Family Matters: Abusing Family Refresh Tokens to Gain Unauthorised Access to Microsoft Cloud Services Exploratory Study of Azure Active Directory Family of Client IDs

Ryan Cobb[1][a], Anthony Larcher-Gore[1][b] and Nestori Syynimaa[1,2][c]
[1]*Secureworks, Counter Threat Unit, U.S.A.*
[2]*Faculty of Information Technology, University of Jyväskylä, Jyväskylä, Finland*

Keywords: Azure Active Directory, Azure AD, OAuth, OIDC, Authentication, Authorisation, Security, FRT, Privilege Escalation.

Abstract: Azure Active Directory (Azure AD) is an identity and access management service used by Microsoft 365 and Azure services and thousands of third-party service providers. Azure AD uses OIDC and OAuth protocols for authentication and authorisation, respectively. OAuth authorisation involves four parties: client, resource owner, resource server, and authorisation server. The resource owner can access the resource server using the specific client after the authorisation server has authorised the access. The authorisation is presented using a cryptographically signed Access Token, which includes the identity of the resource owner, client, and resource. During the authorisation, Azure AD assigns Access and Id Tokens that are valid for one hour and a Refresh Token that is valid for 90 days. Refresh Tokens are used for requesting new Access and Id token after their expiration. By OAuth 2.0 standard, Refresh Tokens should only be able to be used to request Access Tokens for the same resource owner, client, and resource. In this paper, we will present findings of a study related to undocumented feature used by Azure AD, the Family of Client ID (FOCI). After studying 600 first-party clients, we found 16 FOCI clients which supports a special type of Refresh Tokens, called Family Refresh Tokens (FRTs). These FRTs can be used to obtain Access Tokens for any FOCI client. This non-standard behaviour makes FRTs primary targets for a token theft and privilege escalation attacks.

1 INTRODUCTION

1.1 Azure Active Directory

Azure Active Directory (Azure AD) is an identity and access management service (IAM) provided by Microsoft (Microsoft, 2021f). It is used as IAM by Microsoft's own services, such as Microsoft 365 and Azure, and thousands of third-party service providers (Microsoft, 2022a). At least 88 per cent of fortune 500 companies and 95 per cent of top 2000 universities are using Azure AD (Syynimaa, 2022). This makes Azure AD one of the most critical IAM services globally.

1.2 OAuth 2.0 and OIDC

Azure AD uses OpenID Connect (OIDC) and OAuth protocols for authentication and authorisation, respectively. OAuth 2.0 authorisation framework allows third-party applications to access HTTP based services either directly or on-behalf-of users (IETF, 2012). OIDC is an identity layer on top of OAuth 2.0 protocol (OpenID Foundation, 2022). Both protocols have four parties: *OAuth Client* (OC), *Resource Owner* (RO), *Resource Server* (RS), and *Authorisation Server* (AS). Moreover, both protocols use *bearer tokens* to grant access to a *bearer*, which typically refers to the RO. A simple authorisation flow, where RO uses OC to request access from AS to RS is illustrated in Figure 1.

[a] https://orcid.org/0000-0003-3762-9593
[b] https://orcid.org/0000-0002-7188-4160
[c] https://orcid.org/0000-0002-6848-094X

Cobb, R., Larcher-Gore, A. and Syynimaa, N.
Family Matters: Abusing Family Refresh Tokens to Gain Unauthorised Access to Microsoft Cloud Services Exploratory Study of Azure Active Directory Family of Client IDs.
DOI: 10.5220/0011061200003179
In *Proceedings of the 24th International Conference on Enterprise Information Systems (ICEIS 2022) - Volume 2*, pages 62-69
ISBN: 978-989-758-569-2; ISSN: 2184-4992

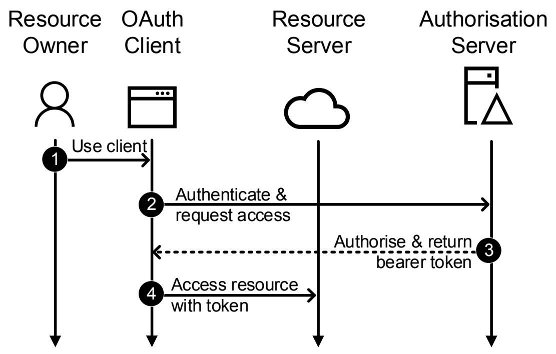

Resource Owner
OAuth Client
Resource Server
Authorisation Server
1
Use client
2
Authenticate & request access
Authorise & return bearer token
3
4
Access resource with token

Figure 1: OAuth 2.0 authorisation flow.

Before authorisation can be requested, the used OC needs to have a *consent*. This process and the granularity of the consent depends on the used AS and RS. Microsoft OAs are generally referred as *first-party clients*. Some of these, such as Exchange Online, are pre-authorised in all Azure AD environments.

1.3 Tokens

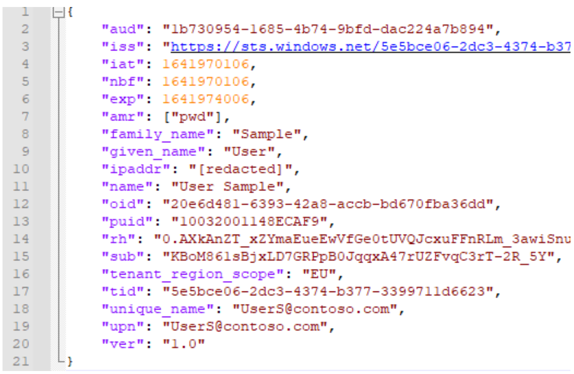
A standard representation of a bearer token is JSON Web Token, or JWT (IETF, 2015). There are two flavours of JWT: JSON Web Signature (JWS) and JSON Web Encrypt (JWE). The former is used to represent bearer tokens and is also generally used as a synonym for JWT. JWS consists of three Base64 URL encoded parts: Javascript Object Signing and Encrypt (JOSE) header, JWS payload, and JWS signature. The JOSE header contains the information about the key used to sign the JWS, the payload contains a set of JWS claims, and the signature contains the cryptographic signature of the header and the payload.

