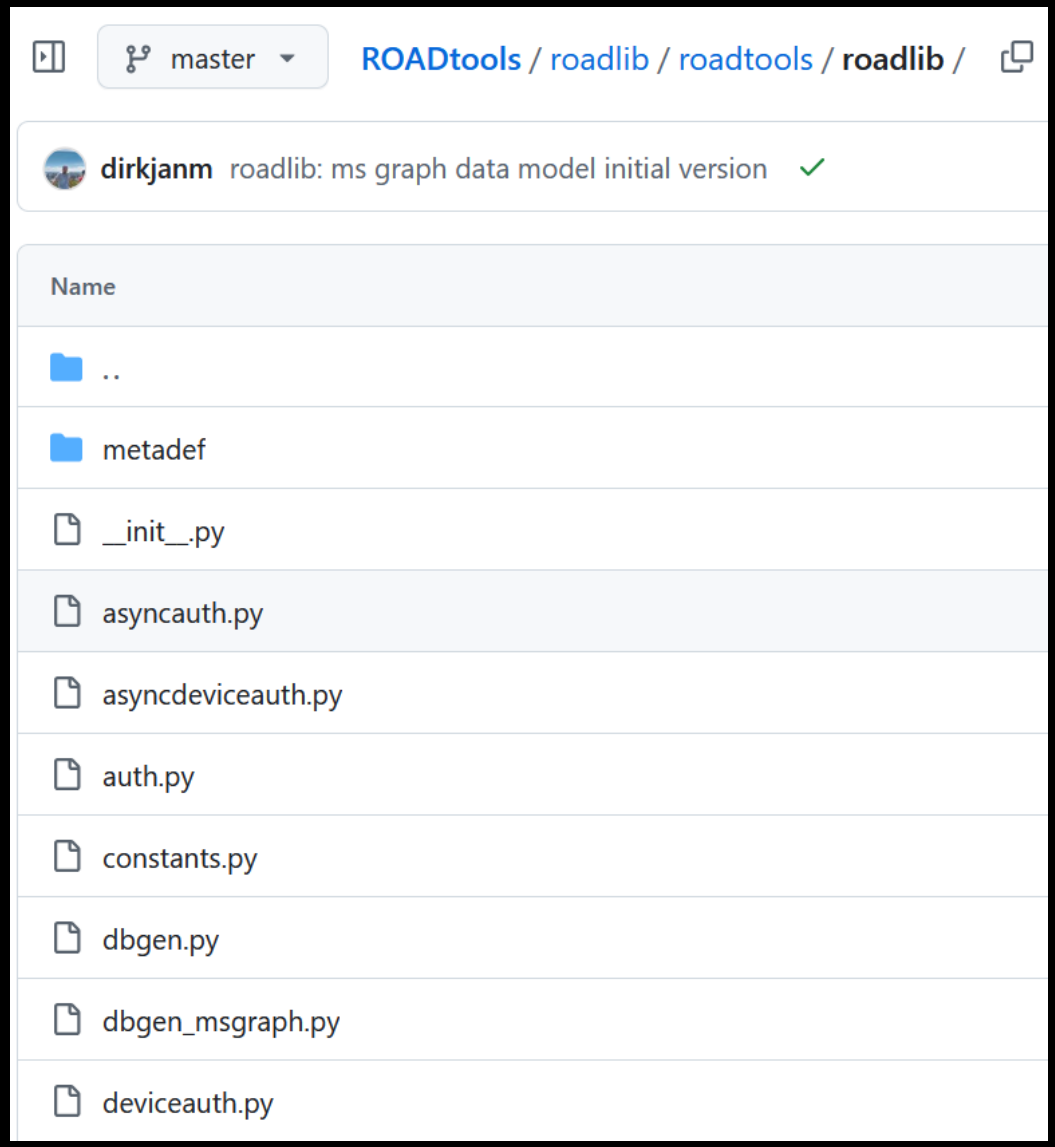


Sign in to all the apps – round 1

- Sign in to public clients
  - ROPC flow allows easy username + password auth
  - Gives a refresh token
- With refresh token, sign in to all possible resources
  - Record scopes for each resource
- Can also re-use refresh token for FOCI clients
  - Brute force all client IDs for FOCI refresh token
  - If auth succeeds, found a new FOCI client ☺

Practical approach

- Wrote a python script that signed in to all the apps
- Problem: doing each app in turn is quite slow
- Solution: rewrite roadtools authentication stack to support async operations, allowing for parallel enumeration


master
ROADtools / roadlib / roadtools / roadlib /
dirkjanm roadlib: ms graph data model initial version
Name
..
metadef
__init__.py
asyncauth.py
asyncdeviceauth.py
auth.py
constants.py
dbgen.py
dbgen_msgraph.py
deviceauth.py