Azure AD uses three types of tokens: *access tokens*, *Id tokens*, and *refresh tokens* (Microsoft, 2021c). Different token types and their properties are listed in Table 1.

Table 1: Azure AD token types.

Type	Standard	Lifetime
Id Token	OIDC	1 hour
Access Token	OAuth2	1 hour
Refresh Token	OAuth2	90 days

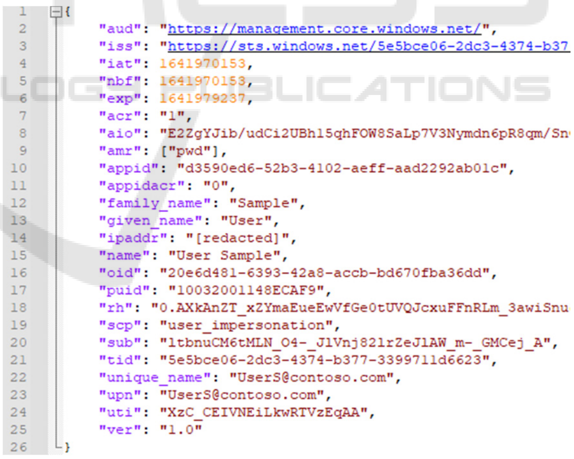
The *Id token* contains the user's (*i.e.*, the bearers) identity information. A sample id token can be seen in Figure 2. As we can see, it contains different information about the user, like *unique_name* on line 18. It also includes information about the issuer (*i.e.*, Authorisation Server) on line 2.



```
 1  {
 2      "aud": "1b730954-1685-4b74-9bfd-dac224a7b894",
 3      "iss": "https://sts.windows.net/5e5bce06-2dc3-4374-b37
 4      "iat": 1641970106,
 5      "nbf": 1641970106,
 6      "exp": 1641974006,
 7      "amr": ["pwd"],
 8      "family_name": "Sample",
 9      "given_name": "User",
10      "ipaddr": "[redacted]",
11      "name": "User Sample",
12      "oid": "20e6d481-6393-42a8-accb-bd670fba36dd",
13      "puid": "10032001148ECAF9",
14      "rh": "0.AXkAnZT_xZYmaEueEwVfGe0tUVQJcxuFFnRLm_3awiSnu
15      "sub": "KBoM861sBjxLD7GRPpB0JqqxA47rUZFvqC3rT-2R_5Y",
16      "tenant_region_scope": "EU",
17      "tid": "5e5bce06-2dc3-4374-b377-339971ld6623",
18      "unique_name": "UserS@contoso.com",
19      "upn": "UserS@contoso.com",
20      "ver": "1.0"
21  }
```

Figure 2: A sample id token.

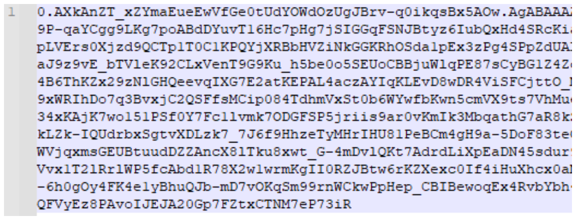
The *access token* contains the same information as the identity token, but also information about the resource the user has been authorised to access. A sample access token can be seen in Figure 3. As we can see, the audience (*i.e.*, Resource Server) is included on line 2. Moreover, the *scope* is included on line 19. The scope is used in Azure AD to further limit access to the resource server by listing different Application Programming Interface (API) scopes. The *user_impersonation* and *.default* scopes mean that the user can access all APIs on the RS but can perform only actions they have permissions on the RS in question.



```
 1  {
 2      "aud": "https://management.core.windows.net/",
 3      "iss": "https://sts.windows.net/5e5bce06-2dc3-4374-b37
 4      "iat": 1641970153,
 5      "nbf": 1641970153,
 6      "exp": 1641979237,
 7      "acr": "1",
 8      "aio": "E2ZgYJib/udCi2UBh15qhFOW8SaLp7V3Nymdn6pR8qm/Sn
 9      "amr": ["pwd"],
10      "appid": "d3590ed6-52b3-4102-aeff-aad2292ab01c",
11      "appidacr": "0",
12      "family_name": "Sample",
13      "given_name": "User",
14      "ipaddr": "[redacted]",
15      "name": "User Sample",
16      "oid": "20e6d481-6393-42a8-accb-bd670fba36dd",
17      "puid": "10032001148ECAF9",
18      "rh": "0.AXkAnZT_xZYmaEueEwVfGe0tUVQJcxuFFnRLm_3awiSnu
19      "scp": "user_impersonation",
20      "sub": "1tbnuCM6tMLN_O4-_J1Vnj821rZeJ1AW_m-_GMCej_A",
21      "tid": "5e5bce06-2dc3-4374-b377-339971ld6623",
22      "unique_name": "UserS@contoso.com",
23      "upn": "UserS@contoso.com",
24      "uti": "XzC_CEIVNEiLkwRTVzEqAA",
25      "ver": "1.0"
26  }
```

Figure 3: A sample access token.