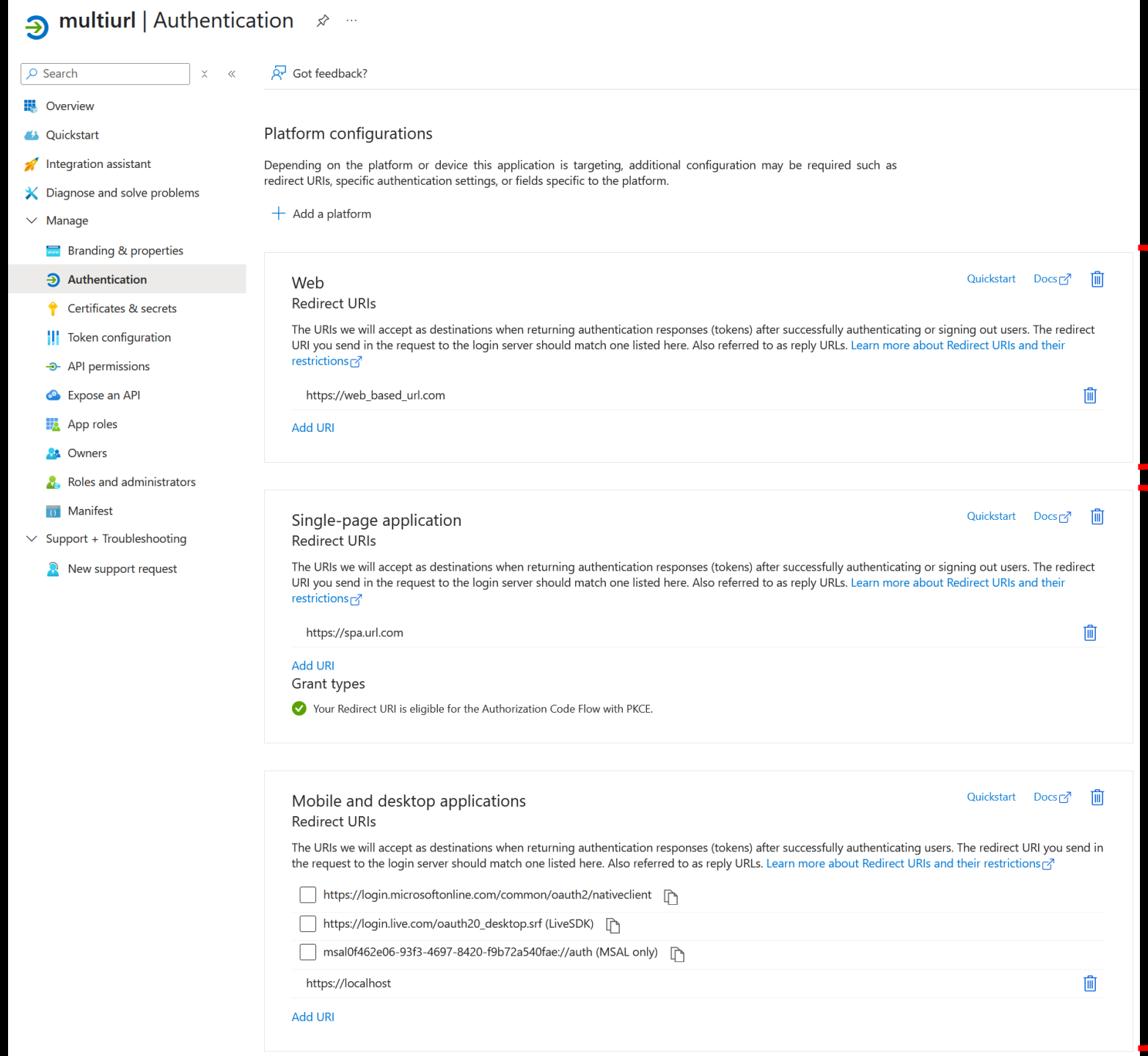
Round 1 results

- Total enumerated clients: 219
- Total FOCI clients: 48 (10 new)
- Total tokens requested: 347k

```
Total token requests 347374
Scope brute force executed in 540.04
```

Sign in to all the apps – round 2

- Not all apps are marked as public clients – means ROPC won't work
- Many apps are "hybrid" apps
  - Act as public clients for some URLs
  - Act as confidential clients for other URLs



Confidential client URLs

Public client URLs

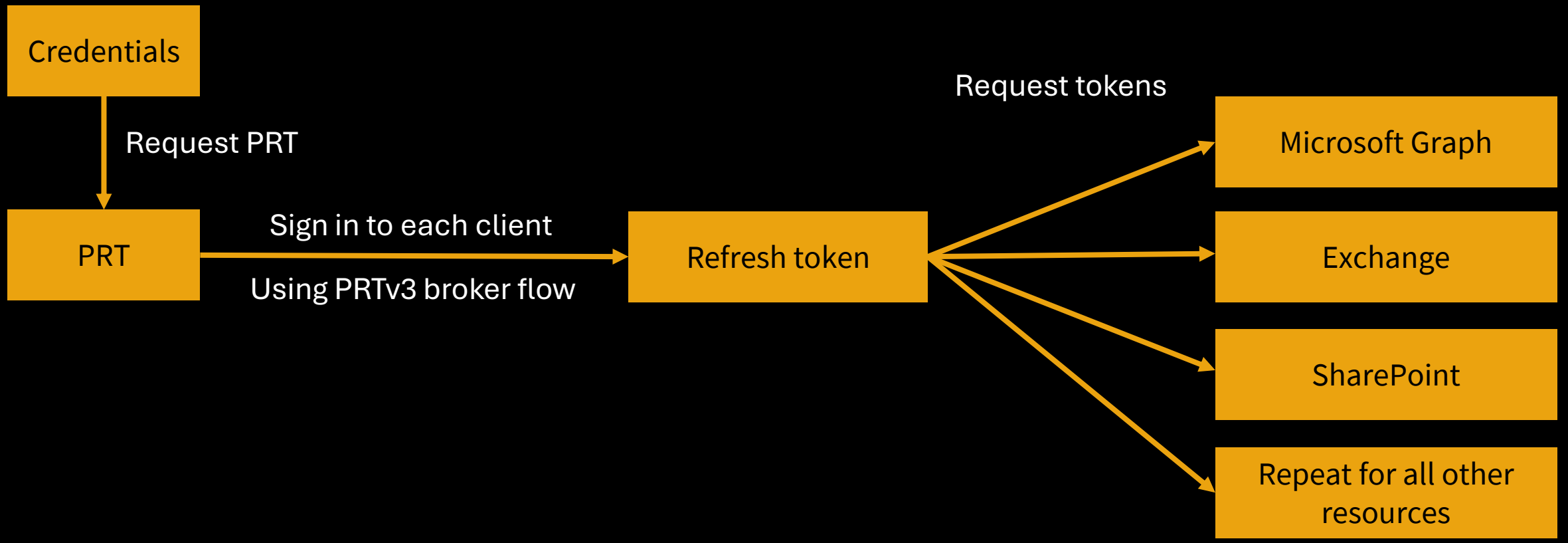
Round 2 challenges

- We cannot query for third-party apps what the URL type is.
- These flows require interactive authentication, such as OAuth2 authorization code flow.
- Spawning a browser is error-prone and slow.

Solution: Primary Refresh Tokens

- Primary Refresh Tokens are SSO tokens.
- Valid for every app.
- PRTv3 protocol (discussed last year at Troopers) works with scope parameter.
- Broker flow uses redirect URL parameter.
- Works for both native app URLs and Single Page App URLs

PRT based approach



Round 2 - subrounds

- Start with obvious public URLs:
  - https://login.microsoftonline.com/common/oauth2/nativeclient
  - Any non-http(s) protocol specifier such as ms-appx-web / msauth://
- Add more likely public URLs:
  - localhost http/https / 127.0.0.x URLs
  - Other non-standard domains
- In the end just attempt all possible redirect URLs until one working is found.

Round 2 - results

- Total enumerated clients: 229
- Total FOCI clients: 50 (2 new)
- Total tokens requested: ~600k

Round 3 – scope troubles

- Problem: not every client has a Microsoft Graph scope.
- Solution: ask only for **openid** scope with **offline_access** to get a refresh tokens

- Problem: apparently not every client has an **openid** scope either.
- Solution: loop over all the resources until we find an allowed scope for that client, then start enumeration with refresh tokens.