Refresh token is used to obtain a new set of tokens when access or Id tokens expires (IETF, 2012). In Azure AD, refresh token is an opaque binary large object (blob) encrypted with a key known only by Microsoft (Microsoft, 2021a), delivered as JWE. As such, its actual content is unknown. A sample refresh token can be seen in Figure 4.



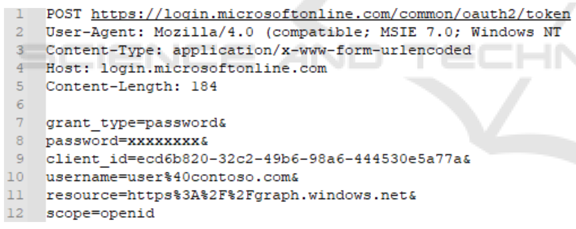
```
0.AXkAnZT_xZYmaEueEwVfGe0tUdYOWdOzUgJBrv-q0ikqsBx5AOw.AgABAAA
9P-qaYCgg9LKg7poABdDYuvTl6Hc7pHg7jSIGGqFSNJBtyz6IubQxHd4SRcKi
pLVErs0Xjzd9QCTplT0ClKPQYjXRBbHVZiNkGGKRhOSdalpEx3zPg4SPpZdUA
aJ9z9vE_bTVleK92CLxVenT9G9Ku_h5be0o5SEUoCBBjuWlqPE87sCyBGlZ4Z
4B6ThKZx29zNlGHQeevqIXG7E2atKEPAL4aczAYIqKLEvD8wDR4ViSFCjttO_
9xWRIhDo7q3BvxjC2QSFfsMCip084TdhmVxSt0b6WYwfbKwn5cmVX9ts7VhMu
34xKAjK7wol5lPSf0Y7Fcllvmk7ODGFSP5jriis9ar0vKmIk3MbqathG7aR8k
kLZk-IQUdrbxSgtvXDLzk7_7J6f9HhzeTyMHrIHU8lPeBCm4gH9a-5DoF83te
WVjqxmsGEUBtuudDZZAncX8lTku8xwt_G-4mDvlQKt7AdrdLiXpEaDN45sdur
VvxlT2lRrlWP5fcAbdlR78X2wlwrmKgII0RZJBtw6rKZXexc0If4iHuXhcx0a
-6h0gOy4FK4elyBhuQJb-mD7vOKqSm99rnWCkwPpHep_CBIBewoqEx4RvbYbh
QFVyEz8PAvoIJEJA20Gp7FZtxCTNM7eP73iR
```

Figure 4: A sample refresh token.

1.4 OAuth 2.0 Authentication Flows

OAuth 2.0 standard (IETF, 2012) defines several different flows for acquiring authorisation: *Authorization Code Grant*, *Implicit Grant*, *Resource Owner Password Credentials Grant*, and *Client Credentials Grant*.

The Authorization Code Grant is commonly used to obtain the initial set of tokens (id token, access token, and refresh token). The Authorization Code Grant allows the RO to use more secure authentication mechanisms, such as multi-factor authentication (MFA), and without the need to share their credentials with the OC. In contrast, the Resource Owner Password Credential (ROPC) flow allows the OC to authenticate using the RO's credentials. A sample ROPC request made to Azure AD's */token* endpoint can be seen in Figure 5 (line breaks added for readability).

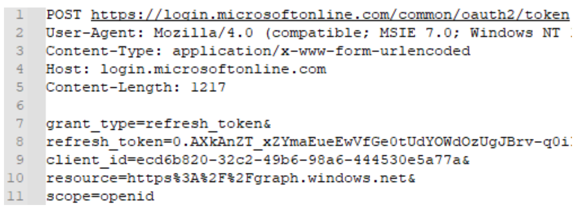


```
POST https://login.microsoftonline.com/common/oauth2/token
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT
Content-Type: application/x-www-form-urlencoded
Host: login.microsoftonline.com
Content-Length: 184

grant_type=password&
password=xxxxxxxx&
client_id=ecd6b820-32c2-49b6-98a6-444530e5a77a&
username=user%40contoso.com&
resource=https%3A%2F%2Fgraph.windows.net&
scope=openid
```

Figure 5: Sample ROPC authorisation request.

After the access and id tokens have expired, a new set of access tokens can be obtained using the refresh token. A sample request using refresh token can be seen in Figure 6 (line breaks added for readability).



```
POST https://login.microsoftonline.com/common/oauth2/token
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT
Content-Type: application/x-www-form-urlencoded
Host: login.microsoftonline.com
Content-Length: 1217

grant_type=refresh_token&
refresh_token=0.AXkAnZT_xZYmaEueEwVfGe0tUdYOWdOzUgJBrv-q0i
client_id=ecd6b820-32c2-49b6-98a6-444530e5a77a&
resource=https%3A%2F%2Fgraph.windows.net&
scope=openid
```

Figure 6: Sample refresh token authorisation request.

The OAuth 2.0 standard (IETF, 2012, section 6) and *OAuth 2.0 Threat Model and Security Considerations* (IETF, 2013, section 5.2.2.2.) specifies that refresh tokens should be bound to the client id it was issued. Moreover, OAuth 2.0 standard specifies that refresh tokens can be used to obtain access tokens "..with identical or narrower scope.." (IETF, 2012, section 1.5.).

The OAuth implementation in Azure AD deviates from the standard regarding refresh tokens. Azure AD does not enforce the requirement that newly issued access tokens must have the same or narrower scope as the original authorisation. Microsoft (2021a) documentation states that:

"Refresh tokens are bound to a combination of user and client, but aren't tied to a resource or tenant. As such, a client can use a refresh token to acquire access tokens across any combination of resource and tenant where it has permission to do so."

A recent research article (Syynimaa, 2020) revealed, however, that the refresh tokens issued to certain Microsoft first-party clients (later to be known as FOCI clients) are redeemable for new access tokens authorised to a different client. This behaviour, or feature, was unexpected given the Microsoft documentation and OAuth specifications.

1.5 Research Questions

Our research sought to answer the following research questions:

1. Which first-party clients are supported by this feature?
2. What is the purpose of granting access tokens for other first-party clients using refresh tokens?

1.6 Structure of the Paper

The rest of the paper is structured as follows. In Section 2, we describe the research methodology used in the study. In Section 3, the results of the study are presented. Finally, in Section 4, implications of the findings and a conclusion are presented.

2 METHODOLOGY

The research had two distinct research lines, one for each research question.

To answer the first research question, a list of 450 first-party clients (Seb8iaan, 2020) was acquired and supplemented with the client ids (~150) harvested from Azure AD sign-in logs. After that, the initial set of tokens for each first-party client using various scopes was obtained, and a new set of tokens was tried to be received with the refresh token. The details of the experiment are as follows:

1. A new Azure AD tenant with two users (one admin, one regular user) was provisioned. Both users were assigned Microsoft E5 license.
2. The initial set of tokens were obtained for each client using a *Jupyter Notebook* and the *Microsoft Authentication Library (MSAL) for Python*.
3. ROPC flow was used to authorise each client for *.default*, *openid*, *profile*, and *offline_access* scopes on 40 different resources including: *Microsoft Graph*, Microsoft 365 APIs (*Exchange Online*, *Sharepoint*, and *Teams)*, Azure Resource Manager APIs (*Azure Storage*, *Vault*, *Database*), and the client itself. The *offline_access* scope instructs AS to issue a refresh token in addition to Id and access tokens.
4. The results of each authorisation attempt was logged, including the issued tokens in JSON format.
5. The refresh tokens obtained in previous steps were used to request new authorisation via *refresh grant* flow for the same combination of clients, scopes, and resources.
6. The issued access tokens were compared to the initial set of tokens for each successful refresh attempt.

To answer the second research question, the public Microsoft documentation and Github were studied, and the various Google searches were conducted. Microsoft was also contacted during the research, including the findings and comments regarding the first research question.

3 RESULTS

A subset of the research is available at Github (see Cobb & Gore, 2022) with interactive demo in form of a Jupyter Notebook. This allows other researchers to reproduce and expand upon findings described in the following sections.

3.1 Family of Client IDs (FOCI) and Family Refresh Tokens (FRTs)

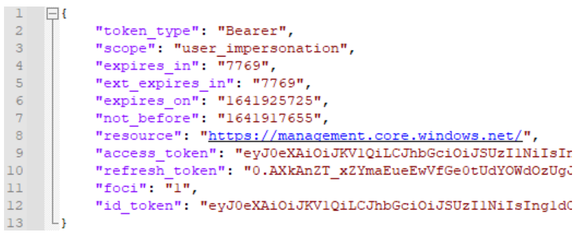
After running an experiment with ~600 first-party clients, only 13 clients were found to be supported by the feature. Later, three additional clients were found by scraping client ids from various Github sites. All 16 clients are listed in Table 2.

Table 2: List of Azure AD FOCI clients.

Client	Client Id
Office 365 Management (mobile app)	00b41c95-dab0-4487-9791-b9d2c32c80f2
Azure CLI	04b07795-8ddb-461a-bbee-02f9e1bf7b46
AZ PowerShell Module	1950a258-227b-4e31-a9cf-717495945fc2
Teams	1fec8e78-bce4-4aaf-ab1b-5451cc387264
Windows Search	26a7ee05-5602-4d76-a7ba-eae8b7b67941
MS MAM Service API	27922004-5251-4030-b22d-91ecd9a37ea4
Microsoft Bing Search for Microsoft Edge	2d7f3606-b07d-41d1-b9d2-0d0c9296a6e8
Authenticator App	4813382a-8fa7-425e-ab75-3b753aab3abb
Microsoft Stream Mobile Native	844cca35-0656-46ce-b636-13f48b0eecbd
Microsoft Teams – Device Admin Agent	87749df4-7ccf-48f8-aa87-704bad0e0e16
OneDrive	ab9b8c07-8f02-4f72-87fa-80105867a763
Microsoft Bing Search	cf36b471-5b44-428c-9ce7-313bf84528de
Office Desktop client	d3590ed6-52b3-4102-aeff-aad2292ab01c
Visual Studio	872cd9fa-d31f-45e0-9eab-6e460a02d1f1
OneDrive iOS App	af124e86-4e96-495a-b70a-90f90ab96707
Edge	ecd6b820-32c2-49b6-98a6-444530e5a77a

When obtaining the initial set of tokens for the aforementioned clients, the received JSON responses had a *foci* attribute with a value set to “1” (see line 11 in Figure 7). These clients are later referred as *FOCI clients*. The *foci* attribute did not exist for other clients. The list of known FOCI clients is maintained in the Github repository associated with this research (see Cobb & Gore, 2022).