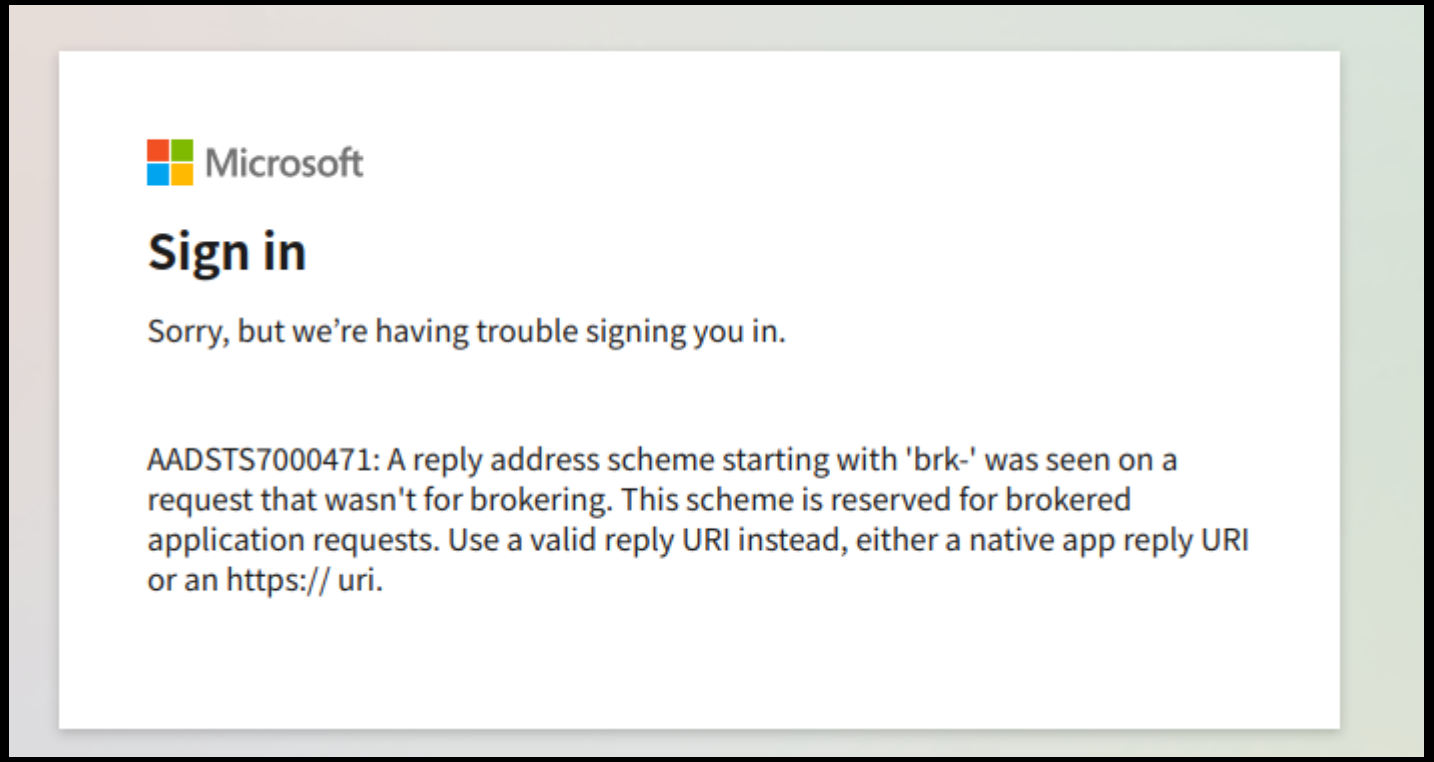
- Total number of clients from 229 to 245

Round 4 - brk

- Weird redirect URLs that start with “brk-”


```
"5926fc8e-304e-4f59-8bed-58ca97cc39a4": {
    "foci": false,
    "name": "Microsoft Intune portal extension",
    "preferred_interactive_redirurl": null,
    "preferred_noninteractive_redirurl": "brk-c44b4083-3bb0-49c1-b47d-974e53cbdf3c://lighthouse.microsoft.com",
    "public_client": false,
    "redirect_uris": [
        "brk-c44b4083-3bb0-49c1-b47d-974e53cbdf3c://lighthouse.microsoft.com",
        "brk-c44b4083-3bb0-49c1-b47d-974e53cbdf3c://canary-endpoint.microsoft.com",
        "brk-c44b4083-3bb0-49c1-b47d-974e53cbdf3c://intuneeducation.portal.azure.com",
        "brk-c44b4083-3bb0-49c1-b47d-974e53cbdf3c://aad.portal.azure.com",
        "brk-c44b4083-3bb0-49c1-b47d-974e53cbdf3c://canary-intune.microsoft.com",
        "brk-c44b4083-3bb0-49c1-b47d-974e53cbdf3c://intune.microsoft.com",
        "brk-c44b4083-3bb0-49c1-b47d-974e53cbdf3c://canary.entra.microsoft.com",
        "brk-c44b4083-3bb0-49c1-b47d-974e53cbdf3c://entra.microsoft.com",
        "brk-c44b4083-3bb0-49c1-b47d-974e53cbdf3c://portal.azure.com",
        "brk-c44b4083-3bb0-49c1-b47d-974e53cbdf3c://canary.portal.azure.com",
        "brk-c44b4083-3bb0-49c1-b47d-974e53cbdf3c://preview.portal.azure.com",
        "brk-c44b4083-3bb0-49c1-b47d-974e53cbdf3c://ms.portal.azure.com",
        "brk-c44b4083-3bb0-49c1-b47d-974e53cbdf3c://canary-ms.portal.azure.com",
        "brk-c44b4083-3bb0-49c1-b47d-974e53cbdf3c://rc.portal.azure.com",
        "brk-c44b4083-3bb0-49c1-b47d-974e53cbdf3c://endpoint.microsoft.com",
```

Broker redirect URL



Broker redirect URLs

brk-c44b4083-3bb0-49c1-b47d-974e53cbdf3c://portal.azure.com

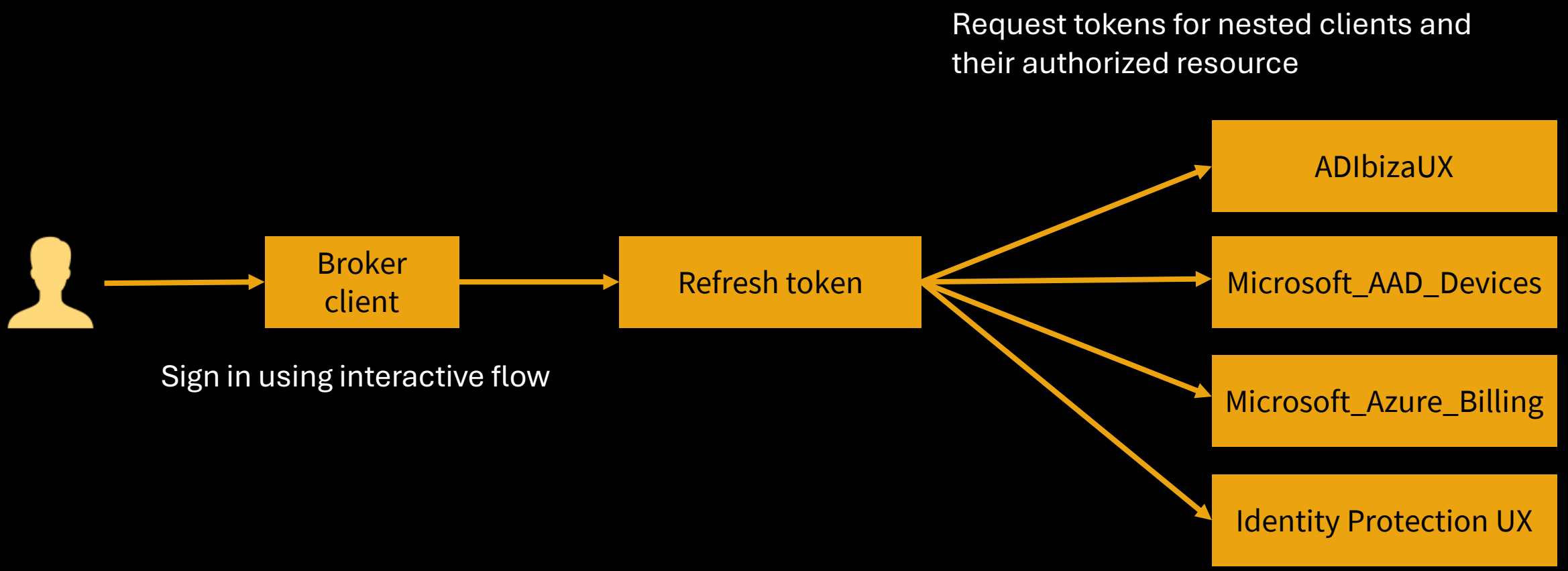
Hardcoded prefix

Allowed client ID

Allowed origin URL

- Main broker clients:
  - Azure Portal (c44b4083-3bb0-49c1-b47d-974e53cbdf3c)
  - Office (specific client ID or “multihub”)

Broker auth flow

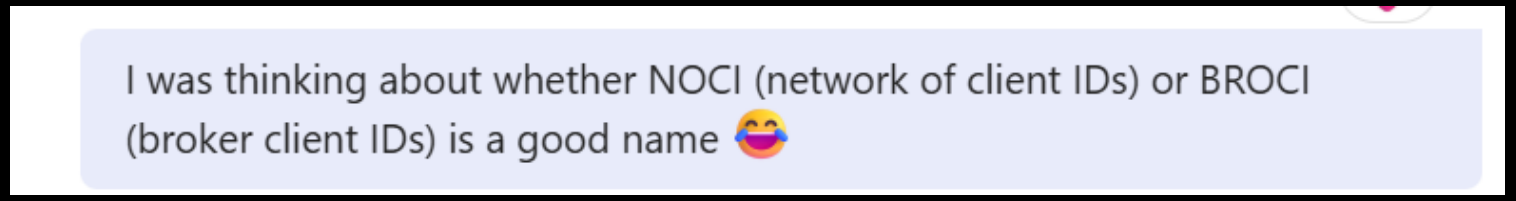


Brokered authentication

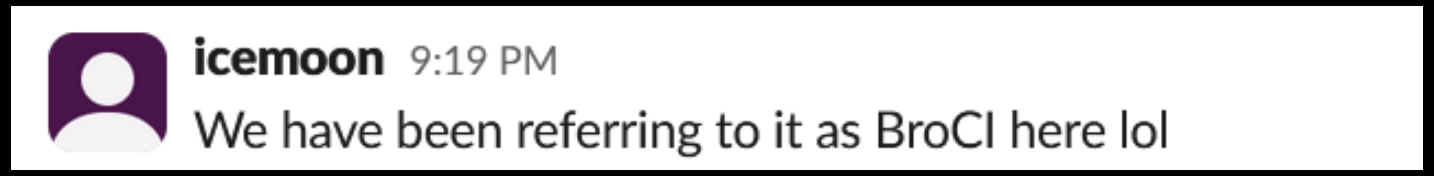
- Kind of FOCI-lite
- Apps that can use their refresh token for sub-apps
- Suggested names:
  - BroCI (Broker Client IDs)
  - NOCI (Network of Client IDs)
- Turns out there is already a name: Nested App Authentication (NAA)
- Sort of documented here: https://learn.microsoft.com/en-us/microsoftteams/platform/concepts/authentication/nested-authentication

Researchers think alike

Me talking to Fabian:



A few weeks later in a chat with Hope from SpecterOps



Brokered auth with roadtx

- Request initial refresh token for Azure Portal:

```
(ROADtools) → ROADtools git:(master) ✗ roadtx interactiveauth -u newlowpriv@iminyour.cloud -c c44b4083-3bb0-49c1-b47d-974e53cbdf3c
 --pkce --origin https://portal.azure.com
Tokens were written to .roadtools_auth
```

- Exchange for nested clients:

```
(ROADtools) → ROADtools git:(master) ✗ roadtx refreshtokento -c 74658136-14ec-4630-ad9b-26e160ff0fc6 --broker-client c44b4083-3bb0-49c1-b47d-974e53cbdf3c -
bru brk-c44b4083-3bb0-49c1-b47d-974e53cbdf3c://engagehub.portal.azure.com -s https://graph.microsoft.com/.default --origin https://portal.azure.com
Requesting token with scope https://graph.microsoft.com/.default
Tokens were written to .roadtools_auth
```

- Or, lazy mode:

```
(ROADtools) → ROADtools git:(master) ✗ roadtx refreshtokento -c 74658136-14ec-4630-ad9b-26e160ff0fc6 --autobroker -s https://graph.microsoft.com/.default
Requesting token with scope https://graph.microsoft.com/.default
Tokens were written to .roadtools_auth
```

Round 4

- Find all brokered clients
- Sign in to their allowed client ID to get “broker” refresh token
- Use brokered refresh token with correct origin to get tokens for broker client.
- Have to keep using the original refresh token for all resources: nested apps do not get their own refresh tokens.

Round 4 – final results

- Total enumerated clients: **537**
- Total FOCI clients: **52** (24 new in total) -1 that got disabled last week
- Total tokens requested in final run: **839k**
- Runtime: around 25 minutes
- Sentinel bill: ???

```
Total token requests 839404
Scope brute force executed in 1420.36
```

Parallel / prior research

- Around this time GraphPreConsentExplorer was released by zh54321 from Compass Security
- Also contained many clients, including 51 FOCI clients
- Also contained research on Broker auth