All FOCI clients were so-called *public clients* (Microsoft, 2021d) which can only access APIs on behalf of the user. FOCI clients could exchange their refresh token for new tokens for any other FOCI client. The scopes in the newly issued access tokens were based on the new client and its *.default* scopes, *i.e.*, the client and scopes from the original authorisation did not matter.



```
{
    "token_type": "Bearer",
    "scope": "user_impersonation",
    "expires_in": "7769",
    "ext_expires_in": "7769",
    "expires_on": "1641925725",
    "not_before": "1641917655",
    "resource": "https://management.core.windows.net/",
    "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsI
    "refresh_token": "0.AXkAnZT_xZYmaEueEwVfGe0tUdYOWdOzUg
    "foci": "1",
    "id_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIngld
}
```

Figure 7: Authorisation response for FOCI client.

Searching for "foci" from internet with Google resulted to only one hit from Microsoft documentation (Microsoft, 2021e), which revealed that FOCI refers to Family of Client IDs. No further explanation was given what is or what is the purpose of FOCI. However, some error messages returned during our experiment led to an Github issue from 2016, which explained the purpose (Pangrle, 2016):

"Future server work will allow client IDs to be grouped on the server side in a way where a RT for one client ID can be redeemed for a AT and RT for a different client ID as long as they're in the same group. This will move us closer to being able to provide SSO-like functionality between apps without requiring the broker (or workplace join)."

Microsoft calls refresh tokens of FOCI clients Family Refresh Tokens (FRTs), which "…can be used for all clients part of the family" (Thompson, 2020, line 2281). Currently, only one FOCI client family, "1", is used (Microsoft, 2021b, line 1171). This statement is supported by our empirical findings.

3.2 Communication with Microsoft

The findings and recommendations were shared with Microsoft Security Response Center (MSRC) on November 23rd 2021 (Cobb, 2021).

In their response on December 14th 2021, MSRC (2021) confirms that FOCI is a feature of Azure AD service. Microsoft "..built this feature to match the existing practice with mobile platforms of sharing authentication artifacts, like refresh tokens, along the publisher (rather than application) boundary". This implies that FOCI feature is meant for mobile devices, but as we have observed, it works on all platforms. Interestingly, Azure AD allready provides single sign-on (SSO) functionality for Azure AD joined and registered (*a.k.a.* workplace joined) devices with Primary Refresh Token (PRT) (Microsoft, 2022b). For these devices, FOCI seems to be a redundant feature.

Moreover, Microsoft did not consider FRTs being capable for privilege escalation, as the "Family refresh tokens can only provide the level of access that the user has".

4 DISCUSSION

4.1 Implications

4.1.1 Privilege Escalation

The key finding of the study is that Family Refresh Tokens (FRTs) can be used to acquire tokens for any FOCI client. The acquired access tokens will have a defined *resource* and *scope* (see lines 2 and 19 in Figure 3, respectively), depending on the used client. The scopes of access tokens of each FOCI client were documented in a scope lookup table (see Figure 8). All returned scopes, resources, and FOCI clients used were included. This helps choosing which FOCI client to use to obtain tokens. For example, if one needs to get *Calendars.ReadWrite* scope, one could use any client listed in Table 3.

Table 3: FOCI clients with *Calendars.ReadWrite* scope.

Client	Client Id
Teams	1fec8e78-bce4-4aaf-ab1b-5451cc387264
Office desktop client	d3590ed6-52b3-4102-aeff-aad2292ab01c
MAM Service API	27922004-5251-4030-b22d-91ecd9a37ea4

NIST (2016) defines *privilege escalation* as the "..exploitation of a bug or flaw that allows for a higher privilege level than what would normally be permitted" which includes the access the *application* (client) has. In the context of an OAuth client, the level of access afforded by FRTs exceeds the authorisation given to any given client. Furthermore, the *OAuth 2.0 Security Best Current Practice* explicitly states (IETF, 2021, p. 39):

"If refresh tokens are issued, those refresh tokens MUST be bound to the scope and resource servers as consented by the resource owner. This is to prevent privilege escalation by the legitimate client and reduce the impact of refresh token leakage."

From the client and attacker perspectives, the level of access provided by FRTs greatly surpasses what the RO authorised. Therefore, we consider abusing FRTs as a form of privilege escalation attack.

4.1.2 Token Theft

Refresh tokens can be considered long-term credentials and, thus, are subject to theft (IETF, 2013). The level of access afforded to an attacker from a stolen refresh token is determined by the resources and scopes authorised to the access tokens obtained using the stolen refresh token. FRTs can be used to acquire access tokens for any FOCI client, resource, and scope, and thus, are much more powerful than ordinary refresh tokens.