- Project link: https://github.com/zh54321/GraphPreConsentExplorer

Structured CA analysis

Find new CA bypasses

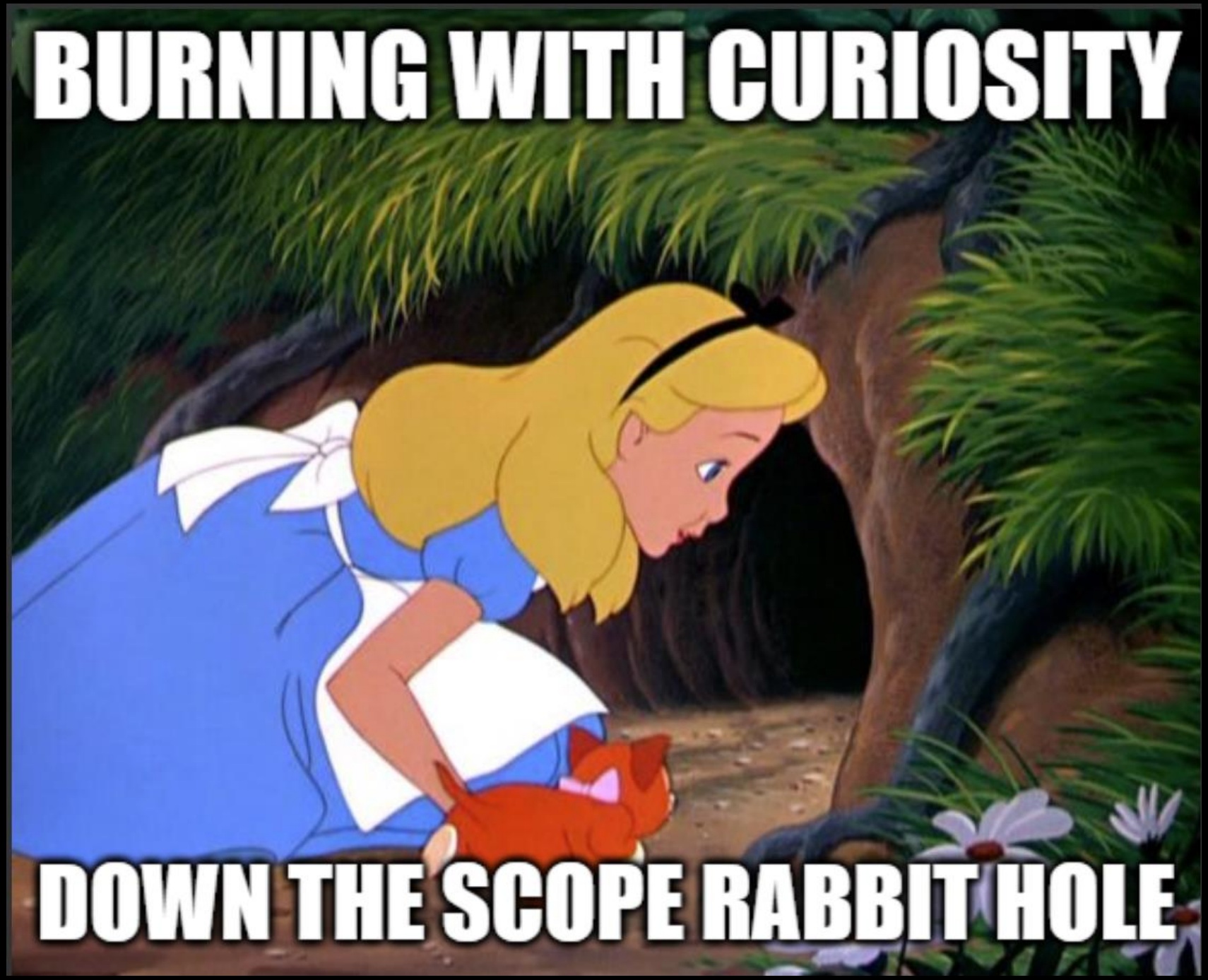
- Available resources
  - List of “all” first party apps
  - List of “all” resources
  - List of some available scopes
- Focus on FOCI applications
  - Use a single refresh token to test all use cases
- Setup Conditional Access Use Cases
  - One user per use case
  - One or more Conditional Access Policies per use case

Use cases

- Baseline use cases
  - No MFA required
  - MFA required and present
- Bypass use cases
  - MFA required for all applications
  - MFA required for all applications and user action „register device“
  - MFA required for all applications and user action „register security information“
  - MFA required for all applications with a single app exclusion
  - Compliant Device required
  - Hybrid Device required
  - Global Secure Access required
  - Network location required
  - Passkey required

Practical approach

- Wrote a PowerShell script to find all FOCI applications using a single refresh token (had not yet talked to Dirk-jan)
  - Result: 52 FOCI Clients (15 new)
- Wrote a second PowerShell script to iterate through
  - All known FOCI applications
  - All known resources
- Run the script for each use case
- Write the results to a local JSON file
- Search for anomalies

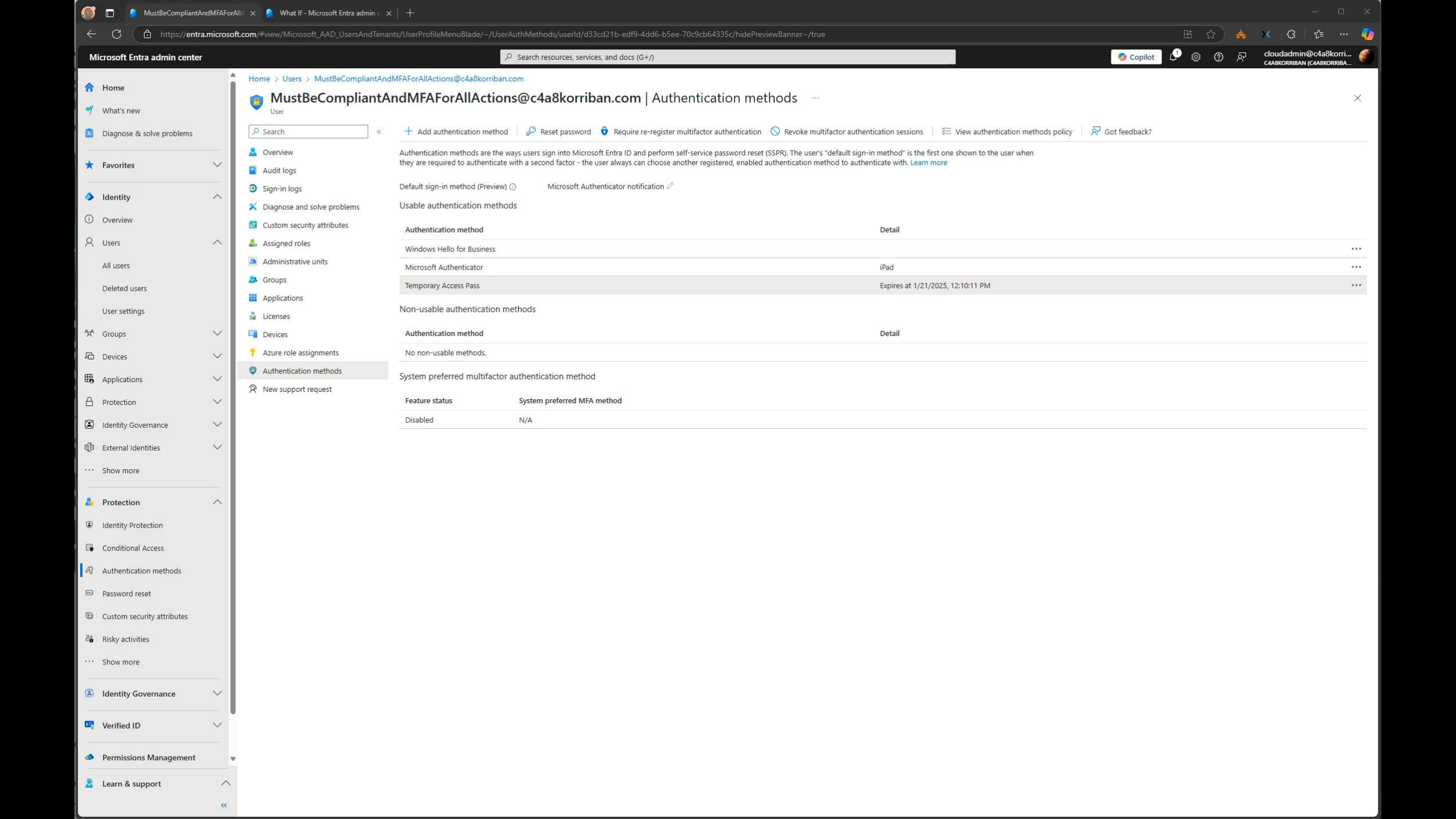

BURNING WITH CURIOSITY
DOWN THE SCOPE RABBIT HOLE

Rabbit hole: Scope based bypass

- Unrelated research into the Microsoft Authenticator App

- Conditional Access enforced MFA and compliant device
- User was able to retrieve a token with a sensitive scope (UserAuthenticationMethod.Read) from a non-compliant device

- A request without this specific scope failed because of the Conditional Access Requirement


MustBeCompliantAndMFAForAll
What If - Microsoft Entra admin
https://entra.microsoft.com/#view/Microsoft_AAD_UsersAndTenants/UserProfileMenuBlade/~/UserAuthMethods/userId/d33cd21b-edf9-4dd6-b5ee-70c9cb64335c/hidePreviewBanner~/true
Microsoft Entra admin center
Search resources, services, and docs (G+/)
Copilot
cloudadmin@c4a8korri...
C4A8KORRIBAN (C4A8KORRIBA...
Home
What's new
Diagnose & solve problems
Favorites
Identity
Overview
Users
All users
Deleted users
User settings
Groups
Devices
Applications
Protection
Identity Governance
External Identities
Show more
Protection
Identity Protection
Conditional Access
Authentication methods
Password reset
Custom security attributes
Risky activities
Show more
Identity Governance
Verified ID
Permissions Management
Learn & support
Home > Users > MustBeCompliantAndMFAForAllActions@c4a8korriban.com
MustBeCompliantAndMFAForAllActions@c4a8korriban.com | Authentication methods
User
Search
Overview
Audit logs
Sign-in logs
Diagnose and solve problems
Custom security attributes
Assigned roles
Administrative units
Groups
Applications
Licenses
Devices
Azure role assignments
Authentication methods
New support request
Add authentication method
Reset password
Require re-register multifactor authentication
Revoke multifactor authentication sessions
View authentication methods policy
Got feedback?
Authentication methods are the ways users sign into Microsoft Entra ID and perform self-service password reset (SSPR). The user's "default sign-in method" is the first one shown to the user when they are required to authenticate with a second factor - the user always can choose another registered, enabled authentication method to authenticate with. Learn more
Default sign-in method (Preview)
Microsoft Authenticator notification
Usable authentication methods
Authentication method
Detail
Windows Hello for Business
Microsoft Authenticator
iPad
Temporary Access Pass
Expires at 1/21/2025, 12:10:11 PM
Non-usable authentication methods
Authentication method
Detail
No non-usable methods.
System preferred multifactor authentication method
Feature status
System preferred MFA method
Disabled
N/A

Scope based bypass - The fix

```
PS C:\Research\TokenTacticsV2> Get-MgContext


ClientId              : 4813382a-8fa7-425e-ab75-3b753aab3abb
TenantId              : e3686c4f-af27-4f22-b9de-062f05b93aac
Scopes                : {email, openid, profile, UserAuthenticationMethod.Read}
AuthType              : UserProvidedAccessToken
TokenCredentialType   : UserProvidedAccessToken
CertificateThumbprint :
CertificateSubjectName :
SendCertificateChain  : False
Account               : MustBeCompliantAndMFAForAllActions@c4a8korriban.com
AppName               : Microsoft Authenticator App
ContextScope          : Process
Certificate           :
PSHostVersion         : 5.1.20348.2849
ManagedIdentityId     :
ClientSecret          :
Environment           : Global



PS C:\Research\TokenTacticsV2> Invoke-MGgraphRequest -Method GET -Uri "https://graph.microsoft.com/beta/me/authentication/methods" | Select-Object -ExpandProperty value
Invoke-MGgraphRequest : GET https://graph.microsoft.com/beta/me/authentication/methods
HTTP/1.1 403 Forbidden
Transfer-Encoding: chunked
Vary: Accept-Encoding
Strict-Transport-Security: max-age=31536000
request-id: 7813be9e-805f-4e65-9f9e-5c77dde2ca8d
client-request-id: 7419c1ab-95ea-497c-881f-d449855df759
```

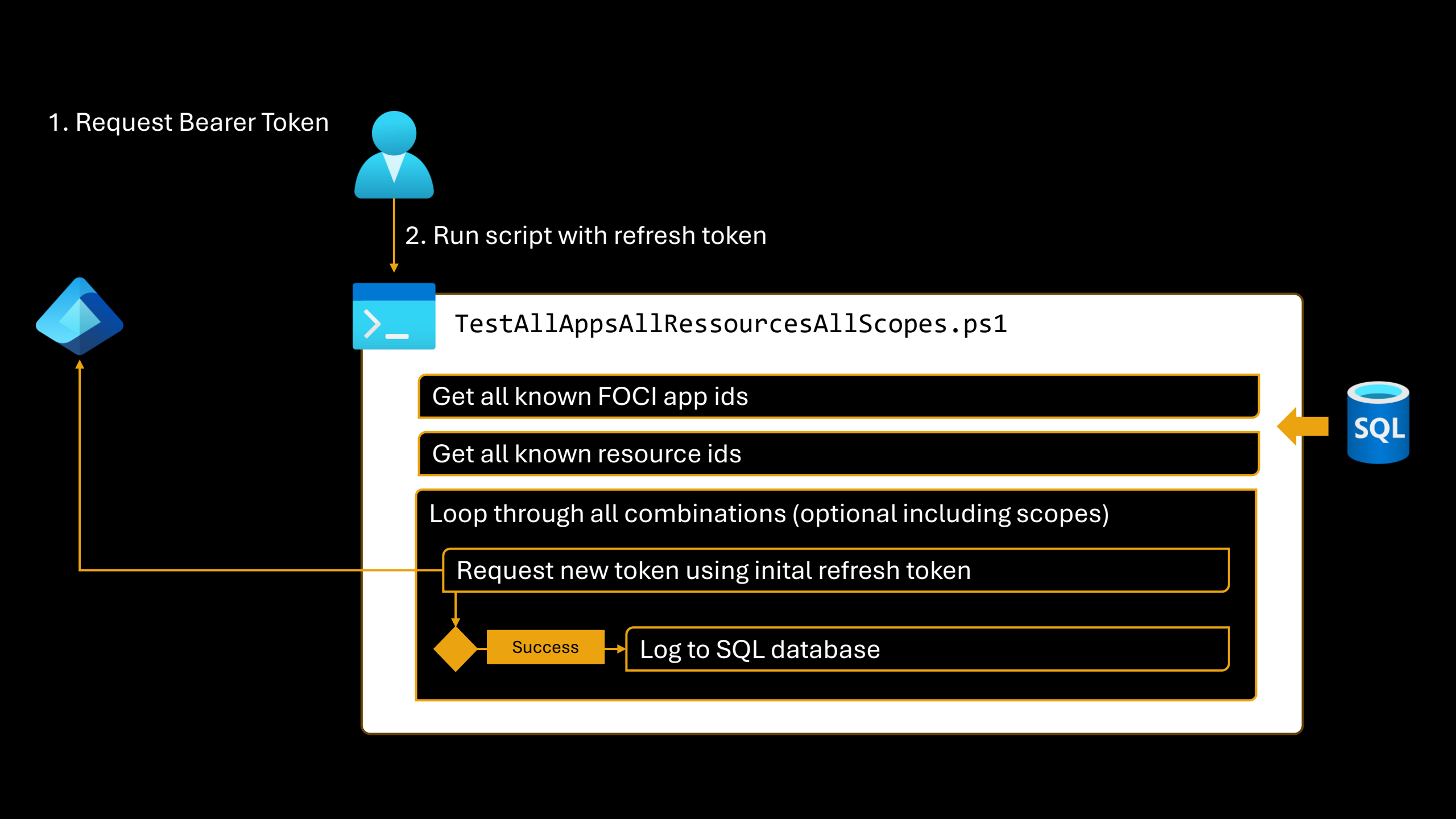
Scope based bypass - Disclosure

- Built-in exclusions in Conditional Access are not only based on the requesting application and the target resource
- The combination of scopes can have an impact on the enforced conditional access policy