Some commonly used attack paths the malicious actors can use to obtain refresh tokens are (IETF, 2013):

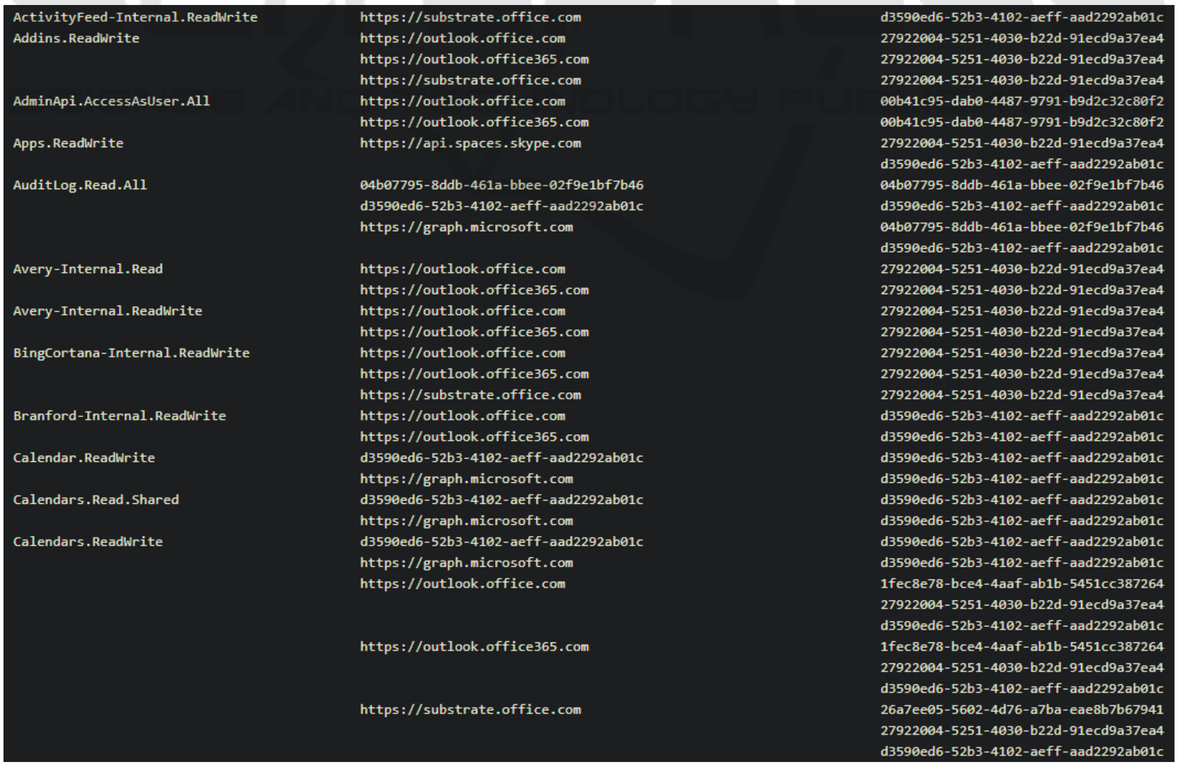
- Steal a previously and legitimately issued refresh token.
- Obtain refresh token through malicious authorisation.

These attack paths also apply to FRTs. It is possible to steal FRTs that were previously issued to FOCI client. For example, if the attacker compromises the cache where the tokens are stored (such as the Windows Web Account Manager), eavesdrops on network traffic during a grant flow, or finds them serialised on disk in files (like *~/.Azure/accessTokens.json*). We focused our attention, however, on how an attacker could obtain FRTs by maliciously authorising a FOCI client.

Device Code Phishing is an attack method where a malicious actor can lure the victim to authorise access to a resource using *device authorisation grant flow* (see IETF, 2013). If the attacker is using a FOCI client, the user consent is not required, and the attacker can use whatever FOCI client is most likely to socially engineer the victim. After successful authorisation, the attacker can redeem the returned FRT for a new access token for a different FOCI client for the desired scopes.

4.1.3 Single Sign-on

Another likely attack path to family refresh tokens is to abuse SSO on Azure AD joined devices. The OAuth 2.0 threat model describes a scenario where an attacker might obtain a refresh token through exploiting some mechanism that automatically authorises client applications without knowledge or intent from the resource owner (IETF, 2013, section 4.4.1.10.). This is trivially possible on Azure AD joined devices. Processes that execute in the context of a logged-in Azure AD user on an Azure AD-joined Windows device can request a pre-signed cookie from a COM service (Christensen, 2020). This cookie



```
ActivityFeed-Internal.ReadWrite      https://substrate.office.com                    d3590ed6-52b3-4102-aeff-aad2292ab01c
Addins.ReadWrite                     https://outlook.office.com                      27922004-5251-4030-b22d-91ecd9a37ea4
                                     https://outlook.office365.com                   27922004-5251-4030-b22d-91ecd9a37ea4
                                     https://substrate.office.com                    27922004-5251-4030-b22d-91ecd9a37ea4
AdminApi.AccessAsUser.All            https://outlook.office.com                      00b41c95-dab0-4487-9791-b9d2c32c80f2
                                     https://outlook.office365.com                   00b41c95-dab0-4487-9791-b9d2c32c80f2
Apps.ReadWrite                       https://api.spaces.skype.com                    27922004-5251-4030-b22d-91ecd9a37ea4
                                                                                     d3590ed6-52b3-4102-aeff-aad2292ab01c
AuditLog.Read.All                    04b07795-8ddb-461a-bbee-02f9e1bf7b46            04b07795-8ddb-461a-bbee-02f9e1bf7b46
                                     d3590ed6-52b3-4102-aeff-aad2292ab01c            d3590ed6-52b3-4102-aeff-aad2292ab01c
                                     https://graph.microsoft.com                     04b07795-8ddb-461a-bbee-02f9e1bf7b46
                                                                                     d3590ed6-52b3-4102-aeff-aad2292ab01c
Avery-Internal.Read                  https://outlook.office.com                      27922004-5251-4030-b22d-91ecd9a37ea4
                                     https://outlook.office365.com                   27922004-5251-4030-b22d-91ecd9a37ea4
Avery-Internal.ReadWrite             https://outlook.office.com                      27922004-5251-4030-b22d-91ecd9a37ea4
                                     https://outlook.office365.com                   27922004-5251-4030-b22d-91ecd9a37ea4
BingCortana-Internal.ReadWrite       https://outlook.office.com                      27922004-5251-4030-b22d-91ecd9a37ea4
                                     https://outlook.office365.com                   27922004-5251-4030-b22d-91ecd9a37ea4
                                     https://substrate.office.com                    27922004-5251-4030-b22d-91ecd9a37ea4
Branford-Internal.ReadWrite          https://outlook.office.com                      d3590ed6-52b3-4102-aeff-aad2292ab01c
                                     https://outlook.office365.com                   d3590ed6-52b3-4102-aeff-aad2292ab01c
Calendar.ReadWrite                   d3590ed6-52b3-4102-aeff-aad2292ab01c            d3590ed6-52b3-4102-aeff-aad2292ab01c
                                     https://graph.microsoft.com                     d3590ed6-52b3-4102-aeff-aad2292ab01c
Calendars.Read.Shared                d3590ed6-52b3-4102-aeff-aad2292ab01c            d3590ed6-52b3-4102-aeff-aad2292ab01c
                                     https://graph.microsoft.com                     d3590ed6-52b3-4102-aeff-aad2292ab01c
Calendars.ReadWrite                  d3590ed6-52b3-4102-aeff-aad2292ab01c            d3590ed6-52b3-4102-aeff-aad2292ab01c
                                     https://graph.microsoft.com                     d3590ed6-52b3-4102-aeff-aad2292ab01c
                                     https://outlook.office.com                      1fec8e78-bce4-4aaf-ab1b-5451cc387264
                                                                                     27922004-5251-4030-b22d-91ecd9a37ea4
                                                                                     d3590ed6-52b3-4102-aeff-aad2292ab01c
                                     https://outlook.office365.com                   1fec8e78-bce4-4aaf-ab1b-5451cc387264
                                                                                     27922004-5251-4030-b22d-91ecd9a37ea4
                                                                                     d3590ed6-52b3-4102-aeff-aad2292ab01c
                                     https://substrate.office.com                    26a7ee05-5602-4d76-a7ba-eae8b7b67941
                                                                                     27922004-5251-4030-b22d-91ecd9a37ea4
                                                                                     d3590ed6-52b3-4102-aeff-aad2292ab01c
```

Figure 8: Excerpt from Scope Lookup Table.

can then be used to complete an authorisation grant flow for arbitrary client applications, including FOCI clients.

Typically, the disadvantage of abusing SSO is that each time the attacker wants access to some scope that was not authorised for some stolen access token, the attacker needs to request a new signed cookie or otherwise complete an authorisation grant flow again to obtain a new access token with the desired scopes. In the case of FRTs, even if the attacker only had the opportunity to generate a single pre-signed cookie, the attacker can silently exchange the FRT multiple times for new access tokens for other FOCI clients and benefit from their authorised scopes.

4.1.4 Zero Trust

FOCI predates the adoption of the *Zero Trust* security model at Microsoft. The guiding principals of Zero Trust require that client authentication and authorization are based on all available information, client access is limited to least privilege for the shortest duration, and that the client is assumed to be breached, so the blast radius must be minimized (Microsoft, 2022c). The current implementation of FOCI is incompatible with the Zero Trust model. FRTs allow long-term persistent access and privilege escalation relative to the client application. As there is only one “family” of Microsoft first-party client applications means that the level of access afforded by FRTs is not segmented according to the needs of legitimate software that require FOCI to function.

4.1.5 Conditional Access Policies

Conditional access policies still apply to FOCI clients and FRTs. Conditional access policies that require multi-factor authentication, however, do not impede attackers from abusing the legitimately issued FRTs since refresh token grants are always non-interactive, and usually inherit the authentication method claims from the original authorisation grant. Furthermore, conditional access policies based on trusting the device are ineffective when a FOCI client is maliciously authorised by abusing SSO because the request “originates” from the trusted device.

Any conditional access policies (or other controls) based purely on the FOCI client identifiers are trivial to bypass if another FOCI client has consent for the desired scopes.

Refresh token grants are logged in Azure AD non-interactive user sign-ins log. Currently, the non-interactive sign-in log events do not contain details about the client application to which the refresh token was originally issued. This prevents detecting exploitation of FRTs.

4.1.6 Anticompetitive Practices

According to United States Federal Trade Commission (FTC), antitrust laws “prohibit conduct by a single firm that unreasonably restrains competition by creating or maintaining monopoly position” (FTC, 2022). FTC uses a previous Microsoft case as an example for monopolisation (FTC, 2022):

Microsoft was able to use its dominant position in the operating systems market to exclude other software developers and prevent computer makers from installing non-Microsoft browser software to run with Microsoft's operating system software.

FOCI establishes a “family” of first-party Microsoft client applications that are given special treatment compared to third-party client applications in Azure AD. Microsoft does not allow third-party developers to benefit from the FOCI functionality, *i.e.*, designate their own “family” of client. As such, it may have provided Microsoft software with a competitive advantage over third-party software even if the third-party used Azure AD as the identity provider.

4.2 Conclusion

In this paper, we reported our findings related to the non-standard behaviour of certain Azure AD’s first-party clients’ refresh tokens.

We found answers to both research questions. First, we found 16 first-party clients supporting these special type of refresh tokens, called Family Refresh Tokens (FRTs). Second, we found out that the clients supporting FRTs were called Family of Client ID (FOCI) clients, and that the purpose of FRTs is to provide singe-sign-on experience without a separate authentication broker for mobile platforms.