- MSRC communication
  - 19.01.2025 - Reported to MSRC
  - 23.01.2025 - Case opened and tracked as VULN-14615
  - 11.04.2025 - Confirmed, fixed, and bounty awarded
- Public disclosure today

Version 2 … ?

- Extended TokenTacticsV2 to support
  - Auth code flow for initial token acquisition
  - V1 version of the Entra ID auth endpoints
  - Optimizations when running in a PowerShell runspace
- Imported additional scopes from Dirk-Jans work
- Switched from JSON to SQLite database for better reporting
- Extended the PowerShell script to also iterate through all possible scopes for each resource
- Added parallelization support for resource scope testing
  - Reduced the runtime for each use case to ~ **30 minutes**


1. Request Bearer Token
2. Run script with refresh token
TestAllAppsAllRessourcesAllScopes.ps1
Get all known FOCI app ids
Get all known resource ids
Loop through all combinations (optional including scopes)
Request new token using inital refresh token
Success
Log to SQL database
SQL

Challenges

- Each CA use case has its own “initial access” bypass
  - Manual token acquisition required
- Version 1 and Version 2 Entra ID endpoint can behave differently
  - Just try one after the other

```
[0] Failed to get initial token for 4813382a-8fa7-425e-ab75-3b753aab3abb / 00000002-0000-0000-c000-000000000000 with scope 'openid' and V1Endpoint - Retrying with V2 Endpoint - 53000
[0] Failed to get initial token for 4813382a-8fa7-425e-ab75-3b753aab3abb / "00000002-0000-0000-c000-000000000000/openid" and V2 Endpoint - 65002
[0] Failed to get initial token for 4813382a-8fa7-425e-ab75-3b753aab3abb / 00000002-0000-0ff1-ce00-000000000000 with scope 'openid' and V1Endpoint - Retrying with V2 Endpoint - 53000
[0] Failed to get initial token for 4813382a-8fa7-425e-ab75-3b753aab3abb / "00000002-0000-0ff1-ce00-000000000000/openid" and V2 Endpoint - 65002
[0] Failed to get initial token for 4813382a-8fa7-425e-ab75-3b753aab3abb / 00000003-0000-0000-c000-000000000000 with scope 'openid' and V1Endpoint - Retrying with V2 Endpoint - 53000
[1] Got token for 4813382a-8fa7-425e-ab75-3b753aab3abb / "00000003-0000-0000-c000-000000000000/openid" and V2 Endpoint
```

V1.0

Error Code	53000
Message	Device is not in required device state: {state}. Conditional Access policy requires a compliant device, and the device is not compliant. The user must enroll their device with an approved MDM provider like Intune.

V2.0

Error Code	65002
Message	Consent between first party application '{applicationId}' and first party resource '{resourceId}' must be configured via preauthorization - applications owned and operated by Microsoft must get approval from the API owner before requesting tokens for that API.

Challenges

- Trying every scope takes a long time
  - Implemented a “brute force scopes on initial success” logic
  - Only brute force if initial token request with **openid** is successful
- Reporting the results is a manual process
- Patience ⏳
  - If you setup a Conditional Access Policy and immediately run your tests, those tests are not worth anything

Results

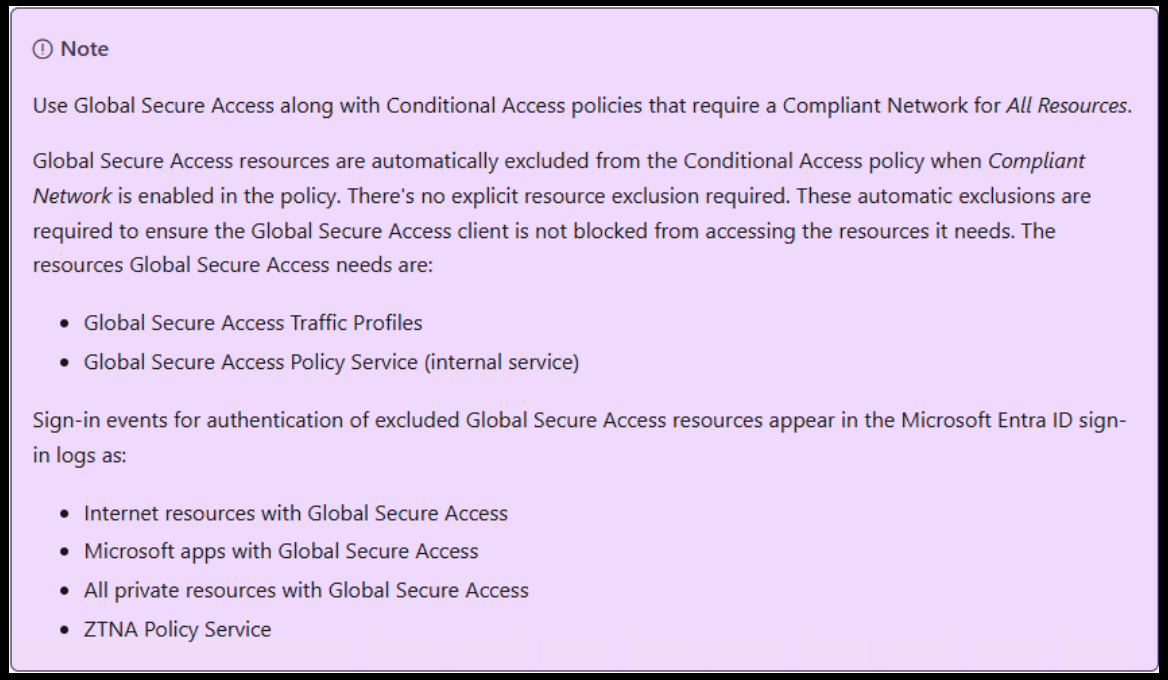
- Device Registration Service resource is not protected by network or compliance requirements
- MSRC: This is expected behavior (VULN-153600)
- Device Registration Service can only be protected by Multi-Factor Authentication / Authentication Strengths

> `Require multifactor authentication` is the only access control available with this user action and all others are disabled. This restriction prevents conflicts with access controls that are either dependent on Microsoft Entra device registration or not applicable to Microsoft Entra device registration.