Based on our findings, we recommend Microsoft to publish the list of FOCI clients, so that Azure AD customers can protect their environments accordingly. Further, as FOCI is created for mobile platforms, its usage should be limited to those platforms.

4.3 Limitations

The used data set of ~600 first-party applications is not exhaustive, so the study may not have revealed all FOCI clients. Also, Microsoft is creating new and removing old FOCI applications (MSRC, 2021).

When building the scope lookup table, only a limited number of scopes were used when obtaining tokens. As such, only the scopes that Azure AD automatically adds were returned. Therefore, the list of scopes may not be exhaustive.

4.4 Directions for Future Research

As new the FOCI clients are introduced, the list of known FOCI clients needs to be updated.

The security implications of FOCI clients and FRTs requires more research, especially in the mobile platforms. For instance, studying how FRTs are stored and accessible in mobile devices would be an interesting research target.

REFERENCES

Christensen, Lee. (2020). RequestAADRefreshToken. Retrieved from https://github.com/leechristensen/RequestAADRefreshToken

Cobb, Ryan (2021). [Nov 23rd 2021. Vulnerability report to MSRC. VULN-057712, Case Number 68674.].

Cobb, Ryan, & Gore, Tony. (2022). Abusing Family Refresh Tokens for Unauthorized Access and Persistence in Azure Active Directory. Retrieved from https://github.com/secureworks/family-of-client-ids-research

FTC. (2022). Federal Trade Commission. Monopolization Defined. Retrieved from https://www.ftc.gov/advice-guidance/competition-guidance/guide-antitrust-laws/single-firm-conduct/monopolization-defined

IETF. (2012). The OAuth 2.0 Authorization Framework. Retrieved from https://datatracker.ietf.org/doc/html/rfc6749

IETF. (2013). OAuth 2.0 Threat Model and Security Considerations. Retrieved from https://datatracker.ietf.org/doc/html/rfc6819

IETF. (2015). JSON Web Token (JWT). Retrieved from https://datatracker.ietf.org/doc/html/rfc7519

IETF. (2021). OAuth 2.0 Security Best Current Practice. draft-ietf-oauth-security-topics-19. Retrieved from https://datatracker.ietf.org/doc/html/draft-ietf-oauth-security-topics

Microsoft. (2021a). Microsoft identity platform refresh tokens. Retrieved from https://docs.microsoft.com/en-us/azure/active-directory/develop/refresh-tokens

Microsoft. (2021b). MSAL application.py on Github. Retrieved from https://github.com/AzureAD/microsoft-authentication-library-for-python/blob/3062770948f1961a13767ee85dd7ba664440feb3/msal/application.py

Microsoft. (2021c). OAuth 2.0 and OpenID Connect protocols on the Microsoft identity platform. Retrieved from https://docs.microsoft.com/en-us/azure/active-directory/develop/active-directory-v2-protocols

Microsoft. (2021d). Public client and confidential client applications. Retrieved from https://docs.microsoft.com/en-us/azure/active-directory/develop/msal-client-applications

Microsoft. (2021e). Sign-in logs in Azure Active Directory - preview. Retrieved from https://docs.microsoft.com/en-us/azure/active-directory/reports-monitoring/concept-all-sign-ins

Microsoft. (2021f). What is Azure Active Directory? Retrieved from https://docs.microsoft.com/en-us/azure/active-directory/fundamentals/active-directory-whatis

Microsoft. (2022a). Azure Active Directory Marketplace. Retrieved from https://azuremarketplace.microsoft.com/en-GB/marketplace/apps/category/azure-active-directory-apps

Microsoft. (2022b). What is a Primary Refresh Token? Retrieved from https://docs.microsoft.com/en-us/azure/active-directory/devices/concept-primary-refresh-token

Microsoft. (2022c). Zero Trust Guidance Center. Retrieved from https://docs.microsoft.com/en-us/security/zero-trust/

MSRC (2021). [Email response on Dec 14th 2021 to VULN-057712 Case Number 68674].

NIST. (2016). Computer Security Resource Center. Glossary: Privilege Escalation. Retrieved from https://csrc.nist.gov/glossary/term/privilege_escalation

OpenID Foundation. (2022). Welcome to OpenID Connect. Retrieved from https://openid.net/connect/

Pangrle, Ryan. (2016). Family of Client IDs Support #453. Retrieved from https://github.com/AzureAD/azure-activedirectory-library-for-objc/issues/453

Seb8iaan. (2020). Microsoft Owned Enterprise Applications Overview. Retrieved from https://github.com/Seb8iaan/Microsoft-Owned-Enterprise-Applications/blob/main/Microsoft%20Owned%20Enterprise%20Applications%20Overview.md

Syynimaa, Nestori. (2020). Introducing a new phishing technique for compromising Office 365 accounts. Retrieved from https://o365blog.com/post/phishing/

Syynimaa, Nestori. (2022). *Exploring Azure Active Directory Attack Surface - Enumerating Authentication Methods with Open-Source Intelligence Tools*. Paper presented at the ICEIS - 24th International Conference on Enterprise Information Systems.

Thompson, Jason. (2020). MSAL.PS 4.7.1.1. Retrieved from https://www.powershellgallery.com/packages/MSAL.PS/4.7.1.1/Content/Microsoft.Identity.Client.4.7.1%5Cnetcoreapp2.1%5CMicrosoft.Identity.Client.xml