Reference: https://learn.microsoft.com/en-us/entra/identity/conditional-access/policy-all-users-device-registration

Results

- “ZTNA Network Access Traffic Profile” is not protected by Compliant Network control
- Exclusion was already documented by Microsoft



> **Note**
>
> Use Global Secure Access along with Conditional Access policies that require a Compliant Network for *All Resources*.
>
> Global Secure Access resources are automatically excluded from the Conditional Access policy when *Compliant Network* is enabled in the policy. There's no explicit resource exclusion required. These automatic exclusions are required to ensure the Global Secure Access client is not blocked from accessing the resources it needs. The resources Global Secure Access needs are:
>
> - Global Secure Access Traffic Profiles
> - Global Secure Access Policy Service (internal service)
>
> Sign-in events for authentication of excluded Global Secure Access resources appear in the Microsoft Entra ID sign-in logs as:
>
> - Internet resources with Global Secure Access
> - Microsoft apps with Global Secure Access
> - All private resources with Global Secure Access
> - ZTNA Policy Service

Reference: https://learn.microsoft.com/en-us/entra/global-secure-access/how-to-compliant-network

Results

- Some resources are completely excluded from any Conditional Access control
  - 26a4ae64-5862-427f-a9b0-044e62572a4f - Microsoft Intune Checkin
  - 04436913-cf0d-4d2a-9cc6-2ffe7f1d3d1c - Windows Notification Service
  - 0a5f63c0-b750-4f38-a71c-4fc0d58b89e2 - Microsoft Mobile Application Management
  - 1f5530b3-261a-47a9-b357-ded261e17918 - Azure Multi-Factor Auth Connector
  - c2ada927-a9e2-4564-aae2-70775a2fa0af - OCaaS Client Interaction Service
  - ff9ebd75-fe62-434a-a6ce-b3f0a8592eaf - Authenticator App
- “Working as expected” according to MSRC
- Not the case when “**Per User MFA**” is enforced or **V2** endpoint is used


SOUTH PARK
1331
DO YOU HAVE MORE?


WAITING FOR MSRC

Results

- [redacted] is not protected by the „[redacted]“ control when using the app [redacted]
- Granted scopes:
  - [redacted]
  - [redacted]

Currently still under investigation by MSRC

Plans for vNext

- Run fully automated including initial token acquisition
- Use output from Dirk-Jans research to only check for pre-consented resources for each application
- Store results in non-local database
- Add more use case
- Automated reporting

Detection approaches

- CA Policy state in SignIn logs is a very strong indicator

```
UnifiedSignInLogs
| where TimeGenerated > ago(90d)
| where ResultType == 0
| where ConditionalAccessPolicies has "failure"
| where ConditionalAccessStatus == "success"
| mv-expand ConditionalAccessPolicies
| where ConditionalAccessPolicies has "failure"
```

UserPrincipalName	AppDisplayName	AppId	ResourceDisplayName	ResourceIdentity	ConditionalAccessPolicyDis...	ConditionalAccessPolicyRes...	enforcedGrantControls
mustbecompliantandmfaforallactions@c4a8korriban.com	Microsoft Authenticator App	4813382a-8fa7-425e-ab75-3b7...	Microsoft Graph	00000003-0000-0000-c000-000000000000	Compliant Device required	failure	["RequireCompliantDevice"]
compliantuser@c4a8korriban.com	Microsoft Intune Company Portal	9ba1a5c7-f17a-4de9-a1f1-617...	Microsoft Graph	00000003-0000-0000-c000-000000000000	Compliant Device required	failure	["RequireCompliantDevice"]
compliantuser@c4a8korriban.com	Microsoft Intune Company Portal	9ba1a5c7-f17a-4de9-a1f1-617...	Windows Azure Active Directory	00000002-0000-0000-c000-000000000000	Compliant Device required	failure	["RequireCompliantDevice"]
compliantuser@c4a8korriban.com	Microsoft Intune Company Portal	9ba1a5c7-f17a-4de9-a1f1-617...	Microsoft Intune	0000000a-0000-0000-c000-000000000000	Compliant Device required	failure	["RequireCompliantDevice"]
compliantuser@c4a8korriban.com	Microsoft Intune Company Portal	9ba1a5c7-f17a-4de9-a1f1-617...	Microsoft Intune Enrollment	d4ebce55-015a-49b5-a083-c84d1797ae8c	Compliant Device required	failure	["RequireCompliantDevice"]
compliantuser@c4a8korriban.com	Microsoft Intune Company Portal	9ba1a5c7-f17a-4de9-a1f1-617...	Microsoft Intune IW Service	b8066b99-6e67-41be-abfa-75db1a2c8809	Compliant Device required	failure	["RequireCompliantDevice"]
compliantuser@c4a8korriban.com	Microsoft Authenticator App	4813382a-8fa7-425e-ab75-3b7...	Microsoft Graph	00000003-0000-0000-c000-000000000000	Compliant Device required	failure	["RequireCompliantDevice"]

Reference: https://cloudbrothers.info/en/unified-sign-logs-advanced-hunting/

Entra ID First Party Apps & Scope Browser

Browse and explore first-party applications including their pre-consented permissions in Microsoft Entra ID

Filters

Search: Search apps...

FOCI Status: FOCI and Non-FOCI

Client Type: All Client Types

Nested app authentication: All Apps

Conditional Access Bypass: Bypass available

App IDs: Select App IDs

Resources: Select Resources

Scopes: Select Scopes

Microsoft Intune Company Portal

FOCI | Public Client | CA Bypass (1)

App ID: 9ba1a5c7-f17a-4de9-a1f1-6178c8d51223

8 Resources | 16 Scopes | 24 URIs

RESOURCES:

ADIbizaUX 74658136-14ec-4630-ad9b-26e160ff0fc6	1 scope
Azure Key Vault cfa8b339-82a2-471a-a3c9-0fc0be7a4093	1 scope
Azure Resource Manager 797f4846-ba00-4fd7-ba43-dac1f8f63013	1 scope

This project is inspired and built upon the work of various contributors in the community. Here are some key resources and references used in this project or that are helpful for further exploration:

- Microsoft Entra First Party App Scopes JSON by Dirk-jan Mollema
- Classification of Roles and Permissions by Thomas Naunheim
- Graph Pre-Consent Explorer by zh54321
- Token Tactics V2 by Fabian Bader

Documentation references

- Microsoft Learn: Enterprise access model
- Microsoft Learn: Nested app authentication

Built with ❤️ by Fabian Bader and Dirk-jan Mollema

www.entrascopes.com

Conclusion

- Conditional Access is complicated
- There are hardcoded bypasses and only some are documented
- Wherever you can, use all resources and no exclusions.
  - But most likely you need to use exclusions

Other research

- https://github.com/secureworks/family-of-client-ids-research
- https://github.com/merill/microsoft-info
- https://github.com/zh54321/GraphPreConsentExplorer
- https://github.com/rvrsh3ll/TokenTactics
- https://github.com/f-bader/TokenTacticsV2

Finding Entra ID CA bypasses the structured way

TROOPERS